Saving Bits Made Easy*

Srinivasa Arikati Anil Maheshwari† Christos Zaroliagis

Max-Planck Institute für Informatik
Im Stadtwald, D-66123 Saarbrücken
Germany
E-mail : {arikati,anil,zaro}@mpi-sb.mpg.de

April 14, 1994

Abstract

Sparse graphs (e.g. trees, planar graphs, relative neighborhood graphs) are among the commonly used data-structures in computational geometry. The problem of finding a compact representation for sparse graphs such that vertex adjacency can be tested quickly is fundamental to several computational geometry and graph algorithms. We provide here simple and optimal algorithms for constructing a compact representation of $O(n)$ size for an n-vertex sparse graph such that the adjacency can be tested in $O(1)$ time. Our sequential algorithm runs in $O(n)$ time, while the parallel one runs in $O(\log n)$ time using $O(n/\log n)$ CRCW PRAM processors. Previous results for this problem are based on matroid partitioning and thus have a high complexity.

1 Introduction

Paths, trees, relative neighborhood graphs, planar graphs, visibility graphs, etc. are some of the numerous graph-theoretic data-structures frequently used by algorithms in computational geometry. For example, dual of a triangulation of a simple polygon is a tree and dual of the Voronoi diagram of a planar point set is a planar graph. A fundamental data structuring question in the design of efficient algorithms in computational geometry is how to represent such an underlying graph in memory using as little space as possible, so that, given two vertices, we can test their adjacency in $O(1)$ time. Compact representation can be used to obtain e.g. an optimal algorithm for the visibility query problem [9] stated as follows. Queries are a pair of vertices of a simple polygon and we are interested to know if the query vertices are visible. It is easy to see that the complexity of reporting queries and the cost of preprocessing depends upon the compact representation of the visibility graph of the simple polygon.

Following [6, 13], we say that a class of graphs has an *implicit representation* if there exists a constant k such that for every n-vertex graph G in the class, there is a labeling of the vertices with $k\lceil\log n\rceil$-bits each, that allows us to decide adjacency in $O(1)$ time. Implicit representation eliminates the need for an adjacency matrix. (Note that in the adjacency matrix representation of G, adjacency can be tested in $O(1)$ time, but n^2 bits are required.) Also, an adjacency list

*This work is partially supported by the EEC ESPRIT Basic Research Action No. 7141 (ALCOM II).
†On leave from CSC Group, Tata Institute of Fundamental Research, Bombay - 400 005, India

representation requires $(n+m)\lceil \log n \rceil$ bits (where m is the number of edges of G), but the test for adjacency takes $O(\log n)$ time.

The *arboricity* of a graph G is defined as $\max_J\{|E(J)|/(|V(J)|-1)\}$, where J is any subgraph of G and $|V(J)|$, $|E(J)|$, are the number of vertices and edges, respectively, of J. Graphs of bounded arboricity are called *sparse*. As observed in [6], an implicit representation can be computed by decomposing the edges of G into edge-disjoint forests, or alternatively by coloring the edges of G with k colors such that there is no monochromatic cycle. If G has this latter property, we say that it is *k-forest colorable*. It follows from a theorem of [10, 11] that if G has arboricity c then G is c-forest colorable, and consequently that G has an implicit representation of $(c+1)n\lceil \log n \rceil$ bits. In such a case, G is said to have an *optimal implicit representation*.

The known sequential and parallel algorithms for obtaining an optimal implicit representation are based on involved techniques such as Edmonds' results on matroid partitioning [1]. Also, the algorithms of Narayanan et al. [8] on matroid union and intersection result in a randomized parallel algorithm for finding a c-forest coloring of graphs with arboricity c (it runs in $O(\log^3 n)$ time using $O(n^{4.5})$ processors on a probabilistic CREW PRAM). Planar graphs, a particular case of sparse graphs with $c \leq 3$, have received a considerable amount of attention, see [2, 6, 12].

The main contribution of this paper is to provide optimal sequential and parallel algorithms for obtaining optimal implicit representations of sparse graphs; our results and their comparison with previous work are summarized in Table 1. Our results are achieved by simple and rather intuitive techniques compared with those used in [1, 2, 12].

Implicit Representation	Planar Graphs		Graphs of Bounded Arboricity c	
Number of bits	$4n\lceil \log n \rceil$	$4n\lceil \log n \rceil$	$(c+1)n\lceil \log n \rceil$	$(c+1)n\lceil \log n \rceil$
Sequential Time	$O(n)$	$O(n)$	$O(n^4)$	$O(n)$
Parallel Time	$O(\log n \log\log n)$	$O(\log n)$	—	$O(\log n)$
Number of Processors	$O(n/\log n \log\log n)$	$O(n/\log n)$	—	$O(n/\log n)$
Results achieved in:	[2, 12]	This paper	[1]	This paper

TABLE 1: Our results and their comparison with previous work. The parallel model of computation is the arbitrary CRCW PRAM [5].

Note that the results in Table 1 require a priori the knowledge of the arboricity of the input graph. Since computing the exact value of arboricity seems to be hard [8, 13], we provide here algorithms that compute good approximations for it. We also show that using the approximate value, we can still obtain an optimal implicit representation of a sparse graph.

2 Preliminaries

The following lemma is central to our discussion.

Lemma 1 *Suppose that the vertices of a graph G can be ordered as $v_1, v_2, \ldots, v_n$ such that each vertex v_i has at most k neighbors before it (i.e., among $v_1, \ldots, v_i$). Then, G is k-forest colorable.*

Proof: Induction on i. Assume that the subgraph of G induced by $v_1, \ldots, v_{i-1}$ can be colored using k colors, say integers from 1 up to k. Let the neighbors of v_i that come before it, be $u_1, \ldots, u_p$, where $p \leq k$. For each $1 \leq j \leq p$, color the edge (v_i, u_j) with color j. □

Suppose that we are given a k-forest coloring of a sparse graph G. We can obtain an optimal implicit representation of G as follows. First, give distinct labels to the vertices with integers from 1 up to n. Then, concatenate to each vertex label, the label of its parent in each of the k forests. In order to decide if two vertices are adjacent, check if one is the parent of the other in any of the k forests. Observe that in this representation we need at most $(1+d(v))\lceil \log n \rceil$ bits for each vertex v, where $d(v)$ is the total number of parents of v in the k forests. It is clear that the total number of bits thus needed is at most $(n+m)\lceil \log n \rceil$ as each edge of G is represented exactly once. Further, $(n+m)\lceil \log n \rceil \leq (c+1)n\lceil \log n \rceil$, since $m \leq c(n-1)$ in graphs with arboricity c. Notice that the number of bits is independent of the number of forests k; k affects only the query time for adjacency. It is easy to see that the above-mentioned procedure to compute an optimal implicit representation from a k-forest coloring of G can be implemented sequentially in $O(n)$ time, or in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors. Hence, for the rest of the paper, we will be concerned with the forest coloring problem.

We refer to the ordering defined in Lemma 1 as a *k-ordering* of the vertices. The following two lemmas will be used in the next section for designing sequential and parallel algorithms to compute k-orderings of sparse graphs.

Lemma 2 *Let $G = (V, E)$, $|V| = n$, be a graph with arboricity c. Then G has a vertex of degree at most $2c - 1$.*

Proof: $|E| \leq c(n-1)$ as G has arboricity c. So the sum of the degrees is at most $2c(n-1)$ and hence G must have a vertex of degree at most $2c - 1$. □

Lemma 3 *Suppose that $G = (V, E)$, $|V| = n$, is a graph with arboricity c. Let $|U|$ be the set of vertices of degree at most $2c$. Then $|U| \geq (\frac{1}{2c+1})n$.*

Proof: As before, $|E| \leq c(n-1)$. There are $n - |U|$ vertices of degree at least $2c + 1$, and summing the degrees of these vertices we get $(n - |U|)(2c + 1) \leq 2|E|$. The lemma follows by rearranging the terms. □

3 Forest Coloring With Known Arboricity

In this section we present algorithms for computing optimal implicit representations of sparse graphs. Lemma 1 implies that in order to find an optimal implicit representation of a sparse graph G, it suffices to find a k-ordering of G. A sequential algorithm for computing a k-ordering of G is given in Algorithm 1.

Theorem 1 *Let $G = (V, E)$, $|V| = n$, be a graph with bounded arboricity, say c. Then a $(2c-1)$-forest coloring of G can be computed in $O(n)$ time.*

Proof: By Lemma 2, G has a vertex of degree at most $2c - 1$; call it v_n and delete it from G. Again, G has arboricity c and hence has a vertex, say v_{n-1}, of degree at most $2c - 1$. By repeating this process, we obtain a sequence $v_1, \ldots, v_n$, which is actually a $(2c-1)$-ordering of the vertices of G. This procedure is formalized in Algorithm 1. Hence, Algorithm 1 correctly computed a $(2c-1)$-forest coloring. We now discuss the complexity of the algorithm. The work

Input: A graph $G = (V, E)$, $|V| = n$, and its arboricity c.
Output: A $(2c-1)$-forest coloring of G.
1. $Low := \{v : \text{degree of } v \text{ in } G \text{ is at most } 2c-1\}$; $i := n$.
2. **while** $Low \neq \emptyset$ **do**

   1. Pick a vertex, say u, from the set Low. **for** each neighbor $w \notin Low$ of u **do**: Decrement the degree of w by one and add w to the set Low if its degree becomes $2c-1$.

   2. $G := G - u$; $v_i := u$; $i := i - 1$.

Algorithm 1: A sequential algorithm to compute forest coloring.

done in each iteration of the while loop is bounded by the degree of the vertex u. So the total work done in the while loop is bounded by the sum of degrees, which is $O(|E|) = O(n)$, since G is sparse. Now using this ordering, we can compute (as shown in the proof of Lemma 1) a $(2c-1)$-forest coloring of G. □

A parallel algorithm to compute a k-ordering of sparse graphs is given in Algorithm 2.

Input: A graph $G = (V, E)$, $|V| = n$ and its arboricity c.
Output: A $2c$-forest coloring of G.

1. $G' := G$; $i := 1$; mark all vertices unlabeled.

2. **while** there is an unlabeled vertex **do**:

   (a) Let U be the set of vertices of G' with degree at most $2c$. **for** each $v \in U$ **do**: label$(v) = i$.

   (b) $G' := G' - U$; update the degrees of neighbors of U accordingly.

   (c) $i := i + 1$.

3. **for** each vertex v in G **do**: delete all the neighbors u from its adjacency list satisfying label$(u) <$ label(v).

4. **for** each vertex v **do**: let its neighbors be $u_1, \ldots, u_\ell$, where $\ell \leq 2c$; color the edge (v, u_i) with color i, $1 \leq i \leq \ell$.

Algorithm 2: A parallel algorithm to compute forest coloring.

Theorem 2 *Given a graph $G = (V, E)$ of bounded arboricity, say c, Algorithm 2 finds a $2c$-forest coloring of G; the algorithm runs in $O(\log n)$ time using $O(n/\log n)$ CRCW PRAM processors, where $|V| = n$.*

Proof (Sketch): The proof for correctness is straightforward: By virtue of Lemma 1, it suffices to generate a $2c$-ordering $v_1, v_2, \ldots, v_n$ of the vertices. It is easy to see that Algorithm 2 generates this ordering. Now we analyse the complexity of the algorithm. By Lemma 3, the number of iterations of the while loop is $O(\log n)$. Note that in each iteration, it is not necessary to recompute the degrees of each vertex after deleting U. Instead, it is sufficient to mark in G' which vertices have degree at most $2c$. This can be done as follows. For every $v \in G'$, assign one

processor to every vertex u in its adjacency list such that u has not been labeled yet. Then all such processors p_i iterate the following two steps, $2c+1$ times: (a) Every p_i writes its id, $id(p_i)$, into a specified memory location $m(v)$. (b) All p_i read the contents of $m(v)$; if $m(v) = id(p_i)$, then p_i does not participate in the next iteration. If the contents of $m(v)$ after the $2c+1$-st iteration is the same as that after the $2c$-th iteration, then the degree of v is at most $2c$. Thus the above procedure needs $O(1)$ time and performs $O(|G'|)$ work on a CRCW PRAM. It is easy to see that the rest of the steps can be done in $O(1)$ time using $O(n)$ processors. Hence, the total resource bounds are as those stated in the theorem. □

4 Approximating Arboricity

The results listed in Table 1 require a priori the knowledge of the arboricity of the input sparse graph in order to obtain its optimal implicit representation. However, computing the exact value of the arboricity seems to be difficult [11, 13]. In this section we present algorithms to compute good approximations for arboricity. The basic idea is to turn around the proof of Theorem 1.

For the rest of this section, let $G = (V, E)$, $|V| = n$, $|E| = m$, denote a graph of unknown arboricity c. Given an integer a, Algorithm 3 tries to generate a $(2a-1)$-ordering of the graph G.

Input: A graph G and an integer a.
Output: A boolean variable ans. The variable ans is set to *true* if and only if the algorithm is able to find a $(2a-1)$-ordering of G.

1. $G_n := G$, $i := n$, ans := *true*.
2. **while** $(i \geq 1)$ and (ans = *true*) **do**
   if G_i does not have a vertex of degree at most $2a-1$ **then** ans := *false*
   else
   (a) Let u be a vertex of G_i with degree at most $2a-1$. Define $G_{i-1} = G_i - u$ and update the degrees of neighbors of u accordingly.
   (b) $v_i := u$ and $i := i-1$.
   fi

Algorithm 3: A sequential algorithm to test arboricity.

Lemma 4 *Let k be the smallest value of a for which Algorithm 3 returns the value* true. *Then (i) $k \leq c$ and (ii) k can be computed in $O(m \log n)$ time.*

Proof (Sketch): By Theorem 1, Algorithm 3 returns a true value for $a = c$. So $k \leq c$. In order to estimate the complexity, observe that for a single value of a, Algorithm 3 takes $O(m)$ time. Therefore, in order to find the particular k, it suffices to perform a binary search in the range $[1, n]$. At each step we apply Algorithm 3 supplied with an appropriate value for a (as determined by the binary search). Note that the binary search will stop as soon as Algorithm 3 returns *true* for k and *false* for $k-1$. The lemma follows. □

A parallel algorithm to test arboricity is given in Algorithm 4.

Input: A graph $G = (V, E)$ and an integer a.
Output: A boolean variable ans. The variable ans is set to *true* if and only if the algorithm is able to find a $2a$-ordering of G.

1. $G' := G$; $V' := V$; ans := *true*; mark all vertices unlabeled.
2. **while** there is an unlabeled vertex **and** (ans = *true*) **do**:
   Let U be the set of vertices of G' with degree at most $2a$.
   If $|U| < (\frac{1}{2a+1})|V'|$ **then** ans := *false*
   else
   (a) mark all vertices of U as labeled.
   (b) $V' := V' - U$; $G' := G' - U$; update the degrees of neighbors of U accordingly.
   fi

Algorithm 4: A parallel algorithm to test arboricity.

Lemma 5 *Let k be the smallest value of a for which Algorithm 4 returns the value* true. *Then (i) $k \leq c$ and (ii) k can be computed in $O(\log^2 n)$ time using $O(m/\log n)$ CRCW PRAM processors.*

Proof: Omitted from this version. □

The following lemma states that our approximation value for arboricity is at most 1/2 away from the exact value.

Lemma 6 $(c/2) \leq k \leq c$.

Proof: The fact $k \leq c$ comes by Lemma 4. Algorithm 3 generates a $(2k-1)$-ordering of G, and hence G is $(2k-1)$-forest colorable by Lemma 1. Thus $c \leq 2k-1$ by the definition of c. (A similar argument holds for Algorithm 4.) The lemma follows. □

By the discussion in Section 2 and the above lemma, it is clear that Algorithms 3 and 4 can be used to compute optimal implicit representations of sparse graphs (since a $(2k-1)$-forest coloring of G results in an optimal implicit representation of G). Therefore, we summarize with the following.

Theorem 3 *Let G be a sparse graph of unknown arboricity. Then an optimal implicit representation of G can be computed in $O(n \log n)$ sequential time, or in $O(\log^2 n)$ parallel time using $O(n/\log n)$ CRCW PRAM processors.*

Conclusion. We have presented simple and optimal algorithms that compute optimal implicit representations of sparse graphs. It is known that many intersection graphs also have implicit representations [6]. The problem of characterizing the classes of graphs having compact representation is open. It will be interesting to find better approximations for arboricity of a graph than what we have presented. Although with our approximation we can compute an optimal implicit representation, our algorithms compute a number of forests which is at most twice the optimal. This is good enough for all practical purposes in which the forest coloring problem has applications; for example, in the design of fault-tolerant networks [4], study of rigidity of structures [7] and analysis of electrical networks [3]. The known algorithms for computing an optimal

forest coloring use matroid partitioning and thus have a high complexity. It is of independent interest to come up with efficient algorithms for computing an optimal forest coloring.

Acknowledgements. We are grateful to Shiva Chaudhuri for many helpful discussions and criticism, and to Kurt Mehlhorn for his encouragement.

References

[1] J. Edmonds. Minimum partition of a matroid into independent sets. *Research of the NBS*, 69B:67–72, 1965.

[2] M. Fürer, X. He, M. Kao, and B. Raghavachari. Parallel algorithms for straight-line grid embeddings of planar graphs. *ACM Symposium on Parallel Algorithms and Architectures.* 1992.

[3] M. Iri and S. Fujishige, Use of matroid theory in operating research, circuits and systems theory, *Int. J. Systems Sci.*, 12,1,1981, pp.27-54.

[4] A. Itai and M. Rodeh, The multi-tree approach to reliability in distributed networks, *Proc. 25th IEEE Symp. on FOCS*, pp.137-147, 1984.

[5] J. JáJá. *An Introduction to Parallel Algorithms.* Addison-Wesley, New York, 1992.

[6] S. Kannan, M. Naor, and S. Rudich. Implicit representation of graphs. In *Proc. 20th ACM STOC*, pp.334-343, 1988.

[7] L. Lovasz and Y. Yemini, On generic rigidity in the plane, *SIAM J. on Alg. Disc. Meth.*, 3, 1982, pp.91-98.

[8] H. Narayanan, H. Saran, and V.V. Vazirani. Randomized parallel algorithms for matroid union and intersection, with applications to arborescences, and edge-disjoint spanning trees. In *Third ACM-SIAM Symposium on Discrete Algorithms*, 1992.

[9] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[10] C. St. J. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal London Math. Soc*, 36:445–450, 1961.

[11] C. St. J. A. Nash-Williams. Decomposition of finite graphs into forests. *Journal London Math. Soc*, 39:12, 1964.

[12] W. Schnyder. Embedding planar graphs on the grid. In *First ACM-SIAM Symposium on Discrete Algorithms*, 1990.

[13] J. van Leeuwen. Graph algorithms. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, pages 525–631. Elsevier, Amsterdam, 1990.