UPMAIL Technical Report No. 97
February 21, 1995

Cycletrees: Flexible Interconnection Graphs for Parallel Computing

Margus Veanes & Jonas Barklund

Box 311, S-751 05 Uppsala, Sweden
Phone: +48−18−18 25 00
Fax: +46−18−51 19 25

Abstract

Natural cycletrees, formally defined in this report, is a subclass of Hamiltonian graphs with maximum degree 3 that contain a binary spanning tree. A natural cycletree used as an interconnection network thus supports directly broadcasting through the binary tree as well as nearest-neighbour communication through the cycle. Natural cycletrees have several other interesting properties, e.g., they are planar, easily extensible and can be contracted using the same methods as for binary trees. The two main results of the paper are: (i) Given an arbitrary basic binary spanning tree, there exists a natural cycletree with a minimal number of edges. (ii) Given a set of vertices, we present an algorithm for constructing a natural cycletree such that it has a minimal number of edges, its binary spanning tree has the minimal total path length and its structure satisfies a given abstract specification. For example, if we wish to construct a natural cycletree connecting k processing elements, we could invoke the algorithm with a set of k distinct vertices and a simple specification (provided as an example in the paper).

1 Introduction

A *cycletree* is a graph that has a basic binary spanning tree and a unique Hamiltonian cycle. The problems addressed in this paper can be stated succinctly:

1. Given a basic binary tree T, can we obtain a cycletree with as low degree as possible by adding a minimal number of edges to T?

2. Given a cycle C, can we construct a cycletree with minimal degree, having a binary spanning tree with minimal total path length, by adding a minimal number of edges to C?

Solutions to these problems will be immediately applicable to process interconnection graphs, because binary trees and Hamiltonian cycles support directly the following important communication patterns. (Let the nodes be numbered from 1 to N.)

1. Broadcasting or distributing data from node 1 to all the other nodes.

2. Collecting or combining data from all nodes to node 1.

3. Communication between nodes i and $i+1$ for all i, $1 \leq i < N$, and possibly between nodes 1 and N.

These communication patterns occur frequently in many parallel programming paradigms [10], but also in computations obtained by automatic parallelization of repetition usually in the form of sequential loops [25, 27, 31, 46, 51].

The theoretically ideal network has a complete interconnection graph, see Figure 1. Clearly, such a network is prohibitively expensive to realize for a large number of nodes. Some parameters that are commonly used to characterize networks are: number of edges, degree of the network (maximum degree of the nodes), diameter (the largest distance between any two nodes), average distance, symmetry, edge- and node connectivities, extensibility, and reliability [1, 2, 7, 38, 39, 50].

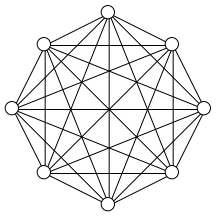

Figure 1: *The ideal network.*

In this paper, the following parameters are treated as the most important, in the order given.

1. *Minimal degree.* When using an interconnection network for communication between computation nodes, the number of physical communication ports of each node is usually limited. For example, INMOS Transputers have four communication ports, where typically one port is needed for connection to an external device, leaving only three ports for the interconnection network.[1] Three is the theoretical minimum of ports required and it is thus imperative to design the network with a degree of three.

2. *Minimal total path length.* The path lengths from the root to every node affects the minimum time for broadcasting information from the root or gathering information to the root, that is, two of the operations listed above. More generally, we wish to map divide-and-conquer computations onto the network, and it is desirable to minimize the depth of such computation trees.

3. *Minimal number of edges* (while the cycletree under consideration fulfills certain criteria). The number of edges is another measure (apart from the degree of the network) of the cost of realizing the network and should be minimized, while maintaining the above two properties.

4. *Flexible structure and size.* Many proposed interconnection graphs require, e.g., that the number of nodes is $2^k - 1$ for some $k > 0$, or fix the structure of the graph in such a way that it is difficult to describe the addition of a new node to the graph. This is undesirable for those applications where the structure should be able to grow in smaller steps, or where physical considerations make some structures expensive to realize (e.g., if the locations of the nodes are fixed, then in some interconnection graphs there could be comparatively long connections).

An inductively defined class of graphs that we call *natural cycletrees*, is introduced. Figure 2 illustrates a natural cycletree. It is shown that

1. natural cycletrees indeed belong to the class of cycletrees;

2. the degree of every nontrivial natural cycletree is three;

3. the class of *ringtrees* [52] is strictly contained by the class of natural cycletrees;

[1] In fact, our first application of natural cycletrees was as an interconnection network between Transputers on a Meiko parallel computer.

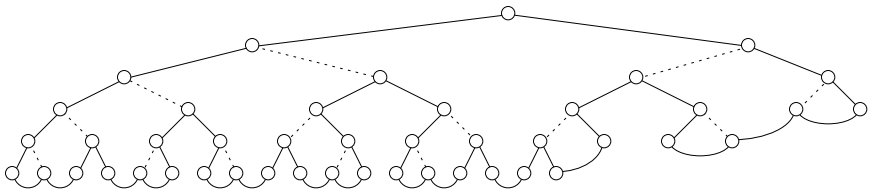

Figure 2: *A natural cycletree. The curved lines correspond to edges that are not part of the binary tree and the dashed lines correspond to edges that are not part of the cycle.*

4. given any basic binary tree T, every natural cycletree which is *minimal* for T has fewer edges than any other Hamiltonian graph having T as spanning tree;

5. given any odd cycle C, every *path-minimal* natural cycletree for C has C as its Hamiltonian cycle and contains a binary tree that has the minimal total pathlength; such a natural cycletree has fewer edges than any other graph with those properties and the exact number of edges can be computed by a formula.

Finally, a recursive algorithm is presented that, given a set of vertices, can be used to produce natural cycletrees with various properties. That algorithm is based on the inductive definition of natural chaintrees.

The problems set forth in the opening are thus answered positively by the class of natural cycletrees and the algorithm for constructing them.

We now outline the remainder of the paper. Section 2 is a brief account for the terminology. In Section 3 the concepts *cycletree* and *natural cycletree* are formally defined and some basic properties are stated and proved. Solutions to the minimality problems stated in the beginning are given in Section 4. In Section 5 the order induced by the Hamiltonian cycle in a natural cycletree is related to the recursive structure of natural cycletrees. That relationship is used in Section 6 for automatic construction of a natural cycletree, parameterized with a formal specification of its structure. Section 7 relates natural cycletrees to similar interconnection graphs. Finally, Section 8 concludes the presentation by summarizing the relationships between the classes of graphs introduced in the paper and their applications.

2 Preliminaries

The reader is assumed to be familiar with elementary concepts in graph theory [15]. Let $G = \langle V, E \rangle$ be a graph. Throughout the paper we will assume that G is simple and undirected. We will sometimes denote the edge-set of G by E_G and the vertex-set by V_G. By writing $E_{G_1 \oplus G_2}$ we mean $E_{G_1} \oplus E_{G_2}$ where '$\oplus$' is some set operation. An edge in E_G is written as a

pair (a, b) where a and b are vertices in V_G. (Note that $(a, b) = (b, a)$, as G is undirected.) We use the notation $G[X_1, X_2, \ldots, X_n]$, to indicate that each X_i, $i \in \{1, \ldots, n\}$, is a distinguished partial subgraph of G; if X_i is an isolated vertex, i.e., $X_i = \langle \{v\}, \emptyset \rangle$ for some vertex v, then we simply write v for X_i.

A *free tree* is a connected, acyclic, undirected graph. We say that a free tree T is a *binary tree* if T has exactly one vertex of degree 0 or 2, called the *root* of T, and all the other vertices have degree 1 or 3. A vertex of degree 1 is called an *external* vertex or a *leaf*. A vertex of degree 2 or 3 is called an *internal* vertex. If a binary tree is not an isolated vertex, i.e., its root has degree 2, then we call it a *basic* binary tree. We make explicit the root r of a binary tree T by $T[r]$.

Note that a binary tree is not ordered, and thus a basic binary tree is not the same as an *extended ordered binary tree* [24], although the concepts are related. (Knuth [24, pp. 309 and 315] uses the term b-trees for (unordered) binary trees as defined above.)

A *Hamiltonian circuit* or *path* in a graph G is a circuit or path which visits all the vertices of G exactly once. We call the partial subgraph of G traversed by a Hamiltonian circuit or path of G simply a *cycle* or *chain* in G, respectively. We identify the terminals r and s of a chain C by $C[r, s]$. We say that a chain $C[r, s]$ in G is *unique with respect to r and s* if there exists no chain $C'[r, s]$ in G such that $C' \neq C$.

3 Natural chaintrees and cycletrees

In the following we define formally two classes of interconnection graphs called *chaintrees* and *cycletrees*. We show how a certain class of chaintrees, which we call *natural chaintrees*, can be defined inductively. A *natural cycletree* is constructed from two natural chaintrees.

Definition 3.1 (*Chaintrees, Cycletrees*) Let $G[C[r, s], T[r]]$ be a graph where $V_C = V_T = V_G$ and $E_G = E_{C \cup T}$. G is a *chaintree* if T is a binary tree and C is a chain in G that is unique with respect to its endpoints. G is a *cycletree* if, in addition, T is a basic binary tree and $r = s$, i.e., C is the unique cycle in G.

We write $G[r, s]$ for a chaintree $G[C[r, s], T[r]]$. The uniqueness criterion in Definition 3.1 yields a unique ordering of the graph and is an important implementation issue which is discussed in Section 5. We say that a chaintree is *trivial* if it is an isolated vertex. We say that a cycletree is *trivial* if it consists of only three vertices, i.e., when it is a "triangle". We say that a graph $G[C, T]$, where $V_C = V_T = V_G$, is a *candidate* cycletree if T is a basic binary tree and C a (not necessarily unique) cycle of G.

Let $G[C,T]$ be a cycletree. We say that an edge e, $e \in E_G$, is a *tree edge* (with respect to T), if $e \in E_T$; a *cycle edge*, if $e \in E_C$; a *nontree edge*, if $e \in E_{C-T}$; a *noncycle edge*, otherwise.

We proceed to show how to construct natural chaintrees inductively and to show that natural chaintrees are indeed chaintrees. A natural cycletree is constructed from two natural chaintrees. First, let us illustrate the form of a natural cycletree by an example (see Figure 3). The edges represented by solid lines form a unique cycle. The edges represented by straight lines form a basic binary tree. The dashed lines represent edges that are not part of the cycle. The curved lines represent edges that are not part of the basic binary tree.

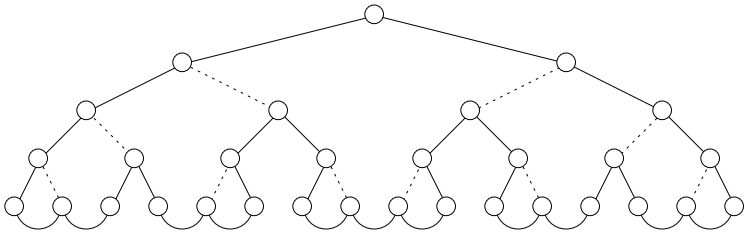

Figure 3: *A full natural cycletree of 31 vertices.*

Definition 3.2 (*Natural chaintrees*)

[**1**] Let r be a vertex. Then $\langle\{r\}, \emptyset\rangle[r,r]$ is a natural chaintree.

[**2(a)**] Let $H_1[r_1, s_1]$ be a natural chaintree, and let r and s be two distinct vertices not in V_{H_1}. Then

$$\langle V_{H_1} \cup \{r,s\},\ E_{H_1} \cup \{(r,r_1),(r,s),(s_1,s)\}\rangle[r,s]$$

is a natural chaintree (see Figure 4).

[**2(b)**] Let $H_i[r_i, s_i]$, $i \in \{1,2,3\}$, be disjoint natural chaintrees, and let r and s be two distinct vertices not in $V_{H_1 \cup H_2 \cup H_3}$. Then

$$\langle V_{H_1\cup H_2\cup H_3} \cup \{r,s\},\ E_{H_1\cup H_2\cup H_3} \cup \{(r,r_1),(r,s),(s,r_2),(s,r_3),(s_1,s_2)\}\rangle[r,s_3]$$

is a natural chaintree (see Figure 5).

[**3**] The only natural chaintrees are those given by clauses 1, 2(a) and 2(b).

Notice that this definition of natural chaintrees is similar to an inductive definition of binary trees, where the inductive case is asymmetric, explicitly separating the cases where one subtree is a leaf and where it is not. (Simply do not add the edge (s_1, s) in case 2(a) and the edge (s_1, s_2) in case 2(b).)

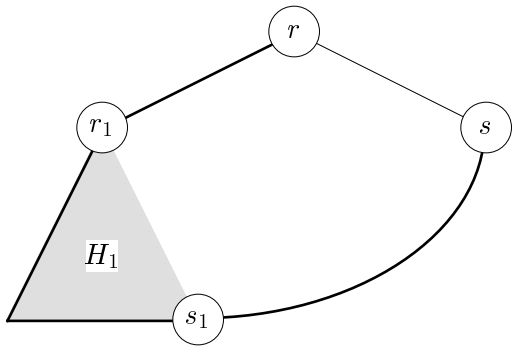

Figure 4: *Case 2(a) of Definition 3.2. The bold lines illustrate a chain.*

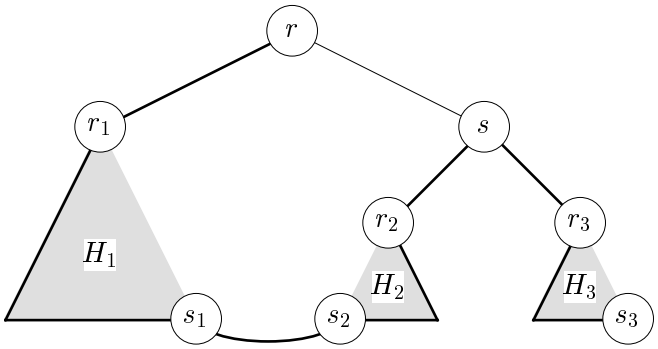

Figure 5: *Case 2(b) of Definition 3.2. The bold lines illustrate a chain.*

It is thus obvious that we can construct a natural chaintree from a binary tree by adding some edges. The choice of which vertex is r_1 in cases 2(a) and 2(b) is arbitrary, so the edge (r, r_1), which is the topmost edge of the chain, can go to either child of r.

We can now construct a natural cycletree from two natural chaintrees as follows.

Definition 3.3 (*Natural cycletrees***)** Let $H_1[r_1, s_1]$ and $H_2[r_2, s_2]$ be disjoint natural chaintrees, and r a vertex not in $V_{H_1 \cup H_2}$. Then

$$\langle V_{H_1 \cup H_2} \cup \{r\},\ E_{H_1 \cup H_2} \cup \{(r, r_1), (r, r_2), (s_1, s_2)\}\rangle$$

is a natural cycletree (see Figure 6).

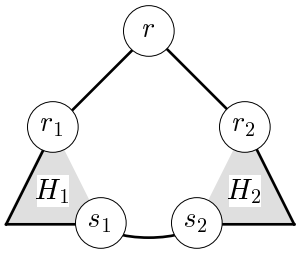

Figure 6: *Construction of a natural cycletree. The bold lines illustrate a cycle.*

An example of a possible natural cycletree is shown by Figure 7. We get

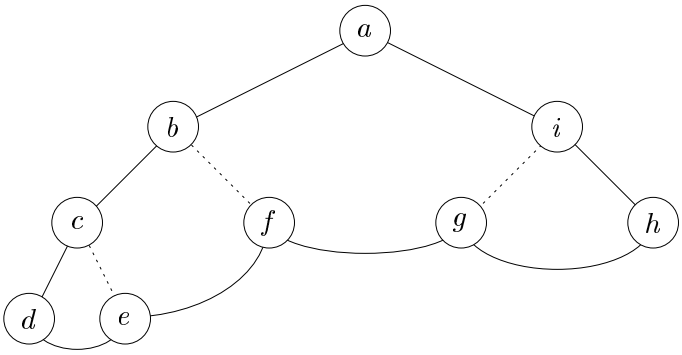

Figure 7: *A natural cycletree.*

the following immediate result.

Theorem 3.4 *The degree of every nontrivial natural cycletree is three.*

Proof. An easy inductive argument can verify that in a nontrivial natural chaintree $H[r, s]$, the degrees of r and s are 2 and the degrees of all other vertices are at most 3. The theorem then follows immediately from Definition 3.3. ∎

The observant reader may have noticed that we cannot always uniquely identify the basic binary tree in a natural cycletree, the smallest example to illustrate this is when the natural cycletree is a triangle. This is, however, only a minor detail and we deal with it in Section 5.

We will now prove that all natural cycletrees are cycletrees, using the following lemma.

Lemma 3.5 *A natural chaintree is a chaintree.*

Proof. Let $H[r, s']$ be a natural chaintree. We must prove that H has a unique chain $C[r, s']$ and that H contains a binary spanning tree. We prove the first statement by induction over natural chaintrees.

[**Base case**] When H is an isolated vertex ($r = s'$) then trivially C ($C = H$) exists and is unique.

[**Induction case**] We prove the lemma for cases 2(a) and 2(b) of Definition 3.2.

[**Case 2(a)**] Let $H_1[r_1, s_1]$ be a natural chaintree as in Definition 3.2.2(a). Assume as induction hypothesis that $C_1[r_1, s_1]$ is a chain in H_1 and that the lemma holds for H_1. Clearly, $C[r, s]$, where $s = s'$, must use the edges (r, r_1) and (s_1, s) (see Figures 4 and 5), and thus the chain through H_1 must have r_1 and s_1 as terminals. According to the induction hypothesis, C_1 is such a unique chain. Consequently,

$$C = \langle V_H,\ \{(r, r_1), (s_1, s)\} \cup E_{C_1} \rangle [r, s]$$

is a chain in H and there exists no other chain $C'[r, s]$, $C' \neq C$, in H.

[**Case 2(b)**] Let $H_i[r_i, s_i]$, $i \in \{1, 2, 3\}$, be natural chaintrees as in Definition 3.2.2(b). Assume as induction hypothesis that $C_i[r_i, s_i]$ is a chain in each H_i and that the lemma holds for every H_i. Obviously, $C[r, s_3]$, where $s_3 = s'$, must use the edges (r, r_1), (s_1, s_2), (r_2, s) and (s, r_3) (see Figures 4 and 5), and thus the chains through H_1, H_2 and H_3 must have r_i and s_i as terminals. According to the induction hypothesis, C_1, C_2 and C_3 are the corresponding unique chains. Consequently,

$$C = \langle V_H,\ \{(r, r_1), (s_1, s_2), (r_2, s), (s, r_3)\} \cup E_{C_1 \cup C_2 \cup C_3} \rangle [r, s_3]$$

is a chain in H and there exists no other chain $C'[r, s_3]$, $C' \neq C$, in H.

According to the induction principle we have proved the first statement for all natural chaintrees. The second statement follows from the discussion following Definition 3.2. ∎

We can now easily prove the following theorem by using Lemma 3.5.

Theorem 3.6 *A natural cycletree G is a cycletree $G[C,T]$.*

Proof. Let G, r, $H_1[r_1,s_1]$ and $H_2[r_2,s_2]$ be as in Definition 3.3. As r has degree 2, both of the edges (r,r_1) and (r,r_2) must be part of any cycle of G. Clearly the edge (s_1,s_2) must also be used. Thus, the chains through H_1 and H_2 must be $C_1[r_1,s_1]$ and $C_2[r_2,s_2]$, respectively. It follows from Lemma 3.5 that C_1 and C_2 exist and are unique. Consequently

$$C = \langle V_G,\ \{(r,r_1),(s_1,s_2),(r_2,r)\} \cup E_{C_1 \cup C_2} \rangle$$

is the unique cycle of G. According to Lemma 3.5, each of H_1 and H_2 contains a binary spanning tree. The tree T having r as root and these trees as immediate subtrees is a binary spanning tree for G. Hence G is a cycletree. ∎

All cycletrees are not natural cycletrees. A non-natural cycletree is illustrated by Figure 8. Another example of a non-natural cycletree is the threaded X-tree in Figure 16.

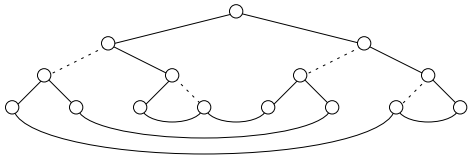

Figure 8: *A non-natural cycletree.*

4 Minimality and path-minimality

In this section, our goal is to construct a Hamiltonian graph with a binary spanning tree having "few" edges. More precisely, there are two complementary goals:

1. Given any binary tree T, add as few edges as possible to obtain a Hamiltonian supergraph. We say that such a cycletree is *minimal* (*for* T).

2. Given a cycle, add as few edges as possible to obtain a supergraph that contains a binary spanning tree with minimum total path length. We say that such a cycletree is *path-minimal*. We also give a formula for the exact number of edges in a path-minimal cycletree.

The *path length* of a path is the number of edges on that path. The *level* of a vertex s in a binary tree $T[r]$ is the path length of the shortest path from r to s, e.g., r has level 0. The *total path length* of T is the sum of the levels of all vertices of T.

Minimality

Given a basic binary tree T, we want to obtain a cycletree $G[C,T]$ by adding as few edges as possible to T.

Definition 4.1 (*Minimality*) A cycletree $G[C,T]$ is *minimal for* T, if $|E_G| \leq |E_{G'}|$ for any cycletree $G'[C',T]$.

Note that the the uniqueness of the cycle in a cycletree does not entail minimality; this is illustrated with the next example.

Example 4.2 Consider the natural cycletree G illustrated in Figure 9. The noncycle edges are (b,c) and (i,g). Clearly it has the same set of tree edges with respect to the binary tree $T[a]$ as the cycletree in Figure 7, but the total number of edges is less in Figure 9 than in Figure 7. ∎

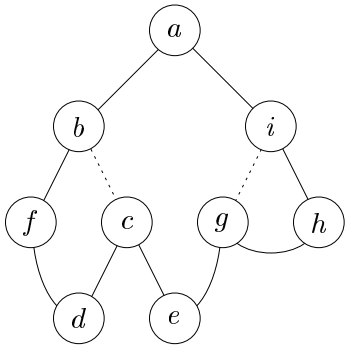

Figure 9: *A minimal (and path-minimal) cycletree.*

We shall now prove that there exists a natural cycletree $G[C,T]$ that is a minimal cycletree for the binary tree T (Theorem 4.4). As an immediate consequence we get that all *full* natural cycletrees are minimal, e.g., the natural cycletree in Figure 3 is minimal. This also confirms the result proved for ringtrees [52], which are in fact full natural cycletrees.

We shall first prove a lemma in which we will make use of the following definitions. Let T be a basic binary tree. Let $S[s]$ be a subtree of T and r the parent of s, if $S \neq T$; any vertex not in V_T, otherwise. Define

$$|S|_{\in} = \min\{\, |E_{S-C}| \mid C \text{ is a cycle} \wedge V_T = V_C \wedge (r,s) \in E_C \,\},$$

and

$$|S|_{\notin} = \min\{\, |E_{S-C}| \mid C \text{ is a cycle} \wedge V_T = V_C \wedge (r,s) \notin E_C \,\}.$$

The norms $|S|_{\in}$ and $|S|_{\notin}$ thus measure the minimal number of noncycle edges in S, the first when the edge to the parent is part of the cycle, the

other when it is not. Informally, this is a measure of to what extent the cycle edges can be used also in the tree. As a special case, $|T|_{\notin}$ is the theoretical lower bound for the number of edges of T that cannot participate in a cycle.

Lemma 4.3 *Let T be a basic binary tree and let $G[C,T]$ be a natural cycletree such that there exists no other natural cycletree for T having less edges. Then G is minimal for T and $|E_{T-C}| = |T|_{\notin}$.*

Proof. Let $S[s]$ be any subtree of T. Let r be the parent of s, if $S \neq T$; any vertex not in V_T, otherwise. We claim that

$$|E_{S-C}| = \begin{cases} |S|_{\in}, & \text{if } (s,r) \in E_C; \\ |S|_{\notin}, & \text{otherwise.} \end{cases} \tag{1}$$

An immediate consequence of Property 1 is that $|T|_{\notin} = E_{T-C}$, which proves the lemma. It remains to prove Property 1, together with the following property,

$$|S|_{\in} \leq |S|_{\notin} + 1 \leq |S|_{\in} + 1, \tag{2}$$

by induction over binary trees (subtrees of T).

[**Base case**] If s is a leaf of T then $|E_{S-C}| = |S|_{\in} = |S|_{\notin} = 0$, and $0 \leq 1 \leq 1$. It is obvious that Property 1 and Property 2 hold for S.

[**Induction case**] Let $S_1[s_1]$ and $S_2[s_2]$ be the immediate subtrees of S and assume that Property 1 and Property 2 hold for S_1 and S_2. There are two subcases: either (r,s) is in C or it is not.

> [$(\boldsymbol{r,s})$ **in** $\boldsymbol{C}$] According to the discussion after Definition 3.2, we can construct natural cycletrees such that either of (s,s_1) and (s,s_2) is in the cycle (clearly, one of them must be). Without loss of generality, assume that S_1 is the immediate subtree of S such that
>
> $$|S_1|_{\in} + |S_2|_{\notin} \leq |S_1|_{\notin} + |S_2|_{\in} \tag{3}$$
>
> holds. We have assumed that Property 2 holds for S_1 and S_2, simple arithmetic gives that S_1 can always be chosen in this way. The minimal number of noncycle edges in S is thus
>
> $$|S|_{\in} = |S_1|_{\in} + |S_2|_{\notin} + 1, \tag{4}$$
>
> where the addition of 1 counts the edge (s,s_2). Furthermore, since Property 1 holds for S_1 and S_2, i.e., $|E_{S_1-C}| = |S_1|_{\in}$ and $|E_{S_2-C}| = |S_2|_{\notin}$, and since $|E_{S-C}| = |E_{S_1-C}| + |E_{S_2-C}| + 1$ (due to the structure of natural cycletrees), it follows that $|E_S| = |S|_{\in}$.

[(r, s) **not in** C] In this case both (s, s_1) and (s, s_2) are in C. From Property 2 it follows that

$$|S|_{\notin} = |S_1|_{\in} + |S_2|_{\in}. \tag{5}$$

Furthermore, since Property 1 holds for S_1 and S_2, i.e., $|E_{S_1-C}| = |S_1|_{\in}$ and $|E_{S_2-C}| = |S_2|_{\in}$, and since $|E_{S-C}| = |E_{S_1-C}| + |E_{S_2-C}|$ (again, obvious from the structure of natural cycletrees), it follows that $|E_S| = |S|_{\notin}$.

Property 1 now follows immediately from the two cases above. Also, Property 2 follows from (4) and (5), because Property 2 holds for S_1 and S_2.

According to the induction principle, we have proved Property 1 and Property 2 for all subtrees of T and in particular for T itself. ∎

We can now easily prove the following theorem, which asserts that there is no graph that contains a given binary spanning tree T and a unique cycle, and has fewer edges than every natural cycletree spanned by T.

Theorem 4.4 *Let $G'[C', T]$ be any candidate cycletree and let $G[C, T]$ be a natural cycletree that is minimal for T. Then $|E_G| \leq |E_{G'}|$.*

Proof. We know, by definition, that $|T|_{\notin} \leq |E_{T-C'}|$. Using Lemma 4.3, $|E_{T-C}| = |T|_{\notin}$, we get that $|E_{T-C}| \leq |E_{T-C'}|$. We know that $|E_G| = |E_{T-C}| + |E_C|$ because there are no superfluous edges in G, and we know trivially that $|E_{G'}| \geq |E_{T-C'}| + |E_{C'}|$. Hence $|E_G| \leq |E_{G'}|$, since $|E_C| = |E_{C'}|$. ∎

Path-minimal cycletrees

Let us assume that one wants to construct a cycletree $G[C, T]$ given only a set of vertices. Clearly, constructing T to have the minimal total path length has several advantages. For example if T is used as the interconnection graph of a process network then the total communication path length of T is minimized. At this point it is worth noting that we are not giving top priority to minimizing the *average* distance between an arbitrary pair of vertices of G. The reason for introducing natural cycletrees in the first place is to provide efficient communication for parallel computations that generally need T for "global" communication and C for "local" communication.

The *depth* of a binary tree T is the maximum level of T. We say that a binary tree T of depth d is *tree-complete* if all the leaves of T are at levels d and $d-1$. We say that a cycletree $G[C, T]$ is tree-complete (with respect to T) if T is tree-complete.

We know that the total path length of a basic binary tree T is minimal if T is tree-complete. (The relation to complete extended ordered binary trees

is obvious, see Knuth [24, pp. 399–400].) When constructing a cycletree $G[C,T]$ we also want to keep the number of additional edges at minimum, which suggests the following definition.

Definition 4.5 (*Path-minimality*) Let $G[C,T]$ be a tree-complete cycletree. We say that G is *path-minimal* if $|E_G| \leq |E_{G'}|$ for every tree-complete cycletree $G'[C,T']$.

Clearly, if a cycletree $G[C,T]$ is path-minimal then it is both minimal for T and tree-complete. (The converse is in general not true, i.e., a cycletree that is both tree-complete and minimal need not be path-minimal.)

Theorem 4.6 *Let $G[C,T]$ be a path-minimal cycletree, then*

$$|E_G| = \begin{cases} (3n-1)/2 - \lfloor (2^k+1)/3 \rfloor, & \text{if } n \geq 4\lfloor (2^k+1)/3 \rfloor - 1; \\ n - 1 + \lfloor (2^k+1)/3 \rfloor, & \text{otherwise.} \end{cases} \quad (6)$$

where $n = |V_G|$ and $k = \lfloor \log_2(n+1) \rfloor$.

Proof. Let $n = |V_G|$. As T is a tree-complete basic binary tree, T is full up to level $k-1$, i.e., T has 2^l vertices at level l, $0 \leq l < k$, and $R = (n+1-2^k)/2$ number of internal vertices at level $k-1$. Let us by I_m denote the number of noncycle edges at a full level[2] m of a natural cycletree, and by J_m that of a natural chaintree. Then

$$I_m = 2J_{m-1}, \quad m > 0,$$

and, by using Definition 3.2, we obtain the following linear recurrence equation for J_m:

$$\begin{aligned} J_0 &= 0, \\ J_1 &= 1, \\ J_m &= J_{m-1} + 2J_{m-2}, \quad m > 1. \end{aligned}$$

Using standard techniques we get the following solution:

$$J_m = \frac{2^m - (-1)^m}{3} = \left\lfloor \frac{2^m+1}{3} \right\rfloor.$$

Let R' be the number of noncycle edges at level k. We know that R' must be minimal, since G is path-minimal and the number of noncycle edges at levels m, $1 \leq m < k$, are fixed by I_m. There are R internal vertices at level $k-1$ and for each of these vertices exactly one edge is a noncycle edge

[2]The level of an edge (r,s), where r is the parent of s, is the level of s.

(assume that $k > 1$). Thus $R - R'$ of these edges are at level $k-1$. We get the following formula for R', since $R - R'$ can be at most I_{k-1},

$$R' = \begin{cases} R - I_{k-1}, & \text{if } R - I_{k-1} \geq 0; \\ 0, & \text{otherwise.} \end{cases}$$

We know that $R = (n+1)/2 - 2^{k-1}$ and $I_{k-1} = 2J_{k-2}$, thus

$$\begin{aligned} R - I_{k-1} &= \frac{n+1}{2} - 2^{k-1} - 2\frac{2^{k-2} - (-1)^{k-2}}{3} \\ &= \frac{n+1}{2} - 2\frac{2^k - (-1)^k}{3} \\ &= \frac{n+1}{2} - I_{k+1}. \end{aligned}$$

Thus the following is an equivalent formula for R':

$$R' = \begin{cases} (n+1)/2 - I_{k+1}, & \text{if } n \geq 2I_{k+1} - 1; \\ 0, & \text{otherwise.} \end{cases}$$

Now, the total number of edges of G is the number of cycle edges, which is n, plus the number of noncycle edges at all levels. Thus

$$\begin{aligned} |E_G| &= n + \sum_{m=1}^{k-1} I_m + R' \\ &= n + 2\sum_{m=0}^{k-2} J_m + R' \\ &= n + 2\sum_{m=0}^{k-2} \frac{2^m - (-1)^m}{3} + R' \\ &= n + \frac{2}{3}\left(\sum_{m=0}^{k-2} 2^m - \sum_{m=0}^{k-2} (-1)^m\right) + R' \\ &= n + \frac{2}{3}\left(2^{k-1} - 1 - \frac{(-1)^{k-1} - 1}{-2}\right) + R' \\ &= n + \frac{2^k - (-1)^k}{3} - 1 + R' \\ &= n + J_k - 1 + R' \\ &= \begin{cases} n + J_k - 1 + (n+1)/2 - 2J_k, & \text{if } n \geq 4J_k - 1; \\ n + J_k - 1, & \text{otherwise.} \end{cases} \\ &= \begin{cases} (3n-1)/2 - J_k, & \text{if } n \geq 4J_k - 1; \\ n - 1 + J_k, & \text{otherwise.} \end{cases} \\ &= \begin{cases} (3n-1)/2 - \lfloor (2^k+1)/3 \rfloor, & \text{if } n \geq 4\lfloor (2^k+1)/3 \rfloor - 1; \\ n - 1 + \lfloor (2^k+1)/3 \rfloor, & \text{otherwise.} \end{cases} \end{aligned}$$

Let $G[C,T]$, $n = |V_G|$, be a cycletree where T is a full binary tree, i.e., $n+1$ is a power of 2. Then Formula 6 reduces to

$$n - 1 + \lfloor (n+2)/3 \rfloor, \tag{7}$$

which was also shown by Xie and Ge [52] to be the number of edges in a ringtree. One can easily verify that the natural cycletree in Figure 9 is path-minimal, by using Formula 6. We also get the following corollary from Theorem 4.6.

Corollary 4.7 *Let $G[C,T]$ be any candidate cycletree where T is tree-complete. Then G has at least as many edges as given by Theorem 4.6.*

Proof. Immediate by using Theorem 4.4, because any path-minimal cycletree is, by definition, also minimal. ∎

As an example of the use of the above formulas, assume that one has a full cycletree of n vertices and wants to double its size to $2n+1$. How many additional edges are required? By using Formula 7, we obtain that $n + 2\lfloor (n+3)/6 \rfloor$ additional edges are required. Thus, doubling the number of vertices roughly doubles the number of edges in a cycletree.

5 Ordered natural cycletrees

Up to this point natural cycletrees have been treated as undirected simple graphs and are therefore not ordered. In order to use a graph for organizing nodes, e.g., in an interconnection graph (as we will propose in Section 6), we must be able to identify each subtree of a binary tree by means of direction, using for example the terms left and right subtree.

Let $T[r]$ be a binary tree. We say that T is *ordered* if it is associated with a mapping $\eta : V_T \setminus \{r\} \to \{\text{left}, \text{right}\}$ such that for each internal vertex v of T with children v_1 and v_2, $\{\eta(v_1), \eta(v_2)\} = \{\text{left}, \text{right}\}$. We call η an *order mapping* for T. We write T^η to emphasize the ordering. If $T^\eta[r]$ is a basic binary tree with immediate subtrees $T_1[r_1]$ and $T_2[r_2]$ such that $\eta(r_1) = \text{left}$ and $\eta(r_2) = \text{right}$ then we call T_1 and T_2 the left and right subtrees of T, respectively. Clearly T can be ordered in many different ways, e.g., any full basic binary tree of depth k can be ordered in 2^{2^k-1} different ways.

Given an order mapping η define $\overline{\eta}$ so that $\overline{\eta}(v) = \overline{\eta(v)}$, where $\overline{\text{left}} = \text{right}$ and $\overline{\text{right}} = \text{left}$. We can now define (by induction over the structure of natural chaintrees) what we mean by the *order mapping $\eta_{r,u}$ induced by* a natural chaintree $H[r,u]$. As the base case let $\eta_{r,r} = \emptyset$. Let s and $H_i[r_i, s_i]$, for $i \in \{1,2,3\}$, be as in Definition 3.2 and assume that we have order

mappings $\eta_i = \eta_{r_i,s_i}$ induced by H_i. If case $2(a)$ is used to form $H\ (u = s)$ then

$$\eta_{r,s} \quad = \quad \eta_1 \cup \{\, r_1 \mapsto \text{left}, s \mapsto \text{right}\,\}.$$

If case $2(b)$ is used to form $H\ (u = s_3)$ then

$$\eta_{r,s_3} \quad = \quad \eta_1 \cup \overline{\eta_2} \cup \eta_3 \cup \{r_1 \mapsto \text{left}, s \mapsto \text{right}, r_2 \mapsto \text{left}, r_3 \mapsto \text{right}\}.$$

Let G be the natural cycletree obtained from r, $H_1[r_1, s_1]$ and $H_2[r_2, s_2]$ as in Definition 3.3, and let $\eta_i = \eta_{r_i,s_i}$ be the order mappings induced by the respective H_i. The order mapping η_{r_1,r_2} for G is defined as follows.

$$\eta_{r_1,r_2} \quad = \quad \eta_1 \cup \overline{\eta_2} \cup \{\, r_1 \mapsto \text{left}, r_2 \mapsto \text{right}\,\}.$$

Observe that there exist exactly two such mappings η_{r_1,r_2} and η_{r_2,r_1}, and that $\eta_{r_2,r_1} = \overline{\eta_{r_1,r_2}}$. When we say that a natural cycletree G is ordered, we assume a fixed order mapping η and we write G^η if we wish to make η explicit. It should be clear that the above notions are well-defined, i.e., that the definitions indeed produce unique order mappings.

We will now present an alternative method of constructing natural cycletrees where the starting point is an ordered basic binary tree. The method is based on the following notion.

Definition 5.1 (*cycle order traversal*) Let T be an ordered basic binary tree. Traverse T in root-mode.

Root-mode Visit the root and mark it with '$\leftarrow$'.

Traverse the left subtree in pre-mode.

Traverse the right subtree in post-mode.

Visit the root again.

Pre-mode Visit the root and mark it with '$\downarrow$'.

Traverse the left subtree (if any) in pre-mode.

Traverse the right subtree (if any) in in-mode.

In-mode Traverse the left subtree (if any) in post-mode.

Visit the root and mark it with '$\rightarrow$'.

Traverse the right subtree (if any) in pre-mode.

Post-mode Traverse the left subtree (if any) in in-mode.

Traverse the right subtree (if any) in post-mode.

Visit the root and mark it with '↑'.

Let $T[r]$ be an ordered basic binary tree. Let $v \in V_T$. We use $v+$ to denote that vertex of T which is the immediate successor of v in a cycle order traversal of T.

Theorem 5.2 *For all η, $G^\eta[C,T]$ is an ordered natural cycletree if and only if T^η is an ordered basic binary tree and $E_C = \{ (v, v+) : v \in V_T \}$.*

Proof.

[$\Rightarrow$] Let $H_1[C_1[r_1,s_1],T_1[r_1]]$ and $H_2[C_2[r_2,s_2],T_2[r_2]]$ be as in Definition 3.3 and consider $G^\eta[C,T[r]]$. Assume that $\eta = \eta_{r_1,r_2}$. (Note that η_{r_1,r_2} tells us that T_1 is the left subtree of T.) We know that the paths $P_1 = (r_1,\ldots,s_1)$ and $P_2 = (s_2,\ldots,r_2)$ corresponding to C_1 and C_2, respectively, are unique with respect to their endpoints. By the below lemma P_1 is the pre-mode traversal of T_1 and P_2 is the post-mode traversal of T_2. Clearly (r,P_1,P_2,r) corresponds to C and is the cycle order traversal of T^η.

[**Lemma**] Let H, r, s, $H_i[C_i[r_i,s_i],T_i[r_i]]$ for $i \in 1,2,3$ be as in Definition 3.2. We prove by induction over the structure of H that $P = (r,\ldots,s')$ is the pre-mode traversal of $T^{\eta_{r,s'}}$ and $\overline{P} = (s',\ldots,r)$ is the post-mode traversal of $T^{\overline{\eta_{r,s'}}}$. ($s'$ is one of $\{r,s,s_3\}$ depending on the case of the definition.) Let $\eta = \eta_{r,s'}$. If H is the isolated vertex, $s' = r$, then this holds trivially.

Assume that P_i is the pre-mode traversal of $H_i[r_i,s_i]$ for $i \in \{1,2,3\}$, and $\overline{P_i}$ is the corresponding post-mode traversal.

If H is constructed using H_1 only, $s' = s$, (by using 2(a)) then clearly $P = (r,P_1,s)$ correponds to $C[r,s]$ and is the pre-mode traversal of T^η. (Note that the right subtree of T^η is just s, so the in-mode traversal of the right subtree of T^η is simply (s').) Also $\overline{P} = (s,\overline{P_1},r)$ is clearly the post-mode traversal of $T^{\overline{\eta}}$.

If H is constructed according to 2(b) then $P = (r,P_1,\overline{P_2},s,P_3)$ corresponds to the chain in H and is the pre-mode traversal of T^η. Note that $(\overline{P_2},s,P_3)$ is indeed the in-mode traversal of the right subtree of T^η. We get also that $\overline{P} = (\overline{P_3},s,P_2,\overline{P_1},r)$ is the post-mode travesal of $T^{\overline{\eta}}$. Note that $(\overline{P_3},s,P_2)$ is indeed the in-mode traversal of the left subtree of $T^{\overline{\eta}}$.

[$\Leftarrow$] Let $T^\eta[r]$ be an ordered basic binary tree. We must prove that Then $G^\eta[C,T[r]]$ is a natural cycletree, where $E_C = \{ (v, v+) : v \in V_T \}$.

The statement follows easily from Definition 5.1 and Definition 3.3 once we haved proved the following lemma.

[**Lemma**] Let $T\eta[r]$ be an ordered binary tree and $P = (r, \ldots, s)$ its pre-mode traversal. Then by adding the edges of P to T we obtain a chaintree $H[r, s]$ and $\eta = \eta_{r,s}$.

The proof is by straightforward induction over T. The base case, i.e., when T is just one vertex is trivial. The case when the right subtree is a leaf corresponds to case 2(a) of Definition 3.3 and the case when the right subtree is not a leaf correponds to case 2(b). There is an obvious correspondence beteen Definitions 3.2 and 3.3 and Definition 5.1. ∎

The theorem provides us with an alternative definition of cycletrees, which we will use in the subsequent sections.

We introduce some more useful terminology. Let v be a vertex of a natural cycletree $G^\eta[C, T[r]]$. We call v a *pre*-, an *in*- or a *post*-vertex if v has mark '$\downarrow$', '$\rightarrow$' or '$\uparrow$', respectively (in the cycle order traversal of T^η). We say that the ith vertex in the cycle order traversal of T has address i, r has address 1. We often identify a vertex with its address and say "vertex a" instead of "vertex v such that v has address a". The addresses and marks are illustarted in Figure 10.

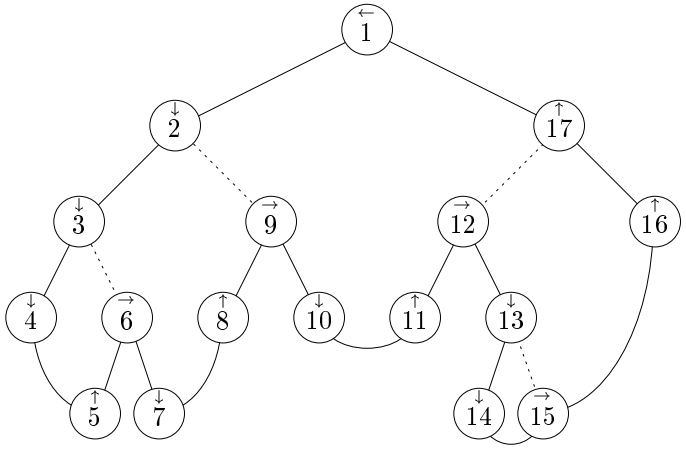

Figure 10: *An ordered natural cycletree.*

Related concepts. The choice of names containing 'pre', 'post' and 'in' has a historical background in ordered binary trees. In fact, there is a connection between ordered natural cycletrees and *threaded* binary trees [24]. Some types of threaded trees, to be used as interconnection graphs, have been studied by Despain and Patterson [11] and are illustrated in Figure 16.

6 Natural cycletrees as interconnection graphs

We have so far studied some basic properties of natural cycletrees. Let us now turn to the more practical issue of using natural cycletrees as in-

terconnection graphs in the MP-RAM model, see for example Almasi and Gottlieb [2] for a formal treatment of the various computational models. We will present an algorithm that recursively configures a natural cycletree from a given collection of nodes.

We will think of our RAMs simply as *nodes*. Each node has a unique address between 1 and N, where N is the total number of nodes. When saying 'node a' we will mean 'the node with address a'.

The *interconnection graph* has a vertex corresponding to each node in the ensemble. An edge in the graph indicates that the nodes at its endpoints can communicate via a bidirectional channel or *link*. Nodes so connected are called *neighbours*. We will be referring to the vertices and the edges of an interconnection graph as nodes and links, when we have that interpretation in mind.

Let us assume having only a collection of N nodes that we want to configure as a natural cycletree. As there exists a vast number of possible natural cycletrees having N vertices (N must be odd and ≥ 3), we assume that certain constraints are given and must be satisfied, e.g., tree-completeness. Let n be the number of internal vertices of a basic binary tree having N vertices. The relationship between n and N is simply $n = (N-1)/2$. Let $\mathcal{M}$ be the set of marks and $\mathcal{N}$ the set of natural numbers. In the following we will assume that the constraints are given in the form of definitions for partial functions split_m, for each mark $m \in \mathcal{M}$, with the type

$$\mathcal{N} \times \mathcal{N} \to 2^{\mathcal{N}\times\mathcal{N}} \cup \{\, \{\langle\rangle\}\,\}$$

(where $\langle\rangle$ denotes an impossible "split"), such that for all $m \in \mathcal{M}$ and $k, n \in \mathcal{N}$,

$$\text{split}_m(0,k) = \{\,\langle\rangle\,\} \quad \text{and} \quad \langle n_1, n_2\rangle \in \text{split}_m(n,k) \Rightarrow n_1 + n_2 + 1 = n.$$

Let $G[C,T]$ be a natural cycletree and $S[s]$ a subtree of T. Let n be the number of internal vertices of S, let m be the mark and k the level of s. Then $\text{split}_m(n,k)$ is the set of allowed partitions, $\langle n_1, n_2\rangle$, of n into the number of internal vertices, n_1, in the left subtree of S, and the number of internal vertices, n_2, in the right subtree of S.

Example 6.1 If we want to divide the vertices as evenly as possible among the subtrees then we can define 'split' as follows. For $m \in \mathcal{M}$ and $k, n \in \mathcal{N}$,

$$\text{split}_m(n,k) = \begin{cases} \{\,\langle n_1, n-1-n_1\rangle \mid \\ \qquad n_1 = \lfloor (n-1)/2 \rfloor \vee n_1 = \lceil (n-1)/2 \rceil\,\}, & \text{if } n > 0; \\ \{\,\langle\rangle\,\}, & \text{otherwise.} \end{cases}$$

∎

Example 6.2 The formula

$$\text{split}_m(n,k) = \begin{cases} \{\,\langle n-1, 0\rangle\,\}, & \text{if } n > 0; \\ \{\,\langle\rangle\,\}, & \text{otherwise,} \end{cases}$$

says that each internal vertex must have a leaf as its right child. The natural cycletree in Figure 11 satisfies this constraint and is, for example, the

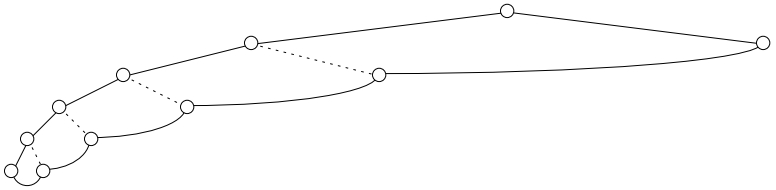

Figure 11: *An extreme example of a natural cycletree.*

interconnection graph of the cyclic-order odd-even transposition bilinear sorter [26]. ∎

We needed neither the mark nor the level in the simple examples 6.1 and 6.2. Using the mark and the level is more powerful and one can define a 'split' that yields, for example, a path-minimal natural cycletree. This is illustrated with the next example.

Example 6.3 We wish to obtain a natural cycletree $G[C,T]$ that is path-minimal, i.e., it is tree-complete and has a minimal number of noncycle edges (i.e., a minimal number of in-vertices).

Let d be the highest level of T such that the number of vertices at level d is 2^d, i.e., the highest "full" level of T. Let n_T be the number of internal vertices in T. T is tree-complete if, and only if,

$$d = \lfloor \log_2(n_T + 1) \rfloor.$$

After two definitions, we shall state a criterion (8) for path-minimality, which must hold for each subtree of T.

Clearly, T is tree-complete if, and only if, each subtree of T is tree-complete. Consider any subtree $S[s]$ of T, let n be the number of internal vertices in S, let m be the mark and k the level of s. Define $g : \mathcal{N} \times \mathcal{N} \to \mathcal{N}$ as follows:

$$g(l,x) = x - (2^l - 1).$$

S is tree-complete if and only if S has $g(d-k,n)$ *internal* vertices at level $d-k$ of S (i.e., at level k of T in S). For each mark m, define $I : \mathcal{N} \to \mathcal{N}$

as follows:

$$\begin{aligned} I_{\downarrow}(l) = I_{\uparrow}(l) &= \frac{2^l - (-1)^l}{3} = \left\lfloor \frac{2^l + 1}{3} \right\rfloor, \\ I_{\rightarrow}(l) &= \begin{cases} 1, & \text{if } l = 0; \\ 2I_{\downarrow}(l-1), & \text{otherwise,} \end{cases} \\ I_{\leftarrow}(l) &= \begin{cases} 0, & \text{if } l = 0; \\ I_{\rightarrow}(l), & \text{otherwise.} \end{cases} \end{aligned}$$

$I_m(l)$ is the number of *in-vertices* at any "full" level l of $S[s]$, where m is the mark of s [47, pp. 47, where J_l denotes $I_{\downarrow}(l)$ and $I_{\uparrow}(l)$].

Let $h(S)$ be the number of *internal in-vertices* at level $d-k$ of S. Now, path-minimality of G is equivalent with $h(S)$ being as large as possible, i.e.,

$$h(S) = \min(g(d-k, n), I_m(d-k)), \tag{8}$$

as S is tree-complete and $d-k$ is a full level of S.

Using the information above, we can define 'split' as follows. Assume, when $n > 0$, that m_1 and m_2 are the marks of the left and right children of s, respectively (m_1 and m_2 are uniquely determined by m), then

$$\text{split}_m(n,k) = \begin{cases} \{\,\langle\rangle\,\}, & \text{if } n = 0; \\ \{\,\langle n_1, n_2\rangle \mid \exists l(l = d-k \wedge \varphi)\,\}, & \text{otherwise,} \end{cases}$$

where φ is the formula

$$\begin{aligned} &g(l,n) = g(l-1,n_1) + g(l-1,n_2) \wedge \\ &g(l-1,n_1), g(l-1,n_2) \leq 2^{l-1} \wedge \\ &\min(g(l,n), I_m(l)) = \\ &\min(g(l-1,n_1), I_{m_1}(l-1)) + \min(g(l-1,n_2), I_{m_2}(l-1)). \end{aligned}$$

Figure 12 shows a path-minimal natural cycletree with 10 internal vertices. We have $\text{split}_{\leftarrow}(10,0) = \{\underline{\langle 4,5\rangle}, \langle 5,4\rangle\}$, $\text{split}_{\downarrow}(4,1) = \{\underline{\langle 2,1\rangle}\}$ and $\text{split}_{\uparrow}(5,1) = \{\langle 1,3\rangle, \underline{\langle 2,2\rangle}\}$. Note that, e.g., $\langle 6,3\rangle$ is not in $\text{split}_{\leftarrow}(10,0)$ because although it would give tree-completeness, the tree would not be path-minimal. ∎

Before turning to the configuration algorithm, let us show how, for a given subtree $S[a]$, the addresses of a's left and right children can be calculated in a "top down" manner. Let $S_1[a_1]$ and $S_2[a_2]$ be the left and right subtrees of S, respectively. We know from Section 5 that the vertices of S_1 shall precede in cycle order those of S_2. We know also the following:

- If a is a pre-vertex then it is the first vertex of S (in cycle order).

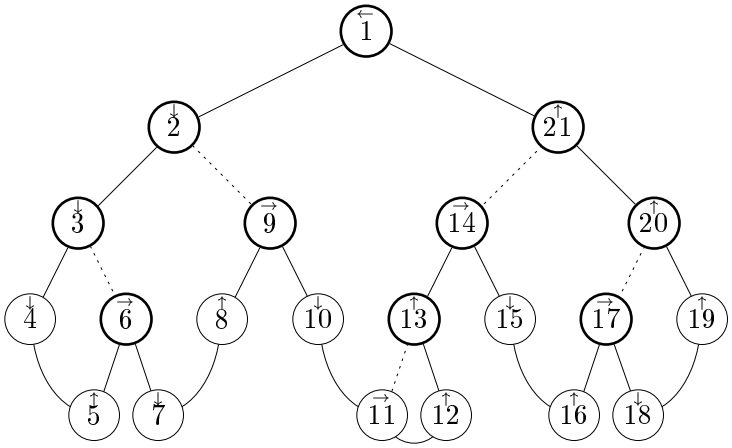

Figure 12: *A path-minimal natural cycletree.*

- If a is an in-vertex then it is between S_1 and S_2.
- If a is a post-vertex then it is the last vertex of S.

For example, if a is a pre-vertex then $a_1 = a+1$ and $a_2 = a+1+|V_{S_1}|+|V_{S'_2}|$, see Figure 13.

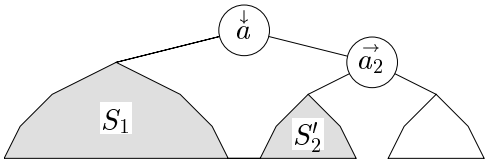

Figure 13: *Calculating the address of the right child of a pre-vertex.*

We can now present the algorithm which constructs a cycletree recursively, using the properties above. As it is a trivial matter to construct a cycle, one is assumed to be given. Define $\nu(\langle x, y\rangle) = 2(x + y + 1) + 1$ and $\nu(\langle\rangle) = 1$, $\pi_1(\langle x, y\rangle) = 2x + 1$ and $\pi_1(\langle\rangle) = 0$, and $\pi_2(\langle x, y\rangle) = 2y + 1$ and $\pi_2(\langle\rangle) = 0$. The intuition behind the above functions is as follows. Consider a binary tree S having $x + y + 1$ internal vertices. Then $\nu(\langle x, y\rangle)$ denotes the *total* number of vertices in S. Accordingly, $\pi_1(\langle x, y\rangle)$ and $\pi_2(\langle x, y\rangle)$ denote the total number of vertices in the left and right subtrees of S, respectively.

Algorithm 6.4 Let C be a cycle of N vertices; N is odd and $N \geq 3$. An enumeration of the vertices is assumed. The algorithm constructs an ordered cycletree

$$G[C, T] = \langle V_C, E_C \cup E\rangle,$$

by adding the missing noncycle edges E to C in such a way that for every subtree S of T having root with mark m and level k, n_1 internal vertices in its left subtree and n_2 internal vertices in its right subtree, $\langle n_1, n_2 \rangle \in \mathrm{split}_m(n_1 + n_2 + 1, k)$.

[**Initialize**] Let $n_T = (N-1)/2$, i.e., n_T is the number of internal vertices in T. Let $p \in \mathrm{split}_{\leftarrow}(n_T, 0)$ (note that $\nu(p) = N$). Now E is given by $\mathrm{cfg}_{\leftarrow}(1, p, 0)$ where 'cfg' is defined recursively as follows.

[**Configure**] $\mathrm{cfg}_m(a, p, k)$.

[**Base case**] If $p = \langle\rangle$ then a is an external vertex at level k with mark m;

$$\mathrm{cfg}_m(a, p, k) = \emptyset.$$

[**Recursive case**] If $p = \langle n_1, n_2 \rangle$, then a is an internal vertex at level k with mark m. The (addresses of the) left child a_1 and right child a_2 of a are calculated and the corresponding subtrees are configured as follows. There are four cases depending on the mark m of a.

[$\boldsymbol{m}$ **is** $\leftarrow$]

$p_1 \in \mathrm{split}_{\downarrow}(n_1, 1)$; $p_2 \in \mathrm{split}_{\uparrow}(n_2, 1)$;
$a_1 = 2$; $a_2 = \nu(p)$;
$\mathrm{cfg}_{\leftarrow}(a, p, 0) = \mathrm{cfg}_{\downarrow}(a_1, p_1, 1) \cup \mathrm{cfg}_{\uparrow}(a_2, p_2, 1)$.

[$\boldsymbol{m}$ **is** $\downarrow$] (see Figure 13)

$p_1 \in \mathrm{split}_{\downarrow}(n_1, k+1)$; $p_2 \in \mathrm{split}_{\rightarrow}(n_2, k+1)$;
$a_1 = a + 1$; $a_2 = a + 1 + \nu(p_1) + \pi_1(p_2)$;

$$\begin{aligned} \mathrm{cfg}_{\downarrow}(a, p, k) \quad = \quad & \mathrm{cfg}_{\downarrow}(a_1, p_1, k+1) \cup \\ & \mathrm{cfg}_{\rightarrow}(a_2, p_2, k+1) \cup \{\, (a, a_2) \,\} \end{aligned}$$

[$\boldsymbol{m}$ **is** $\uparrow$]

$p_1 \in \mathrm{split}_{\rightarrow}(n_1, k+1)$; $p_2 \in \mathrm{split}_{\uparrow}(n_2, k+1)$;
$a_2 = a - 1$; $a_1 = a - 1 - \nu(p_2) - \pi_2(p_1)$;

$$\begin{aligned} \mathrm{cfg}_{\uparrow}(a, p, k) \quad = \quad & \mathrm{cfg}_{\rightarrow}(a_1, p_1, k+1) \cup \\ & \mathrm{cfg}_{\uparrow}(a_2, p_2, k+1) \cup \{\, (a, a_1) \,\} \end{aligned}$$

[$\boldsymbol{m}$ **is** $\rightarrow$]

$p_1 \in \mathrm{split}_{\uparrow}(n_1, k+1)$; $p_2 \in \mathrm{split}_{\downarrow}(n_2, k+1)$;
$a_1 = a - 1$; $a_2 = a + 1$;
$\mathrm{cfg}_{\rightarrow}(a, p, k) = \mathrm{cfg}_{\uparrow}(a_1, p_1, k+1) \cup \mathrm{cfg}_{\downarrow}(a_2, p_2, k+1)$.

Termination of Algorithm 6.4 is guaranteed by the fact that the second argument of 'cfg' is strictly less in each recursive call of 'cfg'. Thus eventually the base case must be reached. For example, by running Algorithm 6.4 with $N = 11$ and assuming a definition of 'split' as given in Example 6.2, the algorithm produces the natural cycletree as shown in Figure 11. The dashed lines correspond to the edge-set E in the algorithm.

7 Embedding of binary tree based networks

Embedding of circular and linear arrays, binary trees and binary tree based networks, like X-trees [11], leap trees [20], back-to-back trees [13], de Bruijn networks [42] and completely linked trees, in other networks such as hypercubes [20, 35, 37, 48, 49], meshes [28, 43, 54], rings [23], pyramids [12], butterflies [40], and VLSI arrays [18, 22, 41, 53] have been studied extensively. Several of those techniques and results apply directly, or with minor modifications, to cycletrees.

Large virtual networks can be simulated by smaller networks by mapping several virtual nodes to one node [21]. The mapping problem, in its most general form, is computationally equivalent to the graph isomorphism problem, as shown by Bokhari [9], and therefore $\mathcal{NP}$-hard.

In a recent paper [5] Barak and Ben-Natan discuss degree and structure preserving partition schemes for mapping (contracting) full trees. In particular, they introduce an ABFS (alternating breadth first search) partition scheme, which for full binary trees yields a bounded contraction of degree three. As a corollary of their Theorem 3.2 [5] we can prove that the same partition scheme can be used to contract natural chaintrees as illustrated by Figure 14.

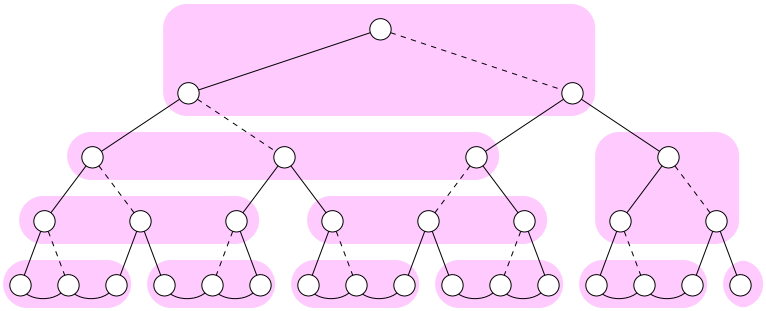

Figure 14: *The ABFS partition scheme applied to a full natural chaintree, each grey area corresponds to a supervertex.*

The same contracted graph is obtained, since the nontree edges occur only locally within each "supervertex" at the leaf level. It is straightforward to modify the ABFS partition scheme for natural cycletrees, to obtain a corresponding bounded contraction of degree three.

Contraction of a natural chaintree using the ABFS partition scheme thus always yields a binary tree. Contraction of a natural cycletree, also with the modified algorithm, generally yields a graph with maximum degree three that is not a binary tree.

8 Properties of natural cycletrees and related networks

In this section we present some properties of natural cycletrees and give a brief survey of related network topologies that have been proposed. For an overview of other static and dynamic connection topologies see, e.g., Almasi and Gottlieb [2].

Our main reason for introducing cycletrees is to support pipeline communication and broadcasting from (or collecting data to) a specific node, i.e., the two communication patterns mentioned in the introduction.

Therefore we have, for now, disregarded issues such as keeping the average distance between an arbitrary pair of nodes as short as possible, fault tolerance and avoidance of congestion points for dense all-to-all communication. It is clear that in case of arbitrary communication patterns natural cycletrees perform only marginally better than binary trees. As in binary trees, the diameter and the average distance is proportional to the depth of the tree, i.e., logarithimic in the best case. Cycletrees have better fault tolerance than binary trees, however, because they are biconnected.

The main properties of a natural cycletree $G[C, T]$ are the following.

- C is the unique Hamiltonian cycle of G and T a basic binary spanning tree of G.
- The maximum degree of G equals the maximum degree of T.
- If G is *minimal for* T then no Hamiltonian graph with T as a spanning tree has fewer edges than G. A minimal natural cycletree exists for any T.
- If G is *path-minimal* then T is tree-complete[3] and E_G is minimal. There exists no Hamiltonian graph with a tree-complete binary spanning tree that has the same set of vertices but fewer edges than G. A path-minimal natural cycletree exists for any odd number of vertices.
- G can be extended incrementally to G' by turning any leaf of T into an internal vertex with two leaves as children and letting those leaves "inherit" its nontree edges. Thus the structure is very flexible. Figure 15 illustrates extension on an in-vertex.

[3] T has the minimal total path length. Then the diameter of G is at most $d_{\min} + 1$, if $d_{\min}$ is the minimal possible diameter.

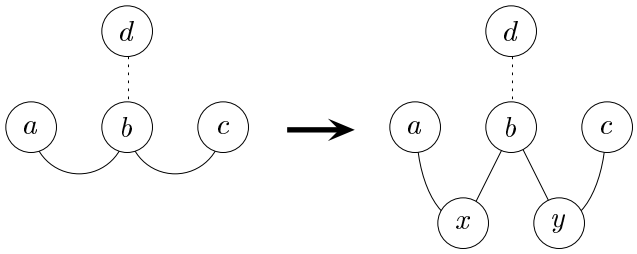

Figure 15: *Extending a natural cycletree on an in-vertex.*

- G is a planar graph. Moreover, there always exists a plane depiction of G where E_C is the contour of the infinite region. These, and other planar properties of natural cycletrees are used in the context of routing in natural cycletrees [47].

The proofs of the properties stated in the last two items are left as easy exercises; the others have been proved in the preceding text.

Binary trees and Linear arrays. A linear array is in itself a powerful interconnection graph for many problems [8] and is used, for example, in the Warp computer [3]. Tseng [46] discusses loop distribution on systolic arrays and presents a systolic array parallelizing compiler for the AL language specially designed for the Warp computer.

Another circular array based interconnection graph is the chordal ring [4]. A binary tree [22] is, for example, a common feature of the DADO architectures [44]. Binary trees are generally used in dictionary machines [17].

X-trees. Most of X-trees introduced by Despain and Patterson [11], e.g., threaded X-trees (having maximum degree 4) and ringed X-trees (having maximum degree 5) are full binary tree based interconnection graphs augmented with extra links to provide uniform message traffic and fault tolerance. A threaded X-tree, see Figure 16, is actually a "preorder threaded" binary tree (assuming left and right as shown in the figure), with an extra thread from the rightmost leaf to the root. Threaded X-trees belong to the class of cycletrees, see also Knuth [24, Exercise 2.3.1–33]. The threaded binary tree in the lower left corner of Figure 16 is what is commonly understood as a threaded binary tree [24, pp. 319–320], i.e., it is threaded in *inorder* (or *symmetric* order). Ringed X-trees are supergraphs of natural cycletrees. Ringed X-trees are interesting in that they do not suffer from data traffic congestion at the root, assuming arbitrary communication patterns.

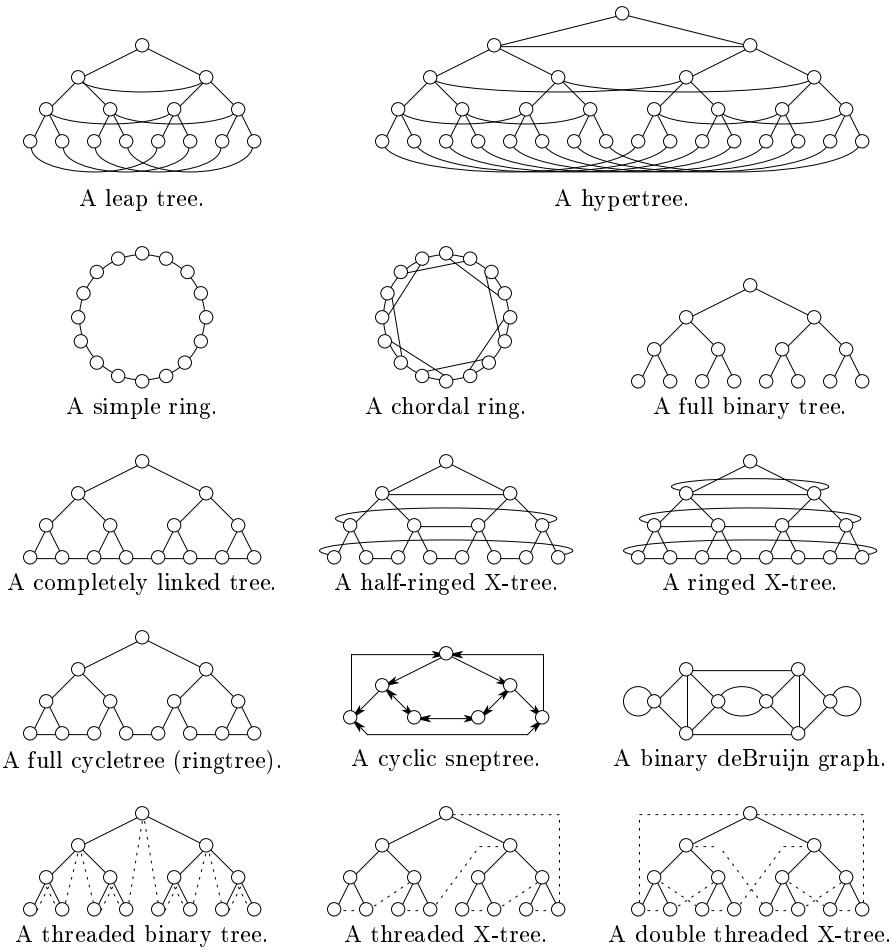

Figure 16: *Some examples of related networks.*

Despain and Patterson point out that a threaded X-tree network is attractive because the threaded links provide a path that linearizes the nodes thus making the network attractive if pipelines of processes are to be distributed onto the tree. Natural cycletrees are also designed with this application in mind, but are even more attractive because, in addition, they minimize the degree and allow arbitrary basic binary spanning trees.

The X-tree structure is used, for example, in the Ottman, Rosenberg and Stockmeyer machine [36]. The Leiserson machine [19, 29] uses the completely linked binary tree structure (also called a semi X-tree [19]), which is also a supergraph of a natural cycletree. In the Leiserson machine, the internal nodes are used for routing only.

Hypertrees. Hypertrees [16] are similar to X-trees in their design. A hypertree is based on a full binary tree structure. There are extra links connecting the nodes on the same level n, forming a set of n-cube connections.

Figure 17 illustrates a 4-cube. There is an edge between the i'th and the j'th vertices whenever the binary representations of i and j differ in exactly one bit (i.e., the Hamming distance between i and j is one). The hypercube

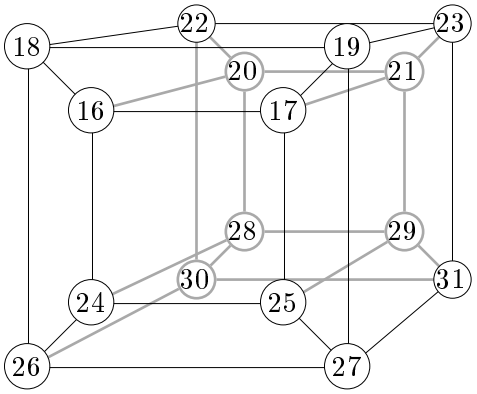

Figure 17: *A 4-cube.*

illustrated in the figure is actually the virtual 4-cube formed at level 4 of the hypertree in figure 16. The vertices are enumerated from left to right in normal order of the binary tree. The leftmost vertex at level 4 then has number 16, which is why the labels in the cube begin with 16 ($= 10000_2$). The vertical edges in the cube are the horizontal edges at level 4 in the hypertree. Any other edge is actually a path through the upper part of the hypertree, which is why we say that the hyper-cube is virtual.

The maximum degree of a hypertree is either four or five. Similarly to X-trees, hypertrees were designed as general purpose interconnection graphs

for problems where arbitrary communication patterns can appear. Hypertrees are better suited than ringed X-trees for problems where remote leaves communicate heavily.

Leap trees [20] have a similar structure as hypertrees. These are full binary trees with additional "leap" edges between the i'th and the j'th vertices (in normal order with the first one being 0'th) at level k, whenever $j = i + 2^{k-1}$.

The interconnection graph of the CM-5 parallel computer [45] is sometimes called a hypertree [33] and is a variant of fat-trees [30]. Although related, this is not the same network.

DeBruijn networks. Binary deBruijn networks [42] of n nodes have a Hamiltonian cycle and and a full binary tree of $n - 1$ nodes. The degree is fixed, i.e., the same for all nodes. In the case of the binary deBruijn network, the degree is four. Those networks have good fault tolerance properties and are well suited as sorting networks, as demonstrated by Samatham and Pradhan [42].

Hyper-deBruijn networks. This interconnection graph [14] is a combination of a hypercube and a deBruijn network. A hyper-deBruijn network admits several other networks, including a circular array and a full binary tree. Unlike hypercubes, hyper-deBruijn networks are bounded degree networks. A hyper-deBruijn network of 2^n nodes can be designed to have a maximum degree k, for any k such that $4 \leq k \leq n$. The diameter is logarithmic. It is a general purpose interconnection graph and the main design goals are fault-tolerance, scalability and simplicity of routing.

Sneptrees. Another class of full binary tree based interconnection graphs are the sneptrees [32]. Each node in a sneptree has degree four. Sneptrees were designed as general purpose interconnection graphs. The main feature of sneptrees is that they can simulate over-sized computation trees through the additional links and are thus particularly suitable for divide-and-conquer algorithms. A deBruijn network of degree 4 is one kind of sneptree. Cyclic sneptrees contain two Hamiltonian circuits.

Ringtrees. Ringtrees [52] form a special case of cycletrees, namely, full natural cycletrees. Natural cycletrees can therefore be seen as a generalization of ringtrees. A ringtree is constructed of two k-linear trees. A k-linear tree corresponds to a full natural chaintree of $2^k - 1$ vertices. Ringtrees were designed largely for the same reasons as natural cycletrees. Unlike the definition of a natural chaintree, the definition of a k-linear tree is not inductive.

9 Conclusions

The most common communication patterns that arise in parallel computations are, arguably, supported by a binary tree structure and a circular array structure. These communication patterns occur frequently in many paradigms for parallel programming [10], but also in computations obtained by automatic parallelization of repetition usually in the form of sequential loops [31, 46, 51]. These communication patterns arise also in parallel computations resulting from parallelization of repetition in declarative programming using, e.g., bounded quantifications [6] or Reform [34], a field where we plan to apply the techniques described herein.

Let us illustrate the relationships between the main classes of graphs that we have introduced in this paper. Let $\mathcal{CCT}$ denote the class of candidate cycletrees, i.e., all Hamiltonian graphs with a basic binary spanning tree, let $\mathcal{CT}$ be the class of cycletrees and let $\mathcal{NCT}$ be the class of natural cycletrees. Let $\mathcal{MCT}$, $\mathcal{TCT}$ and $\mathcal{PCT}$ be the classes of minimal, tree-complete and path-minimal cycletrees, repectively. Let $\mathcal{RT}$ denote the class of ringtrees. See Figure 18. Note that all the intersections in the figure are nonempty.

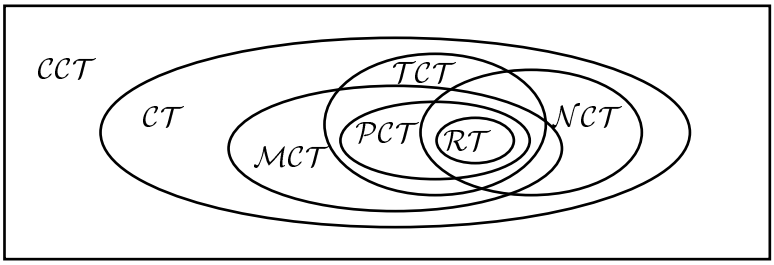

Figure 18: *Some relationships between the various classes of graphs.*

In this paper we focused on natural cycletrees and showed that a natural cycletree includes any basic binary tree, has a unique Hamiltonian cycle, and that the maximum degree of a natural cycletree is 3, which is clearly the lowest possible. We showed that a minimal natural cycletree has the theoretically smallest possible number of nontree edges. Thus, a natural cycletree can be used to realize both a basic binary tree structure and a circular array structure to the lowest possible cost, from an embedding or mapping point of view.

Natural cycletrees have an appealing inductively defined structure. We showed through Algorithm 6.4 how to construct natural cycletrees recursively. This algorithm provides the outlines of how natural cycletree networks can be configured dynamically. Routing in cycletrees is treated elsewhere [47].

References

[1] Agrawal, D. P., Janakiram, V. K. and Pathak, G. C., Evaluating the Performance of Multicomputer Configurations, *IEEE Computer*, 19(5):23–37 (1986).

[2] Almasi, G. S. and Gottlieb, A., *Highly Parallel Computing*, Benjamin/ Cummings, Redwood City, Calif., 1989.

[3] Annaratone, M., Arnould, E., Gross, T., Kung, H. T., Lam, M., Menzilcioglu, O. and Webb, J. A., The Warp Computer: Architecture, Implementation and Performance, *IEEE Transactions on Computers*, C-36(12):1523–1538 (1987).

[4] Arden, B. W. and Lee, H., Analysis of Chordal Ring Network, *IEEE Transactions on Computers*, C-30:291–295 (1981).

[5] Barak, A. and Ben-Natan, R., Bounded Contractions of Full Trees, *J. Parallel and Distributed Computing*, 17:363–369 (1993).

[6] Barklund, J., Bounded Quantifications for Iteration and Concurrency in Logic Programming, *New Generation Computing*, 12:161–182 (1994).

[7] Beivide, R., Herrada, E., Balcázar, E. and Arruabarrena, A., Optimal Distance Networks of Low Degree for Parallel Computers, *IEEE Transactions on Computers*, 40(10):1109–1124 (1991).

[8] Bentley, J. L. and Ottmann, T., On the Power of One Dimensional Vectors of Processors, in: H. Noltemeier (ed.), *Graph-Theoretic Concepts in Computer Science*, LNCS 100, Springer-Verlag, Berlin, 1980.

[9] Bokhari, S. H., On the Mapping Problem, *IEEE Transactions on Computers*, C-30(3):207–214 (1981).

[10] Chaudhuri, P., *Parallel Algorithms: Design and Analysis*, Prentice Hall, Sydney, 1992.

[11] Despain, A. M. and Patterson, D. A., X-tree: A Tree Structured Multiprocessor Computer, *SIGARCH Newsletters*, 6(7):144–151 (1978).

[12] Dingle, A. and Sudborough, I. H., Simulation of Binary Trees and X-Trees on Pyramid Networks, *J. Parallel and Distributed Computing*, 19:119–124 (1993).

[13] Efe, K., Embedding Mesh of Trees in the Hypercube, *J. Parallel and Distributed Computing*, 11:222–230 (1991).

[14] Ganesan, E. and Pradhan, D. K., The Hyper-deBruijn Networks: Scalable Versatile Architecture, *IEEE Transactions on Parallel and Distributed Systems*, 4(9):962–978 (1993).

[15] Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[16] Goodman, J. R. and Séquin, C. H., Hypertree: A Multiprocessor Interconnection Topology, *IEEE Transactions on Computers*, C-30(12):923–933 (1981).

[17] Goodrich, M. T. and Atallah, M. J., On Performing Robust Order Statistics in Tree-Structured Dictionary Machines, *J. Parallel and Distributed Computing*, 9:69–76 (1990).

[18] Gordon, D., Efficient Embedding of Binary Trees in VLSI Arrays, *IEEE Transactions on Computers*, C-36:1009–1018 (1987).

[19] Goyal, P. and Narayanan, T. S., Dictionary Machine with Improved Performance, *The Computer Journal*, 31(6):490–495 (1988).

[20] Gupta, A. K. and Hambrusch, S. E., Multiple Network Embeddings into Hypercubes, *J. Parallel and Distributed Computing*, 19:73–82 (1993).

[21] Heath, L. S., Rosenberg, A. L. and Smith, B. T., The Physical Mapping Problem for Parallel Architectures, *Journal of the ACM*, 35(3):603–634 (1988).

[22] Horowitz, E. and Zorat, A., The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI, *IEEE Transactions on Computers*, C-30:247–253 (1981).

[23] Hromkovitč, J., Müller, V., Sýkora, O. and Vrťo, I., On Embedding Interconnection Networks into Rings of Processors, in: *Proc. PARLE 92*, Springer-Verlag, Berlin, 1992.

[24] Knuth, D. E., *The Art of Computer Programming*, Second edition, volume 1: Fundamental Algorithms, Addison-Wesley, Reading, Mass., 1973.

[25] Kuck, D. J., Muraoka, Y. and Chen, S. C., On the Number of Operations Simultaneously Executable in Fortran-like Programs and their Resulting Speed-up, *IEEE Transactions on Computers*, 21:1293–1310 (1972).

[26] Lam, S. P. S., A Novel Sorting Array Processor, in: *CONPAR 92 – VAPP V*, LNCS 634, Springer-Verlag, Berlin, 1992.

[27] Lamport, L., The Parallel Execution of DO Loops, *Communications of the ACM*, 17:83–93 (1974).

[28] Latifi, S. and El-Amawy, A., Efficient Approach to Embed Binary Trees in 3-D Rectangular Arrays, *IEEE Processors*, 137(2):159–163 (1990).

[29] Leiserson, C. E., Systolic Priority Queues, in: *Proc. CalTech Conf. VLSI*, January 1979.

[30] Leiserson, C. E., Fat-Trees: Universal Networks for Hardware-efficient Supercomputing, *IEEE Transactions on Computers*, C-34(10):892–901 (1985).

[31] Lengauer, C., Towards Systolizing Compilation: An Overview, in: *Proc. PARLE 89*, LNCS 366, Springer-Verlag, Berlin, 1989.

[32] Li, P. P. and Martin, A. J., The Sneptree – A Versatile Interconnection Network, in: K. Hwang, S. M. Jakobs and E. E. Swartzlander (eds.), *Proc. 1986 IEEE Intl. Conf. on Parallel Processing*, IEEE Comp. Soc. Press, August 1986.

[33] Lin, M., Tsang, R. P., Du, D. H. C., Klietz, A. E. and Saroff, S., Performance Characteristics of the Connection Machine Hypertree Network, *J. Parallel and Distributed Computing*, 19:245–254 (1993).

[34] Millroth, H., Reforming Compilation of Logic Programs, in: V. Saraswat and K. Ueda (eds.), *Logic Programming: Proc. 1991 Intl. Symp.*, MIT Press, Cambridge, Mass., 1991.

[35] Monien, B. and Sudborough, I. H., Simulating Binary Trees on Hypercubes, in: J. H. Reif (ed.), *AWOC 88 VLSI Algorithms and Architectures*, LNCS 319, Springer-Verlag, Berlin, 1988.

[36] Ottman, T. A., Rosenberg, A. L. and Stockmeyer, L. J., A Dictionary Machine for VLSI, *IEEE Transactions on Computers*, C-31(9):892–897 (1982).

[37] Provost, F. J. and Melhem, R., A Distributed Algorithm for Embedding Trees in Hypercubes with Modifications for Run-Time Fault Tolerance, *J. Parallel and Distributed Computing*, 14:85–89 (1992).

[38] Reed, D. A. and Grunwald, D. C., The Performance of Multicomputer Interconnection Networks, *IEEE Computer*, 20(6):63–73 (1987).

[39] Reed, D. A. and Schwetman, H. D., Cost-performance Bounds for Multimicrocomputer Networks, *IEEE Transactions on Computers*, C-32(1):83–95 (1983).

[40] Rosenberg, A. L., Graph Embeddings 1988: Recent Breakthroughs, New Directions, in: J. H. Reif (ed.), *AWOC 88 VLSI Algorithms and Architectures*, LNCS 319, Springer-Verlag, Berlin, 1988.

[41] Ruzzo, W. L. and Snyder, L., Minimum Edge Length Planar Embeddings of Trees, in: H. T. Kung, B. Sproull and G. Steele (eds.), *VLSI Systems and Computations*, Springer-Verlag, Berlin, 1981.

[42] Samatham, M. R. and Pradhan, D. K., The de Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI, *IEEE Transactions on Computers*, 38(4):567–581 (1989), corrections in vol. 40, pp. 122, Jan. 1991.

[43] Sitaram, D., Koren, I. and Krishna, C. M., A Random, Distributed Algorithm to Embed Trees in Partially Faulty Processor Arrays, *J. Parallel and Distributed Computing*, 12:1–11 (1991).

[44] Stolfo, S. J., Initial Performance of the DADO2 Prototype, *IEEE Computer*, 20(1):75–83 (1987).

[45] Thinking Machines Corp., Connection Machine CM-5 Technical Summary, 1991.

[46] Tseng, P.-S., A Systolic Array Parallelizing Compiler, *J. Parallel and Distributed Computing*, 9:116–127 (1990).

[47] Veanes, M., Cycletrees: a Novel Class of Interconnection Graphs, Ph.L. thesis, Computing Science Dept., Uppsala University, 1993, Uppsala theses in computing science No. 17/93.

[48] Wagner, A. S., Embedding All Binary Trees in the Hypercube, *J. Parallel and Distributed Computing*, 18:33–43 (1993).

[49] Wagner, A. S., Embedding the Complete Tree in the Hypercube, *J. Parallel and Distributed Computing*, 20:241–247 (1994).

[50] Wittie, L. D., Communication Structures for Large Networks of Microcomputers, *IEEE Transactions on Computers*, C-30:264–273 (1981).

[51] Wolfe, M., New Program Restructuring Technology, in: *Parallel Computation, First Intl. ACPC Conf.*, LNCS 591, Springer-Verlag, Berlin, 1991.

[52] Xie, X. and Ge, Y., An Optimal Structure that Accommodates Both a Ring and a Binary Tree, in: A. Bode (ed.), *Distributed Memory Computing*, LNCS 487, Springer-Verlag, Berlin, 1991.

[53] Youn, H. Y. and Singh, A. D., On Implementing Large Binary Tree Architectures in VLSI and WSI, *IEEE Transactions on Computers*, 38(4):526–537 (1989).

[54] Zienicke, P., Embeddings of Treelike Graphs into 2-dimensional Meshes, in: R. H. Möhring (ed.), *WG'90 Graph-Theoretic Concepts in Computer Science*, LNCS 484, Springer-Verlag, Berlin, 1990.