蜻蜓网络上完全独立生成树的构造算法

卞庆荣，程宝雷，樊建席，潘志勇

引用本文

卞庆荣，程宝雷，樊建席，潘志勇. 蜻蜓网络上完全独立生成树的构造算法[J]. 计算机科学, 2022, 49(11): 284-292.

BIAN Qing-rong, CHENG Bao-lei, FAN Jian-xi, PAN Zhi-yong. Construction Algorithm of Completely Independent Spanning Tree in Dragonfly Network[J]. Computer Science, 2022, 49(11): 284-292.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211000037

蜻蜓网络上完全独立生成树的构造算法

卞庆荣[1,2]　程宝雷[1,2]　樊建席[1]　潘志勇[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
(20195227075@stu.suda.edu.cn)

摘　要　蜻蜓网络(Dragonfly network)是由 Kim 等提出的一种适用于高性能计算系统的拓扑结构。在蜻蜓网络中,网络被组织成两级架构,计算节点与交换机连接,交换机被分为成多个组。在每一组内部的每个交换机之间互相有一条边相连,任意两组之间有一条边相连接。完全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。在实际应用中,随着网络规模的不断增大,信息传输的效率以及安全性等要求越来越高。因此,研究网络的完全独立生成树具有重要意义。目前,有许多关于网络中完全独立生成树的研究,但是缺乏蜻蜓网络上的完全独立生成树的研究成果。文中提出了蜻蜓网络全局链路分别以相对链接、绝对链接以及循环链接下的完全独立生成树划分的构造算法,并在此划分的基础上给出了完全独立生成树边集合的构造算法,并对以上算法的正确性进行了证明。最后分析了算法的时间复杂度。

关键词:蜻蜓网络;拓扑;完全独立生成树;算法
中图法分类号　TP393.0

Construction Algorithm of Completely Independent Spanning Tree in Dragonfly Network

BIAN Qing-rong[1,2],CHENG Bao-lei[1,2],FAN Jian-xi[1] and PAN Zhi-yong[1,2]
1 School of Computer Science and Technology,Soochow University,Suzhou,Jiangsu 215006,China
2 Jiangsu Provincial Key Laboratory for Computer Information Processing Technology,Soochow University,Suzhou,Jiangsu 215006,China

Abstract　Dragonfly network,proposed by Kim et al.,is a topology for high-performance computer systems. In dragonfly network,compute nodes are attached to switches,the switches are organized into groups,and the network is organized as a two-level clique. There is a link between any two nodes in a group,and there is a global link between different groups. Completely independent spanning trees(CISTs) play important roles in reliable information transmission,parallel transmission and safe distribution of information. In practical application,with the continuous increase of network scale,the requirements for information transmission efficiency and security are becoming higher and higher. Therefore,it is meaningful to study the construction of completely independent spanning trees in dragonfly network. At present,there are many results on completely independent spanning trees in networks,but completely independent spanning trees in dragonfly network have not been studied. In this paper,construction algorithm for the global link of dragonfly network is proposed,which is divided into completely independent spanning tree under relative link,absolute link and circulant link. Based on this division,a construction algorithm for the edge set of completely independent spanning tree is given,and the correctness of the above algorithms is proved. Finally,the time complexity of these algorithms is analyzed.

Keywords　Dragonfly network,Topology,Completely independent spanning tree,Algorithm

1　引言

在高性能计算(High Performance Computing,HPC)系统中,网络通信效率和延迟在决定性能和可伸缩性方面起着重要作用[1]。当扩展到数万个节点时,传统拓扑(如环形网格)会将网络能耗增加到整个系统能耗的 50%[2],并引入巨大的通信开销,这是因为消息平均需要传输数十个网络跃点[3]。蜻蜓网络就是为了构建一个百万兆级别的计算系统而提出的[4]。蜻蜓图是一种分层拓扑结构,计算节点与交换机连接,交换机被分为多个组,在每一组内部的每个交换机之间互相有一条链路相连,任意两组之间有一条链路相连接。这意味着每对交换机之间最多间隔 3 个跃点就能连接,一个

到稿日期:2021-10-08　返修日期:2022-03-19
基金项目:国家自然科学基金(62172291,U1905211);江苏高校优势学科建设工程;江苏省教育厅未来网络科研基金(FNSRFP-2021-YB-39).
This work was supported by the National Natural Science Foundation of China(62172291,U1905211),Priority Academic Program Development of Jiangsu Higher Education Institutions and Jiangsu Province Department of Education Future Network Research Fund Project(FNSRFP-2021-YB-39).
通信作者:程宝雷(chengbaolei@suda.edu.cn)

在源组内，一个从源组到目标组，一个在目标组内[5]。目前蜻蜓图已被广泛应用于当前的高性能计算机或高端服务器中[6-8]。

蜻蜓网络的大小由以下参数决定：p，每个交换机中的计算节点数；a，每个分组中交换机的数量；h，每个交换机的全局链路数。具有以上参数的蜻蜓图可以表示为(p,a,h)-蜻蜓网络。由于组与组之间都只有一条链路进行连接，每一组内有 a 个交换机，每个交换机有 h 个全局链路连接到其他的组，因此共有 $g=ah+1$ 组，$ag=a(ah+1)$个交换机，$pag=pa(ah+1)$个计算节点。

关于蜻蜓网络的现有文献侧重于分析路由[9-11]和作业分配算法[12-14]对性能的影响。蜻蜓网络拓扑目前缺乏对其完全独立生成树的研究。本文的研究是将(p,a,h)-蜻蜓网络看作一个图(用 $D(p,a,h)$来表示，顶点和边分别表示蜻蜓图的交换机和链路，其中交换机被视为透明设备)。因此下文中顶点表示蜻蜓网络中的交换机，边表示网络中的链路。

设 $T_1,T_2,\cdots,T_k$是图 G 的 k 棵生成树，u 是图 G 的一个顶点。若对于图 G 中的任意点 $v(\neq u)$，$T_1,T_2,\cdots,T_k$中从顶点 u 到顶点 v 的路径没有相同的顶点也没有相同的边，那么我们就将这 k 棵生成树称为图 G 以 u 为根的顶点独立生成树。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作用。在互连网络中，独立生成树可以被看成以源点作为根的模型，沿着 k 棵独立生成树传输数据。但是，如果我们改变了源点，那么就需要重新构造新的顶点独立生成树。2001 年，Hasunuma 提出了完全独立生成树的概念[15]：设 $T_1,T_2,\cdots,T_k$是图 G 的 k 棵生成树，若对于图 G 中的任意两个点 u 和 $v(u\neq v)$，$T_1,T_2,\cdots,T_k$中从顶点 u 到顶点 v 的路径没有相同的顶点也没有相同的边，那么我们就将这 k 棵生成树称为图 G 的 k 棵完全独立生成树。显然，完全独立生成树是以任意顶点作为根的顶点独立生成树，因此在研究并行计算时，当源点改变之后，就无须重新构造顶点独立生成树。

在 HPC 系统中，网络通信效率和延迟在决定性能和可伸缩性方面起着重要作用。可通过将信息并行分发来提高通信的效率以及降低延迟，因此，在任意两个交换机之间可将信息通过多个内部顶点不相交以及边不相交的路来进行并行传输，从而提高信息传输效率。若找到 k 棵完全独立生成树，假设网络中存在至多 $k-1$ 个故障服务器，则任意两个处理器之间总存在一条无故障路径，从而达到可靠传输的目的。文献[16-19]研究了不同网络的完全独立生成树问题。文献[20-23]研究了完全独立生成树存在的一些充分条件，并且证明了哪怕是想构造出一个图的两棵完全独立生成树都是 NP-困难的。因此，研究完全独立生成树非常必要。

不同的蜻蜓网络的全局链路安排导致其完全独立生成树构造算法也不相同，Hasting 等[24]确定了蜻蜓图的 3 个全局链路安排，之后，Belka 等[25]又提出了两个。对蜻蜓系统的研究通常不指定使用的全局链接安排，但会使用相对或绝对链接方式，因此，本文给出相对链接、绝对链接以及循环链接下完全独立生成树的构造算法。

本文首先介绍了蜻蜓网络的全局链路方法，并给出了在相对链接、绝对链接以及循环链接下的完全独立生成树的构造算法；在此基础上给出了算法正确性的证明；最终分析了各算法的时间复杂度。

2 预备知识

本节主要介绍相关术语和符号定义以及(p,a,h)-蜻蜓网络的结构和性质。

2.1 术语和符号

设图 $G=(V(G),E(G))$是一个简单无向图。令 $V_1\subseteq V(G)$且 $V_1\neq\emptyset$。V_1 在 G 中的导出子图，记作 $G[V_1]$，是以 V_1 为顶点集合、两端点均在 V_1 中的全部边作为边集的子图。我们用 $G-V_1$ 来表示 $G[G\backslash V_1]$。对于 G 中任意两个顶点 x 和 y，x 和 y 是相邻的，当且仅当$(x,y)\in E(G)$。

路径 P 是由$(x_0,x_1,\cdots,x_n)$组成的一条长度为 n 的序列，序列中任意的顶点互不相同且对任意的 $0\leqslant m<n$，都有$(x_m,x_{m+1})\in E(G)$，其中 x_0是此路径的起点，x_n为路径的终点。对于顶点 x 与 y 之间的任意两条不同的路径 P_1和 P_2，若 $V(P_1)\cap V(P_2)=\{x,y\}$，则称 P_1和 P_2是顶点不相交路径；若 $E(P_1)\cap E(P_2)=\emptyset$，则称 P_1和 P_2是边不相交路径。

如果图 G 的一个子图是一棵包含 G 的所有顶点的树，则称该子图为 G 的生成树。设 $T_1,T_2,\cdots,T_k$是图 G 的 k 棵生成树。若对于图 G 中的任意两个顶点 u 和 $v(u\neq v)$，在任意 T_i中，$T_j(1\leqslant i,j\leqslant k)$，如果从顶点 u 到顶点 v 的路径 P_i与 P_j既是顶点不相交路径又是边不相交路径，那么我们就将这 k 棵生成树称为图 G 的 k 棵完全独立生成树。

设$(V_0,V_1,\cdots,V_k)$是边集合 $V(G)$的一个划分。$B(V_i,V_j,G)(i\neq j)$表示由 V_i与 V_j构成的二部图，其中边集为$\{uv|uv\in E(G),u\in V_i,v\in V_j\}$。由于本文讨论的是图 G 中的情形，因此可以用 $B(V_i,V_j)$来代替 $B(V_i,V_j,G)$。

将满足下列两种条件的边集合 $V(G)$的划分$(V_0,V_1,\cdots,V_k)$称为完全独立生成树划分。

(1)对于 $i=0,1,\cdots,k$ 而言，导出子图 $G[V_i]$是连通的。

(2)对于任意的 $i\neq j$，二部图 $B(V_i,V_j)$没有树分支，即对于 $B(V_i,V_j)$任意的连通分支 H 而言，都满足$|E(H)|\geqslant|V(H)|$。

Araki[26]证明了对于一个图 G 而言，其存在 k 棵完全独立生成树等价于存在一个完全独立生成树划分。

引理 1[26] 对于一个连通图 G，有 k 棵完全独立生成树，当且仅当它有一个完全独立生成树划分$(V_0,V_1,\cdots,V_k)$。

2.2 蜻蜓网络 $D(p,a,h)$的结构和性质

首先了解蜻蜓网络拓扑结构。蜻蜓网络的大小由以下参数决定：p，每个交换机中的计算节点数；a，每个分组中交换机的数量；h，每个交换机的全局链路数。具有以上参数的蜻蜓网络可以表示为 $D(p,a,h)$。蜻蜓网络具有两级层次结构，其中每一级中的元素紧密相连，使得网络的直径较小。蜻蜓网络拓扑组与组之间都只有一条链路进行连接，每一组内有 a 个交换机(分别记为 $0,1,\cdots,a-1$)。每个交换机有 h 个全局链路连接其他的组。因此共有 $g=ah+1$ 组(分别记为 $0,1,\cdots,g-1$)。在一个组中，交换机使用电气链路以全对全或扁平蝶形的方式连接构成第一级；不同的组之间通过光链路连接这些组的全局链路构成第二级。总的来说，每个交换机将它们连接到(i)计算节点，(ii)组中的交换机，以及(iii)网络

中的其他组。图 1 给出了由 4 个交换机形成的一个组，每个交换机与组内的其他交换机相连接，同时与其他组的两个交换机相连接。

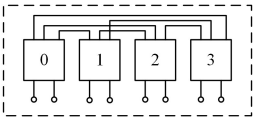

图 1 蜻蜓网络的一个分组

Fig. 1 A dragonfly network group

Kim 等[27]对蜻蜓网络拓扑的定义更加侧重于技术和经济方面，而不是提供基本图形的封闭定义。不同的全局链路安排导致多个不同的拓扑可以被视为蜻蜓网络的变体。对蜻蜓网络的研究通常不指定使用的全局链接安排，但通常使用相对或绝对链接的方式。为了方便描述，我们将 $g=ah+1$ 组编号为 $0,1,\cdots,g-1$，并且将每一组内的交换机编号为 $0,1,\cdots,a-1$。模运算(mod)表示求两个正整数的余数的运算，向下取整(floor)表示对一个数进行向下取整。同时用 $S_{i,j}$ 来表示第 i 组的 j 号交换机。用 $R(S_{i,j})$ 来表示与 $S_{i,j}$ 直接相连的其他组的交换机构成的集合，$R^-(S_{i,j})$ 表示与 $S_{i,j}$ 直接相连且组号大于 i 的交换机构成的集合，$R^-(S_{i,j})$ 表示与 $S_{i,j}$ 直接相连且组号小于 i 的交换机构成的集合。则 $R(S_{i,j})=R^+(S_{i,j})\cup R^-(S_{i,j})$。$G(S)$ 表示 $S_{i,j}$ 的组号 i，$N(s)$ 表示 $S_{i,j}$ 的交换机编号 j。

下面给出蜻蜓网络的相对与绝对链接以及循环链接方式。

相对链接方式：对于第 i 组编号为 k 的交换机 $S_{i,k}$，将其链接向它第 $i+hk+1(\mathrm{mod}\ g),\cdots,i+h(k+1)\ (\mathrm{mod}\ g)$ 组的编号为 $a-k-1$ 的交换机 $S_{i+h(k+1)\ (\mathrm{mod}\ g),a-k-1}$，每个组内的交换机链接向对应组中编号更远的交换机，即 $R(S_{i,k})=\{S_{m,n}\mid i+hk+1(\mathrm{mod}\ g)\leqslant m\leqslant i+h(k+1)\ (\mathrm{mod}\ g),n=a-k-1\}$(例如，在 $D(p,4,2)$ 中第 3 组的 2 号交换机分别链接向第 8 组与第 0 组的 4-2-1=1 号交换机)。图 2 给出了蜻蜓网络全局链路以相对链接方式排列。

绝对链接方式：对于第 i 组编号为 k 的交换机 $S_{i,k}$，下面给出 $S_{i,k}$ 与其他组交换机的链接方式。

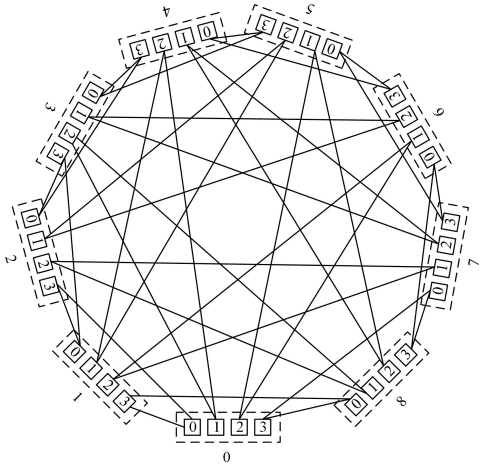

图 2 $D(p,4,2)$ 的相对排列链接

Fig. 2 $D(p,4,2)$-dragonfly with relative arrangement

引理 2[5] 在绝对链接中，与 $S_{i,k}$ 直接链接的其他组交换机为 $R(S_{i,k})=R^+(S_{i,k})\cup R^-(S_{i,k})$，其中：$R^+(S_{i,k})=\{S_{m,n}\mid m>i,i+hk+1(\mathrm{mod}\ g)\leqslant m\leqslant i+h(k+1)\ (\mathrm{mod}\ g),n=\mathrm{floor}(i/h)\}$，$R^-(S_{i,k})=\{S_{m,n}\mid m<i,i+hk+1(\mathrm{mod}\ g)\leqslant m\leqslant i+h(k+1)\ (\mathrm{mod}\ g),n=\mathrm{floor}((i-1)/h)\}$。

例如：在 $D(p,4,2)$ 中第 3 组的 2 号交换机分别链接向第 5 组与第 6 组的 $floor(3/2)=1$ 号交换机)。图 3 给出了蜻蜓网络全局链路以绝对链接方式排列。

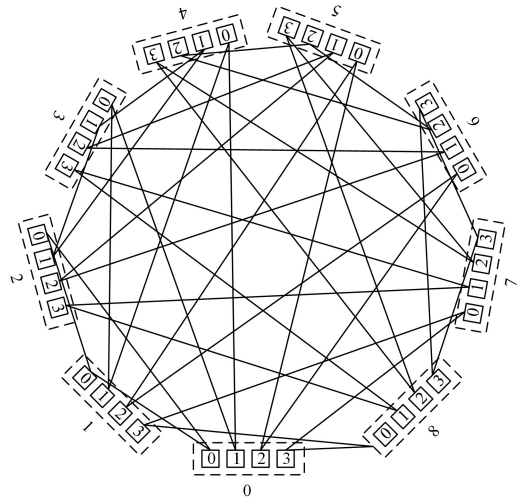

图 3 $D(p,4,2)$ 的绝对排列链接

Fig. 3 $D(p,4,2)$-dragonfly with absolute arrangement

循环链接方式：对于第 i 组编号为 k 的交换机 $S_{i,k.}$，下面给出 $S_{i,k}$ 与其他组交换机的链接方式。

引理 3[5] 在循环链接中，与 $S_{i,k}$ 直接链接的其他组交换机为 $R(S_{i,k})=R^+(S_{i,k})\cup R^-(S_{i,k})$，其中：$R^+(S_{i,k})=\{S_{m,n}\mid i+k*\mathrm{floor}(h/2)<m<i+k*\mathrm{floor}(h/2)+\mathrm{floor}(h/2),n=k\}$，$R^-(S_{i,k})=\{S_{m,n}\mid i-k*\mathrm{floor}(h/2)-\mathrm{floor}(h/2)<m<i+k*\mathrm{floor}(h/2),n=k\}$。

例如：在 $D(p,4,2)$ 中，$S(0,0)$ 与 $S(1,0)$ 和 $S(8,0)$ 连接。图 4 给出了蜻蜓网络全局链路以循环链接方式排列。

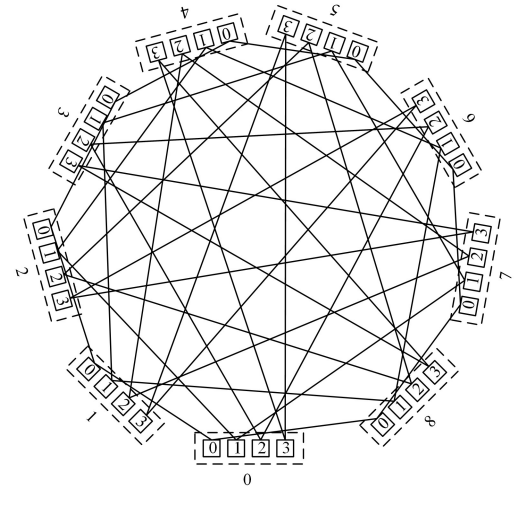

图 4 $D(p,4,2)$ 的循环排列链接

Fig. 4 $D(p,4,2)$-dragonfly with circulant arrangement

3 $D(p,a,h)$ 完全独立生成树的构造算法

本节主要研究 $D(p,a,h)$ 全局链路以相对链接、绝对

链接以及循环链接方式排列下的完全独立生成树的构造算法，并证明了算法的正确性，最后分析了算法的时间复杂度。

“\”指集合删除某个元素的运算，“∪”指集合并集，“[i]”表示数组中第 $i+1$ 个元素，“append”表示从指数组中添加元素，“remove”表示从指数组中删除元素。

3.1 相对链接方式下顶点的完全独立生成树划分

首先给出找到相对链接方式下完全独立生成树划分的算法。

算法 1 CISTP1(a,h)

输入：蜻蜓网络每组交换机数 a 以及每个交换机的全局链路数 h

输出：D(p,a,h)的一个完全独立生成树划分：V[0],V[1],…,V[floor(a/2)]

```
1. g=a*h+1;
2. nextGroup=[];
  /* nextGroup 用于寻找下一个组中交换机 */
3. for i=0 to floor(a/2)do in parallel:
4.    V[i]=∅;
5.    V[i]=V[i]∪S(0,i)∪S(0,a-i-1);
6.    nextGroup.append(S(0,i));
7.    nextGroup.append(S(0,a-i-1));
8.    while nextGroup!=null
9.      next=nextGroup[0];
10.       for j=0 to R(next).length-1
11.         if R(next)[j])not in V[i]
12.           V[i]=V[i]∪R(next)[j]∪S(G(R(next)[j]),a-N
              (R(next)[j])-1);
13.         end if
14.       end for
15.       nextGroup=nextGroup.remove(next);
16.       nextGroup.append(S(G(R(next)[j]),a-N(R(next)[j])-
          1));
17.       if V[i].length >=g
18.         break;
19.       end if
20.   end while
21. end for
22. if a mod 2 ≠ 0
23.   for j=0 to g-1
24.     V[0]=V[0]∪S(j,floor(a/2)+1);
25.   end for
27. end if
28. return V; /* V 中有 V[0],V[1],…,V[floor(a/2)] */
29. end function
```

下面以 $D(p,4,2)$为例来描述算法 1 的过程。对于 $V[0]$与 $V[1]$而言，其表示的是 $D(p,4,2)$中所有顶点的一个子集。首先将第 0 组的第 0 号与第 3 号顶点 $S(0,0)$,$S(0,3)$分配给 $V[0]$，此时 $V[0]=\{S(0,0),S(0,3)\}$；并且在 nextGroup 记录下分配完的顶点，用于寻找下一个组的对应顶点。$R(S(0,0))$中有 $S(1,3)$与 $S(2,3)$，因此将 $S(1,3)$与 $S(2,3)$分配给 $V[0]$，并在 nextGroup 中删除 $S(0,3)$。此时 $V[0]=\{S(0,0),S(0,3),S(1,3),S(2,3)\}$。依次类推，最终得到 $V[0]=\{S(0,0),S(0,3),S(1,0),S(1,3),S(2,0),S(2,3),S(8,0),S(8,3),S(7,0),S(7,3),S(3,0),S(3,3),S(4,0),S(4,3),S(6,0),S(6,3),S(7,0),S(7,3)\}$。同理可得 $V[1]=\{S(0,1),S(0,2),S(1,1),S(1,2),S(2,1),S(2,2),S(8,1),S(8,2),S(7,1),S(7,2),S(3,1),S(3,2),S(4,1),S(4,2),S(6,1),S(6,2),S(7,1),S(7,2)\}$。图 5 给出了 $D(p,4,2)$通过算法 1 得到的两个顶点划分。

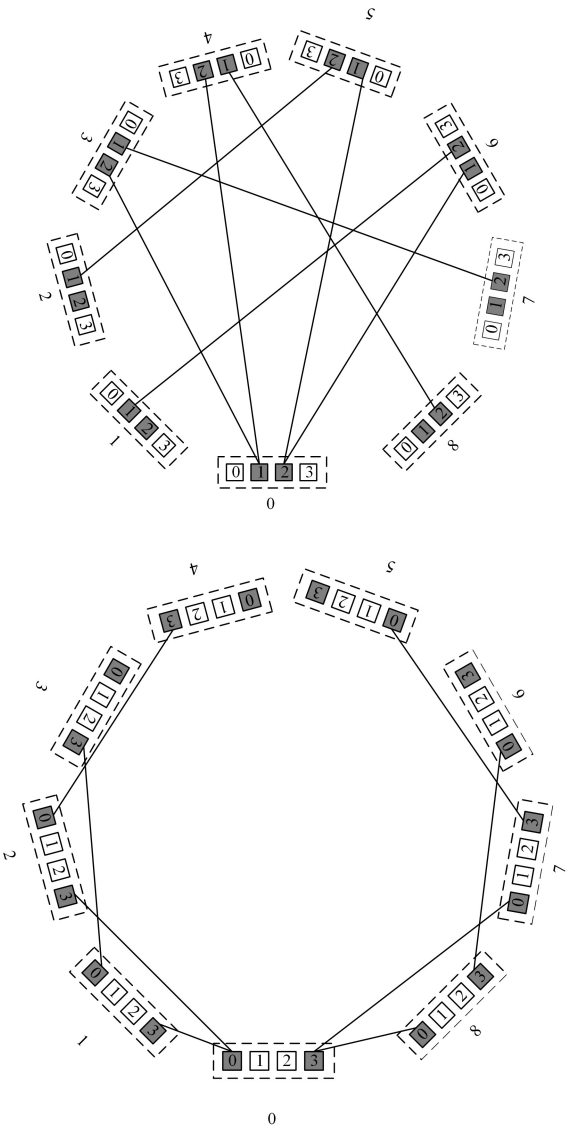

图 5 $D(p,4,2)$在相对链接下的一个顶点划分

Fig. 5 Vertex partition of $D(p,4,2)$ under relative link

显然 $V[0]$与 $V[1]$是 $D(p,4,2)$的一个顶点划分。且由图 4 可看出，$G[V[i]]$,$i\in\{0,1\}$是连通的。在任意的一组中，第 0 和 3 号顶点与第 1 和 2 号顶点都有 4 条边相连接，故对于每一个分支而言，必然是边数大于或等于顶点数。因此得到的两个顶点划分 $V[0]$与 $V[1]$是 $D(p,4,2)$的一个完全独立生成树划分。

引理 4 设蜻蜓网络 $D(p,a,h)$的顶点集合为 V，则通过算法 1 得到的顶点的子集 $V[0],V[1],\cdots,V[\text{floor}(a/2)]$为 V 的一个划分。

证明：对于任意组 j 内的 $S(j,i)$与 $S(j,a-i-1)$($i\in[0,\text{floor}(a/2)]$,$j\in[0,ah]$)，根据蜻蜓网络全局链路相对链接方式排列的规则，它们分别连接向 $S(k,a-i-1)$与 $S(k,i)$($k\neq j$ 且 $k\in[0,ah]$)。因此根据算法 1 得到的 $V[i]$中每组的顶点编号都是 i 与 $a-i-1$。算法中 nextGroup 的作用是将 $V[i]$

中所有顶点所在的组中的第 i 与 $a-i-1$ 号顶点连接向其他组中对应的第 $a-i-1$ 与 i 号顶点。因此，假设 $\exists k\in[0,ah]$，使得 $S(k,a-i-1)\notin V[i]$ 且 $S(k,i)\notin V[i]$，由相对链路安排，$S((g+k-1-(a-i-1)h)(\mathrm{mod}\ g),a-i-1)$ 与 $S(k,i)$ 连接；$S((g+k-1-ih)(\mathrm{mod}\ g),i)$ 与 $S(k,a-i-1)$ 连接，则 $S((g+k-1-(a-i-1)h)(\mathrm{mod}\ g),a-i-1)\notin V[i]$ 且 $S((g+k-1-ih)(\mathrm{mod}\ g),i)\notin V[i]$。以此类推可得 $S(m*g+k-m*1-m*(a-i-1)h)(\mathrm{mod}\ g),a-i-1)\notin V[i]$ 且 $S((m*g+k-m*1-m*ih)(\mathrm{mod}\ g),i)\notin V[i]$，其中 $m=N^*$。则必然存在 $n\in[0,a]$，使得 $S(n,i)\notin V[i]$，即第 0 组到第 a 组之间有一组 n，使得 $S(n,a-i-1)\notin V[i]$。但是根据算法 1 的第一步，$S(0,i)\in V[i]$ 且 $S(0,i)$ 通过 nextGroup 将其所有连接的组中的第 $a-i-1$ 号顶点都并入 $V[i]$ 之中，所以第 0 到 $a-1$ 组中的第 $a-i-1$ 号顶点在 $V[i]$ 之中，且第 1 组中的第 i 个顶点与第 a 组中第 $a-i-1$ 号顶点连接，故第 a 组的第 $a-i-1$ 号顶点也在 $V[i]$ 之中。因此矛盾。假设不成立。

综上，$\forall k\in[0,a-1]$，$S(k,a-i-1)\in V[i]$，$S(k,i)\in V[i]$。且若 $a \bmod 2\neq 0$ 则将每组的中间顶点分配给 $V[0]$，故 $V[0],V[1],\cdots,V[\mathrm{floor}(a/2)]$ 为 V 的一个划分，证毕。

引理 5 通过算法 1 得到的蜻蜓网络 $D(p,a,h)$ 顶点的子集为 $V[0],V[1],\cdots,V[\mathrm{floor}(a/2)]$，则导出子图 $G[V[i]]$ 是连通的，其中 $i\in[0,\mathrm{floor}(a/2)]$。

证明：由于蜻蜓网络组内部是通过全对全的方式连接，因此组内任意两顶点 $S(m,n_1)$ 与 $S(m,n_2)$ 必然是连通的（$m\in[0,g-1]$，$n_1,n_2\in[0,a-1]$）。对于不同组之间的顶点有如下两种情形。

情形 1 $a \bmod 2=0$。

对于 $\forall i\in[0,\mathrm{floor}(a/2)]$，由引理 3 可知，$V[i]$ 中为 $S(m,a-i-1)$ 与 $S(m,i)$，其中 $m\in[0,g-1]$。由相对连接的定义可知，$G[V[i]]$ 是连通的。

情形 2 $a \bmod 2\neq 0$。

对于 $\forall i\in[1,\mathrm{floor}(a/2)]$，与情形 1 同理可得 $G[V[i]]$ 是连通的。当 $i=0$ 时，$\forall j\in[0,a-1]$ 且 $j\neq\mathrm{floor}(a/2)+1$，则同样由引理 3 可得 $V[i]$ 中为 $S(m,a-i-1)$ 与 $S(m,i)$，其中 $m\in[0,g-1]$，因此每组中的 j 与 $a-j-1$ 必然是连通的。将每组中间的第 $\mathrm{floor}(a/2)+1$ 号顶点加入 $V[0]$ 之中，每组内部是全对全连接的，因此每组中的 $\mathrm{floor}(a/2)+1$ 号顶点在导出子图中也必然是连通的。

综上所述，导出子图 $G[V[i]]$ 是连通的，证毕。

引理 6 通过算法 1 得到的蜻蜓网络 $D(p,a,h)$ 顶点的子集为 $V[0],V[1],\cdots,V[\mathrm{floor}(a/2)]$，则二部图 $B(V[i],V[j])$ 没有树分支，其中 $i,j\in[0,\mathrm{floor}(a/2)]$。

证明：有如下两种情形。

情形 1 $a \bmod 2=0$。

对于 $\forall i,j\in[0,\mathrm{floor}(a/2)]$ 且 $i\neq j$，由引理 3 可知，$V[i]$ 中为 $S(m,a-i-1)$ 与 $S(m,i)$，$V[j]$ 中为 $S(m,a-j-1)$ 与 $S(m,j)$，其中 $m\in[0,g-1]$。在第 m 组内部，$V[i]$ 中的 $S(m,a-i-1)$ 与 $S(m,i)$ 与 $V[j]$ 中的 $S(m,a-j-1)$ 与 $S(m,j)$ 必然两两连接，因此对于 $B(V[i],V[j])$ 的每一个连通分支 H 而言，必然满足 $E(H)\geqslant V(H)$。因此 $B(V[i],V[j])$ 没有树分支。

情形 2 $a \bmod 2\neq 0$。

情形 2.1 $\forall i,j\in[1,\mathrm{floor}(a/2)]$ 且 $i\neq j$。

此情形证明同情形 1。得证。

情形 2.2 $\forall j\in[1,\mathrm{floor}(a/2)]$ 且 $i=0$。

由引理 3 可得 $V[j]$ 中为 $S(m,a-j-1)$ 与 $S(m,j)$，其中 $m\in[0,g-1]$。$V[0]$ 中为 $S(m,a-1)$ 与 $S(m,0)$，其中 $m\in[0,g-1]$。在第 m 组内部，$V[j]$ 中的 $S(m,a-j-1)$ 和 $S(m,j)$ 与 $V[0]$ 中的 $S(m,a-1)$ 和 $S(m,0)$ 必然两两连接，同时，$V[0]$ 中的 $S(m,\mathrm{floor}(a/2)+1)$ 与 $V[j]$ 中的 $S(m,j)$ 连接，因此对于 $B(V[i],V[j])$ 的每一个连通分支 H 而言，必然满足 $E(H)>V(H)$。

情形 2.3 $\forall i\in[1,\mathrm{floor}(a/2)]$ 且 $j=0$。

证明同情形 2.2。

综上所述，二部图 $B(V[i],V[j])$ 没有树分支，证毕。

定理 1 通过算法 1 得到的蜻蜓网络 $D(p,a,h)$ 顶点的子集 $V[0],V[1],\cdots,V[\mathrm{floor}(a/2)]$ 是蜻蜓网络 $D(p,a,h)$ 在相对链接下的一个完全独立生成树划分。

证明：由引理 1、引理 4—引理 6 可知，该定理成立。证毕。

3.2 绝对链接方式下顶点的完全独立生成树划分

首先给出找到绝对链接方式下完全独立生成树划分的算法。

算法 2 CISTP2(a,h)

输入：蜻蜓网络每组交换机数 a 以及每个交换机的全局链路数 h

输出：D(p,a,h)的一个完全独立生成树划分：V[0],V[1],…,V[floor(a/2)]

```
1. g=a*h+1;
2.    nextGroup=[];
      /*nextGroup 用于寻找下一个组中交换机*/
3.    for i=0 to floor(a/2)do in parallel:
4.      V[i]=∅;
5.      V[i]=V[i]∪S(0,i)∪S(0,floor(a/2)+i);
6.      nextGroup.append(S(0,i));
7.      nextGroup.append(S(0,floor(a/2)+i));
8.      while nextGroup!=null
9.        next=nextGroup[0];
10.         for j=0 to R(next).length-1
11.         if R(next)[j])not in V[i]
12.           V[i]=V[i]∪R(next)[j]∪S(G(R(next)[j]),
              N(|R(next)[j]-floor(a/2)|);
13.         end if
14.       end for
15.     nextGroup=nextGroup.remove(next);
16.     n=(floor(j/2)+1)mod a;
17.     nextGroup.append((g+i+R(next)[j].length)(modg),n);
18.     if V[i].length >=g
19.       break;
20.     end if
21.   end while
22. end for
23. if a mod 2≠0
24.   for j=0 to g-1
25.     V[0]=V[0]∪S(j,floor(a/2)+1);
26.   end for
27. end if
```

28. return V; /* V 中有 V[0],V[1],…,V[floor(a/2)] */
29. end function

下面以 $D(p,4,2)$为例来描述算法 2 的过程。对于 $V[0]$与 $V[1]$而言,其表示的是顶点的一个子集。首先将第 0 组的第 0 与 2 号顶点 $S(0,0),S(0,2)$分配给 $V[0]$,此时 $V[0]=\{S(0,0),S(0,2)\}$。并且在 nextGroup 记录下分配完的顶点,用于寻找下一个组的对应顶点。由引理 2 可知,$R(S(0,0))$中有 $S(1,0)$与 $S(2,0)$,因此将 $S(1,0)$与 $S(2,0)$分配给 $V[0]$,并在 nextGroup 中删除 $S(0,0)$。与相对链接方式不同的是,我们需要加入 nextGroup 数组的元素是 $R(S(0,0))$中编号最大的组中的(floor(0/2)+1)mod a 号顶点,此处即 $S(2,1)$。此时 $V[0]=\{S(0,0),S(0,2),S(1,0),S(2,0)\}$。依次类推,最终得到 $V[0]=\{S(0,0),S(0,2),S(1,0),S(2,0),S(1,2),S(2,1),S(5,0),S(6,3),S(5,6),S(7,3),S(3,1),S(3,3),S(4,1),S(4,3),S(7,3),S(7,2),S(8,3),S(8,1)\}$。同理可得 $V[1]=\{S(1,1),S(1,3),S(3,0),S(3,2),S(4,0),S(4,2),S(5,2),S(5,1),S(6,2),S(6,1),S(7,1),S(7,0),S(8,0),S(8,2),S(2,3),S(2,2),S(1,1),S(1,3)\}$。图 6 给出了 $D(p,4,2)$通过算法 2 得到的两个顶点划分。

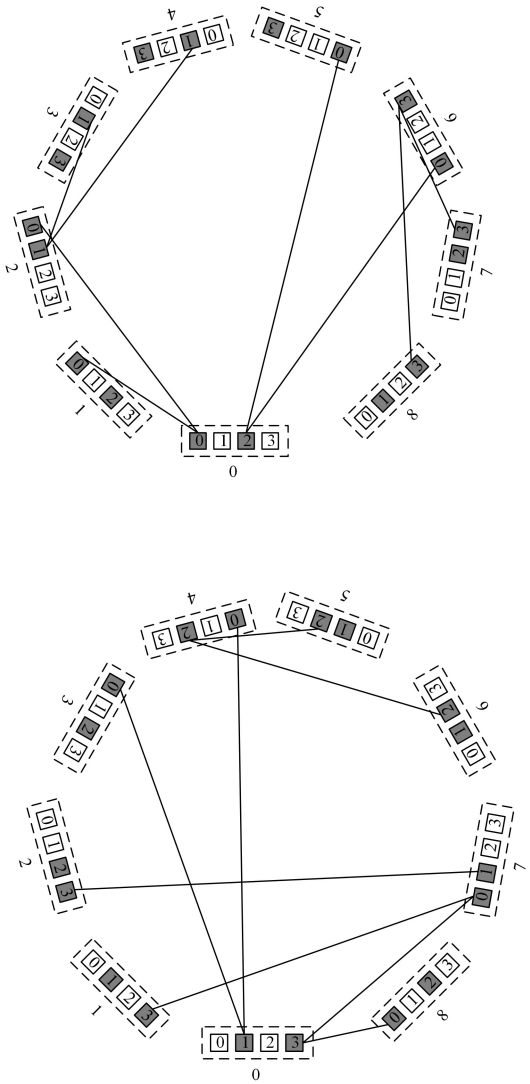

图 6 $D(p,4,2)$在绝对链接下的一个顶点划分

Fig. 6 Vertex partition of $D(p,4,2)$ under absolute link

显然 $V[0]$与 $V[1]$是 $D(p,4,2)$的一个顶点划分。且由图 6 可看出,$G[V[i]],i\in\{0,1\}$是连通的。在任意的一组中,$V[0]$与 $V[1]$各有两个不同的顶点,因此对于每一个分支而言,必然是边数大于或等于顶点数。因此得到的两个顶点划分 $V[0]$与 $V[1]$是 $D(p,4,2)$的一个完全独立生成树划分。

引理 7 设蜻蜓网络 $D(p,a,h)$顶点集合为 V,则通过算法 2 得到的顶点的子集 $V[0],V[1],\cdots,V[\text{floor}(a/2)]$为 V 的一个划分。

证明:设 $mid=\text{floor}(a/2)$,$g=ah+1$。对于任意组 i 内的 $S(i,j)$与 $S(i,mid+i)(j\in[0,mid],i\in[0,ah])$。根据蜻蜓网络全局链路绝对链接方式排列的规则,$R(S_{i,j})=R^{+}(S_{i,j})\cup R^{-}(S_{i,j})$,其中,$R^{+}(S_{i,j})=\{S_{m,n}\mid m>i,i+hj+1 \pmod g\leqslant m\leqslant i+h(j+1) \pmod g,n=\text{floor}(i/h)\}$,$R^{-}(S_{i,j})=\{S_{m,n}\mid m<i,i+hj+1 \pmod g\leqslant m\leqslant i+h(j+1) \pmod g,n=\text{floor}((i-1)/h)\}$。根据算法 2,通过 nextGroup 将 $R(S(i,j))$中编号最大的组作为向外扩张的顶点。显然对于 $m\neq n,S(i,m)\neq S(j,m)$。因此得到的 $V[i]\cap V[j]=\varnothing$。

情形 1 $a \bmod 2=0$。

对于 $\forall i\in[0,mid]$,$V[i]$中必然包含 $D(p,a,h)$每组的两个顶点,$|V[i]|=2g$。因此 $|V[0]\cup V[1]\cup\cdots\cup V[mid]|=2g*mid=ag=V(D(p,a,h))$,又有 $V[i]\cap V[j]=\varnothing$,故 $V[0],V[1],\cdots,V[floor(a/2)]$为 V 的一个划分。

情形 2 $a \bmod 2\neq 0$。

对于 $\forall i\in[1,mid]$,$V[i]$中必然包含 $D(p,a,h)$每组的两个顶点,$|V[i]|=2g$。由算法 2 得到 $|V[0]|=2g+g$。因此 $|V[0]\cup V[1]\cup\cdots\cup V[mid]|=2g*(mid-1)+3g=ag=V(D(p,a,h))$。又有 $V[i]\cap V[j]=\varnothing$。故 $V[0],V[1],\cdots,V[\text{floor}(a/2)]$为 V 的一个划分。

综上,$V[0],V[1],\cdots,V[mid]$为 V 的一个划分,其中 $mid=\text{floor}(a/2)$,证毕。

引理 8 通过算法 2 得到的蜻蜓网络 $D(p,a,h)$顶点的子集为 $V[0],V[1],\cdots,V[\text{floor}(a/2)]$,则导出的子图 $G[V[i]]$是连通的,其中 $i\in[0,\text{floor}(a/2)]$。

证明:由于蜻蜓网络组内部是通过全对全的方式连接,因此组内任意两顶点 $S(m,n_1)$与 $S(m,n_2)$必然是连通的($m\in[0,g-1],n_1,n_2\in[0,a-1]$)。对于不同组之间的顶点有如下两种情形。

情形 1 $a \bmod 2=0$。

对于 $\forall i\in[0,\text{floor}(a/2)]$,由引理 7 可知,$V[i]$中有任意一组中的两个顶点。由相对连接的定义可知,$G[V[i]]$是连通的。

情形 2 $a \bmod 2\neq 0$。

对于 $\forall i\in[1,\text{floor}(a/2)]$,与情形 1 同理可得 $G[V[i]]$是连通的。当 $i=0$ 时,$\forall j\in[0,a-1]$且 $j\neq\text{floor}(a/2)+1$。则同样由引理 7 可得 $V[i]$中有任意一组中的两个顶点。由相对连接的定义可知,$G[V[i]]$是连通的。将每组中间的第 floor(a/2)+1 号顶点加入 $V[0]$之中,每组内部是全对全连接,因此每组中的 floor(a/2)+1 号顶点在导出子图中必然也是连通的。

综上所述,导出子图 $G[V[i]]$是连通的,证毕。

引理 9 通过算法 1 得到的蜻蜓网络 $D(p,a,h)$ 顶点的子集为 $V[0],V[1],\cdots,V[\text{floor}(a/2)]$，则二部图 $B(V[i],V[j])$ 没有树分支，其中 $i,j\in[0,\text{floor}(a/2)]$。

证明：我们有如下两种情形。

情形 1 $a \bmod 2=0$。

对于 $\forall i,j\in[0,\text{floor}(a/2)]$ 且 $i\neq j$，由引理 7 可知，$V[i]$ 中有任意一组中的两个顶点。在任意一组内部，$V[i]$ 中的 $S(m,p)$ 和 $S(m,q)$ 与 $V[j]$ 中的 $S(m,t)$ 与 $S(m,k)$ 必然两两连接，其中 $m\in[0,g]$；$p,q,t,k\in[0,a-1]$ 且 p,q,t,k 两两不相等。因此对于 $B(V[i],V[j])$ 的每一个连通分支 H 而言，必然满足 $E(H)\geqslant V(H)$。因此 $B(V[i],V[j])$ 没有树分支。

情形 2 $a \bmod 2 \neq 0$。

情形 2.1 $\forall i,j\in[1,\text{floor}(a/2)]$ 且 $i\neq j$。

此情形证明同情形 1。得证。

情形 2.2 $\forall j\in[1,\text{floor}(a/2)]$ 且 $i=0$。

由引理 7 可得 $V[j]$ 中为 $S(m,p)$ 与 $S(m,q)$，其中 $m\in[0,g-1]$；$p,q\in[0,a-1]$ 且 $p\neq q$。$V[0]$ 中为 $S(m,t)$ 与 $S(m,k)$，其中 $m\in[0,g-1]$。在第 m 组内部，$V[i]$ 中的 $S(m,p)$ 和 $S(m,q)$ 与 $V[0]$ 中的 $S(m,t)$ 和 $S(m,k)$ 必然两两连接，同时 $V[0]$ 中的 $S(m,\text{floor}(a/2)+1)$ 与 $V[j]$ 中的 $S(m,j)$ 连接，因此对于 $B(V[i],V[j])$ 的每一个连通分支 H 而言，必然满足 $E(H)>V(H)$。

情形 2.3 $\forall i\in[1,\text{floor}(a/2)]$ 且 $j=0$。

证明同情形 2.2。

综上所述，二部图 $B(V[i],V[j])$ 没有树分支。证毕。

定理 2 通过算法 2 得到的蜻蜓网络 $D(p,a,h)$ 顶点的子集 $V[0],V[1],\cdots,V[\text{floor}(a/2)]$ 是蜻蜓网络 $D(p,a,h)$ 在相对链接下的一个完全独立生成树划分。

证明：由引理 1，引理 7—引理 9 可知，该定理成立。证毕。

3.3 循环链接方式下顶点的完全独立生成树划分

循环链接与相对链接方式较为接近，区别在于寻找下一组交换机时相对链接将第 i 组的编号为 k 的交换机 $S(i,k)$ 链接向它第 $i+hk+1(\text{mod } g),\cdots,i+h(k+1)\ (\text{mod } g)$ 组的编号为 $a-k-1$ 的交换机，因此每个顶点划分选取每组的第 $a-k-1$ 号与第 k 号顶点。而循环链接则是链接向对应组的 k 号交换机，则同样选取每组中第 k 与第 $a-k-1$ 号顶点。因此只需要将算法 1 中用来寻找下一个组的第 16 行换成 nextGroup.append($S(G(R(\text{next})[j])),i$)；nextGroup.append($S(G(R(\text{next})[j])),a-i-1$)；

通过修改后的算法 1 我们可以得到 $D(p,4,2)$ 在循环链接下的一个完全独立生成树的划分。图 7 给出了 $D(p,4,2)$ 通过算法 1 得到的两个顶点划分。

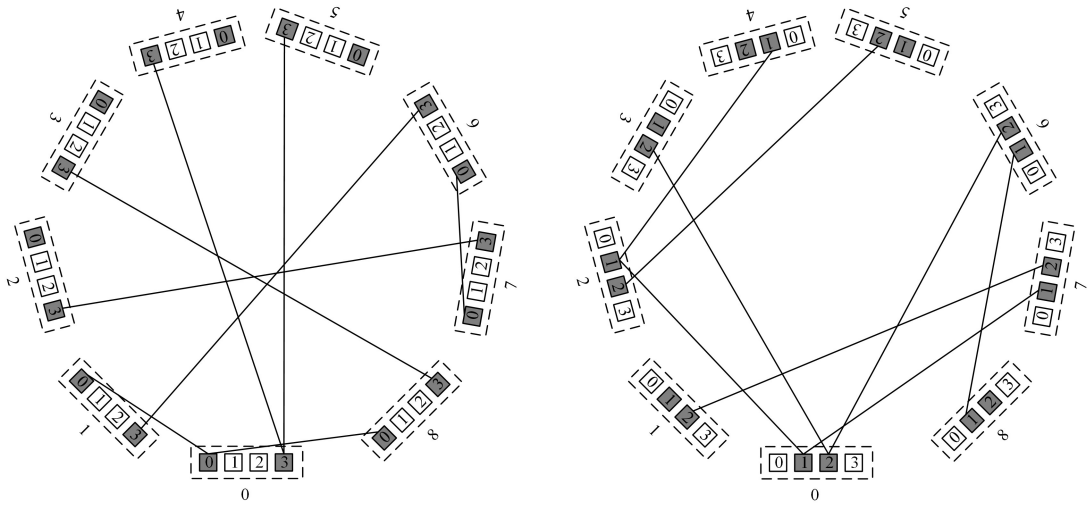

图 7 $D(p,4,2)$ 在循环链接下的一个顶点划分

Fig. 7 Vertex partition of $D(p,4,2)$ under cyclic link

定理 3 通过修改后的算法 1 得到的蜻蜓网络 $D(p,a,h)$ 顶点的子集 $V[0],V[1],\cdots,V[\text{floor}(a/2)]$ 是蜻蜓网络 $D(p,a,h)$ 在循环链接下的一个完全独立生成树划分。

证明：相对链接将第 i 组的编号为 k 的交换机 $S(i,k)$ 链接向它第 $i+hk+1(\text{mod } g),\cdots,i+h(k+1)\ (\text{mod } g)$ 组的编号为 $a-k-1$ 的交换机。而循环链接则是链接向对应组的 k 号交换机。因此该证明与定理 1 证明过程相似，同理可得该定理正确。

3.4 $D(p,a,h)$ 的完全独立生成树构造算法

通过算法 1 和算法 2，我们找到了 $D(p,a,h)$ 的一个完全独立生成树划分，下面给出算法 3，根据所得到的完全独立生成树划分来构造出它的 floor(a/2) 棵完全独立生成树。

算法 3 CIST1(a,h,V)

输入：V[0],V[1],…,V[floor(a/2)]

输出：完全独立生成树的边集合

```
1. g=a*h+1;
2. for i=0 to floor(a/2)
3.     E[i]=∅;
4.     for j=0 to g-1
5.         m=findIndex(j,V[i]);/*findindex 用来查找 V[i]在第 j 组
           中顶点的序号，由于每组中有至少两个顶点在 V[i]中，因此
           m 为数组，设其递增*/
6.         E[i]=E[i]∪(S(j,m[0]),S(j,m[1]))
7.         E[i]=E[i]∪(S(j,m[0]),R(S(j,m[0])))∪(S(j,m[1]),
               R(S(j,m[1])))
8.         for k=0 to a-1
9.         if (k < floor(a/2) and k > m[0]) or(k > m[0])
10.            E[i]=E[i]∪(S(j,m[0]),S(j,k));
11.        elif (k < m[0])or (k > floor(a/2) and k < m[1])
```

```
12.            E[i]=E[i]∪(S(j,m[1]),S(j,k));
13.          end if
14.          if a mod 2≠0
15.            E[i]=E[i]∪(S(j,m[0]),S(j,floor(a/2)+1);
16.      end for
17.      end for
18.   end for
19.   return E; /*E中有 E[0],E[1],…,E[floor(a/2)]*/
20. end function
```

定理 4 通过算法 3 得到的 floor($a/2$)个边集合是蜻蜓网络 $D(p,a,h)$ 的 floor($a/2$)棵完全独立生成树的边集合。

证明：算法 3 是将算法 1 和算法 2 得到的 floor($a/2$)个完全独立生成树划分中的顶点作为内部顶点，其他顶点依次作为叶子结点挂靠在 $V[i]$之中所得到的边集合，故由此边集合构成的树 $T[i]$中内部结点只在 $V[i]$之中，因此，通过算法 4 得到的 floor($a/2$)个边集合是蜻蜓网络 $D(p,a,h)$ 的 floor($a/2$)棵完全独立生成树的边集合。

图 8 给出了算法 4 的构造过程。

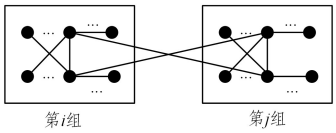

图 8 构造过程

Fig. 8 Construction process

3.5 算法时间复杂度分析

定理 5 算法 1 的时间复杂度为为 O(N)，其中 N 为顶点的个数。

算法 1 通过 nextGroup 数组依次将 $R(i,j)$中的顶点放入 $V[i]$中。每组中有两个顶点进入 nextGroup 中。对于最外层的循环，通过并行构造其 floor($a/2$)棵完全独立生成树。对于内部的循环，所有的顶点至多只进出 nextGroup 一次，故时间复杂度为 O(N)，其中 N 为顶点的个数。

定理 6 算法 2 的时间复杂度为 O(N)，其中 N 为顶点的个数。

该算法与算法 1 类似，同样是通过 nextGro-up 数组来寻找下一个组中的顶点，且每个组中有两个顶点进入 next-Group。因此同算法 1 类似，其时间复杂度为 O(N)，其中 N 为顶点的个数。

定理 7 算法 3 的时间复杂度为 O(N)，其中 N 为顶点的个数。

该算法需要判断每个组中的每个顶点所连接的边，因此时间复杂度为 O(N)，其中 N 为顶点的个数。

结束语 随着信息科技的发展，数据量已经变得非常巨大。特别是在高性能系统中，人们对信息的高效与可靠传输有着严苛的要求。完全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作用。通过完全独立生成树可以并行分发数据，且能保证在任意顶点之间若有 $k-1$ 条路上有服务器发生故障，则至少有 1 条路能够传输信息(假设构造出 k 棵完全独立生成树)，因此研究一个图或者网络的完全独立生成树是非常有必要的。

本文首先介绍了蜻蜓网络相对链接与绝对链接的全局链接方式，其次分别给出了算法 1 与算法 2 来找出相对链接、绝对链接以及循环链接下 $D(p,a,h)$的完全独立生成树划分，并给出算法 3 来构造出 floor($a/2$)棵完全独立生成树的边集合，并对各算法的正确性给出了证明。

参 考 文 献

[1] JACK D. The international exascale software project roadmap [J]. The International Journal of High Performance Computing Applications, 2011, 25(1): 3-60.

[2] DENNIS A, MICHAEL R M, PHILIP M, et al. Energy proportional datacenter networks[J]. ACM SIGARCH Computer Architecture News, 2010, 38(3): 338-347.

[3] BESTA M, HO T. Slim Fly: A cost effective low-diameter network topology[J]. Networking, Storage and Analysis, 2014, 4(4): 348-359.

[4] WILDE T, AUWETER A, SHOUKOURIAN H. The 4 pillar framework for energy efficient HPC data centers[J]. Computer Science—Research and Development, 2014, 29(3/4): 241-251.

[5] CURTSINGER R, BUNDE D. Shortest paths in dragonfly systems[C]// International Workshop of High-Perfomance Interconnection Networks in the Exascale and Big-Data Era. IEEE, 2019: 1-8.

[6] GAHVARI H, GROPP W, JORDAN K E, et al. Algebraic multigrid on a dragonfly network: first experiences on a Cray XC30 [C]// IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, 2015: 3-21.

[7] FAANES G, BATAINEH A, ROWETH D, et al. Cray Cascade: A scalable HPC system based on a dragonfly network[C]// 2012 International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2012: 1-9.

[8] ARIMILLI L B, ARIMILLI R, CHUNG V, et al. The PERCS high-performance interconnect[C]// 18th IEEE Annual Symposium on High Performance Interconnects. IEEE, 2010: 75-82.

[9] WANG X, FAN J X, LIN C K, et al. BCDC: A high-performance, server-centric data center network[J]. Journal of Computer Science and Technology, 2018, 33(2): 400-416.

[10] FUENTES P, VALLEJO E, CAMARERO C, et al. Throughput unfairness in dragonfly networks under realistic traffic patterns [C]// 1st IEEE International Workshop on High-Performance Interconnection Networks Towards the Exascale and Big-Data Era(HiPINEB). IEEE, 2015: 801-808.

[11] YEBENES P, ESCUDERO-SAHUQUILLO J, GARCIA P J, et al. Straightforward solutions to reduce HoL blocking in different Dragonfly fully-connected interconnection patterns[J]. Journal of Supercomputing, 2016, 72(12): 1-23.

[12] YEBENES P, GARCIA P J, QUILES F J, et al. Straightforward modeling of fully-connected dragonfly topologies in HPC-system simulators[C]// 2015 International Conference on High Performance Computing & Simulation (HPCS). IEEE, 2015: 172-178.

[13] CHAKARAVARTHY V T, KATTA N, KEDIA M, et al. Map-

ping Strategies for the PERCS Architecture[C]// 2012 19th International Conference on High Performance Computing. IEEE, 2012:1-10.

[14] PRISACARI B, RODRIGUEZ G, HEIDELBERGERP, et al. Efficient task placement and routing of nearest neighbor exchanges in dragonfly networks[C]// High-performance Parallel and Distributed Computing. 2014.

[15] HASUNUMA T. Completely independent spanning trees in maximal planar graphs[C]// Revised Papers from the International Workshop on Graph-theoretic Concepts in Computer Science. Cham: Springer, 2002:235-245.

[16] QIAN Y, CHENG B L, FAN J X, et al. A general construction method of vertex independent spanning tree in a kind of data center network [J]. Computer Application Research, 2021, 38(7):2130-2134.

[17] PAI K J, CHANG J M. Constructing two completely independent spanning trees in hypercube-variant networks[J]. Theoretical Computer Science, 2016, 652(1):28-37.

[18] YANG M C. Constructing edge-disjointspanning trees in twisted cubes[J]. Information Sciences, 2010, 180(20):4075-4083.

[19] CHENG B L, WANG D J, FAN J X. Constructing completely independent spanning trees in crossed cubes[J]. Discrete Applied Mathematics, 2017, 219:100-109.

[20] LINC K, ZHAO Y, FAN J X, et al. Research on completely independent spanning tree based on vertex degree[J]. Computer Science, 2017(6):93-96.

[21] MAN L J. Some sufficient conditions for the existence of two completely independent spanning trees[D]. Urumqi: Xinjiang University, 2016.

[22] QIN X W, HAO R X, PAI K J, et al. Comments on A Hamilton sufficient condition for completely independent spanning tree [J]. Discrete Applied Mathematics, 2020, 283(3):33-38.

[23] PETERFALVI. Two counterexamples on completely independent spanning trees[J]. DISCRETE MATH, 2012, 312(4):808-810.

[24] HASTINGS E, RINCON-CRUZ D, SPEHLMANN M, et al. Comparing global link arrangements for dragonfly networks [C]// IEEE International Conference on Cluster Computing. IEEE, 2015:361-370.

[25] BELKA M, DOUBET M, MEYERS S, et al. New link arrangements for dragonfly networks[C]// 2017 IEEE 3rd International Workshop on High-Performance Interconnection Networks in the Exascale and Big-Data Era (HiPINEB). IEEE, 2017:325-334.

[26] ARAKI T. Dirac's condition for completely independent spanning trees[J]. Journal of Graph Theory, 2014, 77(3):171-177.

[27] KIM J, DALLY W, SCOTT S, et al. Technology-Driven, Highly-Scalable Dragonfly Topology[J]. Acm Sigarch Computer Architecture News, 2008, 36(3):77-88.

BIAN Qing-rong, born in 1993, postgraduate. His main research interests include parallel and distributed systems, and graph algorithms.

CHENG Bao-lei, born in 1979, Ph.D, associate professor, Ph.D supervisor. His main research interests include parallel and distributed systems, computer networks and graph algorithms.

(责任编辑:何杨)