A One-to-Many Parallel Routing Algorithm on Generalized Petersen Networks*

Wei–Te Yen[1], Shyue–Ming Tang[2] and Jou–Ming Chang[3]

[1,3]Institute of Information and Decision Sciences,
National Taipei University of Business, Taipei, Taiwan.
[2]Department of Psychology and Social Work,
National Defense University, Taipei, Taiwan.

Abstract

The one-to-many routing is to construct internally disjoint paths from one node to other nodes. It can ensure fault-tolerant broadcasting and secure message distribution in a network. Constructing multiple independent spanning trees rooted at one node can guarantee the one-to-many routing of the network.

A generalized Petersen network GP(n,k) is formed by connecting n nodes of a regular polygon to the corresponding n nodes of a star polygon, where k is the skip distance of the star polygon. In this paper, we take into consideration only a class of GP(n,k) where n and k are coprime.

The proposed algorithm in this paper can construct three independent spanning trees based on the individual decision of every node in a generalized Petersen network. That is, the algorithm makes the one-to-many routing parallelized.

Keyword: generalized Petersen network, one-to-many parallel routing, internally disjoint paths, independent spanning trees.

1. Introduction

In network classification, the *generalized Petersen networks* (GP networks for short) are a family of cubic networks [9]. A GP network, denoted by P(n,k), consists of an n-node *outer circulant* (n-cycle), an n-node *inner circulant* (with skip distance k), and n edges connecting corresponding nodes in the outer and inner circulants. As for the definition of circulant networks, see [2, 8]. Thus, P(n,k) has $2n$ nodes and $3n$ edges. This network family was introduced in 1950 by Coxeter [5] and named in 1969 by Watkins [23]. Note that we are concerned with the GP networks of co-prime n and k, where $n \geqslant 3$ and $1 \leqslant k \leqslant \lfloor (n-1)/2 \rfloor$. As a result, the inner circulant is connected.

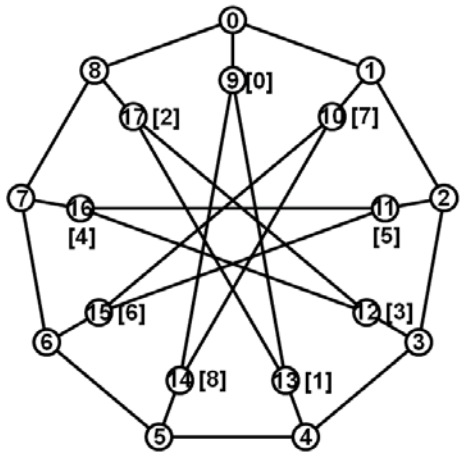
In P(n,k), the n nodes of the outer circulant, called *outer nodes*, are labeled from 0 to $n-1$, while the n nodes of the inner circulant, called *inner nodes*, are labeled from n to $2n-1$. For $0 \leqslant v \leqslant n-1$, nodes v and $v+n$ are connected by an edge. For example, P(9,4) and P(8,3) are shown in Figure 1(a) and 1(b), respectively.

The *skip number* of an inner node v, denoted by $s(v)$, is the sequence number of the consecutive skips, and is enclosed by square brackets in Figure 1. This additional label is used for the parallel routing of one node.

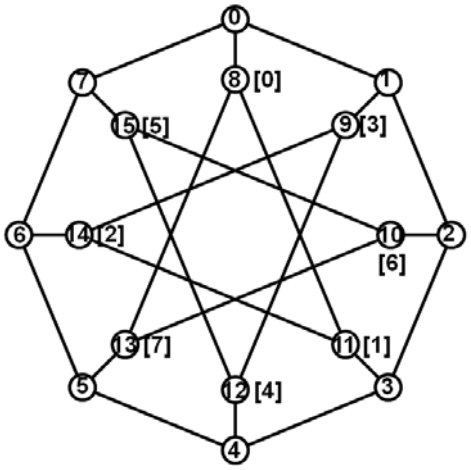
The GP networks have been widely studied, especially in the Hamiltonian cycle problem [1,3], the domination problem [4,19,24], and the isomorphism of GP [14,18]. In [18], the authors gave the inverse property about the parameter d in a GP. Since P(n,k) is isomorphic to P(n,$n-k$), we only take into consideration for $1 \leqslant k \leqslant \lfloor (n-1)/2 \rfloor$. Furthermore, they also proposed the exchange property of the inner and outer circulants when n and k are coprime. That is, P(n,k) is isomorphic to P(n,h) if and only if $kh \equiv \pm 1 \pmod{n}$. For example, P(9,4) is isomorphic to P(9,2) by exchanging the inner and outer circulants.

Two paths connecting two nodes in a network is said to be *internally disjoint* if they have no common node except two end nodes. The *one-to-many routing* is to construct internally disjoint paths in a network from a given node to each of the nodes in a given set that may contain all other nodes. One node can send copies of a message along internally disjoint paths to all other nodes in a network to achieve fault-tolerant broadcasting [10, 15]. One node can also partition the message into multiple parts and send them separately to the destination

*This research is supported by the National Science Council of Taiwan under the Grants MOST103–2410–H–606–004.



(a)



(b)

Figure 1: Two GP networks, (a) P(9,4) and (b) P(8,3).

nodes along internally disjoint paths to achieve secure message distribution [16].

Two spanning trees of a graph are *independent* if they are rooted at the same node r, and for every other node $v \neq r$, the two paths from r to v, one path in each tree, are internally disjoint. A set of spanning trees of a graph is said to be independent if they are pairwise independent. In 1989, Zehavi and Itai conjecture that, for any node r in a k-connected graph G, there exist k independent spanning trees of G rooted at r [28]. Although the conjecture has been proved for k-connected graphs with $k \leqslant 4$ ([10] for $k = 2$, [6,28] for $k = 3$, and [7] for $k = 4$), it is still open for $k > 4$. Notice that constructing multiple independent spanning trees (IST for short) rooted at one node guarantees the one-to-many routing. As a result, lots of research results are presented for solving the IST problem in special graph classes, especially in interconnection networks [11, 13, 17, 20–22, 25–27].

In this paper, we propose an algorithm for one-to-many routing at an arbitrary node of a GP network. Particularly, the proposed algorithm is based on the label of the node and can make the one-to-many routing parallelized. Although the IST problem of general 3-connected graphs was solved in linear time by Cheriyan and Maheshwari [6], the proposed algorithm still has its contribution on parallelized implementation.

The remaining part of this paper is organized as follows. Section 2 gives essential ideas of the algorithm. Section 3 presents the algorithm. Section 4 proves the correctness of the algorithm. The last section contains our concluding remarks.

2. Basic Ideas

To explicitly represent the adjacency of nodes in a GP network P(n,k), we say that node v takes a *skip* to reach one of three neighbors. If v is an outer node, the three skips are $(1\pm)$ and $(n+)$, standing for neighbors $v \pm 1 \pmod n$ and $v+n$, respectively. If v is an inner node, the three skips are $(k\pm)$ and $(n-)$, standing for neighbors $n + (v \pm k \pmod n)$ and $v - n$, respectively. The neighbors of v can be represented as a skip set $\{(1-), (1+), (n+)\}(v$ is an outer node) or $\{(k-), (k+), (n-)\}(v$ is an inner node). The skip set representation is helpful to express the routing algorithm.

Since GP networks are node-symmetric for nodes of the same circulant, without loss of generality, Nodes 0 and n should be considered separately as the root of IST. Due to the exchange property of the inner and outer circulants [18], to determine IST rooted at node n in P(n,k) is equivalent to determine IST rooted at node 0 in P(n,h), where $kh \equiv \pm 1 \pmod n$. For example, to construct IST rooted at node 9 in P(9,4), we firstly construct IST rooted at node 0 in P(9,2), and then exchange the node labels of inner and outer circulants. Therefore, we only considered the root 0 case.

Because of the internally disjoint requirement, it is obvious that the root has only one child in every tree of three IST. If j is the skip taken by the only child for reaching the root, the tree can be denoted by T_j. In Figure 2(a) and 2(b), for example, each tree of the IST of P(9,4) or P(8,3) is named after the unique skip used to reach node 0.

For the IST on a GP network, the skips for reaching the root in IST must be distinct. For a single non-root node, the skips to reach its parents in IST are also distinct. Since the parent-reaching skip set

and the root-reaching skip set have equal cardinality, the parallel routing algorithm is simply to determine a bijection between the two skip sets. For example, in Figure 2(a), node 14 reaches its parents by taking skips (4−), (4+) and (9−) in the IST which are labeled by skips (1−), (9−) and (1+), respectively.

The basic ideas of routing the paths form one node to the root are as follows.
(1) For an outer node, its path in $T_{(1-)}$ is routed counterclockwise, or taking skip (1-) to get its parent. Symmetrically, its path in $T_{(1+)}$ is routed clockwise, or taking skip (1+). Inevitably, its path in $T_{(n-)}$ has to take skip $(n+)$.
(2) For an inner node w, its path in $T_{(n-)}$ is routed according to its skip number $s(w)$. If $s(w) \leqslant \lfloor n/2 \rfloor$, w takes skip $(k-)$ to get its parent, else it takes skip $(k+)$. For $w \leqslant \lfloor n/2 \rfloor + n$, w takes skip $(n-)$ in $T_{(1-)}$, and takes the remained skip in $T_{(1+)}$. Symmetrically, for $w > \lfloor n/2 \rfloor + n$, w takes skip $(n-)$ in $T_{(1+)}$, and takes the remained skip in $T_{(1-)}$.

For an inner node w, we can compute $s(w)$ by $wh \pmod n$, where $kh \equiv 1 \pmod n$. To solve the linear Diophantine equation of two variables, $kh - ny = 1$, a well-known $O(\log n)$ algorithm was proposed by Knuth [12].

3. Parallel Construction of Independent Spanning Trees

The following algorithm is used to construct IST rooted at node 0 in P(n,k). It determines the parent-reaching skips in different trees for every non-root node v.

Algorithm PARENT_DETERMINE(v)
begin
Step 1. **For** $1 \leqslant v \leqslant n-1$ (outer nodes),
  take skip $(1-)$ to get parent in tree $T_{(1-)}$,
  take skip $(n+)$ to get parent in tree $T_{(n-)}$, and
  take skip $(1+)$ to get parent in tree $T_{(1+)}$.
Step 2. **For** $v = n$ (inner node, top),
  take skip $(k+)$ to get parent in tree $T_{(1-)}$,
  take skip $(n-)$ to get parent in tree $T_{(n-)}$, and
  take skip $(k-)$ to get parent in tree $T_{(1+)}$.
Step 3. **For** $n+1 \leqslant v \leqslant n + \lfloor n/2 \rfloor$
    (inner nodes, right side),
Substep 3.1 take skip $(n-)$ to get parent
    in tree $T_{(1-)}$,
Substep 3.2
If $s(v) \leqslant \lfloor n/2 \rfloor$ **then**
  take skip $(k-)$ to get parent in tree $T_{(n-)}$, and
  take skip $(k+)$ to get parent in tree $T_{(1+)}$.
else
  take skip $(k+)$ to get parent in tree $T_{(n-)}$, and
  take skip $(k-)$ to get parent in tree $T_{(1+)}$.
end if
Step 4. **For** $n + \lfloor n/2 \rfloor + 1 \leqslant v \leqslant 2n-1$
    (inner nodes, left side),
Substep 4.1 take skip $(n-)$ to get parent
    in tree $T_{(1+)}$,
Substep 4.2
If $s(v) \leqslant \lfloor n/2 \rfloor$ **then**
  take skip $(k-)$ to get parent in tree $T_{(n-)}$, and
  take skip $(k+)$ to get parent in tree $T_{(1-)}$.
else
  take skip $(k+)$ to get parent in tree $T_{(n-)}$, and
  take skip $(k-)$ to get parent in tree $T_{(1-)}$.
end if
end PARENT_DETERMINE

We use P(9,4) as an example. Table 1 shows the parent-reaching skips of nodes by using Algorithm PARENT_DETERMINE.

Table 1: The parent-reaching skips of node v in three IST on P(9,4)

v	$s(v)$	$T_{(1-)}$	$T_{(n-)}$	$T_{(1+)}$
1		$(1-)$	$(n+)$	$(1+)$
2		$(1-)$	$(n+)$	$(1+)$
3		$(1-)$	$(n+)$	$(1+)$
4		$(1-)$	$(n+)$	$(1+)$
5		$(1-)$	$(n+)$	$(1+)$
6		$(1-)$	$(n+)$	$(1+)$
7		$(1-)$	$(n+)$	$(1+)$
8		$(1-)$	$(n+)$	$(1+)$
9	0	$(k+)$	$(n-)$	$(k-)$
10	7	$(n-)$	$(k+)$	$(k-)$
11	5	$(n-)$	$(k+)$	$(k-)$
12	3	$(n-)$	$(k-)$	$(k+)$
13	1	$(n-)$	$(k-)$	$(k+)$
14	8	$(k-)$	$(k+)$	$(n-)$
15	6	$(k-)$	$(k+)$	$(n-)$
16	4	$(k+)$	$(k-)$	$(n-)$
17	2	$(k+)$	$(k-)$	$(n-)$

For constructing the IST shown in Figure 2(a), we have to transform the skips in Table 1 to the labels of parent nodes.

4. Correctness Proof

To show the correctness of Algorithm PARENT_DETERMINE, we have to prove that the output of the algorithm are spanning trees of the input GP network. Then, we should prove that for every non-root node v, three paths from v to 0 in different spanning trees must be internally disjoint.

When a skip j consecutively occurs t times ($t > 1$), we use j^t to denote the skip sequence. The

following lemma depicts that the output of the algorithm are spanning trees.

Lemma 4.1. *Algorithm* PARENT_DETERMINE *can generate three spanning trees rooted at node 0 in P(n,k).*

Proof. We consider the following four cases:
Case 1: v is an outer node.
In $T_{(1-)}$, v takes $(k+)$, $(1-)^v$ to reach the root.
In $T_{(1+)}$, v takes $(1+)^{n-v}$ to reach the root.
Case 1.1: $s(v+n) \leqslant \lfloor n/2 \rfloor$.
In $T_{(n-)}$, v takes $(n+)$, $(k-)^{s(v+n)}$, and $(n-)$ to reach the root.
Case 1.2: $s(v+n) > \lfloor n/2 \rfloor$.
In $T_{(n-)}$, v takes $(n+)$, $(k+)^{n-s(v+n)}$, and $(n-)$ to reach the root.
Case 2: $v = n$.
In $T_{(1-)}$, v takes $(k+)$, $(n-)$, and $(1-)^k$ to reach the root.
In $T_{(1+)}$, v takes $(k-)$, $(n-)$, and $(1+)^k$ to reach the root.
In $T_{(n-)}$, v takes $(n-)$ to reach the root.
Case 3: $n+1 \leqslant v \leqslant n+\lfloor n/2 \rfloor$.
In $T_{(1-)}$, v takes $(n-)$ and $(1-)^{v-n}$ to reach the root.
Case 3.1: $s(v) \leqslant \lfloor n/2 \rfloor$.
In $T_{(1+)}$, v takes $(k+)^c$, $(n-)$, and $(1+)^{2n-(v+k)}$ to reach the root, where $c = \lceil (3n/2 - v)/k \rceil$.
In $T_{(n-)}$, v takes $(k-)^{s(v)}$ and $(n-)$ to reach the root.
Case 3.2: $s(v) > \lfloor n/2 \rfloor$.
In $T_{(1+)}$, v takes $(k-)^c$, $(n-)$, and $(1+)^{n-v+k}$ to reach the root, where $c = \lceil (v-n)/k \rceil$.
In $T_{(n-)}$, v takes $(k+)^{s(v)}$ and $(n-)$ to reach the root.
Case 4: $n + \lfloor n/2 \rfloor + 1 \leqslant v \leqslant 2n-1$.
In $T_{(1+)}$, v takes $(n-)$ and $(1+)^{2n-v}$ to reach the root.
Case 4.1: $s(v) \leqslant \lfloor n/2 \rfloor$.
In $T_{(1-)}$, v takes $(k+)^c$, $(n-)$, and $(1+)^{2n-(v+k)}$ to reach the root, where $c = \lceil (2n - v)/k \rceil$.
In $T_{(n-)}$, v takes $(k-)^{s(v)}$ and $(n-)$ to reach the root.
Case 4.2: $s(v) > \lfloor n/2 \rfloor$.
In $T_{(1-)}$, v takes $(k-)^c$, $(n-)$, and $(1-)^{v-k-n}$ to reach the root, where $c = \lceil (v - 3n/2)/k \rceil$.
In $T_{(n-)}$, v takes $(k+)^{n-s(v)}$ and $(n-)$ to reach the root.

In any case, the skip sequence forms a unique path from v to 0. □

The following lemma shows the independency of the output spanning trees.

Lemma 4.2. *Three paths from a non-root node v to the root in the output spanning trees on P(n,k) are internally disjoint.*

Proof. Let u be a ancestor of node v in one of the spanning trees. We consider the following four cases:
Case 1: v is an outer node.
In $T_{(1-)}$, $u < v$.
In $T_{(1+)}$, $v < u < n$.
In $T_{(n-)}$, $n \leqslant u < 2n-1$.
Case 2: $v = n$.
In $T_{(1-)}$, $u = n+k$ or $u \leqslant k$.
In $T_{(1+)}$, $u = 2n-k$ or $n-k \leqslant u < n$.
In $T_{(n-)}$, $u = 0$.
Case 3: $n+1 \leqslant v \leqslant n+\lfloor n/2 \rfloor$.
In $T_{(1-)}$, u is an outer node, and $u \leqslant v-n$.
In $T_{(1+)}$, If u is an outer node, $u > v-n$. If u is an inner node and $s(v) \leqslant \lfloor n/2 \rfloor$, $s(u) > s(v)$, otherwise $s(u) < s(v)$.
In $T_{(n-)}$, $u = 0, n$, or u is an inner node. If u is an inner node and $s(v) \leqslant \lfloor n/2 \rfloor$, $s(u) < s(v)$, otherwise $s(u) > s(v)$.
Case 4: $n + \lfloor n/2 \rfloor + 1 \leqslant v \leqslant 2n-1$.
In $T_{(1+)}$, u is an outer node, and $v-n \leqslant u < n$.
In $T_{(1-)}$, If u is an outer node, $u < v-n$. If u is an inner node and $s(v) \leqslant \lfloor n/2 \rfloor$, $s(u) > s(v)$, otherwise $s(u) < s(v)$.
In $T_{(n-)}$, $u = 0, n$, or u is an inner node. If u is an inner node and $s(v) \leqslant \lfloor n/2 \rfloor$, $s(u) < s(v)$, otherwise $s(u) > s(v)$.

In any case, it turns out that every node can routing three internally disjoint paths in the network. □

According to Lemmas 4.1 and 4.2, we give the following theorem.

Theorem 1. *For a single node, Algorithm* PARENT_DETERMINE *can be used to determine its parents in three IST on a GP network in $O(1)$ time.*

5. Concluding Remarks

A one-to-many parallel routing algorithm is proposed to construct three IST on a GP network. Based on the algorithm, each non-root node can determine its parents in the IST in constant time. The height of the IST is $n-1$, where n is the number of outer or inner nodes. Our future work is to design a parallel routing algorithm that can reduce the hight of the IST. Another deserving work is to design an advanced one-to-many routing algorithm for P(n,k) by releasing the restriction of coprime n and k.

References

[1] B. Alspach, P. J. Robinson and M. Rosenfeld, A result on Hamiltonian cycles in gen-

eralized Petersen graphs, *Journal of Combinatorial Theory* B–31 (1981) 225–231.

[2] F. Boesch and A. Felzer, A general class of invulnerable graphs, *Networks* 2 (1972) 261–283.

[3] K. Bannai, Hamiltonian cycles in generalized petersen graphs, *Journal of Combinatorial Theory* B-24 (1978) 181–188.

[4] A. Behzad, M. Behzad and C. E. Praeger, On the domination number of the generalized Petersen graphs, *Discrete Mathematics* 308 (2008) 603–610.

[5] H. S. M. Coxeter, Self–dual configurations and regular graphs, *Bulletin of the American Mathematical Society* 56 (1950) 413–455.

[6] J. Cheriyan and S. N. Maheshwari, Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs, *Journal of Algorithms* 9 (1988) 507–537.

[7] S. Curran, O. Lee, and X. Yu, Finding four independent trees, *SIAM Journal on Computing*, 35 (2006) 1023–1058.

[8] B. Elspas and J. Turner, Graphs with circulant adjacency matrices, *Journal of Combinatorial Theory* 9 (1970) 297–307.

[9] R. M. Foster, Geometrical Circuits of Electrical Networks, *Transactions of the American Institute of Electrical Engineers* 51 (1932) 309-317.

[10] A. Itai and M. Rodeh, The multi-tree approach to reliability in distributed networks, *Information and Computation* 79 (1988) 43–59.

[11] Y. Iwasaki, Y. Kajiwara, K. Obokata, and Y. Igarashi, Independent spanning trees of chordal rings, *Information Processing Letters* 69 (1999) 155–160.

[12] D. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, second Edition, Addison-Wesley (1981) 316-336.

[13] J.-C. Lin, J.-S. Yang, C.-C. Hsu, J.-M. Chang, Independent spanning trees vs. edge-disjoint spanning trees in locally twisted cubes,*Information Processing Letters* 110 (2010) 414–419.

[14] B. Parhami, On isomorphisms and similarities between generalized Petersen networks and periodically regular chordal rings, *Information Processing Letters* 107 (2008) 246–251.

[15] P. Ramanathan, K. G. Shin, Reliable broadcast in hypercube multicomputers, *IEEE Transactions on Computers* 37 (1988) 1654–1657.

[16] M. O. Rabin, Efficient dispersal of information for secursity, load balencing, and fault tolerance, *Journal of the ACM* 36 (1989) 335–348.

[17] A. A. Rescigno, Node-disjoint spanning trees of the star network with applications to fault-tolerance and security, *Information Sciences* 137 (2001) 259–276.

[18] A. Steimle and W. Staton, The isomorphism classes of the generalized Petersen graphs, *Discrete Mathematics* 309 (2009) 231–237.

[19] C. Tong, X. Lin, Y. Yang and M. Luo, 2-rainbow domination of generalized Petersen graphs P(n,2), *Discrete Applied Mathematics* 157 (2009) 1932–1937.

[20] S.-M. Tang, Y.-L. Wang, and Y.-H. Leu, Optimal independent spanning trees on hypercubes, *Journal of Information Science and Engineering* 20 (2004) 605–617.

[21] S.-M. Tang, J.-S. Yang, Y.-L. Wang and J.-M. Chang, independent spanning trees on multidimensional torus networks, *IEEE Transactions on Computers* 59 (2010) 93–102.

[22] S.-M. Tang, J.-S. Yang, J.-M. Chang and Y.-L. Wang, A One-to-Many Parallel Routing Algorithm on a Generalized Recursive Circulant Graph, *Proc. of the 31st Workshop on Combinatorial Mathematics and Computation Theory* (2014) 29–36.

[23] M. E. Watkins, A theorem on tait colorings with an application to the generalized Petersen graphs, *Journal of Combinatorial Theory* 6 (1969) 152–164.

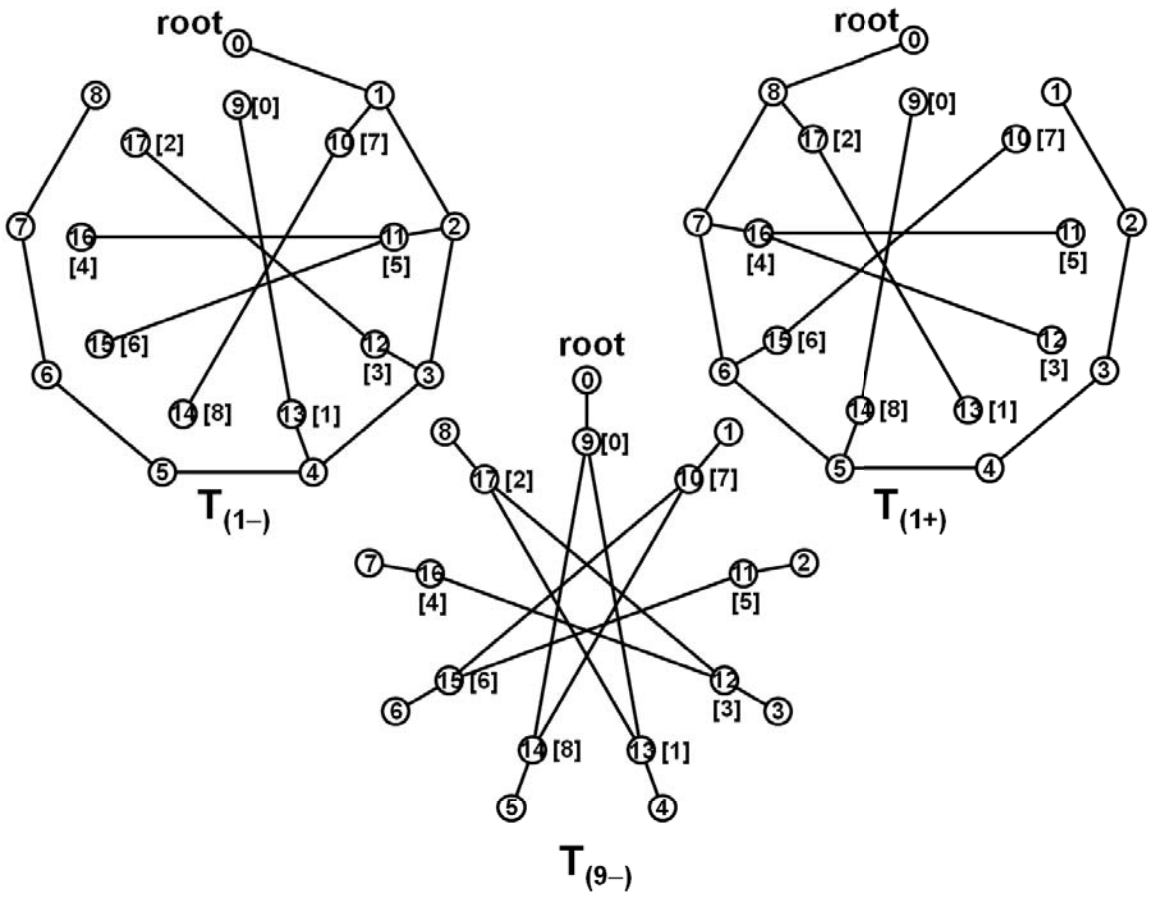
[24] G. Xu and L. Kang, On the power domination number of the generalized Petersen graphs, *Journal of Combinatorial Optimization* 22 (2011) 282–291.

[25] J.-S. Yang, S.-M. Tang, J.-M. Chang, and Y.-L. Wang, Parallel construction of independent spanning trees on hypercubes, *Parallel Computing* 33 (2007) 73–79.

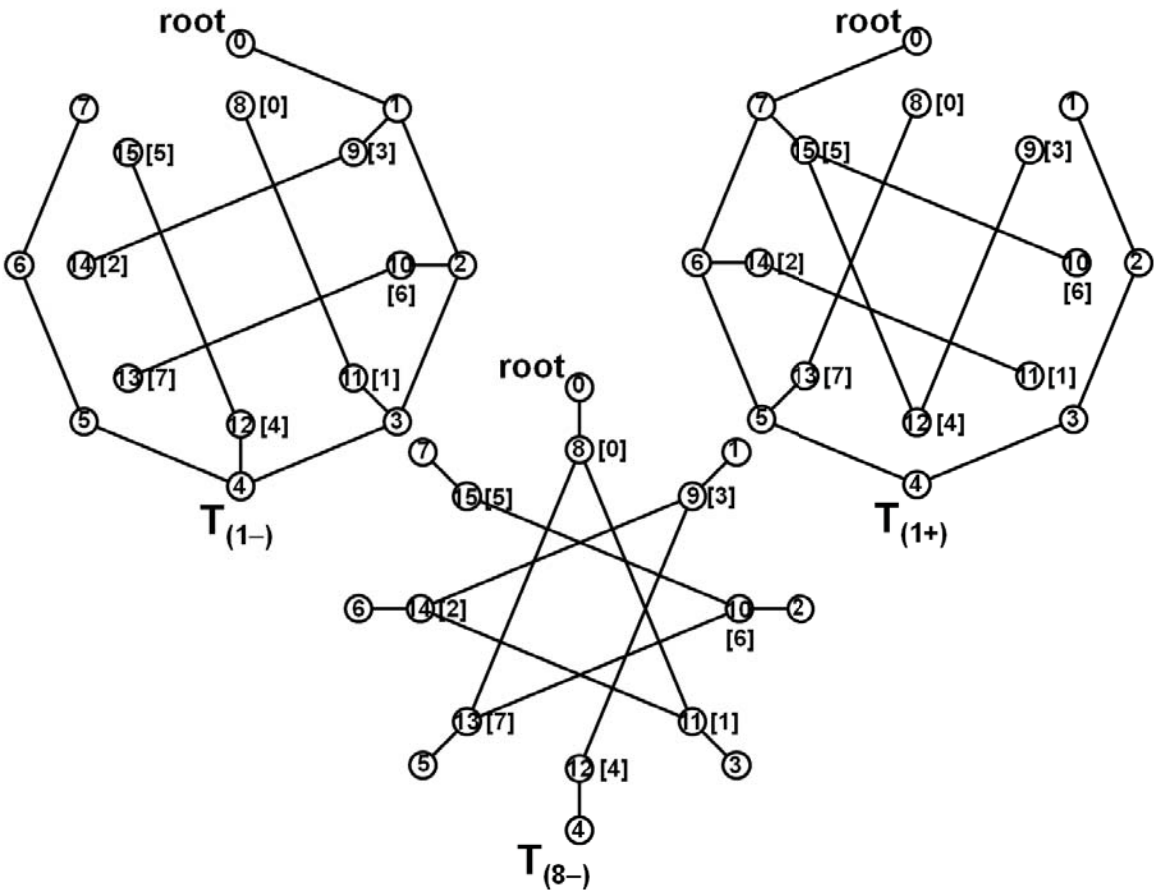
[26] J.-S. Yang and J.-M. Chang, Independent spanning trees on folded hyper-stars, *Networks* 56 (2010) 272–281.

[27] J.-S. Yang, J.-M. Chang, and H.-C. Chan, Broadcasting Secure Messages via Optimal Independent Spanning Trees in Folded Hypercubes, *Discrete Applied Mathematics* 159 (2011) 1254–1263.

[28] A. Zehavi and A. Itai, Three tree-paths, *Journal of Graph Theory* 13 (1989) 175–188.



(a)



(b)

Figure 2: Three IST rooted at node 0, (a) on P(9,4), and (b) on P(8,3).