On Diameter Verification and Boolean Matrix Multiplication

JULIEN BASCH
Department of Computer Science
Stanford University

SANJEEV KHANNA*
Department of Computer Science
Stanford University

RAJEEV MOTWANI†
Department of Computer Science
Stanford University

Abstract

We present a practical algorithm that verifies whether a graph has diameter 2 in time $O\left(n^3/\log^2 n\right)$. A slight adaptation of this algorithm yields a boolean matrix multiplication algorithm which runs in the same time bound; thereby allowing us to compute transitive closure and verify that the diameter of a graph is d, for any constant d, in $O\left(n^3/\log^2 n\right)$ time.

Keywords : Algorithms, analysis of algorithms, boolean matrix multiplication, data structures, design of algorithms, graph diameter.

1 Introduction

We are given a graph $G = (V, E)$ and we would like to verify if the diameter of G is 2. It is easy to see that the complexity of this problem is no more than $O(M(n))$, where $M(n)$ is the complexity of boolean matrix multiplication which at present stands at $O(n^{2.376})$ [4]. However, in almost all $o(n^3)$ matrix multiplication algorithms, the constants hidden in the O-notation are very high. Thus for moderate values of n, it might not be practical to use fast matrix multiplication techniques to perform this verification. Two notable exceptions are Kronrod's algorithm [2] (also known as Four Russians' Algorithm) which runs in time $O(n^3/\log n)$, and a more recent algorithm due to Atkinson and Santoro [3] which runs in $O(n/\log^{1.5} n)$ time; in both algorithms, the hidden constants are relatively small.

In this work we present a practical $O(n^3/\log^2 n)$ time algorithm for verifying that a given graph has diameter 2. An interesting extension of our approach is a boolean matrix multiplication algorithm of the same time complexity. The diameter verification algorithm can be also be extended to computing witnesses (length 2 paths) for the diameter 2 property without altering the asymptotics. We briefly indicate further extensions to verifying diameter d for any constant d, and to the dynamic maintenance of the diameter 2 property. We

*Supported by an OTL grant, NSF NYI Award CCR-9357849, and a Schlumberger Foundation Fellowship.

†Supported by Alfred P. Sloan Research Fellowship, an IBM Faculty Development Award, an OTL grant, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

assume the standard RAM model (see e.g. [1]), where operations on $\log n$ bit numbers can be performed in $O(1)$ time.

2 Diameter Two Verification for Undirected Graphs

Consider the following naive algorithm: start with a $n \times n$ matrix Z initialized to the adjacency matrix A of the given undirected graph G; scan the adjacency list of each vertex and for each pair of vertices u and v in the list, set entry $Z[u, v] = 1$. The graph G has diameter 2 if and only if, at the end of this process, there are no 0 entries in the matrix Z. The problem with this algorithm is that in the worst case it will perform $O(n^2)$ work for each adjacency list, and thus result in an $O(n^3)$ algorithm. Since only $O(n^2)$ entries need to be filled in Z, clearly it must be performing redundant work. Our algorithm constructs a data structure which identifies some redundancy patterns and thus leads to an $O(\log^2 n)$ factor improvement in the running time over the naive algorithm.

Let A be the adjacency matrix of the graph G, and $f(n)$ be a function to be determined later (of the order of $\log n$); further, define $N = 2^{f(n)}$ and $m = n/f(n)$. We adopt the convention that the row and column numbering starts at 0.

We partition the columns of A into m blocks consisting of $f(n)$ columns each; let V_i denote the set of vertices corresponding to the ith block of G. Each row in a given block consists of $f(n)$ bits and we can view these bits as the binary representation of an integer between 0 and $N - 1$. We construct a rectangular integer matrix B with n rows and m columns, where the entry $b_{r,i}$ is the integer represented by the rth row of the ith block of columns of A; $b_{r,i}$ is an encoding of the set of vertices of V_i that are directly connected to vertex r.

Let us now focus on the connections between two given sets V_i and V_j. Given a row r, the pair $(p, q) = (b_{r,i}, b_{r,j})$ encodes the set of pairs in $V_i \times V_j$ that are at distance 2 from each other, having a path of length 2 through vertex r. It can be decoded in time $O(f(n)^2)$ as follows:

```
Decode(i, j, p, q)
{   for all (s, t) ∈ {0, ..., f(n)}^2 do
        if (p_s = 1) and (q_t = 1) then
            Z[i · f(n) + s, j · f(n) + t] ← 1;
}
```

Here p_s is the sth bit of the binary representation of integer p, and the matrix Z holds the desired result.

Consider the set $X = \{(b_{r,i}, b_{r,j}) \mid r = 0 \ldots n-1\}$. This set encodes the pairs in $V_i \times V_j$ that have a path of length 2 between them through *some* vertex r. The key fact is that this set has at most N^2 elements. Thus, if this quantity is less then n, we will save time by computing this set first and then deciding each of its elements, instead of decoding each pair $(b_{r,i}, b_{r,j})$ separately.

We represent X as a boolean matrix, whose entry $X[p, q]$ is 1 if and only if $(p, q) \in X$. It can be constructed in time $O(n + N^2)$ as follows:

```
Construct_X(i, j)
{    for all (p, q) ∈ {0, ..., N − 1}^2 do X[p, q] ← 0        /*Initialize X to 0*/
     for r ← 1 to n do       /*r is the index over the rows of A*/
          X[b_{r,i}, b_{r,j}] ← 1
}
```

The contents of X can be decoded in time $O(N^2 f(n)^2)$, as follows:

```
Decode_X(i, j)
{    for all (p, q) ∈ {0, ..., N − 1}^2 do
          if X[p, q] = 1 then
          Decode(i, j, p, q)
}
```

We complete the description of the algorithm by indicating how it repeats the above steps for all pairs of blocks of A:

```
Diameter_2(A)
{    for all (i, j) ∈ {0, ..., m − 1}^2 do
          Construct_X(i, j);
          Decode_X(i, j)
}
```

The graph G has diameter 2 if and only if there are no 0 entries in the matrix Z constructed by this algorithm. This can be checked in $O(n^2)$ time. Each of the m^2 steps of this procedure is done in time $O(n + N^2 f(n)^2)$. Choosing $f(n) = 0.25 \log n$, we have $N = n^{1/4}$ and $m = 4n/\log n$, which yields the claimed running time of $O\left(n^3/\log^2 n\right)$. The auxiliary space complexity is $N^2 = \sqrt{n}$; it can be reduced to $O(n^\epsilon)$ by choosing $f(n) = (\epsilon \log n)/2$, for any $\epsilon > 0$.

2.1 Witnesses

It is desirable to be able to compute the paths of length 2 between all possible pairs of vertices, rather than merely verifying the existence of such paths as is the case for our diameter 2 verification algorithm. We refer to these length 2 paths as *witnesses* for the diameter 2 property. Obtaining a witness to the existence of a path of length 2 between a pair of vertices is easy in our setup. We need to make the following simple changes:

1. In `Construct_X()`, we replace the assignment $X[b_{r,i}, b_{r,j}] \leftarrow 1$ by $X[b_{r,i}, b_{r,j}] \leftarrow r$. This simply keeps track of the highest number vertex which results in this particular entry being set to true.
2. The procedure `Decode()` is invoked with an additional parameter r and the statement $Z[i \cdot f(n) + s, j \cdot f(n) + t] = 1$ is replaced by $Z[i \cdot f(n) + s, j \cdot f(n) + t] = r$. This indicates that r is a witness to a path of length 2 ¿From the vertex $i \cdot f(n) + s$ to $j \cdot f(n) + t$.
3. Finally, we replace the if-statement in `Decode_X()` by the following:

```
if (X[p,q] ≠ 0) then
Decode(i, j, p, q, X[p,q])
```

Thus, the matrix Z now contains witnesses to all pairs of vertices between which a path of length 2 exists.

2.2 Dynamic Variants

The above algorithms can easily be converted into partially dynamic algorithms which can be used to maintain the property of diameter equal 2 in amortized time $O(n/\log^2 n)$ per edge insertion.

3 Boolean Matrix Multiplication

We now sketch how the above algorithm can in fact be used to perform boolean matrix multiplication in time $O(n^3/\log^2 n)$.

Let A and B be the two given $n \times n$ matrices. We consider the *columns* of A and *rows* of B to be partitioned into blocks of size $f(n)$ each. Let $a_{r,i}$ denote the integer represented by the ith block of the rth column in A, and let $b_{r,j}$ denote the integer represented by the jth block of the rth row in B.

We now need a simple modification in the algorithm described in Section 2. In procedure Construct_X(), the statement $X[b_{r,i}, b_{r,j}] \leftarrow 1$ is to be replaced by $X[a_{r,i}, b_{r,j}] \leftarrow 1$. With this change, the algorithm of Section 2 computes matrix Z as the boolean product of A and B. The index r in procedure `Construct_X()` moves over the columns of matrix A and rows of matrix B. The array X is a compact representation for the set $S_{i,j}$ defined below:

$$\{(x, y) \in A_i \times B_j \mid Z[x, y] = 1\},$$

where A_i and B_j denote the set of indices forming the ith and jth blocks of A and B, respectively. It is easy to verify that the matrix Z indeed gives the boolean product of A and B.

In fact, our algorithm can easily be adapted to multiplying two matrices whose entries are bounded by some constant. In this case, we simply maintain a count at each location $X[a_{r,i}, b_{r,j}]$ in the procedure Construct_X() and use this count to suitably update the entries in Z. The modification is straightforward and we omit further details.

3.1 Applications

Using the above boolean matrix multiplication procedure, we can verify whether a given directed or undirected graph has diameter d for any constant d in $O(n^3/\log^2 n)$ and compute the transitive closure of a graph in $O(n^3/\log^2 n)$ time (for example, see [5]).

Acknowledgements

We would like to thank Jeffrey Oldham and Pierre Demerliac for useful discussions.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1975.

[2] V.L. Arlazarov, E.A. Dinic, M.A. Kronrod, and L.A. Faradzev. On economical construction of the transitive closure of a directed graph. *Dokl. Akad. Nauk SSSR*, 194:487–488 (1970, in Russian).

[3] M.D. Atkinson and N. Santoro. A practical algorithm for Boolean matrix multiplication. *Information Processing Letters*, 29(1):37–38 (1988).

[4] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280 (1990).

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.