增广立方体上边独立生成树的并行构造

李夏晶, 程宝雷, 樊建席, 王岩, 李晓瑞

引用本文

李夏晶, 程宝雷, 樊建席, 王岩, 李晓瑞. 增广立方体上边独立生成树的并行构造[J]. 计算机科学, 2024, 51(9): 346-356.

LI Xiajing, CHENG Baolei, FAN Jianxi, WANG Yan, LI Xiaorui. Parallel Construction of Edge-independent Spanning Trees in Augmented Cubes [J]. Computer Science, 2024, 51(9): 346-356.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230700041

增广立方体上边独立生成树的并行构造

李夏晶[1,2]　程宝雷[1,2]　樊建席[1]　王　岩[1,2]　李晓瑞[1,2]
1 苏州大学计算机科学与技术学院　江苏 苏州 215006
2 苏州大学江苏省计算机信息处理技术重点实验室　江苏 苏州 215006
(20215227119@stu.suda.edu.cn)

摘　要　近年来,围绕互连网络的研究工作越来越多。其中独立生成树(Independent Spanning Trees,ISTs)可以应用于信息的可靠传输、并行传输、安全分发以及故障服务器的并行诊断中,因此受到了许多研究者的关注。在一对多广播、可靠通信、多节点广播、容错广播、安全消息分发、IP 快速重路由等网络通信中,边独立生成树(Edge-Independent Spanning Trees,EISTs)发挥着重要作用。n 维增广立方体 AQ_n 是 n 维超立方体 Q_n 的节点对称变型,它具有超立方体及其变型所没有的一些可嵌入性质。然而,目前增广立方体上边独立生成树的构造方法都是串行构造的。文中首先提出了一种并行算法,用于构造以 AQ_n 中的任意节点为根的 $2n-1$ 棵树。然后证明算法得到的 $2n-1$ 棵树是高度为 n 的边独立生成树,算法的时间复杂度为 $O(N)$,其中 N 表示增广立方体中的节点数。最后通过模拟实验来验证了所提方法的准确性。

关键词:互连网络;增广立方体;边独立生成树;并行算法;高度
中图分类号　TP311.12

Parallel Construction of Edge-independent Spanning Trees in Augmented Cubes

LI Xiajing[1,2],CHENG Baolei[1,2],FAN Jianxi[1],WANG Yan[1,2] and LI Xiaorui[1,2]
1 School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China
2 Provincial Key Laboratory for Computer Information Processing Technology,Soochow University,Suzhou,Jiangsu 215006,China

Abstract　In recent years,more and more research work is being conducted around interconnected networks. Independent spanning trees(ISTs) can be used in reliable transmission,parallel transmission,safe distribution of information,and parallel diagnosis of fault servers,and have attracted many researchers' attention. In network communication,such as one-to-all broadcasting,reliable communication,multi-node broadcasting,fault-tolerant broadcasting,secure message distribution,and IP fast rerouting,edge-independent spanning trees(EISTs) play a significant role. The augmented cube of n dimension AQ_n is a node-symmetric variation of the n-dimensional hypercube Q_n,it has some embeddable properties that the hypercube and its variants do not have. However,the current EISTs construction methods in augmented cubes are serial construction. This paper first proposes parallel algorithms for constructing $2n-1$ trees rooted at any node in AQ_n. Then,it proves that the $2n-1$ trees obtained by algorithms are EISTs with the height n,and the algorithms' time complexity is $O(N)$,where N equals the number of nodes in AQ_n. Finally,its accuracy is verified by simulation experiment.

Keywords　Interconnection network,Augmented cube,Edge-independent spanning trees,Parallel algorithm,Height

1　引言

互连网络最近受到了广泛的关注,其体系结构用图 G 表示。目前已经提出了许多用于平衡性能和成本参数的有用拓扑,其中 n 维超立方体是最流行的拓扑之一,用 Q_n 表示,它具有许多优越的性质,如低直径、对称性、可扩展性、递归构造性以及高容错性等[1]。增广立方体是超立方体的变型[2],具有良好的几何特性,并保留了一些优良的超立方体性质,如节点对称性、线性时间的路由和广播过程。通过一些链接扩展超立方体,可以生成一个 n 维的增广立方体,用 AQ_n 表示。增广立方体是 Cayley 图,满足 $(2n-1)$-正则和 $(2n-1)$-连通。Choudum 等证明了 n 维的增广立方体在 2^n-1 个节点上包含

到稿日期:2023-07-06　返修日期:2023-11-02
基金项目:国家自然科学基金(62272333,62172291);江苏省教育厅未来网络科研基金资助(FNSRFP-2021-YB-39);江苏高校优势学科建设工程资助项目
This work was supported by the National Natural Science Foundation of China(62272333,62172291),Jiangsu Province Department of Education Future Network Research Fund Project(FNSRFP-2021-YB-39) and Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.
通信作者:程宝雷(chengbaolei@suda.edu.cn)

以相同节点为根的两棵边不相交的完全二叉树、两棵边不相交的生成二叉树,以及所有 k 循环($3\leqslant k\leqslant 2n$)[2]。超立方体及其变型则不具有这些属性。

边独立生成树在互连网络中发挥着重要作用,例如一对多广播[3]、可靠通信协议[4-5]、多节点广播[6]、容错广播、安全消息分发[4]、IP 快速重新路由[7-8]和条件 BC 网络[9]。根据容错广播协议[5,10-11],假设在图 G 中有 n 棵以相同源节点 u 为根的边独立生成树。如果根节点 u 需要将消息广播到所有其他节点,则 G 中的每个节点将通过 n 棵边独立生成树发送消息来获取 n 个消息副本。对于每个无故障源节点,此技术最多可以容忍 $n-1$ 条故障边。因此,在给定的互连网络中构造多棵边独立生成树受到了广泛关注。

关于图[10,12]中边独立生成树的存在性,有对于边的猜想。

猜想 1 如果图 G 是 n-边连通的($n\geqslant 1$),则以 G 中的任意节点为根有 n 棵边独立生成树。

然而,猜想 1 仅能在 $n\leqslant 5$[4,13-15]的 n-边连通图中得到解决。同时,该猜想已被证明适用于许多特殊图类,如增广立方体[16]、偶数网络[17]、奇数网络[18]、乘积图[19]、多维环绕网络[20]等。

n 维的增广立方体 AQ_n 既是 $(2n-1)$-边连通又是 $(2n-1)$-节点连通,因此受到了很多研究人员的关注[16,21-23]。由于边连通性为 $2n-1$,因此以 AQ_n 中任意节点为根的边独立生成树的最多数量为 $2n-1$。显然,低高度的边独立生成树具有高性能,并且有助于设计高效的广播算法。Wang 等[16]解决了 AQ_n 上的边猜想。但边独立生成树集合的最大高度大于 n,且边独立生成树集合的高度不相等。Cheng 等[23]随后解决了边独立生成树集合高度不相等的问题,并且提出了串行算法来构造 $2n-1$ 棵边独立生成树,所有树的高度均为 n,算法的时间复杂度为 $O(N\log N)$。这促使我们思考设计一种时间复杂度低的并行方法来构造 $2n-1$ 棵边独立生成树。

本文提出了一种并行算法来构造 $2n-1$ 棵边独立生成树 $T_1,T_2,\cdots,T_{2n-1}$,根节点为 AQ_n 中的任意节点 u,对于任何整数 $n\geqslant 1$,所有边独立生成树的高度都为 n。该并行算法的时间复杂度为 $O(N)$,其中 N 等于 2^n,是 AQ_n 中的节点数。

2 预备知识

本章主要介绍相关术语和符号定义以及增广立方体 AQ_n 的结构和性质。

2.1 术语和符号

用图 $G(V(G),E(G))$表示互连网络,其中 $V(G)$表示节点集合,$E(G)$表示边集合。给定一个简单图 G,图 G_1 和图 G_2 的并集用 $G_1\cup G_2$ 表示,节点集合为 $V(G_1)\cup V(G_2)$,边集合为 $E(G_1)\cup E(G_2)$。如果 W 是 $V(G)$的非空子集,则由集合 W 导出的 G 的子图表示为 $G[W]$。令 $x,y\in V(G)$。从 x 到 y 的路径 R 表示为 $x-y$ 路径。我们使用 $V(R)$和 $E(R)$分别表示 R 中的节点集合和边集合。假设 P 和 Q 为两个 $x-y$ 路径,如果 $E(P)\cap E(Q)=\emptyset$,则路径 P 和 Q 是边不相交的。我们使用 $H(T)$来表示树 T 的高度。如果 T_1 中的 $u-v$ 路径和 T_2 中的 $u-v$ 路径对于每个 $v\in V(G)\backslash\{u\}$是边不相交的,则图 G 中以节点 u 为根的两棵生成树 T_1 和 T_2 是边不相交的。设以节点 u 为根的 $T_1,T_2,\cdots,T_k$ 是图 G 的生成树,如果在 $T_1,T_2,\cdots,T_k$ 中从 u 到 $v(u\neq v,v\in G)$的路径是成对边独立的,则称 $T_1,T_2,\cdots,T_k$ 为图 G 中的边独立生成树(EISTs)。

长度为 n 的二进制字符串 u 表示为 $u_nu_{n-1}\cdots u_1$,其中 u_n 是最高有效位,u_1 是最低有效位。对于一个整数 $i(1\leqslant i\leqslant n)$,我们使用 u_i 来表示字符串 u 的第 i 位,u_i 的补集由 $\bar{u}_i(\bar{0}=1,\bar{1}=0)$表示。给定一个集合 $W\in E(G)$,对于所有 $z\in W$ 且整数 k 满足 $1\leqslant k\leqslant n-1$,如果存在长度为 $k(u_nu_{n-1}\cdots u_{n-k+1})$的二进制字符串 s 是 z 的前缀,则称字符串 s 为集合 W 中所有二进制字符串的共同前缀。

2.2 增广立方体 AQ_n 的结构和性质

定义 1 对于 $n\geqslant 1$,维度为 n 的增广立方体 AQ_n 有 2^n 个节点,每个节点均由 n 位二进制字符串 $u_nu_{n-1}\cdots u_1$ 标记。我们定义 $AQ_1=K_2$。对于 $n\geqslant 2$,AQ_n 是通过增广立方体 AQ_{n-1} 的两个子立方体(用 AQ^0_{n-1} 和 AQ^1_{n-1} 表示)并在两者之间添加 $2\times 2^{n-1}$ 条边来获得的,如下所示:

令 $V(AQ^0_{n-1})=\{0u_{n-1}\ u_{n-2}\cdots u_1\mid u_{n-1}\ u_{n-2}\cdots u_1\in V(AQ_{n-1})\}$且 $V(AQ^1_{n-1})=\{1v_{n-1}\,v_{n-2}\cdots v_1\mid v_{n-1}\,v_{n-2}\cdots v_1\in V(AQ_{n-1})\}$。

AQ^0_{n-1} 中的节点 $u=0u_{n-1}\,u_{n-2}\cdots u_1$ 与 AQ^1_{n-1} 中节点 $v=1v_{n-1}\,v_{n-2}\cdots v_1$ 相邻的条件是:当且仅当对于每个 $1\leqslant i\leqslant n-1$,需要满足下面两个条件之一:

(1)$u_i=v_i$,在这种情形下,边(u,v)被称为超立方体边;

(2)$u_i=\bar{v}_i$,在这种情形下,边(u,v)被称为补边。

显然,Q_n 是 AQ_n 的子图。引理 1 中描述了如何检查给定的一对节点是否在 AQ_n 中相邻,并且可以认为引理 1 是增广立方体的非递归定义。

引理 1[23] 对于所有 $n\geqslant 1$,$(u_nu_{n-1}\cdots u_1,v_nv_{n-1}\cdots v_1)$是一条边当且仅当存在整数 q,且 $1\leqslant q\leqslant n$ 使得:

(1)对于每个 $i,i\neq q,u_q=\bar{v}_q$且 $u_i=v_i$;

或

(2)对于 $q+1\leqslant i\leqslant n,u_i=v_i$ 并且对于 $1\leqslant i\leqslant q,u_i=\bar{v}_i$。

如果引理 1 的条件(1)和条件(2)都成立,则节点 u 和节点 v 在位置 q 有一个最左不同位。如果两个相邻节点 u 和 v 在位置 m 处有一个最左不同位,且 $u_{m-1}u_{m-2}\cdots u_1=v_{m-1}v_{m-2}\cdots v_1$,则节点 v 是节点 u 的 m-邻居,边(u,v)是一条维度为 m 的边。我们使用 $N(u,m)$表示节点 u 的 m-邻居。如果两个相邻节点 x 和 y 在位置 m 处有一个最左不同位,且 $x_{m-1}x_{m-2}\cdots x_1=\overline{y_{m-1}y_{m-2}\cdots y_1}$,则节点 y 是节点 x 的 m-补-邻居,边(x,y)是一条维度为 m 的补边。我们使用 $N^*(x,m)$表示节点 x 的 m-补-邻居。增广立方体 AQ_1,AQ_2,AQ_3 和 AQ_4 如图 1 所示。

引理 2[23] $AQ_n(s)$同构于 $AQ_{n-|s|}$,其中$|s|$表示字符串 s 的长度。对于整数 n 且 $n\geqslant 2$,如果节点 $v=v_nv_{n-1}\cdots v_1$ 是 AQ_n 中的任意节点,那么节点 v 属于 $AQ_n(v_nv_{n-1}\cdots v_{n-i})$的子图,其中 $0\leqslant i\leqslant n-2$。$AQ_n(v_nv_{n-1}\cdots v_{n-i})$可以分为两个子图 $AQ_n(v_nv_{n-1}\cdots v_{n-i})^{v_{n-i-1}}_{n-i-1}$ 和 $AQ_n(v_nv_{n-1}\cdots v_{n-i})^{\overline{v_{n-i-1}}}_{n-i-1}$,例如 $AQ_4(1)$可以分为 $AQ_4(1)^0_3$ 和 $AQ_4(1)^1_3$。

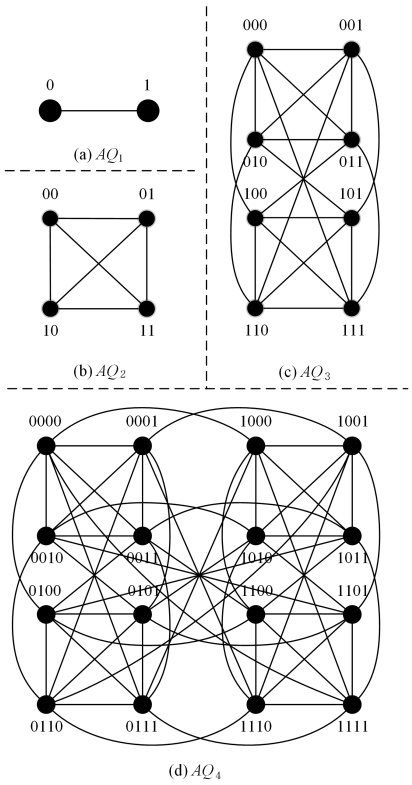


图 1 4 个增广立方体

Fig. 1 Four augmented cubes

3 增广立方体 AQ_n 中边独立生成树的并行构造算法

本章主要研究在 AQ_n 中构造边独立生成树的并行算法，并对算法正确性给出了证明，最后分析了算法的时间复杂度。下文中使用 $r=r_nr_{n-1}\cdots r_1$ 来表示 AQ_n 中的任意根节点。

3.1 AQ_n 中 T_1 与 T_2 的构造算法

算法 1 给出了在 $AQ_n(n\geqslant 2)$ 中构造第一棵树 T_1 与第二棵树 T_2 的过程。

算法 1 EIST1

输入：代表增广立方体维度的整数 n 和 AQ_n 的根节点 $r=r_nr_{n-1}\cdots r_1$

输出：AQ_n 中边独立生成树集合中的 T_1 和 T_2

步骤 1：/ * 构造具有共同前缀为 $r_nr_{n-1}\cdots r_3$ 的子树 * /

1. $V(T_1)=\{r,N(r,1),N(r,2),N^*(r,2)\}$ and $E(T_1)=\{(r,N(r,1)),(N(r,1),N(r,2)),(N(r,1),N^*(r,2))\}$.
2. $V(T_2)=\{r,N(r,2),N(r,1),N^*(r,2)\}$ and $E(T_2)=\{(r,N(r,2)),(N(r,2),N(r,1)),(N(r,2),N^*(r,2))\}$.

步骤 2：/ * 构造具有共同前缀为 $r_nr_{n-1}\cdots r_k(k\geqslant 4)$ 的子树 * /

3. for k=4 to n+1 do
4. $V(T_1')\leftarrow V(T_1),V(T_2')\leftarrow V(T_2),E(T_1')\leftarrow E(T_1),E(T_2')\leftarrow E(T_2)$.
5. $V(T_1'')\leftarrow\{N(v,k-1)|v\in V(T_1')\},E(T_1'')\leftarrow\{(N(x,k-1),N(y,k-1))|(x,y)\in E(T_1')\}$.
6. $V(T_2'')\leftarrow\{N(v,k-1)|v\in V(T_2')\},E(T_2'')\leftarrow\{(N(x,k-1),N(y,k-1))|(x,y)\in E(T_2')\}$.
7. $V(T_1)\leftarrow V(T_1')\cup V(T_1''),E(T_1)\leftarrow E(T_1')\cup E(T_1'')\cup(N(r,1),N(N(r,1),k-1))$.
8. $V(T_2)\leftarrow V(T_2')\cup V(T_2''),E(T_2)\leftarrow E(T_2')\cup E(T_2'')\cup(N(r,2),N(N(r,2),k-1))$.
9. end for
10. return T_1, T_2.
11. end function

在算法 EIST1 中，通过步骤 1，我们构造了具有共同前缀 $r_nr_{n-1}\cdots r_3$ 的子树。通过步骤 2，我们构造了具有共同前缀 $r_nr_{n-1}\cdots r_k(n+1\geqslant k\geqslant 4)$ 的子树。T_1' 和 T_2' 分别代表由第一步构造的子树 T_1 和 T_2，T_1'' 和 T_2'' 代表将 T_1' 和 T_2' 中所有节点取 $(k-1)$-邻居得到的子树。连接 T_1' 和 T_1'' 中根的孩子节点构造 T_1，连接 T_2' 和 T_2'' 中根的孩子节点构造 T_2，直至整数 k 等于 $n+1$。

基于算法 EIST1，我们在例 1 中给出了 AQ_4 中 T_1 和 T_2 的详细构造过程。

例 1 下面以在 AQ_4 中构造 T_1 和 T_2 为例，展示算法 EIST1 的过程。图 2 和图 3 展示了在 AQ_4 中以 0000 为根，构造 T_1 和 T_2 的过程。在图 2(a)中，构造了具有共同前缀 r_4r_3 的子树。在图 2(b)中，T_1' 代表子树 T_1，将子树 T_1' 中的所有节点取 3-邻居得到子树 T_1''，并连接子树 T_1' 和子树 T_1'' 根节点的孩子节点，从而构造了具有共同前缀 r_4 的子树。图 2(c)展示的构造方法与图 2(b)相同，且此时共同前缀为 $r_nr_{n-1}\cdots r_k(n+1\geqslant k\geqslant 4)$，满足整数 k 等于 $n+1$，即 k 等于 5，得到一棵树完整的树 T_1。T_2 的构造过程与 T_1 相同。

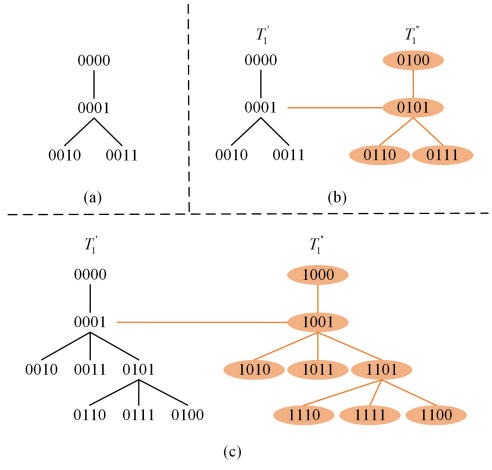


图 2 AQ_4 中 T_1 的构造过程

Fig. 2 Construction process of T_1 in AQ_4

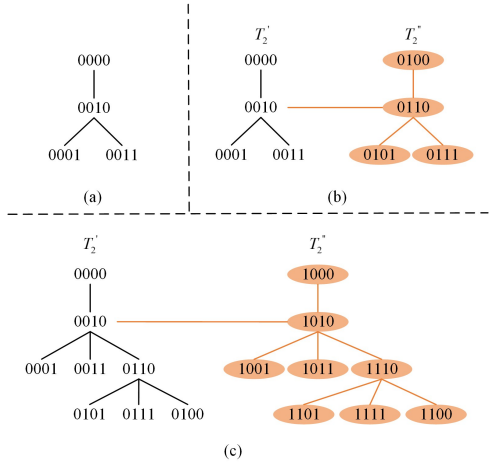


图 3 AQ_4 中 T_2 的构造过程

Fig. 3 Construction process of T_2 in AQ_4

引理 3 通过算法 1 得到的两棵树为 T_1 和 T_2,它们是 AQ_n 中的生成树且高度均为 n,其中 $n\geqslant 2$。

证明:通过算法 EIST1 的步骤 1,构造共同前缀为 $r_nr_{n-1}\cdots r_3$ 的子树,可得 $V(T_1)=V(T_2)=\{r,N(r,1),N(r,2),N^*(r,2)\}$,$E(T_1)=\{(r,N(r,1)),(N(r,1),N(r,2)),(N(r,1),N^*(r,2))\}$,$E(T_2)=\{(r,N(r,2)),(N(r,2),N(r,1)),(N(r,2),N^*(r,2))\}$。通过引理 1,可以验证 $E(T_1)$ 和 $E(T_2)$ 是 $AQ_n(r_nr_{n-1}\cdots r_3)$ 中的边集合,并且 $E(T_1)$ 和 $E(T_2)$ 边不相交。因此 T_1 和 T_2 是 $AQ_n(r_nr_{n-1}\cdots r_3)$ 中的生成树,且 $H(T_1)=H(T_2)=2$。

通过算法 EIST1 的步骤 2,对于 $k=4,5,\cdots,n+1$,构造共同前缀为 $r_nr_{n-1}\cdots r_k$ 的子树,并且有以下 2 种情形:

情形 1 当 $n=2$ 时,即在步骤 1 中构造的树为 AQ_n 的生成树,因此 $H(T_1)=H(T_2)=2=n$。

情形 2 当 $n>2$ 时,可以验证以下几个结果:

(1)$T_1'=T_1$,$T_2'=T_2$;(第一次循环中的 T_1,T_2 由步骤 1 得到);

(2)$T_1''=N(T_1',k-1)$,$T_2''=N(T_2',k-1)$;

(3)$T_1=T_1'\cup T_1''\cup(N(r,1),N(N(r,1),k-1))$,$T_2=T_2'\cup T_2''\cup(N(r,2),N(N(r,2),k-1))$。

可以得知,在第一次循环中(即 $k=4$),T_1' 和 T_2' 表示经过步骤 1 得到的子树 T_1 和 T_2,因此 $E(T_1')$ 和 $E(T_2')$ 边不相交,子树 T_1'' 和 T_2'' 是通过将 T_1' 和 T_2' 中所有节点取 $(k-1)$-邻居而得,很明显 $E(T_1'')$ 和 $E(T_2'')$ 也是边不相交。因此,T_1' 和 T_2' 是 $AQ_n(r_nr_{n-1}\cdots r_k)_{k-1}^{r_{k-1}}$ 中的子树,T_1'' 和 T_2'' 是 $AQ_n(r_nr_{n-1}\cdots r_k)_{k-1}^{\overline{r_{k-1}}}$ 中的子树。根据引理 2 可知,T_1' 和 T_1'' 同构,T_2' 和 T_2'' 同构,故 $H(T_1'')=H(T_2'')$。连接 T_1' 和 T_1'' 中根节点的孩子节点,从而得到 $AQ_n(r_nr_{n-1}\cdots r_k)$ 中的子树 T_1;连接 T_2' 和 T_2'' 中根节点的孩子节点,从而得到 $AQ_n(r_nr_{n-1}\cdots r_k)$ 中的子树 T_2。并且 $H(T_1)$ 和 $H(T_2)$ 会增加高度 1,即 $H(T_1)=H(T_2)=k-1$,且 $E(T_1)$ 和 $E(T_2)$ 边不相交。重复算法 1 中的步骤 2,直到 k 等于 $n+1$,即可得到 AQ_n 中的生成树 T_1 和 T_2,$H(T_1)=H(T_2)=n$,并且 T_1 和 T_2 满足边不相交。

3.2 AQ_n 中 T_3 的构造算法

算法 2 给出了在 $AQ_n(n\geqslant 2)$ 中构造第三棵树 T_3 的过程。

算法 2 EIST2

输入:代表增广立方体维度的整数 n 和 AQ_n 的根节点 $r=r_nr_{n-1}\cdots r_1$

输出:AQ_n 中边独立生成树集合中的 T_3

步骤 1:/* 构造具有共同前缀为 $r_nr_{n-1}\cdots r_3$ 的子树 */

1. $V(T_3)=\{r,N^*(r,2),N(r,1),N(r,2)\}$ and $E(T_3)=\{(r,N^*(r,2)),(N^*(r,2),N(r,1)),(N^*(r,2),N(r,2))\}$.

步骤 2:/* 构造具有共同前缀为 $r_nr_{n-1}\cdots r_4$ 的子树 */

2. $V(T_3)\leftarrow V(T_3)\cup N(N^*(r,2),3)\cup\{N^*(v,3)|v\in V(T_3)\backslash r\}$.

3. $E(T_3)\leftarrow E(T_3)\cup(N^*(r,2),N(N^*(r,2),3))\cup(N^*(r,2),N^*(N^*(r,2),3))\cup(N(r,1),N^*(N(r,1),3))\cup(N(r,2),N^*(N(r,2),3))$

步骤 3:/* 构造具有共同前缀为 $r_nr_{n-1}\cdots r_k(k\geqslant 5)$ 的子树 */

4. for k=5 to n+1 do

5. $V(T_3')\leftarrow V(T_3)$,$E(T_3')\leftarrow E(T_3)$.

6. $V(T_3'')\leftarrow\{N(v,k-1)|v\in V(T_3')\}$,$E(T_3'')\leftarrow\{N(x,k-1),N(y,k-1)|(x,y)\in E(T_3')\}$.

7. $V(T_3)\leftarrow V(T_3')\cup V(T_3'')$,$E(T_3)\leftarrow E(T_3')\cup E(T_3'')\cup(N^*(r,2),N(N^*(r,2),k-1))$.

8. end for

9. return T_3.

10. end function

在算法 EIST2 中,通过步骤 1,构造具有共同前缀 $r_nr_{n-1}\cdots r_3$ 的子树;通过步骤 2,构造具有共同前缀 $r_nr_{n-1}\cdots r_4$ 的子树。将根节点的孩子节点 $N^*(r,2)$ 连接节点 $N(N^*(r,2),3)$,并且将步骤 1 构造的子树中除根节点以外的所有节点 v 连接节点 $N^*(v,3)$。通过步骤 3,构造具有共同前缀 $r_nr_{n-1}\cdots r_k(n+1\geqslant k\geqslant 5)$ 的子树。T_3' 代表由步骤 2 构造的子树 T_3,T_3'' 代表将 T_3' 中所有节点取 $(k-1)$-邻居得到的子树。连接 T_3' 和 T_3'' 中根的孩子节点构造 T_3,直至整数 k 等于 $n+1$。

基于算法 EIST2,通过例 2 给出了 AQ_4 中 T_3 的详细构造过程。

例 2 下面以在 AQ_4 中构造 T_3 为例,展示了算法 EIST2 的过程。图 4 展示了在 AQ_4 中以 0000 为根,构造 T_3 的过程。在图 4(a)中,构造了具有共同前缀 r_4r_3 的子树;在图 4(b)中,构造了具有共同前缀 r_4 的子树;在图 4(c)中,使用 T_3' 代表子树 T_3,T_3'' 是通过将 T_3' 中所有节点取 4-邻居而得到的。连接子树 T_3' 和子树 T_3'' 根节点的孩子节点,且此时共同前缀 $r_nr_{n-1}\cdots r_k(n+1\geqslant k\geqslant 5)$,满足整数 k 等于 $n+1$,即 $k=5$,从而得到一棵完整的树 T_3。

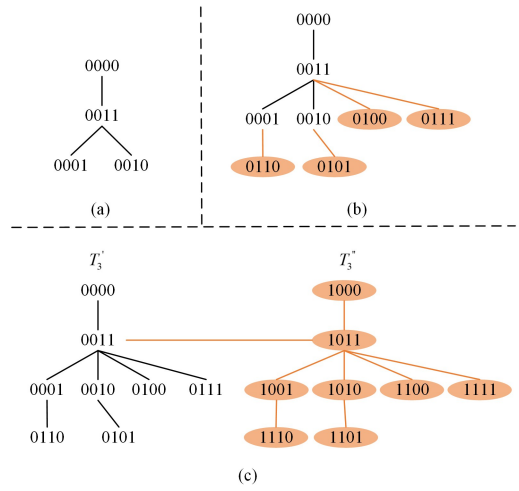


图 4 AQ_4 中 T_3 的构造过程

Fig. 4 Construction process of T_3 in AQ_4

引理 4 通过算法 2 得到的树为 T_3,则它是 AQ_n 中的生成树且高度为 n,其中 $n\geqslant 2$。

证明:通过算法 EIST2 的步骤 1,构造共同前缀为 $r_nr_{n-1}\cdots r_3$ 的子树,可得 $V(T_3)=\{r,N(r,1),N(r,2),N^*(r,2)\}$,$E(T_3)=\{(r,N^*(r,2)),(N^*(r,2),N(r,1)),(N^*(r,2),N(r,2))\}$。通过引理 1,容易验证 $E(T_3)$ 是 $AQ_n(r_nr_{n-1}\cdots r_3)$ 中的边集合。因此,T_3 是 $AQ_n(r_nr_{n-1}\cdots r_3)$ 中的生成树,

且 $H(T_3)=2$。

通过算法 EIST2 的步骤 2，构造共同前缀为 $r_nr_{n-1}\cdots r_4$ 的子树，并且有以下 2 种情形：

情形 1 当 $n=2$ 时，即在步骤 1 中构造的树为 AQ_n 的生成树，因此 $H(T_3)=2=n$。

情形 2 当 $n>2$ 时，连接根节点的孩子节点 $N^*(r,2)$ 与节点 $N(N^*(r,2),3)$，且将 T_3 中除根节点以外的所有节点与其 3-补-邻居节点连接，$H(T_3)$ 会增加高度 1。很显然，T_3 是 $AQ_n(r_nr_{n-1}\cdots r_4)$ 中的生成树，且 $H(T_3)=3=n$。

通过算法 EIST2 的步骤 3，对于 $k=5,6,\cdots,n+1$，构造共同前缀为 $r_nr_{n-1}\cdots r_k$ 的子树，并且有以下 2 种情形：

情形 1 当 $n=3$ 时，即在步骤 2 中构造的树为 AQ_n 的生成树，则 $H(T_3)=3$。

情形 2 当 $n>3$ 时，容易验证以下几种结果：

(1) $T_3'=T_3$；(第一次循环中的 T_3 由步骤 2 得到)

(2) $T_3''=N(T_3',k-1)$；

(3) $T_3=T_3'\cup T_3''\cup(N^*(r,2),N(N^*(r,2),k-1))$。

可以得知，在第一次循环中($k=5$)，T_3' 表示步骤 2 得到的 T_3，T_3'' 是将 T_3' 中所有节点取 $k-1$ 位的补集而得到的。因此，T_3' 是 $AQ_n(r_nr_{n-1}\cdots r_k)_k^{\overline{r_{k-1}}}$ 中的生成树，T_3'' 是 $AQ_n(r_nr_{n-1}\cdots r_k)_{k-1}^{\overline{r_{k-1}}}$ 中的生成树。根据引理 2 可知，T_3' 和 T_3'' 同构，故 $H(T_3')=H(T_3'')$。连接 T_3' 和 T_3'' 中根节点的孩子节点，从而得到 $AQ_n(r_nr_{n-1}\cdots r_k)$ 中的子树 T_3，并且 $H(T_3)$ 会增加高度 1，即 $H(T_3)=k-1$。重复步骤 3，直到 $k-n+1$，即可得到 AQ_n 中的生成树 T_3，$H(T_3)=n$。

3.3 AQ_n 中 T_{2k-2} 的构造算法

下面给出在 $AQ_n(n\geqslant 3)$ 中构造第 $2k-2(k=3,4,\cdots,n)$ 棵树的算法，我们使用 T_{2k-2} 来表示。

定义 2[23] 对于整数 $n\geqslant 3$ 且 $3\leqslant k\leqslant n$，假设 T_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots r_{k+1})$ 中以节点 r 为根的一棵生成树，那么存在一个可扩展节点集合 $W_{2k-2}=V(AQ_n(r_nr_{n-1}\cdots r_{k+1})_k^{\overline{r_k}})\setminus\{N^*(r,k)\}$。

首先给出算法 FCPN(k,v)，通过算法可以得到与节点 v 具有共同前缀 $v_nv_{n-1}\cdots v_k$ 的节点集，并且它们都与节点 v 相邻。

算法 3 FCPN(k,v)

输入：整数 k(k=3,4,…,n)和节点 $v=v_nv_{n-1}\cdots v_1(v\in AQ_n)$

输出：节点集合 CP

```
1. CP=∅, V1, V2=∅.
2. for i=1 to k-1 do
3.    v1=N(v,i), V1=V1∪{v1}.
4. end for
5. for j=2 to k-1 do
6.    v2=N*(v,j), V2=V2∪{v2}.
7. end for
8. CP=V1∪V2.
9. return CP.
10. end function
```

引理 5 在算法 3 输入整数 $k(k=3,4,\cdots,n)$ 和节点 $v(v\in AQ_n)$，得到的是与节点 v 具有共同前缀 $v_nv_{n-1}\cdots v_k$ 的节点集合。

证明：输入一个节点 v，对于 $1\leqslant i\leqslant k-1$，$v_1=N(v,i)=v_nv_{n-1}\cdots v_k\cdots\overline{v_i}\cdots v_1$，即节点 v 的 i-邻居。对于 $2\leqslant j\leqslant k-1$，$v_2=N^*(v,j)=v_nv_{n-1}\cdots v_k\cdots\overline{v_j\cdots v_1}$，即节点 v 的 j-补-邻居。因此，我们可以获得一个共同前缀为 $v_nv_{n-1}\cdots v_k$ 的节点集合，且都与节点 v 互为邻居关系。

接下来给出算法 ADDNodes(k,v,T)，通过该算法可以将 $AQ_n(r_nr_{n-1}\cdots r_{k+1})_k^{\overline{r_k}}$ 中没有被连接在子树中的所有节点添加到子树中。

算法 4 ADDNodes(k,v,T)

输入：整数 k，节点 $v=v_nv_{n-1}\cdots v_1(v\in AQ_n)$和一棵子树

输出：添加节点集合后的子树

```
1. w=vnvn-1…v1.
2. for i=3 to k-1 do
3.    vtemp=N(w,i)
4.    CP=FCPN(i,vtemp).
5.    for each node y in CP do
6.       if y∈V(T) then
7.          delete y from CP.
8.       end if
9.    end for
10.    V(T)=V(T)∪CP and E(T)=E(T)∪{(vtemp,x)|x∈CP}.
11.    if i-3>0 then
12.       call ADDNodes(i,vtemp,T).
13.       end if
14. end for
15. return T.
16. end function
```

算法 5 给出了在 $AQ_n(n\geqslant 3)$ 中构造第 $2k-2$ 棵树 T_{2k-2} 的过程。

算法 5 EIST3

输入：代表增广立方体维度的整数 n；AQ_n 的根节点 $r=r_nr_{n-1}\cdots r_1$；整数 k=3,4,…,n；与 k 值对应的 ENS 集合 $W_4,W_6,\cdots,W_{2k-2}$

输出：AQ_n 中边独立生成树集合中的 T_{2k-2}

步骤 1 /* 构造具有共同前缀为 $r_nr_{n-1}\cdots r_{k+1}$ 的子树 */

步骤 1.1 /* 连接根节点 r 与节点 N(r,k)，并且将与节点 N(r,k)具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}\overline{r_k}$ 的所有邻居节点连接到 N(r,k) */

```
1. w=N(r,k).
2. CP1=FCPN(k,w).
3. V(T2k-2)={r,w}∪CP1 and E(T2k-2)=(r,w)∪{(w,v)|v∈CP1}.
```

步骤 1.2 /* 将 AQ_n 中具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}\overline{r_k}$ 的剩余所有节点添加到子树中 */

```
4. if k>3 then
5.    call ADDNodes(k,w,T2k-2).
6. end if
```

步骤 1.3 /* 将 AQ_n 中具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}r_k$ 的所有节点添加到子树中 */

```
7. V(T2k-2)←V(T2k-2)∪{N*(v,k)|v∈W2k-2}.
```

8. $E(T_{2k-2})\leftarrow E(T_{2k-2})\cup\{(v,N^*(v,k))\mid v\in W_{2k-2}\}$.

步骤 2 /*对于 $k_2=k+1,k+2,\cdots,n$,构造具有共同前缀为 $r_nr_{n-1}\cdots r_{k_1+1}$ 的子树*/

9. if $n\geqslant k+1$ then

10. for $k_2=k+1$ to n do

11. $V(T'_{2k-2})\leftarrow V(T_{2k-2}),E(T'_{2k-2})\leftarrow E(T_{2k-2})$.

12. $V(T''_{2k-2})\leftarrow\{N(v,k_2)\mid v\in V(T'_{2k-2})\},E(T''_{2k-2})\leftarrow\{(N(x,k_2),N(y,k_2))\mid(x,y)\in E(T'_{2k-2})\}$.

13. $V(T_{2k-2})\leftarrow V(T'_{2k-2})\cup V(T''_{2k-2})$.

14. $E(T_{2k-2})\leftarrow E(T'_{2k-2})\cup E(T''_{2k-2})\cup\{(v,N(v,k_2))\mid v\in W_{2k-2}\}\backslash E(T''_{2k-2}[N(W_{2k-2},k_2)])$.

15. end for

16. end if

17. return T_{2k-2}.

18. end function

在算法 EIST3 中,通过步骤 1,我们首先构造具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}$ 的子树。在步骤 1.1 中,连接根节点 r 和节点 $w=N(r,k)=r_nr_{n-1}\cdots r_{k+1}\overline{r_k}\cdots r_1$,并且调用算法 3,将与节点 w 具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}\overline{r_k}$ 的所有邻居节点连接到点 w。在步骤 1.2 中,若 $k>3$,则调用算法 4,将具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}\overline{r_k}$ 的所有节点添加到子树 T_{2k-2} 中,此时 $AQ_n(r_nr_{n-1}\cdots r_{k+1})^{\overline{r_k}}_k$ 中的所有节点都被连接在子树 T_{2k-2} 中,即此时 $V(T_{2k-2})=\{r\}\cup V(AQ_n(r_nr_{n-1}\cdots r_{k+1})^{\overline{r_k}}_k)$。在步骤 1.3 中,将除根节点 r 与节点 $N^*(r,k)$ 以外的所有节点 v,即 $v\in W_{2k-2}(W_{2k-2}=V(AQ_n(r_nr_{n-1}\cdots r_{k+1})^{\overline{r_k}}_k)\backslash\{N^*(r,k)\}=V(T_{2k-2})\backslash\{r,N^*(r,k)\})$,与 $N^*(v,k)$ 连接,由此可以得到具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}$ 的子树。在步骤 2 中,若满足 $n\geqslant k+1$,令 $k_2=k+1,k+2,\cdots,n$,构造共同前缀为 $r_nr_{n-1}\cdots r_{k_1+1}$ 的子树。首先 T'_{2k-2} 表示由步骤 1 得到的子树 T_{2k-2},T''_{2k-2} 代表将 T'_{2k-2} 中所有节点取 k_2-邻居节点得到的子树。将集合 W_{2k-2} 中的节点与集合 $N(W_{2k-2},k_2)$ 中对应的节点相连,即合并 T'_{2k-2} 与 T''_{2k-2},并且将子树 T''_{2k-2} 中 $N(W_{2k-2},k_2)$ 互相连接的边断开,重复以上步骤直到 $k_2=n$,即可得到完整的生成树 T_{2k-2}。

基于算法 EIST3,通过例 3 给出了 AQ_4 中 T_{2k-2} 的详细构造过程。

例 3 下面以在 AQ_4 中构造 T_4 和 T_6 为例,展示算法 EIST3 的过程。图 5 和图 6 展示了在 AQ_4 中以 0000 为根节点构造 T_4 和 T_6 的过程。

首先分析 $T_4(k=3)$ 的构造过程,在图 5(a)中,首先连接根节点 $r=0000$ 与节点 $w=N(r,3)=0100$,其次调用算法 3,将节点 w 和与其共同前缀为 01 的所有邻居节点连接起来,即 $N(w,1)=0101$,$N(w,2)=0110$,$N^*(w,2)=0111$。在图 5(b)中,除根节点 0000 和节点 $N^*(w,2)=0111$ 外的所有节点,即节点 0100,0101,0110 与它们的 3-补-邻居节点相连,从而构造了共同前缀为 0 的子树。在图 5(c)中,首先用 T_4' 代表上一步构造的子树,其次取 T_4' 中所有节点的 4-邻居节点,并且用 T_4'' 来表示。由定义 3 可知 $V(W_4)=\{0100,0101,0110\}$,$E(W_4)=\{(0100,0101),(0100,0110)\}$,则 $V(N(W_4,4))=\{1100,1101,1110\}$,$E(N(W_4,4))=\{(1100,1101),(1100,1110)\}$。将 W_4 与对应的 $N(W_4,4)$ 进行连接,即连接节点 0100 与 1100,连接节点 0101 与 1101,连接节点 0110 与 1110,并且将 $N(W_4,4)$ 中的边断开,最终得到一棵完整的生成树 T_4。

接着分析 $T_6(k=4)$ 的构造过程,在图 6(a)中,首先连接根节点 $r=0000$ 与节点 $w=N(r,4)=1000$,其次调用算法 3,将节点 w 和与其共同前缀为 1 的所有邻居节点连接起来,即 $N(w,1)=1001$,$N(w,2)=1010$,$N(w,3)=1100$,$N^*(w,2)=1011$,$N^*(w,3)=1111$。在图 6(b)中,因为此时满足 $k=4>3$,因此调用算法 4,将共同前缀为 1 的所有节点添加到子树中。由算法 4 可知,将节点 1101 与 1110 连接到节点 1100 上,即满足 $V(T_6)=\{r\}\cup V(AQ_3^1)$。在图 6(c)中,除根节点 0000 和节点 $N^*(w,3)=1111$ 外的所有节点,即节点 1000,1001,1010,1100,1011,1101,1110 与它们的 4-补-邻居节点相连,最终得到一棵完整的生成树 T_6。

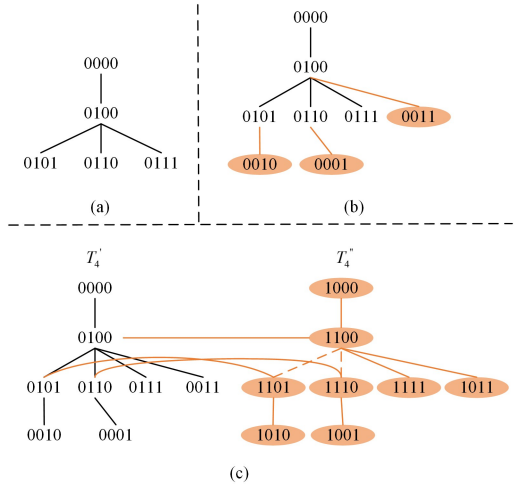


图 5 AQ_4 中 T_4 的构造过程

Fig. 5 Construction process of T_4 in AQ_4

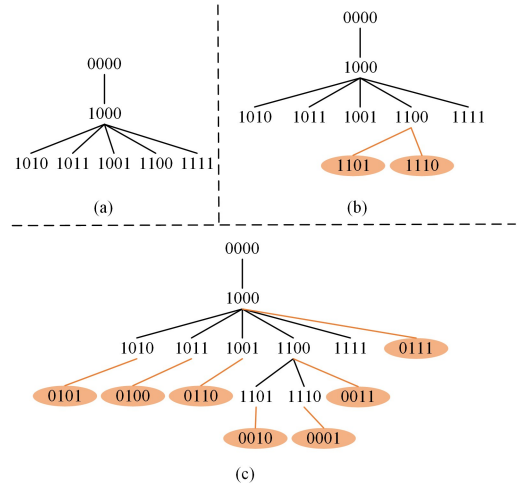


图 6 AQ_4 中 T_6 的构造过程

Fig. 6 Construction process of T_6 in AQ_4

引理 6 通过算法 EIST3 得到的树为 $T_{2k-2}(k=3,4,\cdots,n)$,则 T_{2k-2} 是 AQ_n 中的生成树且高度为 n,其中 $n\geqslant 3$。

证明:通过算法 EIST3 的步骤 1,构造共同前缀为 $r_nr_{n-1}\cdots r_{k+1}$ 的子树。在步骤 1.1 中,连接根节点 r 和节点

$w=N(r,k)=r_nr_{n-1}\cdots r_{k+1}\overline{r_k}\cdots r_1$，可得 $V(T_{2k-2})=\{r,w\}$，$E(T_{2k-2})=\{(r,w)\}$，并且 $H(T_{2k-2})=1$。接着调用算法 3，将与节点 w 具有共同前缀 $r_nr_{n-1}\cdots r_{k+1}\overline{r_k}$ 的所有邻居节点连接到点 w，可得 $V(T_{2k-2})=\{r,w,N(w,1),N(w,2),\cdots,N(w,k-1),N^*(w,2),\cdots,N^*(w,k-1)\}$，$E(T_{2k-2})=\{(r,w),(w,N(w,1)),(w,N(w,2)),\cdots,(w,N(w,k-1)),(w,N^*(w,2)),\cdots,(w,N^*(w,k-1))\}$，并且 $H(T_{2k-2})=2$。

在步骤 1.2 中，给予了条件 $k>3$ 时，执行算法 4，由于算法 4 是将 $AQ_n(r_nr_{n-1}\cdots r_{k+1})_k^{\overline{r_k}}$ 中没有被连接在子树中的所有节点添加到子树中，这里将分为 2 种情形进行证明：

情形 1　当 $k=3$ 时，通过步骤 1.1，已经将 $AQ_n(r_nr_{n-1}\cdots r_4)_3^{\overline{r_3}}$ 中的所有节点添加到子树中，即 $V(T_4)=\{r,w,N(w,1),N(w,2),N^*(w,2)\}$，此时满足 $V(T_{2k-2})=\{r\}\cup AQ_n(r_nr_{n-1}\cdots r_4)_3^{\overline{r_3}}$，并且 T_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots\overline{r_3})\cup\{r\}$ 中的生成树，树的高度为 2，即 $H(T_{2k-2})=k-1$。因此，当 $k=3$ 时，可以跳过步骤 1.2。

情形 2　当 $k>3$ 时，此时不满足 $V(T_{2k-2})=\{r\}\cup AQ_n(r_nr_{n-1}\cdots r_{k+1})_k^{\overline{r_k}}$，因此执行算法 4，将不在子树中的节点添加进去。

情形 2.1　当 $k=4$ 时，首先取整数 $i=3$，取节点 w 的 3-邻居节点作为添加节点 $v_{\text{temp}}=N(w,3)=r_nr_{n-1}\cdots\overline{r_4r_3}r_2r_1$（步骤 1.1 中节点 v_{temp} 与节点 w 连接），并通过调用算法 3，寻找与 v_{temp} 共同前缀为 $r_nr_{n-1}\cdots\overline{r_4r_3}$ 的节点集合 CP，即 $CP=\{N(v_{\text{temp}},1),N(v_{\text{temp}},2),N^*(v_{\text{temp}},2)\}$。由于 CP 中 $\{N^*(v_{\text{temp}},2)\}$ 是节点 w 的邻居节点（即已经与节点 w 连接），因此将它从 CP 中删除，即 $CP=\{N(v_{\text{temp}},1),N(v_{\text{temp}},2)\}$，并且连接节点 v_{temp} 与 CP 中的节点。总结以上步骤可得，$V(T_{2k-2})=\{r,w,N(w,1),N(w,2),N(w,3),N^*(w,2),N^*(w,3),N(N(w,3),1),N(N(w,3),2)\}$，$E(T_{2k-2})=\{(r,w),(w,N(w,1)),(w,N(w,2)),(w,N(w,3)),(w,N^*(w,2)),(w,N^*(w,3)),(N(w,3),N(N(w,3),1)),(N(w,3),N(N(w,3),2))\}$，并且 $H(T_{2k-2})=3$。通过引理 1，容易验证 $E(T_{2k-2})$ 是 $AQ_n(r_nr_{n-1}\cdots\overline{r_4})\cup\{r\}$ 中的边集合，此时满足 $V(T_{2k-2})=\{r\}\cup AQ_n(r_nr_{n-1}\cdots r_5)_4^{\overline{r_4}}$，并且 $H(T_{2k-2})$ 会增加高度 1。因此，T_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots\overline{r_4})\cup\{r\}$ 中的生成树，且 $H(T_{2k-2})=k-1$。

情形 2.2　当 $k>4$ 时，取整数 $i=3,4,\cdots,k-1$，取节点 w 的 i-邻居节点作为添加节点 $v_{\text{temp}}=N(w,i)=r_nr_{n-1}\cdots\overline{r_k}\cdots\overline{r_i}\cdots r_2r_1$（步骤 1.1 中节点 v_{temp} 与节点 w 连接），并通过调用算法 3，寻找与 v_{temp} 共同前缀为 $r_nr_{n-1}\cdots\overline{r_k}\cdots\overline{r_i}$ 的节点集合 CP，即 $CP=\{N(v_{\text{temp}},1),N(v_{\text{temp}},2),N(v_{\text{temp}},3),\cdots,N(v_{\text{temp}},i-1),N^*(v_{\text{temp}},2),\cdots,N^*(v_{\text{temp}},i-1)\}$。由于 CP 中 $\{N^*(v_{\text{temp}},2),\cdots,N^*(v_{\text{temp}},i-1)\}$ 是节点 w 的邻居节点（即已经与节点 w 连接），因此将它们从 CP 中删除，即 $CP=\{N(v_{\text{temp}},1),N(v_{\text{temp}},2),N(v_{\text{temp}},3),\cdots,N(v_{\text{temp}},i-1)\}$，并且连接节点 v_{temp} 与 CP 中的节点。此时，$V(T_{2k-2})=\{r,w,N(w,1),N(w,2),\cdots,N(w,k-1),N^*(w,2),\cdots,N^*(w,k-1),N(N(w,i),1),N(N(w,i),2),\cdots,N(N(w,i),i-1)\}$，$E(T_{2k-2})=\{(r,w),(w,N(w,1)),(w,N(w,2)),\cdots,(w,N(w,k-1)),(w,N^*(w,2)),\cdots,(w,N^*(w,k-1)),(N(w,i),N(N(w,i),1)),(N(w,i),N(N(w,i),2)),\cdots,(N(w,i),N(N(w,i),i-1))\}$。

由于此时 $k>4$，仅执行一次算法 4 将不满足 $V(T_{2k-2})=r\cup AQ_n(r_nr_{n-1}\cdots r_{k+1})_k^{\overline{r_k}}$，例如节点 $N(v_{\text{temp}},3)$，该节点的邻居节点 $N(N(v_{\text{temp}},3),1)$ 和 $N(N(v_{\text{temp}},3),2)$ 未添加到子树中。因此，当满足 $i-3>0$ 时，需要重新执行算法 4（过程与上述相同），直到 $i=3$ 时，将满足 $V(T_{2k-2})=r\cup AQ_n(r_nr_{n-1}\cdots r_{k+1})_k^{\overline{r_k}}$，并且算法 4 每执行一次 $H(T_{2k-2})$ 会增加高度 1，即 $H(T_{2k-2})=k-1$。通过引理 1，容易验证 $E(T_{2k-2})$ 是 $AQ_n(r_nr_{n-1}\cdots\overline{r_k})\cup\{r\}$ 中的边集合。因此，T_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots\overline{r_k})\cup\{r\}$ 中的生成树，且 $H(T_{2k-2})=k-1$。

在步骤 1.3 中，将 T_{2k-2} 中除了根节点 r 和节点 $N^*(r,k)$ 外的所有节点 v，即 T_{2k-2} 的可扩展节点集合 W_{2k-2}，与它的 k-补-邻居节点连接，即 $V(T_{2k-2})=V(T_{2k-2})\cup\{N^*(v,k)\mid v\in W_{2k-2}\}$，$E(T_{2k-2})=E(T_{2k-2})\cup\{(v,N^*(v,k))\mid v\in W_{2k-2}\}$，并且 $H(T_{2k-2})$ 会增加高度 1。通过引理 1，容易验证 $E(T_{2k-2})$ 是 $AQ_n(r_nr_{n-1}\cdots r_{k+1})$ 中的边集合。因此，T_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots r_{k+1})$ 中的生成树，且 $H(T_{2k-2})=k$。

通过算法 EIST3 的步骤 2，对于 $k_2=k+1,k+2,\cdots,n$，构造共同前缀为 $r_nr_{n-1}\cdots r_{k_1+1}$ 的子树，并且有以下 2 种情形：

情形 1　当 $k=n$ 时，不满足条件 $n\geqslant k+1$，即在步骤 1 中构造的树为 AQ_n 的生成树，因此 $H(T_{2k-2})=k=n$。

情形 2　当满足条件 $n\geqslant k+1$ 时，容易验证以下几种结果：

(1) $T'_{2k-2}=T_{2k-2}$；（第一次循环中的 T_{2k-2} 由步骤 1 得到）

(2) $T''_{2k-2}=N(T'_{2k-2},k_2)$；

(3) $T_{2k-2}=T'_{2k-2}\cup T''_{2k-2}\cup\{(v,N(v,k_2))\mid v\in W_{2k-2}\}\setminus E(T''_{2k-2}[N(W_{2k-2},k_2)])$。

可以得知，在第一次循环中（即 $k_2=k+1$），使用 T'_{2k-2} 表示步骤 1 得到的 T_{2k-2}，T''_{2k-2} 是将 T'_{2k-2} 中所有节点取 k_2-邻居而得到的。因此，T'_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots r_{k_1+1})_{k_2}^{\overline{r_{k_2}}}$ 中的生成树，T''_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots r_{k_1+1})_{k_2}^{\overline{r_{k_2}}}$ 中的生成树。根据引理 2 可知，T'_{2k-2} 和 T''_{2k-2} 同构，故 $H(T'_{2k-2})=H(T''_{2k-2})=k_2-1$。将 T'_{2k-2} 中的可扩展节点集合 W_{2k-2} 与 T''_{2k-2} 中的节点集合 $N(W_{2k-2},k_2)$ 连接，断开 T''_{2k-2} 中节点集合 $N(W_{2k-2},k_2)$ 之间组成的边，得到子树 T_{2k-2}，并且 $H(T_{2k-2})$ 会增加高度 1。通过引理 1，容易验证 $E(T_{2k-2})$ 是 $AQ_n(r_nr_{n-1}\cdots r_{k_1+1})$ 中的边集合。因此，T_{2k-2} 是 $AQ_n(r_nr_{n-1}\cdots r_{k_1+1})$ 中的生成树，且 $H(T_{2k-2})=k+1=k_2$。重复上述步骤，直到 k_2 等于 n，即可得到 AQ_n 中的生成树 T_{2k-2}，且 $H(T_{2k-2})=k_2=n$。

3.4　AQ_n 中 T_{2j-1} 的构造算法

算法 6 给出了在 $AQ_n(n\geqslant 3)$ 中构造第 $2j-1(j=3,4,\cdots,n)$ 棵树的过程，使用 T_{2j-1} 来表示这棵树。

算法 6　EIST4

输入：代表增广立方体维度的整数 n；AQ_n 的根节点 $r=r_nr_{n-1}\cdots r_1$；整数 $j=3,4,\cdots,n$

输出：AQ_n 中边独立生成树集合中的 T_{2j-1}

步骤 1　/* 构造具有共同前缀为 $r_n r_{n-1}\cdots r_{j+1}$ 的子树 */

步骤 1.1　/* 连接根节点 r 与节点 $N^*(r,j)$，并且将与节点 $N^*(r,j)$ 具有共同前缀 $r_n r_{n-1}\cdots\overline{r_j r_{j-1}\cdots r_3}$ 的所有邻居节点连接到 $N^*(r,j)$ */

1. w=N^*(r,j).
2. $V(T_{2j-1})=\{r,w,N(w,1),N(w,2),N^*(w,2)\}$ and $E(T_{2j-1})=\{(r,w),(w,N(w,1)),(w,N(w,2)),(w,N^*(w,2))\}$.

步骤 1.2　/* 将 AQ_n 中具有共同前缀 $r_n r_{n-1}\cdots r_{j+1}$ 的剩余所有节点添加到子树中 */

3. $V(T'_{2j-1})\leftarrow V(T_{2j-1})$.
4. for $j_1=3$ to j do
5.　for v_1 in $V(T_{2j-1})\backslash\{r\}$ do
6.　$v_2=N(v_1,j_1)$.
7.　if $v_2\in V(T_{2j-1})$ then
8.　break.
9.　end if
10.　if v_2 is not adjacent to node w then
11.　$V(T'_{2j-1})\leftarrow V(T'_{2j-1})\cup\{v_2\}$, $E(T_{2j-1})\leftarrow E(T_{2j-1})\cup\{(v_1,v_2)\}$.
12.　else
13.　$V(T'_{2j-1})\leftarrow V(T'_{2j-1})\cup\{v_2\}$, $E(T_{2j-1})\leftarrow E(T_{2j-1})\cup\{(w,v_2)\}$.
14.　end if
15.　end for
16.　$V(T_{2j-1})\leftarrow V(T'_{2j-1})$.
17. end for

步骤 2　/* 构造共同前缀为 $r_n r_{n-1}\cdots r_{j+2}$ 的子树 */

18. if n>j then
19.　$V(T_{2j-1})\leftarrow V(T_{2j-1})\cup N(w,j+1)$.
20.　$E(T_{2j-1})\leftarrow E(T_{2j-1})\cup(w,N(w,j+1))$.
21.　$V(T'_{2j-1})\leftarrow V(T_{2j-1})$.
22.　for v_1 in $V(T_{2j-1})\backslash\{r\}$ do
23.　$v_2=N^*(v_1,j+1)$.
24.　$V(T'_{2j-1})\leftarrow V(T'_{2j-1})\cup\{v_2\}$, $E(T_{2j-1})\leftarrow E(T_{2j-1})\cup\{(v_1,v_2)\}$.
25. end for
26.　$V(T_{2j-1})\leftarrow V(T'_{2j-1})$.
27. end if

步骤 3　/* 对于 $j_2=j+2,j+3,\cdots,n$，构造共同前缀为 $r_n r_{n-1}\cdots r_{j_2+1}$ 的子树 */

28. if n≥j+2 then
29.　for $j_2=j+2$ to n do
30.　$V(T'_{2j-1})\leftarrow V(T_{2j-1})$, $E(T'_{2j-1})\leftarrow E(T_{2j-1})$.
31.　$V(T''_{2j-1})\leftarrow\{N(v,j_2)\mid v\in V(T'_{2j-1})\}$, $E(T''_{2j-1})\leftarrow\{(N(x,j_2),N(y,j_2))\mid(x,y)\in E(T'_{2j-1})\}$.
32.　$V(T_{2j-1})\leftarrow V(T'_{2j-1})\cup V(T''_{2j-1})$, $E(T_{2j-1})\leftarrow E(T'_{2j-1})\cup E(T''_{2j-1})\cup(w,N(w,j_2))$.
33.　end for
34. end if
35. return T_{2j-1}.
36. end function

在算法 EIST4 中，通过步骤 1，构造具有共同前缀 $r_n r_{n-1}\cdots r_{j+1}$ 的子树。在步骤 1.1 中，连接根节点 r 和节点 $w=N^*(r,j)$，并且构造以 w 为中心节点且共同前缀为 $r_n r_{n-1}\cdots\overline{r_j r_{j-1}}\cdots r_3$ 的子树。在步骤 1.2 中，对于 $j_1=3,4,\cdots,j$，将子树中除了根节点 r 的所有节点 v 与节点 $N(v,j_1)$ 连接。若节点 $N(v,j_1)$ 与节点 w 互为邻居节点，则连接节点 w 与节点 $N(v,j_1)$；若节点 $N(v,j_1)$ 已存在于子树中，则不再重复连接。通过步骤 2，构造具有共同前缀 $r_n r_{n-1}\cdots r_{j+2}$ 的子树。如果 $n>j$，连接节点 w 与节点 $N(w,j+1)$，并且将子树中除了根节点 r 的所有节点 v 与节点 $N^*(v,j+1)$ 连接。通过步骤 3，对于 $j_2=j+2,j+3,\cdots,n$，构造具有共同前缀 $r_n r_{n-1}\cdots r_{j_2+1}$ 的子树。T'_{2j-1} 表示由步骤 2 得到的子树，其次 T''_{2j-1} 表示将 T'_{2j-1} 中的所有节点取 j_2-邻居得到的子树，并且连接节点 w 与节点 $N(w,j_2)$ 直到 $j_2=n$。

基于算法 EIST4，通过例 4 给出了 AQ_4 中 T_{2j-1} 的详细构造过程。

例 4　下面以在 AQ_4 中构造 T_5 和 T_7 为例，展示了算法 EIST4 的过程。图 7 和图 8 展示了在 AQ_4 中以 0000 为根节点构造 T_5 和 T_7 的过程。

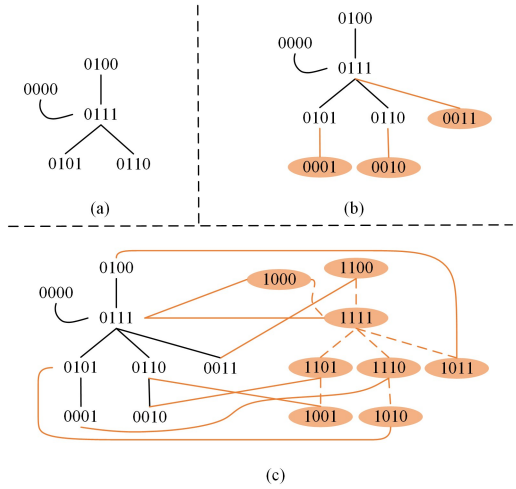


图 7　AQ_4 中 T_5 的构造过程

Fig. 7　Construction process of T_5 in AQ_4

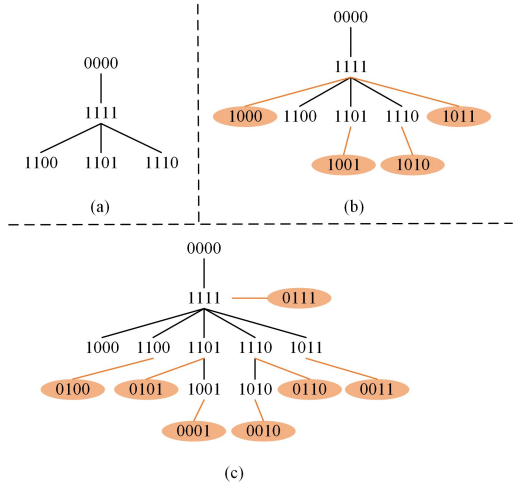


图 8　AQ_4 中 T_7 的构造过程

Fig. 8　Construction process of T_7 in AQ_4

首先分析 $T_5(j=3)$的构造过程，在图 7(a)中，首先连接根节点 $r=0000$ 与节点 $w=N^*(r,3)=0111$，并且以节点 w 为中心构造共同前缀为 01 的子树。在图 7(b)中，将除根节点外的所有节点 v 与其对应的 3-邻居节点连接，即节点 0111 连接节点 0011，节点 0101 连接节点 0001，节点 0110 连接节点 0010，节点 0100 连接节点 0000，然而节点 0000 已存在于子树中，因此不再重复连接。以上步骤构造了共同前缀为 r_4 的子树。在图 7(c)中，首先将中心节点 $w=0111$ 与其对应的 4-邻居节点 $N(w,4)=1111$ 连接，并且将除根节点 r 外的所有节点 v 与其对应的 4-补-邻居节点相连，最终得到一棵完整的生成树 T_5。

接着分析 $T_7(j=4)$的构造过程，在图 8(a)中，首先连接根节点 $r=0000$ 与节点 $w=N^*(r,4)=1111$，并且以节点 w 为中心构造共同前缀为 11 的子树。在图 8(b)中，将除根节点以外的所有节点 v 与其对应的 3-邻居节点连接，即节点 1111 连接节点 1011，节点 1101 连接节点 1001，节点 1110 连接节点 1010。此外，节点 1100 的 3-邻居节点 1000 是中心节点 w 的邻居节点，因此连接节点 w 与节点 1000。在图 8(c)中，与图 8(b)的步骤相似，将除根节点外的所有节点 v 与其对应的 4-邻居节点连接，此时节点 1000 的 4-邻居节点 0000 已存在于子树中，因此不再重复连接。以上步骤构造了共同前缀为空的子树，得到一棵完整的生成树 T_7。

引理 7 通过算法 EIST4 得到的树为 $T_{2j-1}(j=3,4,\cdots,n)$，则 T_{2j-1} 是 AQ_n 中的生成树且高度为 n，其中 $n\geqslant 3$。

证明：通过算法 EIST4 的步骤 1，构造共同前缀为 $r_nr_{n-1}\cdots r_{j+1}$ 的子树。在步骤 1.1 中，连接根节点 r 和节点 $w=N^*(r,j)=r_nr_{n-1}\cdots r_{j+1}\overline{r_j\cdots r_1}$，并且以 w 为中心节点，构造共同前缀为 $r_nr_{n-1}\cdots\overline{r_jr_{j-1}\cdots r_3}$ 的子树。即 $V(T_{2j-1})=\{r,w,N(w,1),N(w,2),N^*(w,2)\}$，$E(T_{2j-1})=\{(r,w),(w,N(w,1)),(w,N(w,2)),(w,N^*(w,2))\}$，并且 $H(T_{2j-1})=2$。

在步骤 1.2 中，对于 $j_1=3,4,\cdots,j$，将子树中除了根节点 r 外的所有节点 v 与其 j_1-邻居节点，即 $N(v,j_1)$ 连接。若节点 $N(v,j_1)$ 与节点 w 互为邻居节点，则连接节点 w 与节点 $N(v,j_1)$；若节点 $N(v,j_1)$ 已存在于子树中，则不再重复连接。上述步骤每执行一次 $H(T_{2j-1})$ 会增加高度 1，因此 $H(T_{2j-1})=j_1$。当 $j_1=j$ 时，此时满足 $V(T_{2j-1})=AQ_n(r_nr_{n-1}\cdots r_{j+1})$，且 $H(T_{2j-1})=j_1=j$。通过引理 1，容易验证 $E(T_{2j-1})$ 是 $AQ_n(r_nr_{n-1}\cdots r_{j+1})$ 中的边集合。因此，T_{2j-1} 是 $AQ_n(r_nr_{n-1}\cdots r_{j+1})$ 中的生成树，且 $H(T_{2j-1})=j$。

通过算法 EIST4 的步骤 2，构造共同前缀为 $r_nr_{n-1}\cdots r_{j+2}$ 的子树，并且有以下 2 种情形：

情形 1 当 $j=n$ 时，不满足条件 $n>j$，即在步骤 1 中构造的树为 AQ_n 的生成树，因此 $H(T_{2j-1})=j=n$。

情形 2 当满足条件 $n>j$ 时，通过步骤 1，得到 $AQ_n(r_nr_{n-1}\cdots r_{j+1})$ 中的生成树 T_{2j-1}。若将 T_{2j-1} 中所有节点取 $(j+1)$-邻居，则得到的是 $AQ_n(r_nr_{n-1}\cdots\overline{r_{j+1}})$ 中的生成树，由引理 2 可知，$AQ_n(r_nr_{n-1}\cdots r_{j+1})$ 与 $AQ_n(r_nr_{n-1}\cdots\overline{r_{j+1}})$ 同构。

因此当满足条件 $n>j$ 时，首先连接节点 w 与其 $(j+1)$-邻居节点，即节点 $N(w,j+1)$，并且节点 $N(w,j+1)\in AQ_n(r_nr_{n-1}\cdots\overline{r_{j+1}})$。其次，将除根节点 r 外的所有节点 v 连接其 $(j+1)$-补-邻居节点，即节点 $N^*(v,j+1)$，并且节点 $N^*(v,j+1)\in AQ_n(r_nr_{n-1}\cdots\overline{r_{j+1}})$。通过上述步骤，将 $AQ_n(r_nr_{n-1}\cdots\overline{r_{j+1}})$ 中的所有节点连接到子树中，并且 $H(T_{2j-1})$ 会增加高度 1。通过引理 1，容易验证 $E(T_{2j-1})$ 是 $AQ_n(r_nr_{n-1}\cdots r_{j+2})$ 中的边集合。因此，T_{2j-1} 是 $AQ_n(r_nr_{n-1}\cdots r_{j+2})$ 中的生成树，且 $H(T_{2j-1})=j+1$。

通过算法 EIST4 的步骤 3，对于 $j_2=j+2,j+3,\cdots,n$，构造共同前缀为 $r_nr_{n-1}\cdots r_{j_2+1}$ 的子树，并且有以下 2 种情形：

情形 1 当 $j+1=n$ 时，不满足条件 $n>j+1$，即在步骤 2 中构造的树为 AQ_n 的生成树，因此 $H(T_{2j-1})=j+1=n$。

情形 2 当满足条件 $n>j+1$ 时，容易验证以下几种结果：

(1) $T'_{2j-1}=T_{2j-1}$；(第一次循环中的 T_{2j-1} 由步骤 2 得到)

(2) $T''_{2j-1}=N(T'_{2j-1},j_2)$；

(3) $T_{2j-1}=T'_{2j-1}\cup T''_{2j-1}\cup(w,N(w,j_2))$。

可以得知，在第一次循环中(即 $j_2=j+2$)，T'_{2j-1} 表示步骤 2 得到的 T_{2j-1}，T''_{2j-1} 是将 T'_{2j-1} 中所有节点取 j_2-邻居得到的子树。因此，T'_{2j-1} 是 $AQ_n(r_nr_{n-1}\cdots r_{j_2+1})_{j_2}^{r_{j_2}}$ 中的生成树，T''_{2j-1} 是 $AQ_n(r_nr_{n-1}\cdots r_{j_2+1})_{j_2}^{\overline{r_{j_2}}}$ 中的生成树。根据引理 2 可知，T'_{2j-1} 和 T''_{2j-1} 同构，故 $H(T'_{2j-1})=H(T''_{2j-1})=j+1=j_2-1$。将 T'_{2j-1} 中的中心节点 w 与 T''_{2j-1} 中的节点 $N(w,j_2)$ 连接，得到子树 T_{2j-1}，并且 $H(T_{2j-1})$ 会增加高度 1。通过引理 1，容易验证 $E(T_{2j-1})$ 是 $AQ_n(r_nr_{n-1}\cdots r_{j_2+1})$ 中的边集合。因此，T_{2j-1} 是 $AQ_n(r_nr_{n-1}\cdots r_{j_2+1})$ 中的生成树，且 $H(T_{2j-1})=j+2=j_2$。重复上述步骤，直到 j_2 等于 n，即可得到 AQ_n 中的生成树 T_{2j-1}，且 $H(T_{2j-1})=j_2=n$。

3.5 AQ_n 中 $2n-1$ 棵边独立生成树的准确性

定理 1 于 $n\geqslant 2$，且根节点 $r=r_nr_{n-1}\cdots r_1$，通过算法 EIST1，EIST2，EIST3，EIST4 得到的 $2n-1$ 棵生成树是在 AQ_n 中以 r 为根节点的 $2n-1$ 棵边独立生成树，并且每棵树的高度均为 n。

证明：由算法 EIST1，EIST2，EIST3，EIST4 得到的 $2n-1$ 棵生成树，通过引理 3、引理 4、引理 6 和引理 7，容易验证每棵生成树之间满足边不相交，因此得到 $2n-1$ 棵边独立生成树，并且每棵生成树的高度均为 n。

3.6 AQ_n 中 $2n-1$ 棵边独立生成树并行算法的时间复杂度

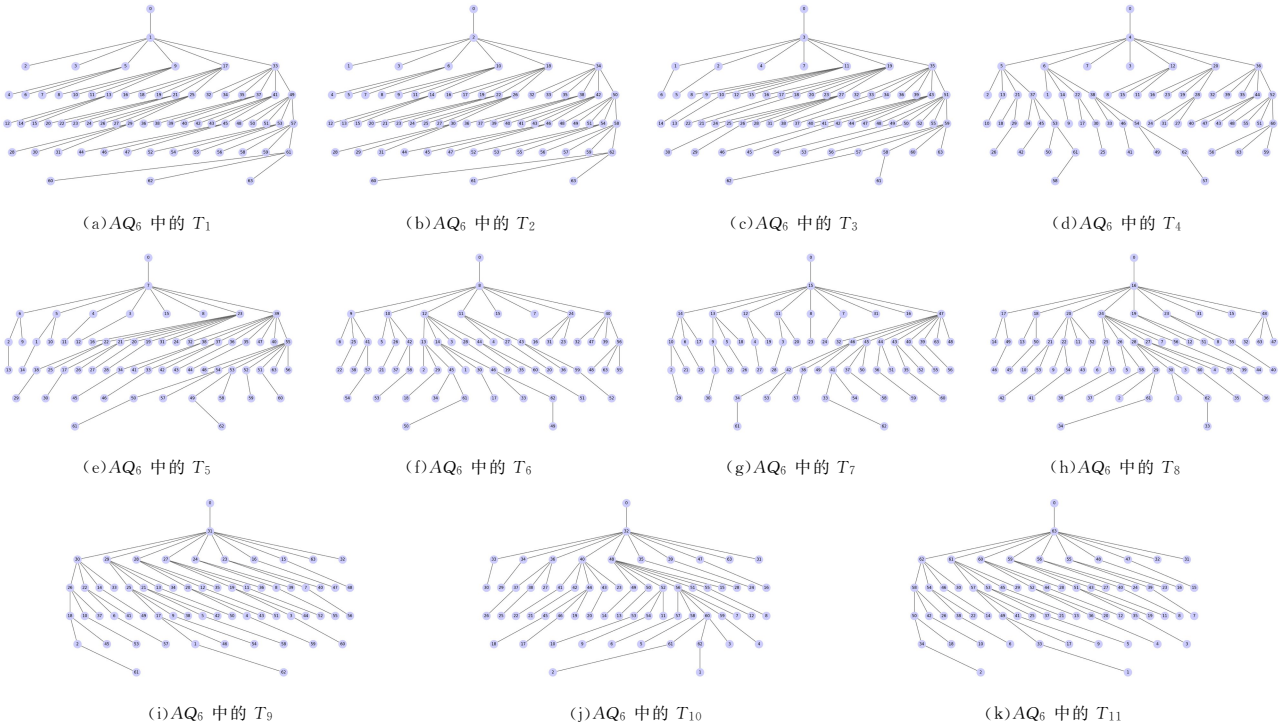
定理 2 AQ_n 中构造 $2n-1$ 棵边独立生成树并行算法的时间复杂度为 $O(N)$，其中 $N=2^n$ 为 AQ_n 中的节点个数。

证明：如上所述，AQ_n 中 $2n-1$ 棵边独立生成树由算法 EIST1，EIST2，EIST3 和 EIST4 并行构造。并且在算法 EIST1，EIST2，EIST3 和 EIST4 中，我们需要遍历 AQ_n 中的所有节点，其中节点的数量为 2^n。因此，本文中的并行算法时间复杂度为 $O(2^n)$。

4 模拟实验

我们通过编程语言 Python 和软件包 treelib，networkx，matplotlib 来验证算法 EIST1，EIST2，EIST3 和 EIST4 的有效性。程序中的变量 n 表示增广立方体的维度，变量 r

表示根节点。输入 $n=6$,$r=0$,以 AQ_6 为例构造 11 棵边独立生成树。图 9 给出了构造 AQ_6 中以节点 0 为根节点、高度为 6 的边独立生成树的程序输出示例。实验结果如图 9 所示。



(a)AQ_6 中的 T_1 (b)AQ_6 中的 T_2 (c)AQ_6 中的 T_3 (d)AQ_6 中的 T_4

(e)AQ_6 中的 T_5 (f)AQ_6 中的 T_6 (g)AQ_6 中的 T_7 (h)AQ_6 中的 T_8

(i)AQ_6 中的 T_9 (j)AQ_6 中的 T_{10} (k)AQ_6 中的 T_{11}

图 9 AQ_6 中以 0 为根节点的 11 棵边独立生成树

Fig. 9 11 EISTs with 0 as root node in AQ_6

通过模拟实验,得到 6 维增广立方体中 11 棵以 0 为根的边独立生成树。从中可以发现具体节点的连接方式,例如根节点 0 到目标节点 1 的路径,在 T_1 中为{(0,1)},在 T_2 中为{(0,2),(2,1)},在 T_3 中为{(0,3),(3,1)},在 T_4 中为{(0,4),(4,6),(6,1)},在 T_5 中为{(0,7),(7,5),(5,1)},在 T_6 中为{(0,8),(8,12),(12,14),(14,1)},在 T_7 中为{(0,15),(15,13),(13,9),(9,1)},在 T_8 中为{(0,16),(16,24),(24,28),(28,30),(30,1)},在 T_9 中为{(0,31),(31,29),(29,25),(25,17),(17,1)},在 T_{10} 中为{(0,32),(32,48),(48,56),(56,60),(60,62),(62,1)},在 T_{11} 中为{(0,63),(63,61),(61,57),(57,49),(49,33),(33,1)}。可以发现,以上 11 个路径是成对边独立的,且 11 棵树的高度均为 6。

结束语 随着网络规模的扩大,在并行计算机系统中,处理器和链路发生故障的情形是不可避免的,而经过故障处理器和故障链路传递的数据将是不可靠的。边独立生成树在信息的可靠传输、信息的并行传输和安全分发以及并行故障服务器诊断算法中具有非常重要的应用。因此研究互连网络的边独立生成树是非常有必要的。

本文首先给出了在增广立方体上构造边独立生成树的并行算法,并验证了并行算法得到的 $2n-1$ 棵生成树为边独立生成树的正确性。通过模拟实验验证了该方法的有效性。实验结果表明,本文提出的并行算法是准确的。

参 考 文 献

[1] SAAD Y, SCHULTZ M H. Topological properties of hypercubes [J]. IEEE Transactions on Computers, 1988, 37(7): 867-872.

[2] CHOUDUM S A, SUNITHA V. Augmented cubes [J]. Networks, 2002, 40(2): 71-84.

[3] TSENG Y C, WANG S Y, HO C W. Efficient broadcasting in wormhole-routed multicomputers: A network-partitioning approach [J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(1): 44-61.

[4] BAO F, FUNYU Y, HAMADA Y, et al. Reliable broadcasting and secure distributing in channel networks [J]. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 1998, E81-A(5): 796-806.

[5] BAO F, IGARASHI Y, ÖHRING S R. Reliable broadcasting in product networks [J]. Discrete Applied Mathematics, 1998, 83(1/2/3): 3-20.

[6] CHEN Y S, CHIANG C Y, CHEN C Y. Multi-node broadcasting in all-ported 3-D wormhole-routed torus using an aggregation-then-distribution strategy [J]. Journal of Systems Architecture, 2004, 50(9): 575-589.

[7] ELHOURANI T, GOPALAN A, RAMASUBRAMANIAN S. IP fast rerouting for multi-link failures [J]. IEEE/ACM Transactions on Networking, 2016, 24(5): 3014-3025.

[8] GOPALAN A, RAMASUBRAMANIAN S. IP fast rerouting and disjoint multipathrouting with three edge-independent spanning trees [J]. IEEE/ACM Transactions on Networking, 2016, 24(3): 1336-1349.

[9] PAN Z Y, CHENG B L, FAN J X, et al. A parallel algorithm to construct edge independent spanning trees on the line graphs of conditional bijective connection networks [J]. Theoretical Com-

puter Science, 2023, 942: 33-46.
[10] ITAI A, RODEH M. The multi-tree approach to reliability in distributed networks [J]. Information and Computation, 1988, 79: 43-59.
[11] HSIEH S Y, WU C Y. Edge-fault-tolerant Hamiltonicity of locally twisted cubes under conditional edge faults [J]. Journal of Combinatorial Optimization, 2010, 19(1): 16-30.
[12] ZEHAVI A, ITAI A. Three tree-paths [J]. Journal of Graph Theory, 1989, 13(2): 175-188.
[13] GOPALANA. On constructing three edge independent spanning trees[EB/OL]. https://www. semanticscholar. org/paper/ON-CONSTRUCTING-THREE-EDGE-INDEPENDENT-SPANNING-Gopalan/b458a3f665cd41f1ec42fb3cebb5176c369a87ec?p2df.
[14] HOYER A, THOMAS R. Four edge-independent spanning trees [J]. SIAM Journal on Mathematics, 2018, 32(1): 233-248.
[15] ALI A, LEE O. Five edge-independent spanning trees [J]. Procedia Computer Science, 2023, 223: 223-230.
[16] WANG Y, SHEN H, FAN J X. Edge-independent spanning trees in augmented cubes [J]. Theoretical Computer Science, 2017, 670: 23-32.
[17] KIM J S, LEE H O, CHENG E, et al. Independent spanning trees on even networks [J]. Information Sciences, 2011, 181(13): 2892-2905.
[18] KIM J S, LEE H O, CHENG E, et al. Optimal independent spanning trees on odd graphs [J]. The Journal of Supercomputing, 2011, 56(2): 212-225.
[19] BAO F, OBOKATA K, IGARASHI Y, et al. Independent spanning trees of product graphs and their construction [J]. IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences, 1996, E79-A(11): 1894-1903.
[20] TANG S M, YANG J S, WANG Y L, et al. Independent spanning trees on multidimensional torus networks [J]. IEEE Transactions on Computers, 2010, 59(1): 93-102.
[21] DONG Q, WANG X. How many triangles and quadrilaterals are there in an n-dimensional augmented cube [J]. Theoretical Computer Science, 2019, 771: 93-98.
[22] MANE S A, KANDEKAR S A, WAPHARE B N. Constructing spanning trees in augmented cubes [J]. Journal of Parallel and Distributed Computing, 2018, 122: 188-194.
[23] CHENG B L, FAN J X, LIN C K, et al. An improved algorithm to construct edge-independent spanning trees in augmented cubes [J]. Discrete Applied Mathematics, 2020, 277: 55-70.

LI Xiajing, born in 1999, postgraduate, is a member of CCF(No. J1607G). Her main research interests include parallel and distributed systems and graph algorithms.

CHENG Baolei, born in 1979, professor, is a member of CCF(No. 16200S). His main research interests include parallel and distributed systems, algorithms, interconnection architectures and software testing.

(责任编辑:何杨)