Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Discrete Mathematics 28 (2007) 153–160
Electronic Notes in DISCRETE MATHEMATICS
www.elsevier.com/locate/endm

Short Labels by Traversal and Jumping

Nicolas Bonichon, Cyril Gavoille, and Arnaud Labourel [1,2]

LaBRI
University Bordeaux 1
Bordeaux, France

Abstract

In this paper, we propose an efficient implicit representation of caterpillar and bounded degree trees of n vertices. Our scheme, called *Traversal & Jumping*, assigns to the n vertices of any bounded degree tree distinct binary labels of $\log_2 n + O(1)$ bits in $O(n)$ time such that we can compute adjacency between two vertices only from their labels. We use our result to improve previous known upper bound for size of labels of implicit representation of outerplanar graphs (respectively planar graphs) to $2\log_2 n$ (respectively $(3\log_2 n)$.

Keywords: graphs, trees, adjacency labelling, induced-universal.

1 Introduction

The two basic ways of representing a graph are adjacency matrices and adjacency lists. The latter representation is space efficient for sparse graphs, but adjacency queries require searching in the list, whereas matrices allow fast queries to the price of a super-linear space. Another technique, called *implicit representation* or *adjacency labeling scheme*, consists in assigning labels to each vertex such that adjacency queries can be computed alone from the labels of the two involved vertices without any extra information source. The goal is to minimize the maximum length of a label associated with a vertex while keeping fast adjacency queries.

[1] Email: bonichon@labri.fr, gavoille@labri.fr, labourel@labri.fr
[2] The three authors are supported by the project "GeoComp" of the ACI Masses de Données

1571-0653/$ – see front matter
doi:10.1016/j.endm.2007.01.022

Adjacency labeling schemes, introduced by [Bre66,BF67], have been investigated by [KNR88,KNR92]. They construct for several families of graphs adjacency labeling schemes with $O(\log n)$-bit labels. In particular, for trees the scheme consists in: 1) choosing an arbitrary prelabeling of the n vertices, a permutation of $\{1,\ldots,n\}$; 2) choosing a root; and 3) setting the label of a vertex to be the pair formed by its prelabel and the prelabel of its parent. The adjacency test checks whether the prelabel for one vertex equals the parent prelabel of the other vertex. Such labels are of $2\lceil \log n \rceil$ bits [3], whereas $\lceil \log n \rceil$ bits are clearly necessary since labels must be different.

Improving the label length of this straightforward scheme is not an easy task. It has been however improved in a non trivial way by [AKM01] to $1.5 \log n + O(\log \log n)$ bits, and more recently to $\log n + O(\log^* n)$ bits [4] [AR02], leaving open the question of whether trees enjoy a labeling scheme with $\log n + O(1)$ bit labels.

An important observation of [KNR88] is that, every family $\mathcal{F}$ of graphs supporting an adjacency labeling scheme with labels of at most $c \log n$ bits, has a graph $\mathcal{U}_n(\mathcal{F})$ with at most n^c vertices such that every graph of $\mathcal{F}$ with n vertices is an induced subgraph of $\mathcal{U}_n(\mathcal{F})$. The graph $\mathcal{U}_n(\mathcal{F})$ is called an induced-universal graph for $\mathcal{F}$.

1.1 Related work

Motivated by applications in XML search engines, and distributed applications as peer-to-peer networks or network routing, several other distributed data-structures with optimal $O(\log n)$-bit labels, have been developed.

For instance, routing in trees [FG01,TZ01], near-shortest path routing in specific networks [BG05,DL02,DL04], distance queries for interval, circular-arc, and permutation graphs [BG05,GP03a], etc. have $O(\log n)$-bit distributed data-structures. And, specifically for several queries on trees, we have: nearest common ancestor [AGKR04] with $O(\log n)$-bit labels, ancestry [AAK+05] with $\log n + O(\sqrt{\log n})$ bit labels, and small distance queries and other related functions with $\log n + \Theta(\log \log n)$ bit labels [KM01,ABR05]. Interestingly, it is shown in [ABR05] that for sibling queries in trees of maximum degree Δ, $\log n + \Theta(\log \log \Delta)$ bit labels are necessary and sufficient. A survey on labeling schemes can be founded in [GP03b]. All these schemes achieve labeling of

[3] All the logarithms are in base two.

[4] $\log^* n$ denotes the number of times log should be iterated to get a constant.

length [5] $\log n + \omega(1)$.

To our best knowledge, for reasonably large families of graphs, no distributed data-structure is known to have an optimal label size up to an additive constant. In particular, for adjacency queries in trees, the current lower bound is $\log n$ and the upper bound is $\log n + O(\log^* n)$ [AR02]. This latter scheme, based on a recursive decomposition of the tree in $\Theta(\log^* n)$ levels, has adjacency query time of $\Omega(\log^* n)$.

1.2 Our contributions

In this paper we present adjacency labeling schemes for caterpillars (i.e., a tree whose nonleaf vertices induce a path), and bounded degree trees with n vertices. Both schemes assign distinct labels of $\log n + O(1)$ bits, and support constant time adjacency queries. Moreover, all the labels can be constructed in $O(n)$ time. We observe that the recursive scheme of [AR02] for general trees does not simplify for caterpillars or bounded degree trees. The worst-case label length remains $\log n + O(\log^* n)$ and the adjacency query time $\Omega(\log^* n)$. From [KNR88], this implies that caterpillar and bounded degree trees have induced-universal graphs of $O(n)$ vertices. Finally, we use our result on bounded degree trees and special edge decomposition of planar and outerplanar graphs

As far as we know, this is the first $\log n + O(1)$ bit adjacency labeling supporting constant query time for a family of trees including trees with an arbitrary numbers of arbitrary degree vertices (caterpillars). The technique, called *Traversal & Jumping*, is interesting on its own, and we believe that it might be extended to larger families of graphs, and to other queries.

1.3 Outline of techniques

Roughly speaking, the Traversal & Jumping technique consists in:

(i) Selecting a suitable traversal of the tree (or of the graph);
(ii) Associating with each vertex x some information $C(x)$;
(iii) Performing the traversal and assign the labels with increasing but non necessarily consecutive numbers to the vertices.

Intuitively, the adjacency test between x and y is done on the basis of $C(x)$ and $C(y)$. Actually, the jumps achieved in Step 3 are done by selecting an

[5] $f(n) = \omega(g(n))$ if and only if $g(n) = o(f(n))$.

interval associated with each vertex in which its label must be. It is important to note that the intervals are ordered in the same way as the corresponding vertices in the traversal. Moreover, all vertex intervals must be disjoint. The position of the label of x in its interval is tuned in order to encode $C(x)$ in the label in a self-extracting way. In general, the information $C(x)$ determines the intervals length of all the neighbours of x which are after in the traversal.

The main difficulty is to design the minimal information $C(x)$ and to tune the jumps, i.e., the interval length. The maximum label length is simply determined by the value of the last label assigned during the traversal.

This technique fundamentally differs from previous schemes, in which a label is essentially viewed as a unique prelabel of $\lceil \log n \rceil$ bits plus some small extra fields, inevitably leading to labels of $\log n + \omega(1)$ bits. On the contrary, Traversal & Jumping abandons this representation, and uses the full range of values $[0, O(n)]$ to get labels of length $\log n + O(1)$.

Section 2 presents the scheme for caterpillars, Section 3 for bounded degree trees, planar and outerplanar graphs. We propose further works in Section 4.

2 Caterpillars

A leaf is a vertex of degree one, and an inner vertex is a nonleaf vertex. A tree is a *caterpillar* if the subgraph induced by its inner vertices is a path.

Theorem 2.1 *The family of caterpillars with n vertices enjoys an adjacency labeling scheme with labels of length at most $\lceil \log n \rceil + 6$ bits, supporting constant time adjacency query. Moreover, all the labels can be constructed in $O(n)$ time.*

Consider a caterpillar G of n vertices. We denote by $X = \{x_1, \ldots, x_k\}$ the inner vertices of G (ordered along the path). For every i, let $Y_i = \{y_{i,1}, \ldots, y_{i,d_i}\}$ be the set of leaves attached to x_i, with $d_i = 0$ if $Y_i = \varnothing$.

The traversal used in our scheme is a prefix traversal of the caterpillar rooted at x_1 where the vertices of Y_i are traversed before the vertex x_{i+1}. According to this traversal, the inner vertex x_i stores necessary information to determine the adjacency with the vertices of $Y_i \cup \{x_{i+1}\}$. The leaves do not store any specific information in their label.

With each inner vertex x_i, we associate an interval of length p_i, for some suitable integer p_i, in which its label $\ell(x_i)$ must be. For some technical reasons, impose that $p_i = 2^{t_i+3}$ with t_i is an integer $\geqslant 0$. With the set of the labels of

Y_i we associate an interval of same length: $(\ell(x_i), \ell(x_i) + p_i]$. In this interval $\ell(y_{i,j}) = \ell(x_i) + j$. Finally, the interval associated with vertex x_{i+1} is $(\ell(x_i) + p_i, \ell(x_i) + p_i + p_{i+1}]$.

The information encoded by x_i is the ordered pair (t_i, t_{i+1}). To encode this information, we use suffix-free code, i.e., a set of word such that no words of the code is the ending of another one. A simple suffix-free code is defined by $\text{code}_0(x) = 1 \circ 0^x$, where 0^x is the binary string composed of x zeros. This code extends to more succinct codes defined recursively by $\text{code}_{i+1}(x) = \text{bin}(x) \circ \text{code}_i(|\text{bin}(x)| - 1)$ for every $i \geqslant 0$. It is easy to check that, for every $i \geqslant 0$, code_i is suffix-free. E.g., $\text{code}_0(5) = 100000$, $\text{code}_1(5) = 101\,100$, and $\text{code}_2(5) = 101\,10\,10$. The suffix code associated to ech inner vertices of the caterpillar is the following:

$$C(x_i) \;=\; \text{code}_0(t_i + 3 - |\text{code}_1(t_{i+1})|) \circ \text{code}_1(t_{i+1}) \;.$$

Three conditions on p_i (and so on t_i) have to be satisfied to ensure that the code is valid. The value p_i must be large enough to encode the information, large enough so that all the labels of the vertices of Y_i can be placed in the interval $(\ell(x_i), \ell(x_i) + p_i]$, and $p_i \geqslant 8$. The following relation ensures such conditions:

$$t_i \;=\; \max\{|\text{code}_1(t_{i+1})| - 3, \lceil \lg d_i \rceil - 3, 0\}\,, \text{ with } t_{k+1} = 0 \;.$$

So, given $\ell(x_i)$, $\ell(x_{i+1})$ is computed with $w = C(x_{i+1})$ and $z = \ell(x_i) + p_i$.

For all i, we can compute from $C(x_i)$ and $\ell(x)$ the intervals containing the leaves of x_i and its following inner vertex and so compute adjacency between vertices. Moreover, due to our encoding using log (the value t_i) instead of real values (p_i), we obtain labels of $\lceil \log n \rceil + 6$ bits.

3 Bounded degree trees and application for planar and outerplanar graphs

Using traversal and jumping, we can achieve a similar scheme for tree of bounded degree.

Theorem 3.1 *The family of binary trees with n vertices enjoys an adjacency labeling scheme with labels of length at most $\log n + O(1)$ bits, supporting constant time adjacency query. Moreover, all the labels can be constructed in $O(n)$ time.*

Using the edge decomposition of Gonçalves [Gon] for planar graph into three trees which one has degree bounded by 4, we can improve the previous upper bound known for planar graph.

Theorem 3.2 *The family of planar with n vertices enjoys an adjacency labeling scheme with labels of length at most $3\log n+O(1)$ bits, supporting constant time adjacency query. Moreover, all the labels can be constructed in $O(n)$ time.*

Similary, we can improve the previous known upper bound for outerplaar graph using the fact they can be edge decomposed into two trees which one has degree bounded by 3.

Theorem 3.3 *The family of outerplanar with n vertices enjoys an adjacency labeling scheme with labels of length at most $2\log n + O(1)$ bits, supporting constant time adjacency query. Moreover, all the labels can be constructed in $O(n)$ time.*

4 Conclusion

The unsolved *implicit graph representation conjecture* of [KNR88,KNR92] asks whether every hereditary[6] family of graphs with $2^{O(n\log n)}$ labeled graphs of n vertices enjoys a $O(\log n)$-bit adjacency labeling scheme. This is motivated by the fact that every family with at least $2^{cn\log n}$ labeled graphs of n vertices requires adjacency labels of at least $c\log n$ bits.

Our schemes suggest that, at least for trees, labels of $\log n+O(1)$ bits may be possible. Therefore, we propose to prove or to disprove the following:

Every hereditary family of graphs with at most $n!2^{O(n)} = 2^{n\log n+O(n)}$ labeled graphs of n vertices enjoys an adjacency labeling scheme with labels of $\log n + O(1)$ bits.

We observe that several well-known families of graphs are concerned by this proposition: trees, planar graphs, bounded treewidth graphs, graphs of bounded genus, graphs excluding a fixed minor (cf. [NRTW05] for counting such graphs). Proving the latter conjecture appears to be hard, e.g., the best upper bound for planar graphs is only $3\log n + O(\log^* n)$.

[6] That is a family of graphs closed under induced subgraph taking.

References

[AAK+05] Serge Abiteboul, Stephen Alstrup, Haim Kaplan, Tova Milo, and Theis Rauhe. Compact labeling schemes for ancestor queries. *SIAM Journal on Computing*, 2005.

[ABR05] Stephen Alstrup, Philip Bille, and Theis Rauhe. Labeling schemes for small distances in trees. *SIAM Journal on Discrete Mathematics*, 19(2):448–462, 2005.

[AGKR04] Stephen Alstrup, Cyril Gavoille, Haim Kaplan, and Theis Rauhe. Nearest common ancestors: A survey and a new algorithm for a distributed environment. *Theory of Computing Systems*, 37:441–456, 2004.

[AKM01] Serge Abiteboul, Haim Kaplan, and Tova Milo. Compact labeling schemes for ancestor queries. In 12^{th} *Symposium on Discrete Algorithms (SODA)*, pages 547–556. ACM-SIAM, January 2001.

[AR02] Stephen Alstrup and Theis Rauhe. Small induced-universal graphs and compact implicit graph representations. In 43^{rd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 53–62. IEEE Computer Society Press, November 2002.

[BF67] Melvin A. Breuer and Jon Folkman. An unexpected result on coding the vertices of a graph. *Journal of Mathematical Analysis and Applications*, 20:583–600, 1967.

[BG05] Fabrice Bazzaro and Cyril Gavoille. Localized and compact data-structure for comparability graphs. In 16^{th} *Annual International Symposium on Algorithms and Computation (ISAAC)*, volume 3827 of Lecture Notes in Computer Science, pages 1122–1131. Springer, December 2005.

[Bre66] Melvin A. Breuer. Coding the vertexes of a graph. *IEEE Transactions on Information Theory*, IT-12:148–153, 1966.

[DL02] Feodor F. Dragan and Irina Lomonosov. New routing schemes for interval graphs, circular-arc graphs, and permutation graphs. In 14^{th} *IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS)*, pages 78–83, November 2002.

[DL04] Feodor F. Dragan and Irina Lomonosov. On compact and efficient routing in certain graph classes. In 15^{th} *Annual International Symposium on Algorithms and Computation (ISAAC)*, volume 3341 of

Lecture Notes in Computer Science, pages 402–414. Springer, December 2004.

[FG01] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, July 2001.

[Gon] D. Gonalves. Covering planar graphs with 3 forests, one being of maximum degree 4. submited.

[GP03a] Cyril Gavoille and Christophe Paul. Optimal distance labeling schemes for interval and circular-arc graphs. In G. Di Battista and U. Zwick, editors, 11^{th} *Annual European Symposium on Algorithms (ESA)*, volume 2832 of Lecture Notes in Computer Science, pages 254–265. Springer, September 2003.

[GP03b] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Journal of Distributed Computing*, 16:111–120, May 2003. PODC 20-Year Special Issue.

[KM01] Haim Kaplan and Tova Milo. Short and simple labels for small distances and other functions. In 7^{th} *International Workshop on Algorithms and Data Structures (WADS)*, volume 2125 of Lecture Notes in Computer Science, pages 32–40. Springer, August 2001.

[KNR88] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. In 20^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 334–343. ACM Press, May 1988.

[KNR92] Sampath Kannan, Moni Naor, and Steven Rudich. Implicit representation of graphs. *SIAM Journal on Discrete Mathematics*, 5:596–603, 1992.

[NRTW05] Serguei Norine, Neil Robertson, Robin Thomas, and Paul Wollan. Proper minor-closed families are small. *Journal of Combinatorial Theory, Series B*, 2005. To appear.

[TZ01] Mikkel Thorup and Uri Zwick. Compact routing schemes. In 13^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 1–10. ACM Press, July 2001.