Independent Spanning Trees on Recursive Circulant Graphs

遞迴環形圖上獨立擴展樹之建構*

唐學明 楊進雄, 王有禮, 林孟玉

政治作戰學校 國立台灣科技大學資訊管理系

stang1@pchome.com.tw ccdir@mail.ntust.edu.tw

Abstract

Two spanning trees of a given graph G are said to be *independent* if they are rooted at the same vertex r and for each vertex v, $v \neq r$, the two paths from r to v, one path in each tree, are internally disjoint. A set of spanning trees of G is said to be independent if they are pairwise independent. A *recursive circulant graph* $R(N,d)$ has $N=cd^m$ vertices, where $0<c<d$, and every vertex v in $R(N,d)$ is adjacent to vertices $v \pm d^k \pmod N$, where $k = 0, 1, 2, ..., m-1$. Since $R(cd^m,d)$ can be recursively partitioned into d induced subgraphs $R(cd^{m-1},d)$, this family of circulant graphs is named as "recursive". $R(cd^m,d)$ is regular with degree δ, where δ is $2m-1$, $2m$, $2m+1$ or $2m+2$, depending on the value of parameters c and d. In this paper, we shall propose efficient algorithms to construct δ independent spanning trees rooted at any vertex in a recursive circulant graph.

Keywords: recursive circulant graphs, independent spanning trees, internally disjoint paths, fault-tolerant broadcasting.

1. Introduction

In this paper, we deal with the independent spanning trees on a special family of interconnection network, called *recursive circulant graphs*. A recursive circulant graph $R(N,d)$ has $N=cd^m$ vertices labeled from 0 to $N-1$, and every vertex v in $R(N,d)$ is adjacent to vertices $[v \pm d^k]_N$, where $0<c<d$, $m>0$, and $k = 0, 1, 2, ..., m-1$. Notice that $[u]_c$ denotes u modulo c. Recursive circulant graphs are vertex symmetric, and thus regular [13]. We denote by δ the degree of vertices in $R(cd^m,d)$. Then, δ in closed-form is shown below:

$$\delta = \begin{cases} 2m-1 & \text{if } c=1, d=2, m \geq 2; \\ 2m & \text{if } c=1, d>2, m \geq 1; \\ 2m+1 & \text{if } c=2, m \geq 1; \\ 2m+2 & \text{if } c>2, m \geq 1. \end{cases}$$

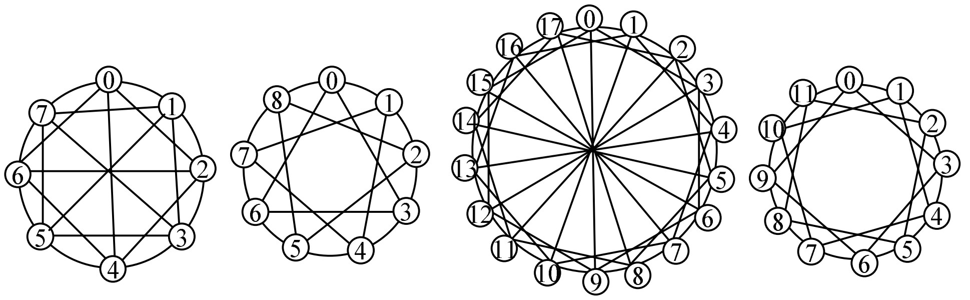
For example, Figure 1 shows the graphs $R(8,2)$, $R(9,3)$, $R(18,3)$ and $R(12,4)$, which stand for distinct cases of parameters c and d.

Recursive circulant graph $R(cd^m,d)$ has a recursive structure since the graph can be partitioned into d induced subgraphs isomorphic to $R(cd^{m-1},d)$. For example, $R(18,3)$ shown in Figure 1(c) contains three disjoint copies of $R(6,3)$. Notice that each induced subgraph $R(6,3)$ of $R(18,3)$ contains exactly those vertices having the same remainder of division by 3. In Figure 1(c), vertices 0, 3, 6, 9, 12 and 15 induce an $R(6,3)$ subgraph of $R(18,3)$. Besides, the *basic cycle* of a recursive circulant graph is the cycle that consists of all the edges not in the induced subgraphs [4]. In $R(18,3)$, the basic cycle contains edges (0,1), (1,2), ..., (16,17), (17,0) which form a Hamiltonian cycle.

In 1994, Park and Chwa first proposed recursive circulant graphs [13]. This family of graphs is important due to its flexibility and extensibility. In addition, they are suitable for developing algorithms [4, 10, 11, 14, 16, 18].

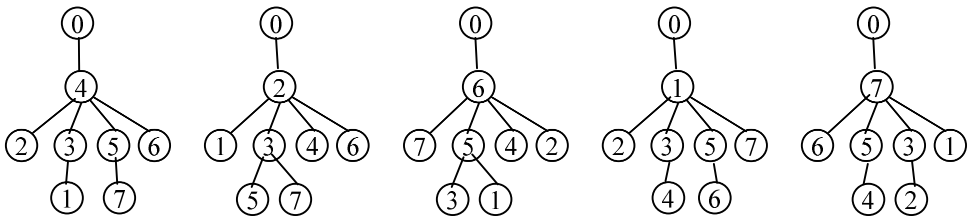
Next, we introduce the definition of independent spanning trees. Considering a graph $G=(V,E)$, a tree T is called a *spanning tree* of G if T is a subgraph of G and T contains all vertices in V. Two spanning trees of G are said to be *independent* if they are rooted at the same vertex, say r, and for each vertex $v \in V \setminus \{r\}$, the two paths from r to v, one path in

* This work was supported by the National Science Council, Republic of China, under Contract 92-2416-H-135-001.



(a) $R(8,2)$ c=1,d=2,m=3 (b) $R(9,3)$ c=1,d=3,m=2 (c) $R(18,3)$ c=2,d=3,m=2 (d) $R(12,4)$ c=3,d=4,m=1

Figure 1. Examples of recursive circulant graphs.



$T(8,4)$ $T(8,2)$ $T(8,6)$ $T(8,1)$ $T(8,7)$

Figure 2. Five independent spanning trees of $R(8,2)$.

each tree, are internally disjoint. A set of spanning trees of a graph is said to be independent if they are pairwise independent.

Finding independent spanning trees of a given graph has applications in the fault-tolerant broadcasting protocols of distributed computing networks [2,7,12,15]. The fault-tolerance is achieved by sending k copies of the message from the root (source node) of k independent spanning trees. If the source node is faultless, this broadcasting protocol can tolerate up to k-1 faulty nodes.

In [7], Itai and Rodeh gave a linear time algorithm for finding two independent spanning trees in a biconnected graph. In [5], Cheriyan and Maheshwari showed that, for any 3-connected graph G and for any vertex r of G, three independent spanning trees rooted at r can be found in $O(|V||E|)$ time. In [19] and [9], the authors conjectured that any k-connected graph has k independent spanning trees rooted at an arbitrary vertex r. This conjecture is still open for arbitrary k-connected graphs with $k > 3$. In [6], Huck has proved that the conjecture is true for planar graphs.

In [17], Tang et al. studied the independent spanning trees of k-connected and k-regular graphs. They found several common properties, which are helpful for finding independent spanning trees in these graphs. Let G be a k-connected and k-regular graph and let *IST* denote a set of k independent spanning trees, if exists, rooted at vertex r in G. Then, the following properties must hold.

Property 1. *The root vertex r has only one child in every tree of the IST.*

Property 2. *The root vertex r has k−1 grandchildren in every tree of the IST.*

For example, an IST of $R(8,2)$ is shown in Figure 2. Since $R(8,2)$ is a 5-connected and 5-regular graph, the IST contains five independent spanning trees rooted at one vertex. The root vertex of every tree has one child and four grandchildren. Throughout this paper, we denote a tree in an IST of $R(N,d)$ by $T(N,t)$, where t is the only child of the root.

The *neighborhood* of vertex v, denoted by $N(v)$, is the set of vertices adjacent to v in a graph. Let parent(v,t) denote the *parent* of vertex v in tree $T(N,t)$ and let ancestor(v,t) denote the *ancestor set* of a vertex v in $T(N,t)$. The following properties also hold in an IST of $R(N,d)$.

Property 3. *For every non-root vertex v, the union of all parent(v,t) in an IST is the* *neighbor of vertex v.*

Property 4. *For every non-root vertex v, the intersection of ancestor(v,i) and ancestor(v,j) is the root vertex, where T(N,i) and T(N,j) are two distinct trees in the IST.*

Using vertex 4 in the IST shown in Figure 2 as an example, parent(4,4) ∪ parent(4,2) ∪ parent(4,6) ∪ parent(4,1) ∪ parent(4,7) = {0,2,6,3,5} = $N(4)$. For any two trees $T(8,i)$ *and* $T(8,j)$, ancestor(4,i) ∩ ancestor(4,j) = {0}. By the way, Property 4 can be used to verify the independency of a set of spanning trees rooted at one vertex.

A *chordal ring*, denoted by $C(N,d)$, is a 4-connected and 4-regular graph with vertex set $V = \{0,1,2,...,N-1\}$ and edge set $E = \{(u,v) | [v-u]_N = 1 \text{ or } d\}$, where $2 \le d < N/2$ [1,3]. Chordal rings have close relation to recursive circulant graphs. For example, $R(9,3)$ and $R(12,4)$ shown in Figure 1 are both chordal rings, while $R(8,2)$ and $R(18,3)$ are not. On the other hand, $C(12,4)$ is a recursive circulant graph, while $C(12,3)$ is not. In [8], Iwasaki et al. proposed an algorithm to find an IST of $C(N,d)$. Their algorithm also works for recursive circulant graphs with degree 4. Furthermore, like chordal rings, some IST's of a recursive circulant graph have the following property.

Property 5. *For all $t \le N/2$, $T(N,N-t)$ is constructed symmetrically from $T(N,t)$ by changing each non-root vertex v to $[N-v]_N$.*

For example, see Figure 2 again. $T(8,6)$ is obtained from $T(8,2)$, while $T(8,7)$ is obtained from $T(8,1)$ using the symmetrical property of $R(8,2)$. Particularly, $T(8,4)$ is symmetrical to itself.

As we mentioned at the beginning of this section, recursive circulant graphs are δ-connected and δ-regular, but δ is varied according to the value of parameters c, d and m. In this paper, we shall prove that Zehavi's conjecture is true for all recursive circulant graphs. That is, we shall propose efficient algorithms to construct δ independent spanning trees rooted at any vertex in a recursive circulant graph. Since recursive circulant graphs are vertex-symmetric. Without loss of generality, we simply consider independent spanning trees rooted at vertex 0 of a recursive circulant graph.

The remainder of this paper is organized as follows. In Section 2, we shall propose an algorithm for constructing an IST of $R(2^m,2)$. In Section 3, we shall propose an algorithm for constructing an IST of $R(d^m,d)$, for all $d > 2$. Section 4 contains our concluding remarks.

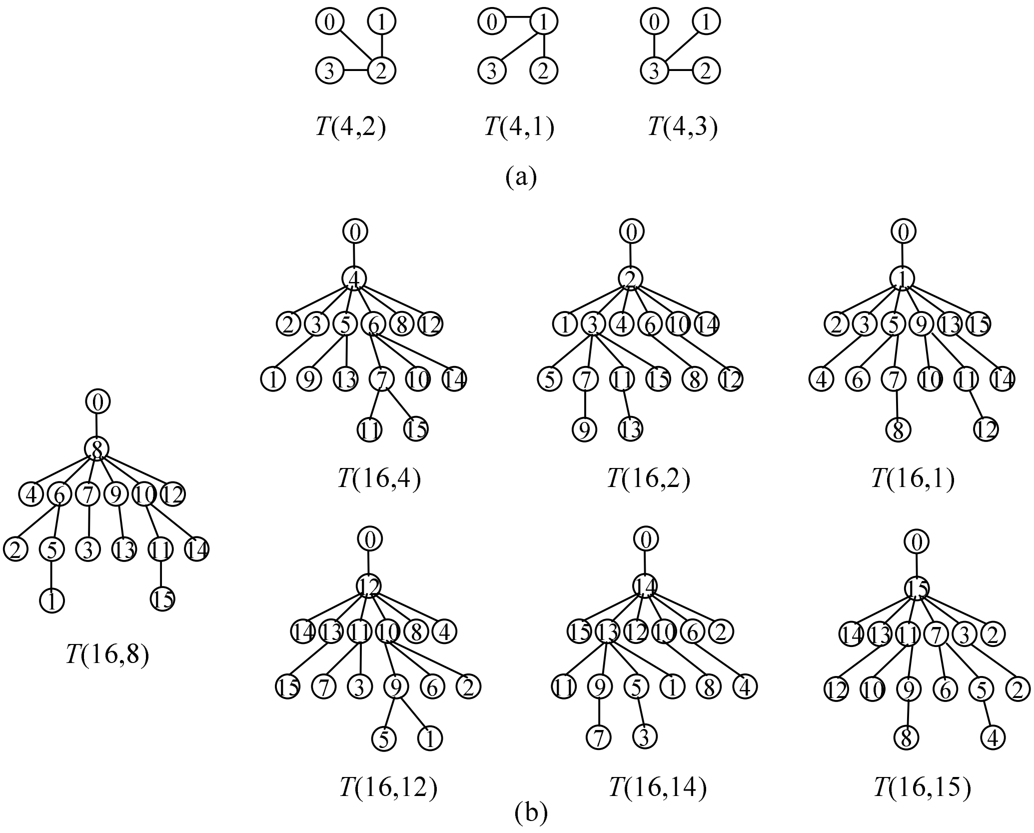


Figure 3. (a) The IST of $R(4,2)$; (b) an IST of $R(16,2)$.

2. Constructing independent spanning trees on $R(2^m, 2)$

In this section, recursive circulant graph $R(2^m,2)$ is taken into consideration. $R(2^m,2)$ is $(2m-1)$-connected and $(2m-1)$-regular, where $m \geq 2$. In case of $m=2$, $R(4,2)$ is a 4-clique. The IST of $R(4,2)$ is shown in Figure 3(a).

Note that the degree of each vertex in $R(2^m,2)$ is two greater than the degree of each vertex in $R(2^{m-1},2)$. Thus, an IST of $R(2^m,2)$ contains two more spanning trees than an IST of $R(2^{m-1},2)$. The construction algorithm consists of three steps. First, we use a recursive procedure to construct $T(2^m,2^{m-1})$, $T(2^m,2^{m-2})$, ..., $T(2^m,2^2)$ and $T(2^m,2)$. Next, we construct $T(2^m,2^m-2^{m-1})$, $T(2^m,2^m-2^{m-2})$, ..., $T(2^m,2^m-2^2)$ and $T(2^m,2^m-2)$ symmetrically. Finally, we use a bit-comparing scheme to determine the parent of each vertex in $T(2^m,1)$ and $T(2^m, 2^m-1)$. For some conflicting edges among these spanning trees, necessary transformation is performed in order to hold the independent property. We give the construction algorithm of an IST of $R(2^m,2)$ as follows.

Algorithm *IST_R2*(m)
Input: m.
Output: An IST of $R(2^m,2)$.
Method:
Step 1. **If** $m = 2$, **then** return the IST of $R(4,2)$;
  else call ***IST_R2***$(m-1)$
  endif
Step 2. **(** construct $T(2^m,2^i)$, where $i = 1,2, \ldots, m-1$ **)**
  For $i = 1$ **to** $m-1$ **do**
   Substep 2.1 (Create the only child of the root)
    parent$(2^i,2^i) = 0$
   Substep 2.2 (Create $2m-2$ grandchildren)
    For each vertex v is a neighbor of vertex 2^i **and** $v \neq 0$ **do**
     parent$(v, 2^i) = 2^i$
    enddo
   Substep 2.3 (determine the parents)
    For each vertex v is not a neighbor of vertex 2^I **do**
     If v is even, **then**
      let p = parent$(v/2,2^{i-1})$ in IST of $R(2^{m-1},2)$,
      parent$(v,2^i) = 2p$
     else (v is odd)
      if $v < 2^i$, **then**
       let p = parent$((v+1)/2,2^{i-1})$ in IST of $R(2^{m-1},2)$,
       parent$(v,2^i) = 2p-1$;
      else **(**$v > 2^i$**)**
       let p = parent$((v-1)/2,2^{i-1})$ in IST of $R(2^{m-1},2)$,
       parent$(v,2^i) = 2p+1$
      endif
     endif
    enddo
  enddo
Step 3. (symmetrically construct $T(2^m, 2^m-2^i)$, where $i = 1,2, \ldots, m-2$)
For $i = 1$ **to** m**–2 do**
  For every vertex v in $T(2^m,2^i)$ **and** $v \neq 0$ **do**
   replace the label of v with 2^m-v
  enddo
enddo
Step 4. (construct $T(2^m,1)$ and $T(2^m,2^m-1)$**)**
  Substep 4.1 (Create the only child of the root)
   parent$(1,1) = 0$;
   parent$(2^m-1, 2^m-1) = 0$.
  Substep 4.2 (create $2m-2$ grandchildren of the root)
   For each vertex v is a neighbor of vertex 1 **and** $v \neq 0$ **do**
    parent$(v,1) = 1$;
    parent$(2^m-v,2^m-1) = 2^m-1$
   enddo
  Substep 4.3 (determine parent and do transformation)
   For each vertex v is not a neighbor of vertex 1 **do**
    Let $v_{m-1}v_{m-2} \ldots v_1v_0$ be the m-bit binary string of vertex v, and let v_p be the right-most different bit between vertices v and 1. We set
     parent$(v,1) = v-2^p$;
     parent$(2^m-v, 2^m-1) = 2^m-v+2^p$
    If $p \neq 0$ **then** we can find the transformed tree $T(2^m,x)$ where parent$(v,x) = v-2^p$. We set
     parent$(v,x) = v-1$;
     parent$(2^m-v, 2^m-x) = 2^m-v+1$
    endif
   enddo
End of Algorithm *IST_R2*

We use the IST shown in Figure 3(b) to illustrate Algorithm IST_R2. In Step 2, $T(16,8)$, $T(16,4)$ and $T(16,2)$ are directly obtained from $T(8,4)$, $T(8,2)$ and $T(8,1)$ (as shown in Figure 2). In Step 3, $T(16,12)$ and $T(16,14)$ are obtained symmetrically from $T(16,4)$ and $T(16,2)$, respectively. In Step 4, $T(16,1)$ and $T(16,15)$ are constructed simultaneously. By comparing the binary string of vertex v with the binary string of vertex 1, the parent of every vertex v in $T(16,1)$ is determined. For example, the binary string of vertex 11 is 1011, the second bit (p=1) is the right-most different bit between 1011 and 0001. Thus, parent$(11,1) = 11-2^1 = 9$. Furthermore, if parent$(v,1) \neq 1$ (v is not a grandchild of the root) and $v-$parent$(v,1) > 1$ (v is not a leaf or edge (parent(v),v) is not in the basic cycle of $R(16,2)$), there must be a conflicting edge in tree $T(16,x)$ in which parent(v,x) = parent(v,1). This is a violation to both Properties 3 and 4. Therefore, parent(v,x) should be changed to $v-1$. For example, parent(11,1) = 9 in Substep 4.3 causes a conflict with parent(11,8) = 9 in Substep 2.3 and, thus, parent(11,8) is changed to 10.

In Algorithm IST_R2(m), an IST of $R(2^m,2)$ is constructed by recursively calling IST_R2($m-1$) twice, computing the symmetrical trees, computing $T(2^m,1)$ and $T(2^m,2^m-1)$, and then doing some transformation.

Let $f(N)$ denote the running time of Algorithm IST_R2, where $N=2^m$ is the number of vertices in $R(2^m,2)$. Then, we have a recurrence equation that bounds $f(N)$:

$$f(N) = 2f(N/2) + cN,$$

where c is a constant. By means of repeated substitution, we can prove that $f(N)$ is $O(N \log_2 N)$, or $O(mN)$. Thus, Algorithm IST takes $O(mN)$ time to construct $2m-1$ independent spanning trees of $R(2^m,2)$.

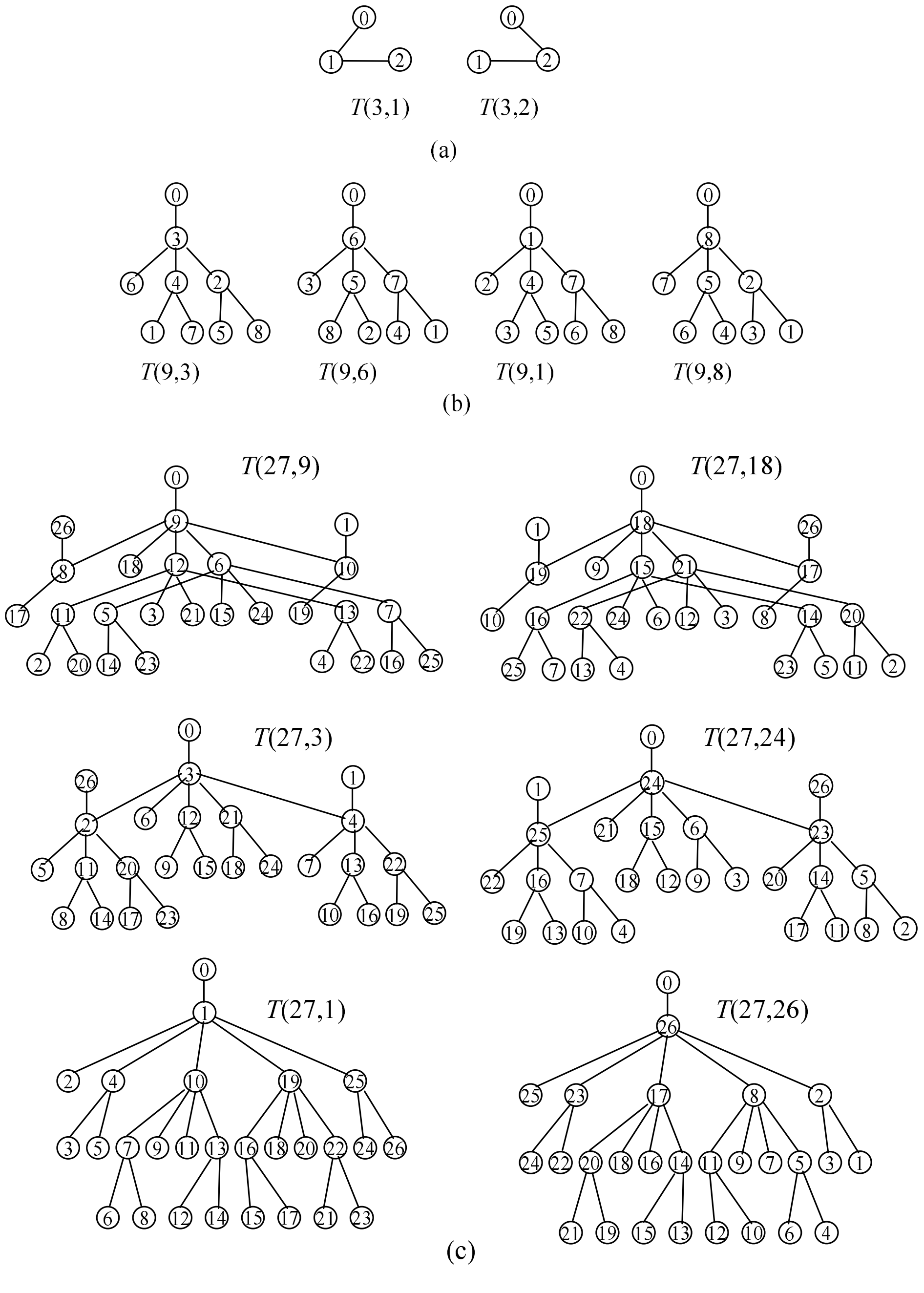


Figure 4. (a) The IST of $R(3,3)$; (b) an IST of $R(9,3)$; (c) an IST of $R(27,3)$.

Lemma 1. *The $2m-1$ trees constructed using IST_R2 are spanning trees of $R(2^m,2)$.*

Proof: We will prove this lemma by induction on m. For m=2, it is obviously true, see the three trees shown in Figure 3(a). Suppose that $T(2^{m-1},2^i)$ ($\mathbf{0} \le i \le m\mathbf{-2}$) are spanning trees of $R(2^{m-1},2)$. Then, $T(2^m,2^i)$ ($\mathbf{1} \le i \le m\mathbf{-1}$) are also spanning trees of $R(2^m,2)$ since they are obtained directly from $T(2^{m-1},2^{i-1})$ and, mostly important, the transformation step (Substep 4.3) does not change the tree property of any transformed tree. $T(2^m,1)$ is created in such a regular manner that for every vertex v (v is not a neighbor of vertex 1), parent(v,1) is less than v. Since 2^m-1 edges are created in Step 4 and no cycle is formed, $T(2^m,1)$ is also a spanning tree of $R(2^m,2)$. As for $T(2^m,2^m-2^i)$ ($\mathbf{0} \le i \le m\mathbf{-2}$), they are spanning trees of $R(2^m,2)$ due to the fact that they are obtained directly form $T(2^m,2^i)$. **Q.E.D.**

Next, we will prove the spanning trees constructed using Algorithm IST_R2 are mutually independent. We only give a concise proof.

Theorem 2. *Algorithm IST_R2 correctly constructs an IST of $R(2^m,2)$ in $O(mN)$ time, where $N = 2^m$ is the number of vertices in $R(2^m,2)$.*

Proof: Let $T(2^m,t)$ be a spanning tree constructed using Algorithm IST_R2. Let $t_{m-1}t_{m-2}\ldots t_1t_0$ denote the m-bit binary string of vertex t, and let t_p is the right-most non-zero bit in t. We can prove that every vertex in the path from v to t in $T(2^m,t)$ changes according to an ordered sequence like $\{\pm 2^p, \pm 2^{p+1}, \ldots, \pm 2^{m-2}, \pm 2^{m-1}, \pm 2^0, \pm 2^1, \ldots, \pm 2^{p-1}\}$. Note that some integers in the sequence are replaced with 0 for a specific vertex. Let $v_{m-1}v_{m-2}\ldots v_1v_0$ be the binary string of vertex v. In case of $v_i = t_i$, the integer $\pm 2^i$ is replaced with 0 in the sequence. We can prove this by induction on m. If this property holds in every tree, then, for any two trees $T(2^m,s)$ and $T(2^m,t)$, ancestor(v,s) $\cap$ ancestor(v,t) = {0}. **Q.E.D.**

3. Constructing independent spanning trees on $R(d^m, d)$, d>2

Suppose $d > 2$, $R(d^m,d)$ is $2m$-connected and $2m$-regular. Thus, $2m$ independent spanning trees should be constructed on $R(d^m,d)$. In the case of d=3 and m=1, $R(3,3)$ is a 3-clique. The IST of $R(3,3)$ is shown in Figure 4(a).

The basic idea of creating an IST of $R(d^m,d)$ is similar to that of $R(2^m,2)$. The most important step is to construct $T(d^m,1)$, which is achieved by a bit-comparing scheme. In addition, the construction algorithm for $R(d^m,d)$ with odd d is a little different from that with even d. In this paper, we only propose the algorithm deals with odd d for conciseness' sake.

Let $v_{m-1}v_{m-2}\ldots v_1v_0$ be the m-bit string of v in base d representation. Then, we assume that v_p is the right-most different bit between vertices v and 1 (in base d representation). In addition, assume v_q is the left bit of v_p, i.e., $q = p+1$ ($p < m-1$) or q=0 (p=$m-1$). We give the construction algorithms for $R(d^m,d)$ as follows.

Algorithm *IST_ Rd*(m)
Input: m.
Output: An IST of $R(d^m,d)$, where d>2 and d is odd)
Method:
Step 1. (recursive procedure)
  If m = 1, **then**
    return the IST of $R(d,d)$;
  else
    call ***IST_Rd***(m–1)
  endif
Step 2. (construct $T(d^m,d^i)$, where i = 1,2, …, m–1)
  For i = 1 **to** m–1 **do**
    Substep 2.1. Construct d trees by copying $T(d^{m-1},d^{i-1})$ and then changing the label of vertex v in d trees from $[dv-(d-1)/2]_N$ to $[dv+(d-1)/2]_N$, respectively.
    Substep 2.2. Construct $T(d^m,d^i)$ by connecting d vertices which are labeled from $d^i-(d-1)/2$ to $d^i+(d-1)/2$.
    Substep 2.3. Symmetrically construct $T(d^m, d^m-d^i)$.
  enddo
Step 3. (construct $T(d^m,1)$)
  For each vertex $v \neq 0$ **do** (determine parent for every vertex)
    Substep 3.1. do case
      case 3.1.1. $v_p = 0$: parent(v,1) = v+1
      case 3.1.2. $v_q = 0$: parent(v,1) = v–1
      case 3.1.3. $0 < v_p \le (d-1)/2$ and $0 < v_q \le (d-1)/2$: parent(v,1) = $v-d^p$
      case 3.1.4. $0 < v_p \le (d-1)/2$ and $(d-1)/2 < v_q$: parent(v,1) = $[v+d^p]_N$
      case 3.1.5. $(d-1)/2 < v_p$: parent(v,1) = v–1
    endcase

Substep 3.2. Find the corresponding transformed tree for cases 3.1.3 and 3.1.4 and do the transformation.

enddo

Step 4. Symmetrically construct $T(d^m, d^m-1)$.

End of Algorithm *IST_Rd*

We use the IST shown in Figure 4(c) to illustrate Algorithm IST_Rd. In Step 2, $T(27,9)$ and $T(27,3)$ are obtained from $T(9,3)$ and $T(9,1)$ (shown in Figure 4(b)), respectively. Meanwhile, $T(27,18)$ and $T(27,24)$ are obtained from $T(27,9)$ and $T(27,3)$, respectively. In Step 3, the parent of every non-root vertex v in $T(27,1)$ is determined by comparing the bit strings of v and 1. For example, By comparing the 3-bit string of vertex 12 (110 in base 3) with the 3-bit string of vertex 1 (001 in base 3), parent(12,1) = 12+1 = 13 since the right-most different bit is v_0 and case 3.1.1 hold. For vertex 13 (111 in base 3), the right-most different bit is v_1 and case 3.1.3 hold, thus parent(13,1) = $13-3^1$ = 10. Since vertex 13 is neither a grandchild of the root nor a leaf in $T(27,1)$, there must be a transformed tree. $T(27,9)$ is the transformed tree in which parent(13,9) is also 10. As a result, parent(13,9) is changed to 12 in order to hold the independent property. In Step 4, $T(27,26)$ is obtained symmetrically from $T(27,1)$. Another example of Algorithm IST_Rd is shown in Figure 5.

In Algorithm IST_Rd, an IST of $R(d^m,d)$ is also recursively constructed. Let $f(N)$ denote the running time of Algorithm IST_Rd, where $N=d^m$ is the number of vertices in $R(d^m,d)$. Then, we can prove that $f(N)$ is also $O(mN)$.

Lemma 3. *The 2m trees constructed using IST_Rd are spanning trees of $R(d^m,d)$.*

The proof of Lemma 3 and Theorem 4 is similar to Lemma 1.

Theorem 4. *Algorithm IST_Rd correctly constructs an IST of $R(d^m,d)$ in $O(mN)$ time, where $N = d^m$ is the number of vertices in $R(d^m,d)$.*

The proof of Theorem 4 is similar to Theorem 2, but more complex. We omit it for conciseness' sake.

4. Concluding remarks

In this paper, we present different algorithms for constructing independent spanning trees on recursive circulant graphs $R(2^m,2)$ and $R(d^m,d)$. We can generalize these results to all recursive circulant graphs. Intuitively, by pruning some vertices, an IST of $R(d^m,d)$ can be transformed to an IST of $R(cd^{m-1},d)$ with $1<c<d$.

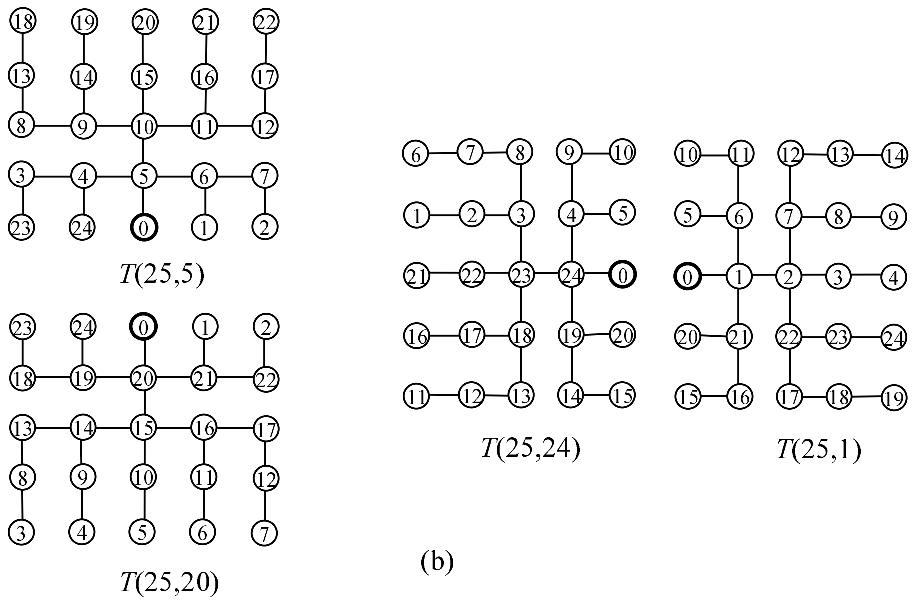


Figure 5. An IST of $R(25,5)$.

References

[1] B. W. Arden and H. Lee, Analysis of Chordal Ring Network, *IEEE Transactions on* Computers, Vol. C-30, No. 4, April 1981, pp. 291-295.

[2] F. Bao, Y. Igarashi and S.R. Öhring, Reliable Broadcasting in Product Networks, *IEICE Technical Report COMP* 95(18), 1995, pp.57-66. (also in *Discrete Applied Mathematics* 83, 1998, pp.3-20.)

[3] J. C. Bermond, F. Comellas, and D. F. Hsu, Distributed Loop Computer Networks: A Survey, *Journal* of *Parallel and Distributed Computing*, Vol. 24, 1995, pp. 2-10.

[4] Daniel K. Biss, Hamiltonian Decomposition of Recursive Circulant Graphs, *Discrete Mathematics* 214, 2000, pp.89-99.

[5] J. Cheriyan and S. N. Maheshwari, Finding Nonseparating Induced Cycles and Independent Spanning Trees in 3-Connected Graphs, *Journal of Algorithms* 9, 1988, pp.507-537.

[6] Andreas Huck, Independent Trees in Planar Graphs, *Graphs and Combinatorics* 15, 1999, pp.29-77.

[7] Alon Itai and Michael Rodeh, The Multi-Tree Approach to Reliability in Distributed Networks, *Proceedings of the 25th Annual IEEE Symposium on Fundamental Computer Science*, 1984, pp.137-147. (also in *Information and Computation* 79, 1988, pp.43-59.)

[8] Y. Iwasaki, Y. Kajiwara, K. Obokata and Y. Igarashi, Independent Spanning Trees of Chordal Rings, *Information Processing Letters* 69, 1999, pp.155-160.

[9] Samir Khuller and Baruch Schieber, On Independent Spanning Trees, *Information Processing Letters* 42, 1992, pp.321-323.

[10] H. S. Lim, J. H. Park and K. Y. Chwa, Embedding Trees in Recursive Circulants, *Discrete Applied Mathematics* 69, 1996, pp.83-99.

[11] C. Micheneau, Disjoint Hamiltonian Cycles in Recursive Circulant Graphs, *Information Processing Letters* 61, 1997, pp.259-264.

[12] K. Obokata, Y. Iwasaki, F. Bao and Y. Igarashi, Independent Spanning Trees of Product Graphs, *Lecture Notes in Computer Science* 1197, 1996, pp.338-351. (also in *IEICE Transaction on* Fundamentals *of Electronics, Communications and Computer Sciences*, Vol. E79-A, No. 11, 1996, pp.1894-1903.)

[13] J. H. Park and K. Y. Chwa, Recursive Circulant: A New Topology for Multicomputer Networks, *Proceedings of International Symposium on Parallel Architectures, Algorithms and Networks* (*ISPAN*), 1994, pp.73-80.

[14] J. H. Park and K. Y. Chwa, Recursive Circulants and Their Embeddings among Hypercubes, *Theoretical Computer Science* 244, 2000, pp.35-62.

[15] P. Ramanathan and K. G. Shin, Reliable Broadcast in Hypercube Multicomputers, *IEEE Transactions on computers* 37(12), 1988, pp.1654-1657.

[16] I. Stojmenovic, Multiplicative Circulant Networks Topological Properties and Communication Algorithms, *Discrete Applied Mathematics* 77, 1997, pp.281-305.

[17] S.-M. Tang, Y.-L. Wang and J.-X. Lee, On the Height of Independent Spanning Trees of A *k*-connected *k*-regular Graph, *Proceedings of National Computer Symposium*, Taipei, 2001, pp.A159-A164.

[18] C.-H. Tsai, Fault-tolerant Hamiltonian Properties on Butterflies, Recursive Circulant graphs and Hypercubes, Dissertation of National Chiao Tung University, 2002.

[19] Avram Zehavi and Alon Itai, Three Tree-paths, *Journal of Graph Theory* 13, 1989, pp.175-188.