# 1-identifying codes on trees

#### NATHALIE BERTRAND\*

ENS Cachan
61, avenue du Président Wilson
94235 Cachan Cedex
FRANCE
bertrand@lsv.ens-cachan.fr

## IRÈNE CHARON OLIVIER HUDRY ANTOINE LOBSTEIN

CNRS & ENST 46, rue Barrault 75634 Paris Cedex 13 FRANCE

charon@infres.enst.fr hudry@infres.enst.fr lobstein@infres.enst.fr

#### Abstract

Consider a connected undirected graph G=(V,E), a subset of vertices  $C\subseteq V$ , and an integer  $r\geq 1$ ; for any vertex  $v\in V$ , let  $B_r(v)$  denote the ball of radius r centered at v, i.e., the set of all vertices within distance r from v. If for all vertices  $v\in V$ , the sets  $B_r(v)\cap C$  are all nonempty and different, then we call C an r-identifying code. We study the smallest cardinalities or densities of these codes in trees. In particular, we prove that, in a tree with n vertices, any 1-identifying code contains at least  $\frac{3(n+1)}{7}$  vertices, and, investigating the complete q-ary trees, we also prove that the minimum cardinality of a 1-identifying code in a complete binary tree with  $2^h-1$  vertices is exactly  $\lceil 20(2^h-1)/31 \rceil$ .

#### 1 Introduction

Given a connected undirected graph G = (V, E) and an integer  $r \geq 1$ , we define  $B_r(v)$ , the ball of radius r centered at  $v \in V$ , by

$$B_r(v) = \{x \in V : d(x,v) \le r\},\$$

where d(x, v) denotes the number of edges in any shortest path between v and x. Whenever  $d(x, v) \leq r$ , we say that x and v r-cover each other (or simply cover if

<sup>\*</sup> Work done during a stay at the ENST.

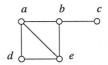


Figure 1: A graph G admitting no 1-identifying code.

there is no ambiguity). A set  $X \subseteq V$  covers a set  $Y \subseteq V$  if every vertex in Y is covered by at least one vertex in X.

A code C is a nonempty set of vertices, and its elements are called codewords. For each vertex  $v \in V$ , we denote by

$$K_{C,r}(v) = C \cap B_r(v)$$

the set of codewords which r-cover v. Two vertices  $v_1$  and  $v_2$  with  $K_{C,r}(v_1) \neq K_{C,r}(v_2)$  are said to be r-separated, or separated, by code C.

A code C is called r-identifying if the sets  $K_{C,r}(v), v \in V$ , are all nonempty and distinct [6]. It is called r-locating-dominating if the same is true for all  $v \in V \setminus C$  [5]. In other words, in the first case all vertices must be covered and pairwise separated by C, in the latter case only the noncodewords need to be covered and separated.

**Remark 1**. For given graph G = (V, E) and integer r, there exists an r-identifying code  $C \subseteq V$  if and only if

$$\forall v_1, v_2 \in V \ (v_1 \neq v_2), \ B_r(v_1) \neq B_r(v_2).$$

Indeed, if for all  $v_1, v_2 \in V$ ,  $B_r(v_1)$  and  $B_r(v_2)$  are different, then C = V is ridentifying. Conversely, if for some  $v_1, v_2 \in V$ ,  $B_r(v_1) = B_r(v_2)$ , then for any code  $C \subseteq V$ , we have  $K_{C,r}(v_1) = K_{C,r}(v_2)$ . For instance, there is no r-identifying code in a complete graph. See also Example 1 below.

**Remark 2.** For given graph G = (V, E) and integer r, an r-locating-dominating code always exists (simply take C = V), and any r-identifying code is r-locating-dominating.

**Example 1.** Consider the graph G in Figure 1. We see that  $B_1(a) = \{a, b, d, e\}$ ,  $B_1(b) = \{a, b, c, e\}$ ,  $B_1(c) = \{b, c\}$ ,  $B_1(d) = \{a, d, e\}$ ,  $B_1(e) = \{a, b, d, e\}$ ; consequently, because  $B_1(a) = B_1(e)$ , there is no 1-identifying code in G (cf. Remark 1 above). On the other hand,  $C = \{a, b\}$  is 1-locating-dominating, since the sets  $K_{C,1}(c) = \{b\}$ ,  $K_{C,1}(d) = \{a\}$ , and  $K_{C,1}(e) = \{a, b\}$ , are all nonempty and different.

The motivations come, for instance, from fault diagnosis in multiprocessor systems. Such a system can be modeled as a graph where vertices are processors and edges are links between processors. Assume that at most one of the processors is malfunctioning and we wish to test the system and locate the faulty processor. For

this purpose, some processors (constituting the code) will be selected and assigned the task of testing their neighbourhoods (i.e., the vertices at distance at most r). Whenever a selected processor (= a codeword) detects a fault, it sends an alarm signal, saying that one element in its neighbourhood is malfunctioning. We require that we can uniquely tell the location of the malfunctioning processor based only on the information which ones of the codewords gave the alarm, and in this case an identifying code is what we need.

If the selected codewords are assumed to work without failure, or if their only task is to test their neighbourhoods (i.e., they are not considered as processors anymore) and we assume that they perform this simple task without failure, then we shall search for locating-dominating codes. These codes can also be considered for modeling the protection of a building, the rooms of which are the vertices of a graph.

Locating-dominating codes were introduced in [5], identifying codes in [6], and they constitute now a topic of their own: both were studied in a large number of various papers, investigating particular graphs or families of graphs (such as planar graphs, certain infinite regular grids, or the *n*-cube), dealing with complexity issues, or using heuristics such as the noising methods for the construction of small codes. See, e.g., [3], [4], and references therein.

Here, we shall study 1-identifying codes in trees. More specifically:

- In Section 2, we address the following question: in a tree with a given number of vertices, how small can a 1-identifying code be? The same question for 1-locating-dominating codes had been answered in [7].
- In Section 3, we study the complete q-ary trees, and give the exact value of the smallest size of a 1-identifying code in a complete q-ary tree of any size,  $q \geq 2$ . Observe that the case q = 1 corresponds to chains, which are studied in [1], [2].

# 2 Bounds on trees

In [7], it is shown that in a tree with n vertices, a 1-locating-dominating code has at least n/3 vertices. We have a similar result for identifying codes.

**Theorem 1** For all  $n \geq 3$  and all trees with n vertices, a 1-identifying code has at least

$$\frac{3(n+1)}{7}$$

vertices.

**Proof.** Assume that T=(V,E) is a tree with n vertices, and that C is a 1-identifying code in T. Let E' be the set of edges having codewords at both ends, and e'=|E'|. For every  $v \in V$ , let  $f(v)=|\{c \in C: d(v,c)=1\}|$ , and finally let  $C^*=\{c \in C: f(c)=0\}$ .

Looking at the edges linking one noncodeword and one codeword, or two codewords, we see that

$$\sum_{v \in V} f(v) \le |E| + e' \le n - 1 + e'. \tag{1}$$

Obviously,

$$\sum_{c \in C \setminus C^*} f(c) = 2e'. \tag{2}$$

Finally, there are at most  $|C \setminus C^*|$  noncodewords v with f(v) = 1, and therefore

$$\sum_{v \in V} f(v) \ge 2|V \setminus C| - |C \setminus C^*| + \sum_{c \in C \setminus C^*} f(c). \tag{3}$$

Combining (1), (2) and (3), we get

$$n-1 \ge 2n-3|C|+|C^*|+\frac{1}{2}\sum_{c \in C \setminus C^*} f(c).$$

We now estimate the sum on the right-hand side. If  $F = (C \setminus C^*, E')$ , then a connected component in F with i vertices contributes 2(i-1) to the sum, so the average f(c) in this component is 2(i-1)/i. Because C is 1-identifying, the size of each connected component in F is at least three, and hence the overall average of f(c) over  $C \setminus C^*$  is at least 4/3. We therefore get

$$n-1 \ge 2n-3|C|+|C^*|+\frac{2}{3}(|C|-|C^*|) \ge 2n-\frac{7}{3}|C|,$$

and the claim follows.

Δ

For infinitely many values of n, we can construct a tree with n vertices admitting a 1-identifying code of size 3(n+1)/7.

**Theorem 2** If n = 6 + 7(m - 1),  $m \ge 2$ , then there exists a tree with n vertices which admits a 1-identifying code having

$$\frac{3(n+1)}{7} = 3m$$

elements.

**Proof.** The construction is the following: first take pattern (a) in Figure 2, then pattern (b) m-1 times, linking each time the leftmost vertex of pattern (b) to the top rightmost vertex of the previous construction. This construction obviously yields a 1-identifying code.

Δ

Note that some trees require as many as n-1 codewords for a 1-identifying code; this is the case for the "star", i.e., the tree consisting of one root linked to n-1 leaves  $(n \ge 3)$ .

Finally, we show that a possible constant lower bound cannot be reached by a finite tree, and that a density of 3/7 can be achieved for a (periodic) 1-identifying code in an infinite tree.

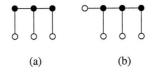


Figure 2: Patterns for 1-identifying codes in trees. Codewords are in black.

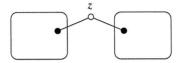


Figure 3: The construction of  $T_2$ .

**Theorem 3** If the density of every 1-identifying code in every finite tree is at least c, where c is a positive constant, then there is no finite tree whose density is c.

**Proof.** Assume on the contrary that there is a tree  $T_1$ , with p vertices  $x_1, \ldots, x_p$ , which admits a 1-identifying code  $C_1 = \{x_{i_1}, \ldots, x_{i_{|C_1|}}\}$ , whose density meets the lower bound c. Then we can construct a second tree,  $T_2$ , with 2p+1 vertices, which admits a 1-identifying code with smaller density: take a copy  $T_1'$  of  $T_1$ , and define  $T_2$  as the union of  $T_1$  and  $T_1'$ , with an additional vertex z and additional edges  $\{x_{i_1}, z\}$  and  $\{x_{i_1}', z\}$  (see Figure 3).

It is straightforward to check that  $C_2 = C_1 \cup C_1'$  is a 1-identifying code in  $T_2$ ; but the density of  $C_2$  is smaller than the density of  $C_1$ :

$$\frac{|C_2|}{2p+1} = \frac{2|C_1|}{2p+1} < \frac{|C_1|}{p} = c,$$

a contradiction.  $\triangle$ 

**Theorem 4** There exists an infinite tree admitting a 1-identifying code with density 3/7.

**Proof.** Repeat an infinite number of times the construction of Theorem 2.  $\triangle$ 

# 3 The complete q-ary tree

For  $q \geq 2$ , we denote by  $\mathcal{CT}_h^q$  the complete q-ary tree of depth h (see Figure 4 for q = 4, h = 3); this tree has

$$X_h^q = \sum_{1 \le i \le h} q^{i-1} = \frac{q^h - 1}{q - 1}$$

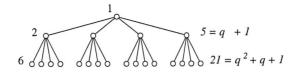


Figure 4: The complete quaternary tree of depth 3.



Figure 5: Codes in complete trees. Codewords are in black.

vertices, which we will number from 1 to  $X_h^q$ , going from left to right and from top to bottom.

We shall say that line i  $(1 \le i \le h)$  consists of the  $q^{i-1}$  vertices numbered from  $\sum_{1 \le i \le i-1} q^{j-1} + 1$  to  $\sum_{1 \le i \le i} q^{j-1}$ .

Let  $M_1^I(\mathcal{CT}_h^q)$  and  $M_r^{LD}(\mathcal{CT}_h^q)$  be the smallest cardinalities of a 1-identifying code and of an r-locating-dominating code, respectively, in  $\mathcal{CT}_h^q$  — it's no use defining  $M_r^I(\mathcal{CT}_h^q)$ , since no r-identifying code exists in  $\mathcal{CT}_h^q$  for r > 1: there exist two leaves  $\ell_1$  and  $\ell_2$  such that  $B_r(\ell_1) = B_r(\ell_2)$ , cf. Remark 1.

In the binary case, we shall drop the superscript and simply write  $\mathcal{CT}_h$  for  $\mathcal{CT}_h^2$  and  $X_h$  for  $X_h^2$ .

Note that there is a fundamental difference for identifying codes between the cases q=2 and q>2, whereas there is no such difference for locating-dominating codes. This is illustrated by Figure 5, which, although very simple, is crucial for the understanding of identifying and locating-dominating codes in complete trees. In this figure, two codes are represented in the complete binary and ternary trees of depth two, each code consisting of the root and all, but one, vertices in the second line of vertices. To the left, one has a code which is 1-locating-dominating, not 1-identifying. To the right, one has a code which is 1-locating-dominating and 1-identifying.

This is why we first study what we think is the most difficult case, that is, the case of 1-identifying codes in the complete binary tree.

Our method will consist in cutting a complete tree into slices having a small number of lines. Inside these slices, we can have a view of the best codes, and finally find the best cardinality in the whole tree.

We consider (see Figure 6) one slice with four lines or fewer, together with one slice with 5k + 4 lines  $(k \ge 0)$ , the latter slice being divided into k slices with five

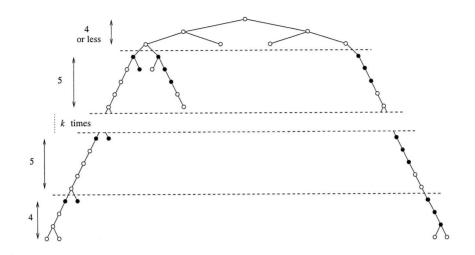


Figure 6: A partial view of the complete binary tree, partitioned into slices. Black vertices represent some of the codewords.

lines and one slice with four lines.

We first consider  $\mathcal{CT}_{5k+4}$   $(k \geq 0)$  and give some lemmas and one corollary involving  $\mathcal{CT}_4$  and  $\mathcal{CT}_5$ .

**Lemma 1** In  $CT_4$ , there are at least ten codewords in a 1-identifying code, and a construction with exactly ten codewords can be achieved.

**Proof.** We number the vertices of  $\mathcal{CT}_4$  from 1 to 15 starting from left to right and from top to bottom. We denote by  $T_L$  the "left part" of  $\mathcal{CT}_4$ , consisting of the vertices  $\{2, 4, 5, 8, 9, 10, 11\}$ , and by  $T_R$  the "right part" of  $\mathcal{CT}_4$ , consisting of the vertices  $\{3, 6, 7, 12, 13, 14, 15\}$ .

First, the code

$$C = \{1, 3, 6, 7, 8, 9, 10, 11, 12, 14\} \tag{4}$$

contains ten elements and is clearly 1-identifying.

Next, we try to construct a 1-identifying code C with nine elements only. If such a code exists, then, without loss of generality, we can assume that there are at most four codewords in  $T_L$ .

Because 8 and 9 have to be covered and separated by C, there are at least two codewords among 4, 8, and 9. The same is true with 5, 10, and 11, so we have already four codewords. As a consequence, 2 is not a codeword.

Note that the same argument for  $\{6, 12, 13\}$  and  $\{7, 14, 15\}$  shows that there are at least four codewords in  $T_R$ .

Now suppose that 4 is a codeword. How to separate 8 and 9 with only one new codeword? One of them, say 8, must be a codeword, but this is not sufficient to

separate 4 and 8. Therefore, 4 is not a codeword, and the same is true with 5, so

$$C \cap T_L = \{8, 9, 10, 11\}.$$

So 2 is not covered by  $C \cap T_L$ , and necessarily  $1 \in C$ . Since  $|C| \leq 9$ , we see that  $|C \cap T_R| = 4$ . Applying to  $T_R$  the argument used for  $T_L$  shows that 3, 6, and 7 cannot be codewords.

This however leads to a contradiction, since 2 and 3 are covered only by the codeword 1, which is not sufficient to separate them.  $\triangle$ 

Observe that in the second part of the previous proof, we made no assumption on how vertex 1 is covered and separated by the code from the other vertices. The contradiction came only by considering vertices 2 to 15. Therefore we have the following, stronger, corollary.

**Corollary 1** In  $CT_4$ , a code which covers all vertices except maybe the root, and separates all vertices from one another except maybe from the root, contains at least ten elements.  $\triangle$ 

**Lemma 2** Consider the tree  $\mathcal{CT}_5$  as a possible part of a larger complete binary tree  $\mathcal{CT}_h$ ,  $h \geq 5$ . A code C in  $\mathcal{CT}_h$ , which covers and pairwise separates all vertices in  $\mathcal{CT}_5$ , contains at least ten codewords in  $\mathcal{CT}_5$ .

**Proof.** We number the vertices of  $\mathcal{CT}_5$  from 1 to 31 starting from left to right and from top to bottom. The fact that  $\mathcal{CT}_5$  can belong to a larger tree means that the vertices 1, 16, 17, ..., 31, can be the extremities of other edges, i.e., may be covered by codewords outside  $\mathcal{CT}_5$ .

Again, denote by  $T_L = \{2, 4, 5, 8, 9, 10, 11, 16, 17, 18, 19, 20, 21, 22, 23\}$  the left part of  $\mathcal{CT}_5$  and  $T_R = \{3, 6, 7, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31\}$  the right part. If  $|C \cap \mathcal{CT}_5| \leq 9$ , we can assume without loss of generality that  $|C \cap T_L| \leq 4$ .

Because 8 and 9 must be covered and separated by C, there are at least two codewords in  $\{4, 8, 9, 16, 17, 18, 19\}$ . The same is true with  $\{5, 10, 11, 20, 21, 22, 23\}$ . Hence there are exactly two codewords in  $\{4, 8, 9, 16, 17, 18, 19\}$  and exactly two codewords in  $\{5, 10, 11, 20, 21, 22, 23\}$ . This shows that 2 is not a codeword, and also, in a similar way, that  $|C \cap T_R| \geq 4$ .

Now suppose that  $4 \in C$ . How to separate 8 and 9 with only one new codeword? Either one of them, say 8, is a codeword, in which case 4 and 8 are not separated by C; or none of them is a codeword: then one, say 9, must be covered by a codeword taken in  $\{18,19\}$ , and again 4 and 8 are not separated. So necessarily  $4 \notin C$ , and similarly  $5 \notin C$ , which means that 2 is not covered by C yet and implies that  $1 \in C$ . In turn, this means that  $|C \cap T_R| = 4$ , that  $3,6,7 \notin C$ , and finally that  $K_{C,1}(2) = K_{C,1}(3) = \{1\}$ , a contradiction.

We now show that a construction with ten elements can be achieved.

**Lemma 3** Consider the tree  $\mathcal{CT}_5$  as a possible part of a larger complete binary tree  $\mathcal{CT}_h$ ,  $h \geq 5$ . Under certain conditions, a code with exactly ten codewords in  $\mathcal{CT}_5$  can be constructed, which covers and pairwise separates all vertices in  $\mathcal{CT}_5$ .

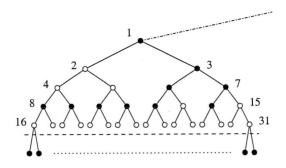


Figure 7: The intersection of a code with  $\mathcal{CT}_5$ . Codewords are in black.

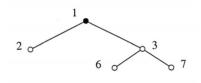


Figure 8: The neighbourhood of the root in  $\mathcal{CT}_4$  or  $\mathcal{CT}_5$ . The black vertex is a codeword. Other vertices are or are not codewords.

**Proof.** Let us assume that  $\mathcal{CT}_5$  is included in a larger complete binary tree, in which we have a code C containing each of the 32 sons of 16, 17, ..., 31; if

$$C \cap \mathcal{CT}_5 = \{1, 3, 6, 7, 8, 9, 10, 11, 12, 14\}$$
 (5)

(see Figure 7), we easily see that all vertices inside  $\mathcal{CT}_5$  are covered and pairwise separated by C.

Note that in the codes given by (4) and (5), vertex 1 is a codeword, i.e., the neighbourhood of the root is as in Figure 8.

Roughly speaking, the information given by Lemma 1 and Corollary 1 on one hand, by Lemmas 2 and 3 on the other hand, is the following: if  $\mathcal{CT}_4$  is in the bottom part of  $\mathcal{CT}_h$  in Figure 6, then ten codewords are necessary and sufficient to take care of its vertices; if  $\mathcal{CT}_5$  is not in the bottom part of  $\mathcal{CT}_h$  in Figure 6, then ten codewords are necessary and can be sufficient. This is more precisely detailed in the proofs of the following two lemmas, involving  $\mathcal{CT}_{5k+4}$ , where  $k \geq 0$ .

Assume that  $\mathcal{CT}_{5k+4}$  is part of a larger tree  $\mathcal{CT}_h$ ,  $5k+8 \geq h \geq 5k+4$ , where only the root of  $\mathcal{CT}_{5k+4}$  can be linked to a vertex of  $\mathcal{CT}_h \setminus \mathcal{CT}_{5k+4}$ ; in other words, the set of leaves of  $\mathcal{CT}_{5k+4}$  is included in the set of leaves of  $\mathcal{CT}_h$ . This keeps the possibility of later adding a slice with four lines or fewer on top of  $\mathcal{CT}_{5k+4}$ . Under this condition, we have the following lower bound.

**Lemma 4** If  $\mathcal{CT}_{5k+4}$  is part of a larger tree  $\mathcal{CT}_h$ ,  $5k+8 \geq h \geq 5k+4$ , where only the root of  $\mathcal{CT}_{5k+4}$  can be linked to a vertex of  $\mathcal{CT}_h \setminus \mathcal{CT}_{5k+4}$ , a 1-identifying code in  $\mathcal{CT}_h$  contains at least

$$\frac{10(2X_{5k+4}+1)}{31}$$

elements in  $\mathcal{CT}_{5k+4}$ , where  $X_{5k+4} = 2^{5k+4} - 1$  is the number of vertices in  $\mathcal{CT}_{5k+4}$ .

**Proof.** Note that  $2X_{5k+4} + 1 = 2^{5k+5} - 1$  is divisible by  $31 = 2^5 - 1$ , so the bound  $10(2X_{5k+4} + 1)/31$  is an integer.

The case k=0 has been solved by Lemma 1 and Corollary 1, so we assume here that k>0.

Let C be a 1-identifying code in  $\mathcal{CT}_h$ . As before, we number the vertices of  $\mathcal{CT}_{5k+4}$  from 1 to  $X_{5k+4}=2^{5k+4}-1$ . We partition  $\mathcal{CT}_{5k+4}$  (cf. Figure 6) into – trees  $\mathcal{CT}_5$ , with roots  $2^{5i}+j$ ,  $0 \le i \le k-1$ ,  $0 \le j \le 2^{5i}-1$ ;

- trees  $\mathcal{CT}_4$ , with roots  $2^{5k} + j$ ,  $0 \le j \le 2^{5k} - 1$ .

By Corollary 1 and Lemma 2, each of these subtrees contains at least ten codewords; therefore,

$$|C \cap \mathcal{CT}_{5k+4}| \ge 10 \sum_{0 \le i \le k} 2^{5i} = 10 \frac{2^{5(k+1)} - 1}{31} = \frac{10(2X_{5k+4} + 1)}{31}.$$

Δ

On the other hand, for  $\mathcal{CT}_{5k+4}$ , we can give a construction with  $\frac{10(2X_{5k+4}+1)}{31}$  codewords.

**Lemma 5** In  $CT_{5k+4}$ , there is a 1-identifying code with

$$\frac{10(2X_{5k+4}+1)}{31}$$

elements, where  $X_{5k+4}=2^{5k+4}-1$  is the number of vertices in  $\mathcal{CT}_{5k+4}$ .

**Proof.** Again we can assume that k > 0. We partition  $\mathcal{CT}_{5k+4}$  in the same way as in the proof of the previous lemma. In each of the trees  $\mathcal{CT}_5$ , we take as codewords the ten vertices corresponding to (5). In each of the trees  $\mathcal{CT}_4$ , we take as codewords the ten vertices corresponding to (4).

We claim that the code C thus constructed is 1-identifying in  $\mathcal{CT}_{5k+4}$ . Because we always take the root, 1, as a codeword in the constructions of (5) and (4), we see that we are in the conditions of the proof of Lemma 3. Therefore, using Lemma 3 and Lemma 1, we see that inside each  $\mathcal{CT}_4$  or  $\mathcal{CT}_5$ , each vertex is covered and each pair of vertices is separated by C.

Problems can arise only at a border between two slices. So all we have to check is that moreover, a vertex  $2^{5i} + j$   $(1 \le i \le k, 0 \le j \le 2^{5i} - 1)$  is separated by C from its father f and its grandfather g, and, for j even, from its brother  $2^{5i} + j + 1$ , and that its two sons,  $s_1$  and  $s_2$ , are separated by C from f. Since the neighbourhood of

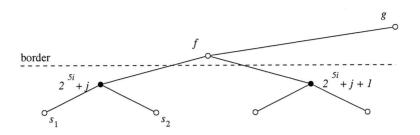


Figure 9: The neighbourhood of  $2^{5i} + j$  (j even). Black vertices are codewords. Other vertices are or are not codewords.

a root in  $\mathcal{CT}_4$  or  $\mathcal{CT}_5$  is as in Figure 8, the neighbourhood of  $2^{5i} + j$  is as in Figure 9. The checking is now immediate.

**Remark 3**. Observe that in the previous construction,  $1 \in C \cap \mathcal{CT}_{5k+4}$ , so the root of  $\mathcal{CT}_{5k+4}$  also has its neighbourhood as in Figure 8.

Now we have to add a slice with four lines or fewer on top of a tree  $\mathcal{CT}_{5k+4}$ , in order to deal with  $\mathcal{CT}_h$ . We need the following three lemmas, which state that if the first slice is a complete binary tree of depth four, three, two, or one, then, under some conditions, it is necessary and sufficient to have five, two, one, and zero codeword(s), respectively.

**Lemma 6** Consider the tree  $\mathcal{CT}_4$  as a possible part of a larger complete binary tree  $\mathcal{CT}_h$ ,  $h \geq 4$ , with the assumption that the root of  $\mathcal{CT}_4$  is the root of  $\mathcal{CT}_h$ . A code C in  $\mathcal{CT}_h$ , which covers and pairwise separates all vertices in  $\mathcal{CT}_4$ , contains at least five codewords in  $\mathcal{CT}_4$ .

**Proof.** The fact that  $\mathcal{CT}_4$  can belong to a larger tree with same root means that the vertices 8, 9, ..., 15, can have descendants, i.e., may be covered by codewords outside  $\mathcal{CT}_4$ .

Because 4 and 5 must be covered and separated by C, there are at least two codewords in  $T_L = \{2, 4, 5, 8, 9, 10, 11\}$ . The same is true for 6 and 7, therefore, with  $T_R = \{3, 6, 7, 12, 13, 14, 15\}, |C \cap T_R| \geq 2$ . If we try with only four codewords, this means that  $|C \cap T_L| = |C \cap T_R| = 2$ , and  $1 \notin C$ . Since 1 must be covered by C, we can assume without loss of generality that  $2 \in C$ . Since 4 and 5 have to be separated by C, one of them, say 4, has to be covered by a second codeword. Either  $4 \in C$  is covered by itself, and 2 and 4 are not separated, or 4 is covered by  $8 \in C$  or  $9 \in C$ , in which case 2 and 5 are not separated. In both cases, we see that it is impossible to have only four codewords.

**Lemma 7** Consider the tree  $\mathcal{CT}_4$  as a possible part of a larger complete binary tree  $\mathcal{CT}_h$ ,  $h \geq 4$ , with the assumption that the root of  $\mathcal{CT}_4$  is the root of  $\mathcal{CT}_h$ . Under certain conditions, a code with exactly five codewords in  $\mathcal{CT}_4$  can be constructed, which covers and pairwise separates all vertices in  $\mathcal{CT}_4$ .

**Proof.** Let us assume that  $\mathcal{CT}_4$  is included in a larger complete binary tree, in which we have a code C containing each of the 16 sons of 8, 9, ..., 15; if

$$C \cap \mathcal{CT}_4 = \{1, 4, 5, 6, 7\},\tag{6}$$

we easily see that all vertices inside  $\mathcal{CT}_4$  are covered and pairwise separated by  $\mathcal{C}$ .

The following easy lemma deals with the trees of depth three, two, and one, and we omit the proof.

**Lemma 8** For i = 3, 2, 1, consider the tree  $CT_i$  as a possible part of a larger complete binary tree  $CT_h$ ,  $h \ge i$ , with the assumption that the root of  $CT_i$  is the root of  $CT_h$ .

A code in  $\mathcal{CT}_h$ , which covers and pairwise separates all vertices in  $\mathcal{CT}_i$ , contains at least f(i) codewords in  $\mathcal{CT}_i$ , where

$$f(3) = 2, f(2) = 1, and f(1) = 0.$$

Assume that a code  $C_i \subset \mathcal{CT}_h$  contains each of the  $2^i$  sons of  $2^{i-1}$ ,  $2^{i-1} + 1$ , ...,  $2^i - 1$  in  $\mathcal{CT}_i$ . If

$$C_3 \cap \mathcal{CT}_3 = \{2, 3\}, C_2 \cap \mathcal{CT}_2 = \{1\}, C_1 \cap \mathcal{CT}_1 = \emptyset,$$

then  $C_i$  has exactly f(i) codewords in  $\mathcal{CT}_i$ , and  $C_i$  covers and pairwise separates all vertices in  $\mathcal{CT}_i$ .  $\triangle$ 

We are now ready to give our result in the binary case.

**Theorem 5** For all  $h \ge 1$ ,

$$M_1^I(\mathcal{CT}_h) = \left\lceil \frac{20X_h}{31} \right\rceil,$$

where  $X_h = 2^h - 1$  is the number of vertices in  $\mathcal{CT}_h$ .

**Proof.** We consider the five cases h = 5k + j, j = 0, 1, 2, 3, or 4. The case j = 4 was dealt with in Lemmas 4 and 5. We now show how to handle the remaining cases.

If k = 0 (seven, three, or one vertices), it is straightforward to check that, respectively, five, two, or one codeword(s) are necessary and sufficient.

If  $k \geq 1$ :

- If j=3, partition  $\mathcal{CT}_{5k+3}$  into one  $\mathcal{CT}_4$  and sixteen  $\mathcal{CT}_{5k-1}$  linked, two by two, to the eight leaves  $8, 9, \ldots, 15$  of  $\mathcal{CT}_4$ . Let  $X'=X_{5k-1}=2^{5k-1}-1=(X_h-15)/16$ .

Since 5k-1=5(k-1)+4 and since we are placed in the conditions of Lemma 6 and of Lemma 4, we have in  $\mathcal{CT}_{5k+3}$  a number of codewords which is at least

$$5 + 16 \frac{10(2X'+1)}{31} = \frac{20X_h + 15}{31} = \left\lceil \frac{20X_h}{31} \right\rceil.$$

On the other hand, using Remark 3 and the same argument as in the proof of Lemma 5, we see that taking as codewords in  $\mathcal{CT}_4$  the five vertices given by (6) and

in each  $\mathcal{CT}_{5k-1}$  the 10(2X'+1)/31 vertices of Lemma 5, we obtain a 1-identifying code in  $\mathcal{CT}_{5k+3}$ , which has the announced size.

– If j=2, partition  $\mathcal{CT}_{5k+2}$  into one  $\mathcal{CT}_3$  and eight  $\mathcal{CT}_{5k-1}$  linked to the four vertices 4, 5, 6, 7 of  $\mathcal{CT}_3$ . Then, using the lower bound from Lemma 8, the computation goes

$$2 + 8 \frac{10(\frac{2(X_h - 7)}{8} + 1)}{31} = \frac{20X_h + 2}{31} = \left\lceil \frac{20X_h}{31} \right\rceil.$$

– If j=1, partition  $\mathcal{CT}_{5k+1}$  into one  $\mathcal{CT}_2$  and four  $\mathcal{CT}_{5k-1}$ . Then, using the lower bound from Lemma 8, the computation goes

$$1 + 4 \frac{10(\frac{2(X_h - 3)}{4} + 1)}{31} = \frac{20X_h + 11}{31} = \left[\frac{20X_h}{31}\right].$$

– If j=0, partition  $\mathcal{CT}_{5k}$  into one  $\mathcal{CT}_1$  and two  $\mathcal{CT}_{5k-1}$ . Then, using the lower bound from Lemma 8, the computation goes

$$0 + 2 \frac{10(\frac{2(X_h - 1)}{2} + 1)}{31} = \frac{20X_h}{31} = \left\lceil \frac{20X_h}{31} \right\rceil.$$

Δ

The same technique could, more or less, be easily adapted in order to deal with the cases q > 2 (1-identifying codes) and  $q \ge 2$  (1-locating-dominating codes). There exists however a shortcut: in [7], there is an algorithm which, given a tree, outputs a 1-locating-dominating code with the smallest possible cardinality. Examining how the algorithm runs on complete trees, one can see that the result consists of the following code in  $\mathcal{CT}_b^q$ :

- if a line has index  $i \neq 1$  congruent to h modulo 3, then each vertex on line i-1 has exactly q-1 of its q sons in C; if  $1=h \pmod 3$ , then  $1 \in C$ ;
- if a line has index congruent to h-1 modulo 3, then C contains all the vertices of this line;
- if a line has index congruent to h-2 modulo 3, then this line contains no codewords.

Figure 10 illustrates these constructions in three small complete trees. We can compute the number of codewords and determine  $M_1^{LD}(\mathcal{CT}_h^q)$  for all h and q:

- If h = 3s, then no vertex in lines 1, 4, ..., h-2 is a codeword, all vertices in lines 2, 5, ..., h-1 are codewords, and a proportion (q-1)/q of the vertices in lines 3, 6, ..., h are codewords. Therefore

$$|C| = q^{1} + q^{4} + \dots + q^{h-2} + \frac{q-1}{q} \left( q^{2} + q^{5} + \dots + q^{h-1} \right)$$

$$= q^{2} \left( 1 + q^{3} + \dots + q^{h-3} \right) = q^{2} \frac{q^{h} - 1}{q^{3} - 1}$$

$$= \frac{q^{2}}{q^{2} + q + 1} \frac{q^{h} - 1}{q - 1} = \frac{q^{2} X_{h}^{q}}{q^{2} + q + 1}.$$

- If h = 3s + 1, then similarly

$$|C| = \frac{q^2 X_h^q + q + 1}{q^2 + q + 1} = \left[ \frac{q^2 X_h^q}{q^2 + q + 1} \right].$$

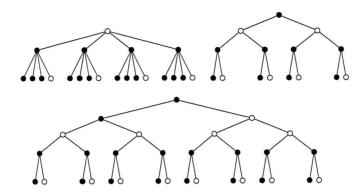


Figure 10: Optimal 1-locating-dominating codes in  $\mathcal{CT}_3^4$ ,  $\mathcal{CT}_4$ , and  $\mathcal{CT}_5$ . Codewords are in black.

- If h = 3s + 2, then

$$|C| = \frac{q^2 X_h^q + q}{q^2 + q + 1} = \left[ \frac{q^2 X_h^q}{q^2 + q + 1} \right].$$

Therefore, we have the following proposition and corollary on locating-dominating codes.

**Proposition 1** For all  $q \geq 2$ , for all  $h \geq 1$ ,

$$M_1^{LD}(\mathcal{CT}_h^q) = \left\lceil \frac{q^2 X_h^q}{q^2 + q + 1} \right\rceil,$$

Δ

Δ

where  $X_h^q$  is the number of vertices in  $\mathcal{CT}_h^q$ .

Corollary 2 For all  $h \ge 1$ ,

$$M_1^{LD}(\mathcal{CT}_h) = \left\lceil \frac{4X_h}{7} \right\rceil,$$

where  $X_h$  is the number of vertices in  $\mathcal{CT}_h$ .

Apart from showing that the binary and nonbinary cases behave differently for identifying codes but similarly for locating-dominating codes, Figure 5 has another interesting feature: in Figure 5(b), where q > 2, we also see that in  $\mathcal{CT}_2^q$ , a 1-locating-dominating code requires as many codewords as a 1-identifying code (and not more, since any identifying code is a fortiori locating-dominating, cf. Remark 2).

Actually, it appears that in  $\mathcal{CT}_h^q$ , the 1-locating-dominating codes described above also are 1-identifying codes when q > 2. The checking is easy, going line by line from the leaves to the root.

 $\triangle$ 

**Theorem 6** For all q > 2, for all  $h \ge 1$ ,

$$M_1^I(\mathcal{CT}_h^q) = \left[\frac{q^2 X_h^q}{q^2 + q + 1}\right],$$

where  $X_h^q$  is the number of vertices in  $\mathcal{CT}_h^q$ .

In general, it is not true however that an optimal 1-locating-dominating code is necessarily also 1-identifying in  $\mathcal{CT}_h^q$  (q > 2), and we have a counter-example in  $\mathcal{CT}_6^3$ .

# Acknowledgments

We wish to thank Iiro Honkala and the anonymous referee for some helpful comments.

### References

- N. Bertrand, I. Charon, O. Hudry and A. Lobstein, Identifying or locating-dominating codes for some families of graphs, Rapport Interne Télécom Paris-2003 C001, Paris, France, 48 pp, Feb. 2003.
- [2] N. Bertrand, I. Charon, O. Hudry and A. Lobstein, Identifying and locating-dominating codes on chains and cycles, *European J. Combin.* 25/7 (2004), 969–987.
- [3] I. Charon, O. Hudry and A. Lobstein, Identifying codes with small radius in some infinite regular graphs, *Electron. J. Combin.* **9(1)** (2002), R11.
- [4] I. Charon, O. Hudry and A. Lobstein, Minimizing the size of an identifying or locating-dominating code in a graph is NP-hard, Theoret. Comput. Sci. 290(3) (2003), 2109-2120.
- [5] C. J. Colbourn, P. J. Slater and L. K. Stewart, Locating dominating sets in series parallel networks, *Congr. Numer.* **56** (1987), 135–162.
- [6] M. G. Karpovsky, K. Chakrabarty and L. B. Levitin, On a new class of codes for identifying vertices in graphs, *IEEE Trans. Inform. Theory* 44(2) (1998), 599-611.
- [7] P. J. Slater, Domination and location in acyclic graphs, Networks 17 (1987), 55-64.

(Received 25 Mar 2003)