Distinguishing trees in linear time

Antoni Lozano*
Dpt. Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Barcelona, Spain
antoni@lsi.upc.edu

Mercè Mora† Carlos Seara
Dpt. Matemàtica Aplicada II
Universitat Politècnica de Catalunya
Barcelona, Spain
{merce.mora, carlos.seara}@upc.edu

Submitted: Apr 15, 2011; Accepted: Apr 26, 2012; Published: May 21, 2012
Mathematics Subject Classifications: 05C85, 68R05, 05C15, 05C78

Abstract

A graph is said to be *d-distinguishable* if there exists a d-labeling of its vertices which is only preserved by the identity map. The *distinguishing number* of a graph G is the smallest number d for which G is d-distinguishable. We show that the distinguishing number of trees and forests can be computed in linear time, improving the previously known $\mathcal{O}(n \log n)$ time algorithm.

1 Introduction

Let G be a connected graph with n vertices[1]. A d-labeling of G is a total function $\phi : V(G) \longrightarrow \{1, 2, \ldots, d\}$. We say that ϕ *distinguishes* G if G has no label-preserving automorphism different from the identity map. In this case, we say that ϕ is a *distinguishing d-labeling* of G. Such a labeling is said to break or destroy the symmetries of G. The *distinguishing number* of G, $D(G)$, is the minimum number d of labels needed so that G has a distinguishing d-labeling. A graph G having a distinguishing d-labeling is said to be *d-distinguishable*.

Distinguishing numbers were first introduced by Albertson and Collins [2]. The parameter can be thought of as a measure of the symmetry of a graph, i.e., if G and G'

*Antoni Lozano is supported by projects MTM2011-28800-C02-01 and BASMATI, Biological and Social Mining: Algorithms, Theory, and Implementation TIN2011-27479-C04-03.

†Mercè Mora and Carlos Seara are partially supported by projects MTM2009-07242, Gen Cat DGR2009GR1040, and the ESF EUROCORES programme EuroGIGA-ComPoSe IP04-MICINN Project EUI-EURC-2011-4306.

[1]Graphs in this paper are finite, undirected and simple. The vertex-set and the edge-set of a graph G are denoted by $V(G)$ and $E(G)$, respectively. The order of G is the number of its vertices, denoted by $|V(G)|$. For more terminology we follow [19].

have the same number of vertices but $D(G) > D(G')$, then G is more symmetric than G' because more colors are needed to destroy its automorphisms than those of G'.

It is not known if the problem of computing $D(G)$ is polynomially-time solvable or NP-hard. Russell and Sundaram [17] showed that determining if $D(G) > d$ belongs to the class AM, i.e., the set of languages for which there are Arthur-Merlin games. However, when G is restricted to certain graph families such as cycles, hypercubes, acyclic graphs, and planar graphs, the problem can be solved efficiently [2, 4, 5, 6, 7, 10]. See [1, 5, 8, 9, 11, 14, 15, 16, 18] for other works on distinguishing number problems.

For the computation of the distinguishing number of trees and forests with n vertices, Cheng [10] and Arvind and Devanur [4] presented an $\mathcal{O}(n \log n)$ time algorithm which uses a binary search to compute the distinguishing number of a tree. Improving this time complexity was our main motivation for the design of an optimal linear-time algorithm for computing the distinguishing number of trees and forests, and this is the main result of our paper.

In Section 2 we will focus on the design of a linear time algorithm for rooted trees which is based on properties proved by Cheng [10] and follow her notation whenever possible. As a consequence of our result, we show in Section 3 that there are linear-time algorithms for computing the distinguishing numbers of trees and forests. Finally, in Section 4 we conjecture logarithmic factor improvements for other graph classes.

2 Distinguishing Rooted Trees

2.1 Preliminaries

We start with some notation. By $\mathrm{Aut}(G)$ we denote the automorphism group of a graph G. As usual, two graphs G and H are *isomorphic*, denoted by $G \cong H$, if there is a permutation $\pi : V(G) \to V(H)$ which preserves adjacencies, that is, $\{u, v\} \in E(G)$ if and only if $\{\pi(u), \pi(v)\} \in E(H)$ for any $u, v \in V(G)$.

Given a graph G and a labeling ϕ of G, we represent the corresponding labeled graph by (G, ϕ). In this case, $\mathrm{Aut}(G, \phi)$ consists of all the automorphisms of $\mathrm{Aut}(G)$ which preserve the labeling ϕ, that is, $\pi \in \mathrm{Aut}(G, \phi)$ if and only if $\phi \in \mathrm{Aut}(G)$ and $\phi(v) = \phi(\pi(v))$. We also consider the extension of isomorphism to labeled graphs. Given two labeled graphs (G, ϕ) and (H, φ), we say that they are isomorphic if there is a permutation $\pi : V(G) \to V(H)$ which preserves adjacencies as defined above, but also preserves labels, that is, $\phi(v) = \varphi(\pi(v))$ for each $v \in V(G)$. In this case, we write $(G, \phi) \cong (H, \varphi)$. Two distinguishing d-labelings ϕ and φ of a graph G are said to be *equivalent* if $(G, \phi) \cong (G, \varphi)$.

Given a rooted tree T, we denote its root by $r(T)$. We also denote by T_u the subtree of T rooted at vertex u of T, and we call *components* of T to all the subtrees T_u of T where u is a child of $r(T)$. Any isomorphism between two rooted trees T_1 and T_2 must map $r(T_1)$ into $r(T_2)$. In the same way, any automorphism of a rooted tree must map the root into itself.

As we will see, the distinguishing number of a rooted tree can be computed using a recursive formula. Call $D(T, d)$ to the number of inequivalent distinguishing d-labelings

of a rotated tree T. For instance, if T is a rotated tree consisting of a single path of k vertices, then $D(T,d) = d^k$, but if T is a full binary tree of (any) depth k, then $D(T,2)$ is just 2 since we can assign two possible labels to the root while the rest of the tree has a unique distinguishing 2-labeling up to isomorphism. Clearly, for any graph G,

$$D(G) = \min\{d \mid D(G,d) > 0\},$$

and, therefore, computing $D(T,d)$ for any d is all that is needed to compute $D(T)$ for a rotated tree T. We also observe the following relation between $D(T)$ and $D(T,d)$.

Proposition 1. *For any rotated tree T and for any $d \geqslant D(T)$, it holds $D(T,d) \geqslant d$.*

Proof. For any $d \geqslant D(T)$, the rotated tree T is clearly d-distinguishable, so suppose ϕ is some d-distinguishing labeling of T. Changing the label assigned by ϕ to the root gives d inequivalent labelings of T, that is,

$$\phi_i(u) = \begin{cases} i, & \text{if } u = r(T) \\ \phi(u), & \text{if } u \neq r(T). \end{cases}$$

for every $i \in \{1,\ldots,d\}$. To see that the labelings are inequivalent, suppose on the contrary that two such labelings, say ϕ_i and ϕ_j (where $1 \leqslant i < j \leqslant d$), are equivalent. Then, there would be a mapping π between the labeled copies (T,ϕ_i) and (T,ϕ_j) such that $\phi_i(r(T)) = \phi_j(\pi(r(T))) = \phi_j(r(T))$, where the last equality holds because any isomorphism between labeled copies of a rooted tree must map the root to itself. But then, by definition of ϕ_i and ϕ_j, we get $i = j$, contradicting our assumption that $i < j$.

The existence of the inequivalent d-labelings $\phi_1,\ldots,\phi_d$ for T implies that $D(T,d) \geqslant d$. □

Now, we consider the recursive formula developed in [4] and [10] which counts the number of inequivalent distinguishing d-labelings of a rooted tree, and how to derive the distinguishing number from it in two ways.

Proposition 2. ([10], Th. 3.2, Cor. 3.3, Th. 4.2) *Let T be a rooted tree and $\mathcal{T}$ be the set of the components of T. Suppose that $\mathcal{T}$ has exactly g distinct isomorphism classes of trees where the ith isomorphism class consists of m_i copies of the rooted tree T_{u_i}; i.e., $\mathcal{T} = m_1T_{u_1} \cup m_2T_{u_2} \cup \cdots \cup m_gT_{u_g}$. Then,*

1. $D(T,d) = d\prod_{i=1}^{g}\binom{D(T_{u_i},d)}{m_i}$.
2. $D(T) = \min\{d \mid \forall i \in \{1,\ldots,g\}\ \ D(T_{u_i},d) \geqslant m_i\}$.
3. $D(T) = \max\{\min\{d \mid D(T_{u_i},d) \geqslant m_i\} \mid 1 \leqslant i \leqslant g\}$.

By Proposition 2(1), in order to compute $D(T)$ for a rooted tree T, it is enough to know a list of values $\{(m_1,u_1),\ldots,(m_g,u_g)\}$, where the degree of $r(T)$ equals $\sum_{i=1}^{g} m_i$ and for each pair (m_i,u_i), u_i is a child of $r(T)$ and m_i is the multiplicity of the isomorphic copies of T_{u_i} which appear as components of T. We take advantage of the fact that Cheng [10] shows

a method to compute exactly this information. We will assume that a new procedure called COMPUTE-LIST, given a rooted tree T, returns the list $\{(m_1, u_1), \ldots, (m_g, u_g)\}$ defined above (in [10], this can be accomplished by calling procedure FIND-ISOMORPH, then ESSENTIAL and, finally, taking the first output). Cheng [10] shows that this can be done in linear time.

2.2 Linear Time Algorithm

We will describe the procedures used in our main algorithm. In the first place, procedure COLORINGS(T, d), given a rooted tree T and a constant d, computes $D(T, d)$ in linear time. We will not detail the algorithm here since it is already described by Cheng [10] and by Arvind and Devanur [4]. This procedure is called EVALUATE in [10] and *Inequiv* in [4].

Note that, according to Proposition 2(2), given a rooted tree T of order n, $D(T)$ can already be computed by making calls to COLORINGS(T, d) for different values of d, $1 \leqslant d \leqslant n-1$. Since each call takes linear time, using binary search in d, it is possible to find $D(T)$ in time $\mathcal{O}(n \log n)$, as it is argued in [4] and [10]. This is precisely the common method in [4, 10] which works in time $\mathcal{O}(n \log n)$. In order to lower it to an overall linear time, we need to carefully call this procedure for the subtrees of T only when it is needed. To do so, we need the information contained in the list $\{(m_1, u_1), \ldots, (m_g, u_g)\}$, which will be obtained in linear time by calling to a procedure COMPUTE-LIST, as stated at the end of Subsection 2.1. Our algorithm is the following (see Figure 1 for an example).

DISTINGUISHING(T)
Input: A rooted tree T with n vertices
Output: $D(T)$

if $|V(T)| = 1$ **then**
  return 1
else
  $col \leftarrow 0$
  $\{(u_1, m_1), \ldots, (u_g, m_g)\} \leftarrow$ COMPUTE-LIST(T)
  for $i = 1$ **to** g **do**
    $d \leftarrow$ DISTINGUISHING(T_{u_i})
    if $d < m_i$ **then**
      while COLORINGS$(T_{u_i}, d) < m_i$ **do**
        $d \leftarrow d + 1$
      end while
    end if
    $col \leftarrow \max\{col, d\}$
  end for
  return *col*
end if

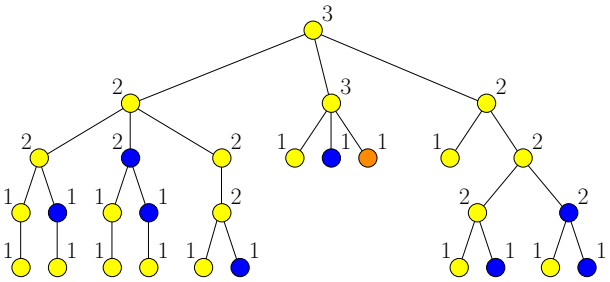

Figure 1: Each vertex is labeled with the distinguishing number of the rooted subtree it defines.

Theorem 3. (Correctness) *Given a rooted tree T as input, procedure* DISTINGUISHING *returns $D(T)$.*

Proof. We show it by induction on the order of T. If the rooted tree T has only one vertex, DISTINGUISHING returns 1 at line 2, which is correct. Suppose now that the order of T is $n > 1$. For each i, procedure DISTINGUISHING makes a recursive call on the subtree T_{u_i} in line 7. By induction hypothesis, we can assume that the result is $d = D(T_{u_i})$. Now we can distinguish two cases:

1. If $d \geqslant m_i$, then by Proposition 1, $D(T_{u_i}, d) \geqslant d \geqslant m_i$.

2. If $d < m_i$, then after the **while** loop in lines 9–11, d is the smallest value satisfying $D(T_{u_i}, d) \geqslant m_i$.

In any case, the value of d at line 13 for a given i is the smallest value such that $D(T_{u_i}, d) \geqslant m_i$. Since the value *col* returned by DISTINGUISHING is the maximum of such values for $1 \leqslant i \leqslant g$, it must equal $D(T)$ by Proposition 2(3). □

Theorem 4. (Complexity) *Procedure* DISTINGUISHING *works in $\mathcal{O}(n)$ time.*

Proof. Let T be the rooted tree of order n given as input and let $T_{u_1}, \ldots, T_{u_g}$ be its different components up to isomorphism, with orders $n_1, \ldots, n_g$, respectively (In [3] the authors provided a linear time isomorphism test for trees). Note that since m_i is the number of isomorphic copies for T_{u_i}, it holds that

$$n = 1 + \sum_{i=1}^{g} m_i \cdot n_i. \tag{1}$$

Since line 5 takes linear time, let a stand for a constant such that an bounds the time required by procedure DISTINGUISHING to execute lines 1–5 and return *col* at line 15. Additionally, the **for** loop between lines 6–14 does some work in constant time, say b, plus a maximum of m_i calls to COLORINGS(T_{u_i}, d). The reason why COLORINGS is called at most m_i times for a given i is that the **while** loop in lines 9–11 is never executed for $d = m_i$ since, by Proposition 1, $D(T_{u_i}, m_i) \geqslant m_i$ and the condition of the loop would not

hold. Now, since COLORINGS works in linear time, we can bound the overall work done between lines 9 and 11 by cn_i for a constant c. Now, if we denote the running time of DISTINGUISHING by $R(n)$, we have

$$R(n) \leqslant an + \sum_{i=1}^{g}(b + c(m_i - 1)n_i + R(n_i)). \tag{2}$$

Let e be a constant such that $e \geqslant a + b + c$. We will show by induction that the running time of DISTINGUISHING(T) is bounded by $an + e(n-1)$. For $n = 1$, we have $R(n) \leqslant a = an + e(n-1)$. In the general case $n > 1$, we unfold the summation in Equation (2) and get:

$$R(n) \leqslant an + bg + c\sum_{i=1}^{g}((m_i - 1)n_i) + \sum_{i=1}^{g}(an_i + e(n_i - 1)),$$

where the last term comes from the induction hypothesis. Now,

$$\begin{aligned}
R(n) &\leqslant an + bg + (e-a)\sum_{i=1}^{g}((m_i - 1)n_i) + a\sum_{i=1}^{g} n_i - eg + e\sum_{i=1}^{g} n_i \\
&\leqslant an + bg + -a\sum_{i=1}^{g}((m_i - 1)n_i) + a\sum_{i=1}^{g} n_i - eg + e(n-1) \\
&\leqslant an - a\sum_{i=1}^{g}(m_i - 1)n_i + a\sum_{i=1}^{g} n_i + e(n-1) \\
&\leqslant an - a(n-1) + a\sum_{i=1}^{g} n_i + e(n-1) \\
&= a(1 + \sum_{i=1}^{g} n_i) + e(n-1) \leqslant an + e(n-1),
\end{aligned}$$

where both Equation (1) and the fact that $e \geqslant a + b + c$ have been used. Therefore, we conclude that $R(n)$ is $\mathcal{O}(n)$. □

3 Distinguishing Trees and Forests

There is an easy way to transform the problem of computing $D(T)$ for a general tree T into the one of computing $D(T')$ for a rooted tree T'. This can be done using the concept of tree center, as is done in [4] and [10]. A *center* of a tree T is a vertex v such that the maximal distance of v to the other vertices is minimized. It is well known that every tree has either one center or two adjacent centers. In the first case, the tree can already be considered a rooted tree with root at its center, while in the second case, a new vertex can be inserted between the centers and then used as its root. As mentioned in [4] and [10], this transformation can be done in linear time.

Proposition 5. *Given a tree T, it is possible to compute a rooted tree T' in linear time such that $D(T) = D(T')$.*

As a direct consequence of Proposition 5 and Theorems 3 and 4, we conclude the following.

Corollary 6. *The distinguishing number of a tree with n vertices can be computed in $\mathcal{O}(n)$ time.*

In the case of forests, we use the transformation from Cheng's paper [10], which is as follows. Suppose that F is a forest and define the following trees T_1 and T_2. In the first place, create two vertices v_1, and v_2 which will act as the respective roots of T_1 and T_2, respectively. In the second place, transform each connected component of F into a rooted tree as indicated before Proposition 5. Since this can be done in two ways, depending on whether the original tree is unicentral or bicentral, call F_1 (F_2) to the set of rooted trees obtained from the unicentral (bicentral) trees in F in the way indicated before Proposition 5. Finally, join v_j to the roots of all trees in F_j, for $1 \leqslant j \leqslant 2$. We can now state the following regarding the above construction.

Proposition 7. *Given a forest F, it is possible to compute two rooted trees T_1 and T_2 in linear time such that $D(F) = \max\{D(T_1), D(T_2)\}$.*

Proof. It is well known that the centers of a tree can be computed in linear time. So, given a forest F with n vertices, the above transformation into trees T_1 and T_2 can be done in time $\mathcal{O}(n)$. Moreover, since no automorphism of F can map a unicentral tree into a bicentral tree or viceversa, nontrivial automorphisms of F induce separate nontrivial automorphisms in T_1 and T_2. Suppose that $d = \max\{D(T_1), D(T_2)\}$. Then, d labels are enough to break symmetries in both T_1 and T_2, that is, there must be two d-labelings ϕ_1, ϕ_2 such that both $\mathrm{Aut}(T_1, \phi_1)$ and $\mathrm{Aut}(T_2, \phi_2)$ are trivial. Then, one can define a distinguishing d-labeling ϕ of F: Given a vertex u of F, if $u \in T_1$, set $\phi(u) = \varphi_1(u)$ and, otherwise, set $\phi(u) = \varphi_2(u)$. In case $\mathrm{Aut}(F, \phi)$ was nontrivial, then one of $\mathrm{Aut}(T_1, \varphi_1)$ or $\mathrm{Aut}(T_2, \varphi_2)$ would be nontrivial. Thus, $\mathrm{Aut}(F, \phi)$ must be trivial, and $D(F) \leqslant d$. But we can discard the case when $D(F) = d' < d$ since it would make it possible to use the distinguishing d'-labeling of F to define a distinguishing d'-labeling of T_1 and T_2, contradicting the definition of d as the maximum between $D(T_1)$ and $D(T_2)$. Therefore, $D(F) = \max\{D(T_1), D(T_2)\}$. □

Now, it is clear that our algorithm DISTINGUISHING from Section 2 can be used twice in combination with Proposition 7 to yield the following result.

Corollary 8. *The distinguishing number of a forest with n vertices can be computed in $\mathcal{O}(n)$ time.*

4 Conclusions and applications

We have shown that the distinguishing number of trees and forests can be computed in linear time, improving the previously known $\mathcal{O}(n \log n)$ time algorithm. We *believe* that our algorithmic technique in Section 2 can be applied to improve by a logarithmic factor (caused by a binary search in the last step of the algorithms) the complexities of computing distinguishing numbers and distinguishing chromatic numbers of the following graph classes: (1) *the distinguishing number* of (i) planar graphs computed by Arvind et al. [4, 5] and (ii) interval graphs computed by Cheng [11]; (2) *the distinguishing chromatic number* (due to Collins and Trenk [12], see also [13]) of: (i) trees computed by Cheng [11] and (ii) interval graphs computed by Cheng [11].

References

[1] M. O. Albertson. Distinguishing cartesian powers of graphs. *Electron. J. Combin.*, Vol. 12, N17, 2005.

[2] M. O. Albertson and K. Collins. Symmetry breaking in graphs. *Electron. J. Combin.*, Vol. 3, R18, 1996.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley Pub. Co. 1974.

[4] V. Arvind and N. R. Devanur. Symmetry breaking in trees and planar graphs by vertex coloring. *Proceedings of the Nordic Combinatorial Conference*, 2004.

[5] V. Arvind, C. T. Cheng, and N. R. Devanur. On computing the distinguishing numbers of planar graphs and beyond: a counting approach. *SIAM Journal on Discrete Mathematics*, 22:4, pp. 1297–1324, 2008.

[6] B. Bogstad and L. J. Cowen. The distinguishing number of the hypercube. *Discrete Math.*, Vol. 283, No. 1-3, pp. 29–35, 2004.

[7] M. Chan. The distinguishing number of the augmented cube and hypercube powers. *Discrete Math.*, Vol. 308, pp. 2330–2336, 2008.

[8] M. Chan. The distinguishing number of the direct product and wreath product action. *Journal of Algebraic Combinatorics*, Vol. 24, pp. 331–345, 2006.

[9] M. Chan. The maximum distinguishing number of a group. *Electron. J. Combin.*, Vol. 13, R70, 2006.

[10] C. Cheng. On computing the distinguishing numbers of trees and forests. *Electron. J. Combin.*, Vol. 13, R11, 2006.

[11] C. Cheng. On computing the distinguishing and distinguishing chromatic numbers of interval graphs and other results. *Discrete Math.*, 309:16, pp. 5169–5182, 2009.

[12] K. L. Collins and A. N. Trenk. The distinguishing chromatic number. *Electron. J. Combin.*, Vol. 13, R16, 2006.

[13] E. M. Eschen, C. T. Hoàng, R. Sritharan, and L. Stewart. On the complexity of deciding whether the distinguishing chromatic number of a graph is at most two. *Discrete Math.*, Vol. 311, Issue 6, pp. 431–434, 2011.

[14] W. Imrich and S. Klavžar. Distinguishing cartesian powers of graphs. *Journal of Graph Theory*, Vol. 53, Issue 3, pp. 250–260, 2006.

[15] W. Imrich, J. Jerebicb, and S. Klavžar. The distinguishing number of cartesian products of complete graphs. *European J. Combin.*, Vol. 29, Issue 4, pp. 922–929, 2008.

[16] S. Klavžar and X. Zhu. Cartesian powers of graphs can be distinguished by two labels. *European J. Combin.*, Vol. 28, pp. 303–310, 2007.

[17] A. Russell and R. Sundaram. A note on the asymptotics and computational complexity of graph distinguishability. *Electron. J. Combin.*, Vol. 5, R23, 1998.

[18] J. Tymoczko. Distinguishing numbers for graphs and groups. *Electron. J. Combin.*, Vol. 11(1), R63, 2004.

[19] D. B. West. *Introduction to graph theory.* Prentice Hall, 1996.