Improved Algorithms for the All-pairs Lowest Common Ancestor Problem in Directed Acyclic Graphs[⋆]

Artur Czumaj[1] and Andrzej Lingas[2]

[1] Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102, USA. E-mail: aczumaj@acm.org

[2] Department of Computer Science, Lund University, 22100 Lund, Sweden. E-mail: Andrzej.Lingas@cs.lth.se

Abstract. We present a new algorithm for solving the *all-pairs lowest common ancestor* problem in *directed acyclic graphs* (dags). Our algorithm runs in time $O(n^{2+\lambda})$, where λ satisfies the equation $\omega(1, \lambda, 1) = 1 + 2\,\lambda$ and $\omega(1, \lambda, 1)$ is the exponent of the multiplication of an $n \times n^{\lambda}$ matrix by an $n^{\lambda} \times n$ matrix. By the currently best bounds on $\omega(1, \lambda, 1)$, the running time of our algorithm is $O(n^{2.575})$. Our algorithm improves upon the recent $O(n^{2.616})$-time algorithm by Kowaluk and Lingas (ICALP'2005) and the previous $O(n^{2.688})$-time algorithm by Bender *et al.* (SODA'2001).

Our result is obtained by using a close relationship between the all-pairs lowest common ancestor problem in dags and the problem of computing the *maximum witnesses* of Boolean matrix, as well as fast Boolean multiplications of rectangular matrices. We precise the relationship by completing the proof of $O(n^{\omega})$-time *equivalence* between the all-pairs lowest common ancestor problem and the problem of computing maximum witnesses of Boolean matrix product, where $\omega = \omega(1, 1, 1) < 2.376$.

Our additional contribution is a faster algorithm for solving the all-pairs lowest common ancestor problem in dags of small height, where the height of a dag is defined as the length of the longest path in the dag. For all dags of height at most $h \leq n^{\alpha}$, where $\alpha \approx 0.294$, our algorithm runs in time asymptotically the same as that of multiplying two $n \times n$ matrices, that is, $O(n^{\omega})$; we also prove that this running time is optimal even for dags of height 1. For dags with height $h > n^{\alpha}$, the running time of our algorithm is at most $O(n^{\omega} \cdot h^{0.468})$. This algorithm is faster than our algorithm for arbitrary dags for all values of $h \leq n^{0.42}$.

1 Introduction

In this paper we investigate the classical algorithmic problem of finding a lowest common ancestor (LCA) for any pair of vertices in a directed acyclic graph. This problem has been very extensively studied in the past in the context of (rooted) trees (see, e.g., [3,12,16,18]), where it appears naturally in various settings and where it found many applications in design of efficient algorithms and data structures (e.g., for the problem of computing maximum matching in graphs and for various string problems). Harel and Tarjan [12] were the first who showed that in rooted trees, the LCA queries can be

[⋆] Research supported in part by NSF ITR grant CCR-0313219 and by VR grant 621-2002-4049.

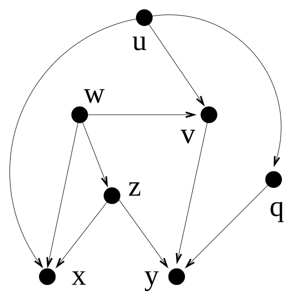

Fig. 1. A dag with 7 vertices. The LCA of vertices x and y are both, vertex u and vertex z; vertex w is a common ancestor of x and y but it is not the LCA of x and y. There is no common ancestor of vertices w and q.

answered in constant time after a linear time preprocessing of the input tree. This work has been later extended in many ways, including the simplified algorithm from [3] and the recent dynamic algorithm [6].

Recently, Bender *et al.* [5] extended these study to more general classes of graphs and initiated investigations of the LCA problem in the contexts of arbitrary directed acyclic graphs. A *lowest common ancestor* (*LCA*) of vertices u and v in a *directed acyclic graph* (dag) is an ancestor of both u and v which has no descendant that is an ancestor of u and v, see Figure 1 for example. As it has been observed in [5], the LCA queries in dags appear naturally in a variety of applications, e.g., in inheritance analysis in programming languages, analysis of genealogical data, and lattice operations in complex systems (for more details, see e.g., [10,15,17], and especially [4,5] and the references therein).

Unlike for trees, where it is well known that after linear-time preprocessing LCA queries can be answered in constant time, no such results are known for arbitrary dags. Even more, it is known that the all-pairs LCA problem in dags is not simpler than transitive closure in arbitrary directed graphs and the problem of multiplying two $n \times n$ Boolean matrices, where n is the number of vertices in the dag. Bender *et al.* [5] were the first who showed that the all-pairs LCA problem in dags can be solved in $o(n^3)$ time and gave an $O(n^{(3+\omega)/2}) = O(n^{2.688})$-time preprocessing that is sufficient to answer LCA queries in constant time (where n is the number of vertices and $\omega < 2.376$ is the exponent of the fastest matrix multiplication algorithm). This upper bound has been recently improved to $O(n^{2+\frac{1}{4-\omega}}) = O(n^{2.616})$ in [14]. An alternative $O(n\,m)$-time preprocessing, where m is the number of edges, superior for sparser dags, has been also presented in [14].

New contributions. In this paper we make further progress in designing fast algorithms for the all-pairs LCA problem in dags and we improve the aforementioned results in two ways. Our first algorithm solves the all-pairs LCA problem in an arbitrary dag on n vertices in time $O(n^{2.575})$. It uses fast rectangular matrix multiplication algorithms (in a similar flavor as in [19]) to improve upon previously existing bounds. The precise running time of our algorithm is $O(n^{2+\lambda})$, where λ satisfies $\omega(1, \lambda, 1) = 1 + 2\,\lambda$ and

$\omega(1,\lambda,1)$ is the exponent of the product of an $n \times n^{\lambda}$ matrix by an $n^{\lambda} \times n$ matrix. By the currently best bounds on $\omega(1,\lambda,1)$, the running time is $O(n^{2.575})$.

Our second algorithm is superior for shallow dags, that is for dags whose height h (the length of the longest directed path) is small. For all dags of height at most $h \leq n^{\alpha}$, where $\alpha \approx 0.294$, it runs in time asymptotically the same as that of multiplying two $n \times n$ matrices, that is, time $O(n^{\omega})$; we also prove that this running time is optimal even for dags of height 1. For dags with height $h > n^{\alpha}$, the running time of our second algorithm is at most $O(n^{\omega} \cdot h^{0.468})$.

Our results are obtained by exploring the close relationship between the all-pairs LCA problems in dags and the problems of computing *witnesses* and *maximum witnesses* of Boolean matrix product (see [1,2,11] and [14], respectively). In fact, we also complete the proof of $O(n^{\omega})$-time equivalence between the all-pairs lowest common ancestor problem and the problem of computing maximum witnesses of Boolean matrix product. This equivalence shows that the all-pairs LCA problem in dags is inherently related to fast algorithms on matrices.

Organization. Our paper is structured as follows. In the next section, we define the concepts of witnesses, maximum witnesses of Boolean matrix product, the height of a vertex in a dag, the level in a dag, and the height of a dag. In Section 3, we demonstrate the relationships between the problems of computing common ancestors and LCA in dags and those of finding witnesses and maximum witnesses for Boolean matrix product. In Section 4, we present our $O(n^{2.575})$-time method for the maximum witness problem and the all-pairs LCA problem for dags. In Section 5, we derive a more efficient solution to the all-pairs LCA problem in dags of small height.

2 Preliminaries

We shall frequently refer to fast algorithms for matrix multiplication and related problems, as we describe below.

Let ω denote the exponent of square matrix multiplication, that is, the smallest constant for which the product of two $n \times n$ matrices can be performed in $O(n^{\omega})$ time. The best asymptotic upper bound on ω currently known is $\omega < 2.376$, by Coppersmith and Winograd [8].

The following fact that relates graph problems to fast matrix multiplication is folklore.

Fact 1 *The transitive closure of any directed graph with n vertices, in particular a dag, can be computed in time $O(n^{\omega})$.*

We also need to define the concept of witnesses and maximum witness in Boolean matrix multiplication.

Definition 1. *If an entry $C[i,j]$ of the Boolean product of two Boolean matrices A and B is equal to 1 then any index k such that $A[i,k]$ and $B[k,j]$ are equal to 1 is a* ***witness*** *for $C[i,j]$. If k is the largest possible witness for $C[i,j]$ then it is called the* ***maximum witness*** *for $C[i,j]$.*

The following fact is due to Alon and Naor [1].

Fact 2 *The witnesses of the Boolean product of two $n \times n$ Boolean matrices can be computed in time $\widetilde{O}(n^{\omega})$.* [3]

As it has been observed in [5], the concept of the height of a vertex and of a dag plays an important role in the study of LCA in dags.

Definition 2. *For a vertex v in a dag, the **height** of v is the maximum length of a path from a source (vertex of indegree zero) of the dag to v. The **level** i of a dag is the set of all its vertices of height i. The **height of a dag** is the maximum height of its vertices, or equivalently, the number of its non-empty levels decreased by one.*

By using standard single-source shortest path algorithm in dags, we can easily obtain the following.

Lemma 1. *For a dag G with n vertices and m edges, one can compute the partition of the vertices of G into the levels and the height of G in time $O(n+m)$.*

Proof. Let us add a new vertex s to G and for every zero-indegree vertex v in G, add directed edge (s, v). Next, we assign weights to all edges in the new dag. Every new edge of the form (s, v) will have weight 0 and every other edge will have weight -1. Then, it is easy to see that the length of the shortest path from s to a vertex u is equal to *minus* the height of u. Therefore, we can find the height of all vertices in G by applying a single-source shortest path algorithm in dags. Since it is known that a variant of the Bellman-Ford algorithm in dags solves the single-source shortest path problem in time $O(n+m)$ (see, e.g., Chapter 24.2 in [9]), the claim follows. $\square$

3 Common ancestors versus Boolean matrix product witnesses

In this section we discuss the relationship between the problems of finding common ancestors and LCA for all pairs of vertices in a dag and the problems of computing witnesses and maximum witnesses of Boolean product of two Boolean $n \times n$ matrices. The relationship between these concepts has been first observed in [14], but here we make it more explicit and prove that some of these problems are equivalent with respect to asymptotic time complexity.

We begin with a reduction of the problem of computing witnesses to the problem of computing all-pairs (non-necessarily lowest) common ancestors in a dag of height 1.

Theorem 1. *The problem of computing witnesses of Boolean product of two Boolean $n \times n$ matrices can be reduced to the problem of computing all-pairs (non-necessarily lowest) common ancestors in a dag with $3\,n$ vertices and height 1 in time $O(n^2)$.*

[3] Throughout the paper, the notation $\widetilde{O}(f(n))$ stands for $O(f(n)\,\log^c n)$ for some positive constant c.

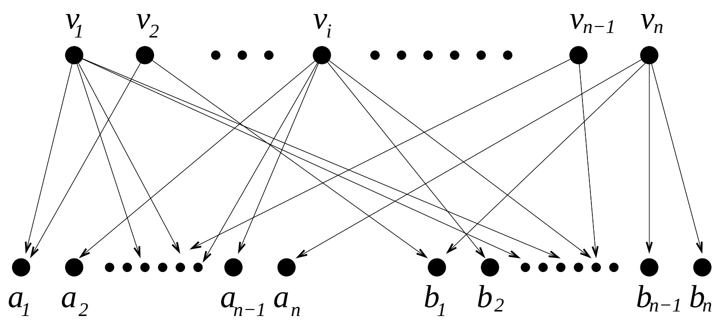

Fig. 2. A scheme of the construction used in the proof of Theorem 1.

Proof. Let A and B be two Boolean $n \times n$ matrices. Construct a two-level dag G with vertices $v_1, v_2, \ldots, v_n$ on the zero level and vertices $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ on the first level. Create an edge (v_k, a_i) if and only if $A[i, k] = 1$. Analogously, form an edge (v_k, b_j) if and only if $B[k, j] = 1$.

Let C be the Boolean product of matrices A and B. By the definition of G, k is a witness of an entry $C[i, j]$ if and only if v_k is a common ancestor of the vertices a_i and b_j. $\square$

Note that the all-pairs common ancestor problem (but not the *least* common ancestor one) for an arbitrary dag can be solved in time $O(n^\omega)$ by computing the transitive closure of the dag, using Fact 1, and then computing the witnesses of the Boolean product of the transitive closure matrix and its transpose using Fact 2. This running time is asymptotically optimal, since Theorem 1 implies that the problem of computing all-pairs LCA in a dag with n vertices requires time $\Omega(n^\omega)$, even for dags of height 1.

By slightly modifying the dag constructed in the proof of Theorem 1, we obtain the following theorem that relates computing *maximum witnesses* to computing all-pairs LCA in a dag.

Theorem 2. *The problem of computing maximum witnesses of Boolean product of two Boolean $n \times n$ matrices can be reduced to the problem of computing all-pairs LCA in a dag on $3\,n$ vertices in time $O(n^2)$.*

Proof. Assume the notation from the proof of Theorem 1. Additionally link the vertices $v_1, v_2, \ldots, v_n$ by the edges $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$ in the dag G. (For an example, see Figure 3.)

By the construction of the extended dag, k is a maximum witness of an entry $C[i, j]$ if and only if v_k is a lowest common ancestor of the vertices a_i and b_j. $\square$

In [14], an $O(n^\omega)$-time reduction of the all-pairs LCA problem for a dag on n vertices to the problem of computing maximum witnesses of the Boolean product of two $n \times n$ Boolean matrices has been also given. We present this result here, and for the sake of completeness, we sketch its proof.

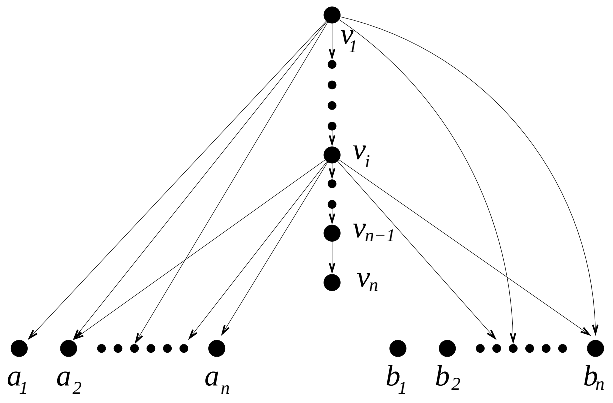

Fig. 3. A scheme of the construction used in the proof of Theorem 2.

Fact 3 *The problem of computing all-pairs LCA in a dag on n vertices can be reduced to the problem of computing maximum witnesses of Boolean product of two Boolean $n \times n$ matrices in time $O(n^{\omega})$.*

Sketch of the proof. Compute the transitive closure of the input dag. Number the vertices of the dag by their rank in the topological ordering of the transitive closure. Observe that a common ancestor of vertices u and v having the largest number among common ancestors of u and v is an LCA of u and v. Form the ancestor matrix A such that in its ith row there is 1 on the kth position if and only if the kth vertex is an ancestor of the ith vertex. Compute the maximum witnesses of the Boolean product of the matrix A and its transpose A^T. By the observation, for any (i, j) entry of the product matrix, if the entry is positive then its maximum witness is the number of an LCA of the ith and jth vertex. □

By combining Theorem 2 with Fact 3, we obtain the following equivalence corollary.

Corollary 1 *The problem of computing all-pairs LCA in a dag on n vertices and the problem of computing maximum witnesses of Boolean product of two Boolean $n \times n$ matrices are $O(n^{\omega})$-time equivalent.*

Bender *et al.* [5] showed that the problem of computing all-pairs LCA in a dag with n vertices is not easier than that of computing transitive closure in a directed graph with n vertices. This implies the lower bound of $\Omega(n^{\omega})$ for the all-pairs LCA in a dag with n vertices. However, our result on the relationship between the all-pairs LCA problem in dags and that of computing maximum witnesses of Boolean product of two Boolean matrices shows that these two problems have asymptotically identical complexity, and thus it yields a stronger relationship. Furthermore, our result (Theorem 1) shows that the problem of computing all-pairs LCA in a dag with n vertices requires time $\Omega(n^{\omega})$ even for dags of height 1.

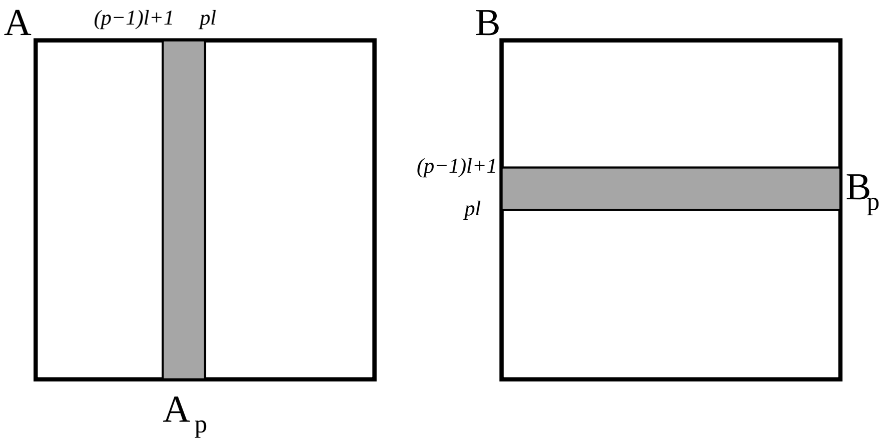

Fig. 4. Rectangular matrices A_p and B_p.

4 $O(n^{2.575})$-time method for maximum witnesses and the all-pairs LCA problem in dags

By Corollary 1, it is sufficient to compute maximum witnesses of Boolean product of two Boolean $n \times n$ matrices in order to solve the all-pairs LCA problem in dags. In this section, we present a new algorithm for the maximum witness problem using fast rectangular matrix multiplication.

Let ℓ be a positive integer smaller than n. Partition the matrix A into $n \times \ell$ sub-matrices A_p and the matrix B into $\ell \times n$ sub-matrices B_p, such that $1 \le p \le n/\ell$, and the sub-matrix A_p covers the columns $(p-1)\,\ell + 1$ through $p\,\ell$ of A whereas the sub-matrix B_p covers the rows $(p-1)\,\ell + 1$ through $p\,\ell$ of B. (For an example, see Figure 4.)

For $p = 1, \ldots, n/\ell$, let C_p be the Boolean product of A_p and B_p. Then, $C_p[i,j] > 0$ if and only if there is an index k, $(p-1)\,\ell < k \le p\,\ell$, such that $A[i,k] = B[k,j] = 1$. Therefore the following claim follows.

Claim 1 *Suppose that a $C[i,j]$ entry of the Boolean product C of A and B is positive. Let p' be the maximum value of p such that $C_{p'}[i,j] > 0$. The maximum witness of $C[i,j]$ belongs to the interval $[(p'-1)\,\ell + 1, p'\,\ell]$.*

By this claim, after computing all the products $C_p = A_p \cdot B_p$, $1 \le p \le n/\ell$, we need $O(n/\ell + \ell)$ time per positive entry of C to find the maximum witness: $O(n/\ell)$ time to determine p' and then $O(\ell)$ time to locate the maximum witness.

Let $\omega(1,r,1)$ denote the exponent of the multiplication of an $n \times n^r$ matrix by an $n^r \times n$ matrix. It follows that the total time taken by our method for maximum witnesses is

$$O((n/\ell) \cdot n^{\omega(1,\log_n \ell,1)} + n^3/\ell + n^2\,\ell) \ .$$

By setting r to $\log_n \ell$ our upper bound transforms to $O(n^{1-r+\omega(1,r,1)} + n^{3-r} + n^{2+r})$. Note that by assuming $r \geq \frac{1}{2}$, we can get rid of the additive n^{3-r} term. Hence, by solving the equation $1 - \lambda + \omega(1, \lambda, 1) = 2 + \lambda$ implying $\lambda \geq \frac{1}{2}$ by $\omega(1, \lambda, 1) \geq 2$, we obtain our main result.

Theorem 3. *Let λ be such that $\omega(1, \lambda, 1) = 1 + 2\,\lambda$. The maximum witnesses for all positive entries of the Boolean product of two $n \times n$ Boolean matrices can be computed in time $O(n^{2+\lambda})$.*

Coppersmith [7] and Huang and Pan [13] proved the following fact.

Fact 4 [7,13] *Let $\omega = \omega(1, 1, 1) < 2.376$ and let $\alpha = sup\{0 \leq r \leq 1 : \omega(1, r, 1) = 2 + o(1)\} > 0.294$. Then $\omega(1, r, 1) \leq \beta(r)$, where $\beta(r) = 2 + o(1)$ for $r \in [0, \alpha]$ and $\beta(r) = 2 + \frac{\omega-2}{1-\alpha}(r - \alpha) + o(1)$ for $r \in [\alpha, 1]$.*

Note that by Fact 4, the solution λ of the equation $\omega(1, \lambda, 1) = 1 + 2\,\lambda$ is satisfied by $\lambda = \frac{1-\alpha\,(\omega-1)}{2\,(1-\alpha)-(\omega-2)} + o(1) < 0.575$. Therefore, we obtain the following concrete corollary.

Corollary 2 *The maximum witnesses for all positive entries of the Boolean product of two $n \times n$ Boolean matrices can be computed in time $O(n^{2.575})$.*

By combining Fact 3 with Theorem 3 and Corollary 2, we obtain also an $O(n^{2.575})$-time solution to the all-pairs LCA problem in dags.

Theorem 4. *Let $\omega(1, \lambda, 1) = 1 + 2\,\lambda$. The all-pairs LCA problem for an arbitrary dag with n vertices can be solved in time $O(n^{2+\lambda})$, which is upper bounded by $O(n^{2.575})$.*

5 All-pairs LCA in dags of bounded height

In this section we describe an algorithm for solving the all-pairs LCA problem for an arbitrary dag G of height bounded by h. The algorithm has a similar flavor as that discussed in the previous section, but now we shall additionally use the fact that the height is bounded to speed up the process.

First, we compute the transitive closure of G and create the ancestor matrix A similarly as in the proof of Fact 3. Next, using Lemma 1, we partition G into $h + 1$ levels and extend the partial order induced by this partition to a linear order and number the vertices according to the linear order so the numbering is increasing with respect to vertex height. These steps can be performed in time $O(n^\omega)$.

Observe that the numbering naturally decomposes into $h+1$ continuous intervals in one-to-one correspondence with the levels of G. Our approach relies on the following generalization of the observation that the common ancestor of vertices u and v that has the highest number (in the topological ordering) is a lowest common ancestor of u and v.

Claim 2 [5] *Any common ancestor of vertices u and v which is of highest level among common ancestors of u and v is a lowest common ancestor of u and v.*

Claim 2 implies directly that the maximum witnesses of the product of the Boolean matrix of ancestors A and its transpose A^T yield the solution to the all-pairs LCA problem for a dag. (If $C = A \cdot A^T$ and k is the maximum witness of $C[i,j]$ then k is an LCA of (i,j).) However, since it is expensive to compute maximum witnesses, we modify this construction and reduce the problem to that of computing *witnesses* (instead of *maximum* witnesses).

Relying on Claim 2 and the bounded height h, we decompose A and its transpose A^T into $h+1$ rectangular sub-matrices in one-to-one correspondence with the level intervals and compute witnesses for the products of the pairs of sub-matrices in A and A^T corresponding to the same level interval. Now, similarly as in the previous section, we observe that if vertices i and j have a common ancestor at level ℓ, then if we multiply the sub-matrix of A corresponding to level ℓ by the sub-matrix of A^T corresponding to level ℓ, then the resulting matrix C_ℓ will have $C_\ell[i,j] > 0$ and the witness for $C_\ell[i,j] > 0$ will be a vertex from level ℓ that is an ancestor of both i and j.

Let us estimate the time taken by finding the witnesses for the $h+1$ products of sub-matrices of A and A^T. Recall the definition of function β in Fact 4 due to Coppersmith, and Huang and Pan. By using the straightforward reduction of Boolean matrix multiplication to the arithmetic one and by generalizing the derandomization method of Alon and Naor [1] for witnesses of Boolean matrix multiplication to include rectangular matrices, we obtain the following lemma.

Lemma 2. *The witnesses of the Boolean product of two Boolean matrices of sizes $n \times n^r$ and $n^r \times n$ can be computed in time $\widetilde{O}(n^{\beta(r)})$.*

For $k = 0, \ldots, h$, let ℓ_k be the number of vertices on level k in the dag G. By Lemma 2, the total time taken by computing the witnesses for $h+1$ sub-matrix products is

$$\widetilde{O}\left(\sum_{k=0}^{h} n^{\beta(\log_n \ell_k)}\right) \quad . \tag{1}$$

Finally, once we determined witnesses for every pair of vertices numbered i and j and for every level ℓ, it remains to find for each pair i and j, the largest witness among the witnesses for the pair i,j in the products of the sub-matrices. It takes $O(h)$ time per pair i,j and hence, $O(n^2\,h)$ time in total.

Next, by straightforward calculations, Jensen inequality implies that the value of (1) is maximized if the levels are of equal size $n/(h+1)$. (Indeed, we observe that the function $n^{\beta(x)-2}$ is concave and therefore $\sum_{k=0}^{h} n^{\beta(\log_n \ell_k)} = n^2 \sum_{k=0}^{h} n^{\beta(\log_n \ell_k)-2} \leq n^2\,(h+1)\,n^{\beta(\log_n n/(h+1))-2} = (h+1)\,n^{\beta(\log_n n/(h+1))}$.) Hence, we obtain the following theorem.

Theorem 5. *The all-pairs LCA problem for an arbitrary dag with n vertices and height n^q can be solved in time $\widetilde{O}(n^\omega + n^{q+\beta(1-q)})$.*

In particular, for dags of height at most $n^\alpha \approx n^{0.294}$ the running time is $\widetilde{O}(n^\omega)$.

For larger values of the height n^q, the running time of this algorithm is

$$\widetilde{O}\left(n^{\omega+q(1-\frac{\omega-2}{1-\alpha})+o(1)}\right) \approx O\left(n^{2.376+0.468\,q}\right) \quad .$$

Remark 1 *Note that for all values of $q \leq \alpha \approx 0.294$, the running time of our algorithm from Theorem 5 is asymptotically optimal due to our result in Theorem 1. Even for larger values of q, up to $q \leq 0.42$, our algorithm from Theorem 5 is faster than that general from Theorem 4.*

6 Conclusions

We have clarified the close relationship between the problem of computing the maximum witnesses of Boolean matrix product and that of finding LCAs for all pairs of vertices in a dag and substantially improved upper time-bounds for these problems. We have also obtained better time-bounds for dags of small height. Our new upper time-bounds rely on the use of fast algorithms for rectangular matrix multiplication.

It is an intriguing open problem whether or not the complexity gap between the problems of computing witnesses and computing maximum witnesses of Boolean matrix product, or alternatively, the problems of finding common ancestors and finding LCAs in a dag, can be further decreased.

Acknowledgments

The authors are grateful to Mirosław Kowaluk and Martin Wahlén for some discussions.

References

1. N. Alon and M. Naor. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica*, 16: 434–449, 1996. A preliminary version of that paper formed a part of [2].
2. N. Alon, Z. Gali, O. Margalit, and M. Naor. Witnesses for Boolean matrix multiplication and for shortest paths. *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science (FOCS'92)*, pp. 417–426, 1992.
3. M. A. Bender and M. Farach-Colton. The LCA problem revisited. *Proceedings of the 4th Latin American Symposium on Theoretical Informatics (LATIN'00)*, pp. 88–93, 2000.
4. M. A. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2): 75–94, 2005. Full version of [3] and [5].
5. M. A. Bender, G. Pemmasani, S. Skiena and P. Sumazin. Finding least common ancestors in directed acyclic graphs. *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pp. 845–853, 2001.
6. R. Cole and R. Hariharan. Dynamic LCA queries in trees. *SIAM Journal on Computing*, 34(4): 894–923, 2005.
7. D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Symbolic Computation*, 13: 42–49, 1997.
8. D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. *Journal of Symbolic Computation*, 9: 251–290, 1990.
9. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* 2nd edition, McGraw-Hill Book Company, Boston, MA, 2001.
10. R. W. Cottingham Jr., R. M. Idury, and A. A. Shäffer. Genetic linkage computations. *American Journal of Human Genetics*, 53: 252–263, 1993.

11. Z. Galil and O. Margalit. Witnesses for Boolean matrix multiplication and for transitive closure. *Journal of Complexity*, 9: 201–221, 1993.
12. D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2): 338–355, 1984.
13. X. Huang and V.Y. Pan. Fast rectangular matrix multiplications and applications. *Journal of Complexity*, 14: 257–299, 1998.
14. M. Kowaluk and A. Lingas. LCA queries in directed acyclic graphs. *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, pp. 241-248, 2005.
15. M. Nykänen and E. Ukkonen. Finding lowest common ancestors in arbitrarily directed trees. *Information Processing Letters*, 50(6): 307–310, 1994.
16. B. Schieber and U. Vishkin. On finding lowest common ancestors: Simplification and parallelization. *SIAM Journal on Computing*, 17: 1253–1262, 1988.
17. A. A. Shäffer, S. K. Gupta, K. Shriram, and R. W. Cottingham Jr. Avoiding recomputation in linkage analysis. *Human Heredity*, 44: 225–237, 1994.
18. R.E. Tarjan. Applications of path compression on balanced trees. *Journal of the ACM*, 26(4): 690–715, 1979.
19. U. Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM*, 49(3): 289–317, 2002.