T U M

I N S T I T U T F Ü R I N F O R M A T I K

A Path Cover Technique for LCAs in Dags

Andrzej Lingas   Miroslaw Kowaluk
Johannes Nowak

TUM-I0809
April 08

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-04-I0809-0/1.-FI
Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©2008

Druck: Institut für Informatik der
Technischen Universität München

A Path Cover Technique for LCAs in Dags

Mirosław Kowaluk [*] Andrzej Lingas [†] Johannes Nowak [‡]

Abstract

We develop a path cover technique to solve lowest common ancestor (LCA for short) problems in a directed acyclic graph (dag).

Our method yields improved upper bounds for two recently studied problem variants, computing one (representative) LCA for all pairs of vertices and computing all LCAs for all pairs of vertices. The bounds are expressed in terms of the number n of vertices and the so called width $\mathrm{w(G)}$ of the input dag G. For the first problem we achieve $\widetilde{O}(n^2\mathrm{w(G)})$ time which improves the upper bound of [?] for dags with $\mathrm{w(G)} = O(n^{0.376-\delta})$ for a constant $\delta > 0$. For the second problem our $\widetilde{O}(n^2\mathrm{w(G)}^2)$ upper time bound subsumes the $O(n^{3.334})$ bound established in [?] for $\mathrm{w(G)} = O(n^{0.667-\delta})$.

As a second major result we show how to combine the path cover technique with LCA solutions for dags with small depth [?]. Our algorithm attains the best known upper time bound for this problem of $O(n^{2.575})$. However, most notably, the algorithm performs better on a vast amount of input dags, i.e., dags that do not have an almost linear-sized subdag of extremely regular structure.

Finally, we apply our technique to improve the general upper time bounds on the worst case time complexity for the problem of reporting LCAs for each triple of vertices recently established by Yuster[?] and to develop space-efficient (subquadratic) LCA algorithms.

1 Introduction

A *lowest common ancestor* (LCA) of vertices u and v in a *directed acyclic graph* (dag) is an ancestor of both u and v that has no descendant which is an ancestor of u and v, see Fig. 1 for an example. Fast algorithms for finding lowest common ancestors (LCAs) in trees and – more generally – directed acyclic graphs (dags) are indispensable computational primitives. Whereas LCA computations in trees are well studied, see, e.g., [?, ?, ?], the case of dags has been found an independent subject of research only recently, initiated by the paper of Bender et al. [?]. Due to the limited expressive power of trees they are often applicable only in restrictive or over-simplified settings. There are numerous applications for LCA queries in dags, e.g., object inheritance in programming languages, lattice operations for complex systems, lowest common ancestor queries in phylogenetic networks, or queries concerning customer-provider relationships in the *Internet*. For a more detailed description of possible applications, we refer to [?, ?]. The fact that lowest common ancestors in dags are not unique leads to a variety of problem variants [?], e.g., computing one (representative) LCA or computing all LCAs.

[*]Institute of Informatics, Warsaw University, Warsaw, Poland. Email: kowaluk@mimum.edu.pl. Research supported by the grant of the Polish Ministry of Science and Higher Education N20600432/0806 .

[†]Department of Computer Science, Lund University, 22100 Lund, Sweden. Email: Andrzej.Lingas@cs.lth.se. Research supported in part by VR grant 621-2005-4806.

[‡]Fakultät für Informatik, Technische Universität München, München, Germany. Email: nowakj@in.tum.de.

Figure 1: A dag with 7 vertices. The LCAs of vertices x and y are both vertex u and vertex z, i.e., $\mathrm{LCA}\{x,y\} = \{u,z\}$; vertex w is a common ancestor of x and y but it is not an LCA of x and y since $w \rightsquigarrow z$ and $z \in \mathrm{CA}\{x,y\}$, i.e., z is a witness, see Definition 1.

Known results on LCAs in dags. LCA algorithms have been extensively studied in the context of trees with most of the research rooted in [?, ?]. The first asymptotically optimal algorithm for the all-pairs LCA problem in trees, with linear preprocessing time and constant query time, was given in [?]. The same asymptotics was reached using a simpler and parallelizable algorithm in [?]. Recently, a reduction to range minimum queries has been used to obtain a further simplification [?].

In the more general case of dags, a pair of nodes may have more than one LCA, which leads to the distinction of *representative* versus *all* LCA problems. In early research both versions still coincide by considering dags with each pair having at most one LCA. Extending the work on LCAs in trees, in [?], an algorithm was described with linear preprocessing and constant query time for the LCA problem on arbitrarily directed trees (or, causal polytrees). Another solution was given in [?], where the representative problem in the context of object inheritance lattices was studied. The approach based on poset embeddings into Boolean lattices [?] yielded $O(n^3)$ preprocessing and $O(\log n)$ query time on lower semi-lattices.

The All-Pairs Representative LCA problem on general dags has been studied recently in [?, ?, ?, ?]. The works rely on fast matrix multiplications (currently the fastest known algorithm needs $O(n^{\omega})$ operations, with $\omega < 2.376$ [?]) to achieve $\widetilde{O}(n^{\frac{\omega+3}{2}})$ [?] [1] and $\widetilde{O}(n^{2+\frac{1}{4-\omega}})$ [?] upper bounds on the running time. The technique developed in [?] was slightly improved in [?] by applying rectangular matrix multiplication[2] [?] to achieve an upper bound for the representative LCA problem of $O(n^{2+\mu})$.

More efficient solutions have been developed for special classes of dags. For sparse dags, algorithms with running time $O(nm)$ given in [?, ?], where m is the number of edges in the input dag, improve the general upper bound. In [?] it was shown that this can be improved to $O(n\,m_{\mathrm{red}})$, where m_{red} is the number of edges in the the transitive reduction of the input dag. In [?] a more efficient solution is described for the All-Pairs Representative LCA problem in dags of small depth h. More specifically, the algorithm is shown to improve upon the general upper bound whenever $h \le n^{0.42}$. Lately, in [?], it is shown that the problem can also be solved in time $\widetilde{O}(n^2 \mathrm{w(G)} + \mathrm{n}^{\omega})$,

[1] Throughout this work, we use $\widetilde{O}(f(n))$ for $O(f(n) \cdot \mathrm{polylog}(n))$

[2] Throughout this work, $\omega(x,y,z)$ is the exponent of the algebraic matrix multiplication of a $n^x \times n^y$ with a $n^y \times n^z$ matrix. Let μ be such that $\omega(1,\mu,1) = 1 + 2\mu$ is satisfied. The fastest known algorithms for rectangular matrix multiplication imply $\mu < 0.575$ and $\omega(2,1,1) < 3.334$

where w(G) is the width of the dag G, see Definition 2.

The authors of [?] study variants of the representative LCA problem, namely (L)CA computations in weighted dags and the ALL-PAIRS ALL LCA problem. For the latter problem, an upper bound of $O(\mathrm{w(G)n}^{2+\mu})$ is established which is $O(n^{3+\mu})$ in the worst case. The solution makes use of a minimum path cover of the vertices. The general upper bound for ALL-PAIRS ALL LCA was improved to $O(n^{\omega(2,1,1)})$ in [?]. In the same work it is shown that ALL-PAIRS REPRESENTATIVE LCA and ALL-PAIRS ALL LCA can be solved with running times of $O(n^2 \log n)$ and $O(n^3 \log\log n)$ respectively in the average case. For the average case analysis it is assumed that the input space is distributed according to the $G_{n,p}$ model for random dags which was introduced by Barak and Erdős [?]. In [?] the problem of finding unique lowest common ancestors is shown to be solvable in time $O(n^{\omega} \log n)$.

Our contributions. We elaborate in-depth on using path cover techniques for the solution of LCA problems in dags. To this end, we present a multi-purpose decomposition technique that can be applied to a variety of LCA problems.

We apply our technique to the ALL-PAIRS REPRESENTATIVE LCA problem improving recently developed solutions for dags of small width [?]. Our result implies an upper bound of $\widetilde{O}(n^2\mathrm{w(G)})$ and improves the result in [?] for dags with width w(G) bounded by $O(n^{\omega-2-\delta})$ for a constant $\delta > 0$. Similarly, the application of our approach to the ALL-PAIRS ALL LCA problem yields an upper time bound of $\widetilde{O}(n^2\mathrm{w(G)}^2)$. This improves the general upper bound of $O(n^{\omega(2,1,1)})$ ([?]) for $\mathrm{w(G)} = O(n^{\frac{\omega(2,1,1)-2}{2}-\delta})$ as well as the upper bound of $O(n\, m_{\mathrm{red}} \min\{\kappa^2, n\})$ [?], where κ is the maximum size of an LCA set, if the size of the transitive reduction cannot be bounded by $O(n \cdot \mathit{polylog}(n))$.

We show that it is possible to combine the path cover approach with the efficient method for low depth dags given in [?]. Our algorithm has the same asymptotic worst-case time complexity as the fastest up to now algorithm for this problem, i.e., $O(n^{2+\mu})$ [?]. However, the class of dags for which the algorithm needs $\Omega(n^{2+\mu})$ time is considerably limited.

Finally, we describe some further applications of our approach. Firstly, we improve the general upper bound for the ALL-K-SUBSETS REPRESENTATIVE LCA problem (i.e., compute representative LCAs for each vertex subset of size k) for $k = 3$ which was recently established by Yuster [?]. Secondly, we show that our decomposition technique leads to space-efficient solutions for dags of small width. That is, we consider LCA computations from a data-structural point of view and present first non-trivial solutions that occupy subquadratic space and enable sublinear LCA queries in dags of small width.

2 Preliminaries

Let $G = (V, E)$ be a directed acyclic graph (dag). Throughout this work we denote by n the number of vertices and by m the number of edges in G. Let G_{clo} denote the reflexive transitive closure of G, i.e., the graph having an edge (u, v) if and only if $u \rightsquigarrow v$, i.e., v is reachable from u over some directed path in G. G_{clo} is in one-to-one correspondence with a partial order (poset) $P = (V, \leq)$ on the ground set V where the relation of the poset corresponds to the edges of G_{clo}. In this sense, by slight abuse of notation, we refer to G_{clo} also as a poset. We consider dags equipped with some topological ordering [?].

For a dag $G = (V, E)$ and $x, y, z \in V$, the vertex z is a *common ancestor* (CA) of a pair $\{x, y\}$ if both x and y are reachable from z. By CA$\{x, y\}$, we denote the set of all CAs of x and y. A vertex z is a *lowest common ancestor* (LCA) of x and y if and only if $z \in$ CA$\{x, y\}$ and no other vertex $z' \in$ CA$\{x, y\}$ is reachable from z, that is, there exists no witness for z and $\{x, y\}$.

Definition 1 (Witness) *A* witness *for z and $\{x,y\}$ such that $z \in \mathrm{CA}\{x,y\}$ is a vertex $w \in V$ such that $w \in \mathrm{CA}\{x,y\}$ and $z \rightsquigarrow w$.*

We denote the set of all LCAs of a pair $\{x,y\}$ by $\mathrm{LCA}\{x,y\}$. If it is not clear from the context we use a subscript to indicate the graph under consideration, e.g., $\mathrm{LCA}_G\{x,y\}$.

An arbitrary lowest common ancestor z of $\{x,y\}$ is also called a *representative* LCA of x and y. It is a well known and used fact, first observed by Bender et al. [**?**], that the vertex with the maximum topological number among all vertices in the set $\mathrm{CA}\{x,y\}$ is an LCA of x and y. We denote this special representative LCA by *maximum* LCA. (L)CA problems on dags come in various favors, i.e., computing one (representative) LCA vs. computing all LCAs or computing LCAs for all pairs vs. computing LCAs only for special pairs.

The width of G, denoted by $\mathrm{w(G)}$, is of particular interest in this work.

Definition 2 (Width) *The width $\mathrm{w(G)}$ of a dag G is defined as the cardinality of a maximum antichain, i.e., incomparable vertices with respect to G_{clo}, in G.*

Note that $\mathrm{LCA}\{x,y\}$ is an antichain by definition and hence $|\mathrm{LCA}\{x,y\}|$ is upper bounded by $\mathrm{w(G)}$.

For a dag $G = (V,E)$, a *path cover* $\mathcal{P}$ of G is a set of directed paths in G such for every $v \in V$ there exists at least one path $P \in \mathcal{P}$ including v. A *minimum path cover* is a path cover $\mathcal{P}$ such that $|\mathcal{P}|$ is minimized. The sizes of a maximal antichain and a minimum path cover of G are related by the famous Dilworth Theorem [**?**]:

Theorem 3 (Dilworth '50) *Let G be a dag. Then, $\mathrm{w(G)}$ equals the size of a minimum path cover of G.*

In the context of posets one considers usually *chain covers* instead of path covers. The paths in a chain cover are required to be vertex disjoint. However, it is easy to see that each chain cover in G_{clo} corresponds to a path cover in G and *vice versa*.

3 Path Cover Technique

In this section we present a natural approach to computing LCAs in dags based on decomposing the dag $G = (V,E)$ into a set of paths covering G. The efficiency of this technique depends mainly on the width ($\mathrm{w(G)}$) of the underlying dag.

We start by giving an intuitive description of our approach. Let $\{x,y\}$ be any vertex pair of G and let $z \in \mathrm{LCA}_G\{x,y\}$. Suppose now that we start a depth first search (DFS) in G at vertex z. Let T_z be the corresponding DFS tree [**?**]. Then, it is not difficult to verify that $\mathrm{LCA}_{T_z}\{x,y\} = z$. Observe that the partial order induced by the tree is a suborder of G_{clo}. Moreover, for all vertices $w \in T_z$ it holds that w is reachable from z. Since $z \in \mathrm{LCA}\{x,y\}$, the only common ancestor of x and y in T_z is z. Recall again, that the vertex with the highest topological number among $\mathrm{CA}\{x,y\}$ is a lowest common ancestor. This leads to a first naive decomposition solution for computing representative LCAs: For all $v \in V$, compute DFS trees T_v and preprocess these trees for LCA queries. In order to determine a representative LCA for $\{x,y\}$, compute $Z = \bigcup_{v \in V} \mathrm{LCA}_{T_v}\{x,y\}$ and return the maximum vertex in Z.

The correctness of the above approach follows from the fact that the maximum LCA z is returned at least once by the $O(n)$ LCA queries in the trees, i.e., by the query $\mathrm{LCA}_{T_z}\{x,y\}$. Furthermore, since all returned vertices z_v are common ancestors of $\{x,y\}$, the topological number of z is maximal among the numbers of vertices in Z. Recall that trees can be preprocessed in linear time and space for constant time LCA queries [?]. The preprocessing time is $O(nm)$. Subsequent queries for LCA in G can be answered in time $O(n)$ implying the trivial deterministic upper bound of $O(n^3)$ for the ALL-PAIRS REPRESENTATIVE LCA problem. However, we show below that in fact only w(G) DFS trees have to be considered.

Definition 4 *For a directed path $P = \{v_1, \ldots, v_l\}$ in G, T_P is a special DFS tree obtained by starting the DFS at vertex v_1 and first exploring the edges along the path P.*

In the following we denote by $T_P(v_i)$ the subtree of T_P rooted at vertex $v_i \in P$.

Lemma 5 *For a directed path $P = \{v_1, \ldots, v_l\}$ in G, $T_P(v_i)$ corresponds to some DFS tree T_{v_i} for all $1 \le i \le l$.*

Proof. Consider again the special dfs, i.e., the dfs which results in T_P, starting at v_1. The following is true for all $v_i \in P$. When v_i is visited for the first time by the dfs, all vertices that have been explored up to that point are predecessors of v_i in G. This follows from the fact that the dfs proceeds directly down the path P. This, in turn, implies that no successor of v_i has been reached since G is acyclic. The claim now follows from the recursive nature of depth first traversals. □

The above lemma implies that given a path cover $\mathcal{P}$ of G, only $|\mathcal{P}|$ DFS trees have to be considered. We recapitulate our decomposition technique:

1. Compute a path cover $\mathcal{P} = \{P_1, \ldots, P_r\}$ of G.
2. Compute DFS-trees $T_{P_1}, \ldots T_{P_r}$
3. Preprocess the DFS-trees $T_{P_1}, \ldots T_{P_r}$ for LCA queries in trees.
4. For given vertices $x, y \in V$, compute $Z = \bigcup z_i$ where $z_i = LCA_{T_{P_i}}\{x, y\}$ for $1 \le i \le r$.
5. Derive the solution of $LCA_G\{x, y\}$ from Z.

Lemma 6 *Let $\mathcal{P} = \{P_1, \ldots, P_r\}$ be a path cover of G. Next, let $Z = \bigcup z_i$, where $z_i = LCA_{T_{P_i}}\{x, y\}$ for $1 \le i \le r$. Then, the following chain of inclusion holds:* $\mathrm{LCA}\{x, y\} \subseteq Z \subseteq \mathrm{CA}\{x, y\}$.

Proof. The second inclusion follows again from the fact that each tree T_{P_i} induces a suborder on the poset induced by G. Hence a common ancestor of $\{x, y\}$ in a tree is also a common ancestor of $\{x, y\}$ in G. Let z be an LCA of x and y. Let P_i be chosen such that $z \in P_i$. Since by Lemma 5, $T_{P_i}(z)$ corresponds to (some) DFS tree T_z, by the previous discussion we have $\mathrm{LCA}_{T_{P_i}}\{x, y\} = z$. □

Remark 7 *By Lemma 6 both answers to representative LCA queries and all LCA queries can be derived from the set Z. For representative LCAs, we simply have to find the vertex with the maximum number in Z in time $O(|Z|)$. In order to find all LCAs, it is necessary to identify the set of vertices without outgoing edges in the subdag induced by Z. This can be achieved in time $O(|Z|^2)$. To this end, we assume that we have a mapping from V to $\mathcal{P}$ such that for each v a path P_i including v is known. With this mapping reachability queries of two vertices $z_1, z_2 \in Z$ can be answered in constant time by querying the LCA of $\{z_1, z_2\}$ in the corresponding trees. Observe that $z_1 \rightsquigarrow z_2$ implies* $\mathrm{LCA}_{P_i}\{z_1, z_2\} = z_1$ *if $z_1 \in P_i$.*

The following lemma follows immediately from the previously established facts.

Lemma 8 *Let $T_{\mathrm{PC}}(G, r)$ be the time needed to compute a path cover $\mathcal{P} = \{P_1, \ldots, P_r\}$ of G. Then,* ALL-PAIRS REPRESENTATIVE LCA *can be solved in time $O(T_{\mathrm{PC}}(G, r) + rm + rn^2)$.*

Proof. The special DFS trees for $P_1,\ldots,P_r$ can be constructed in time $O(rm)$. The preprocessing of the trees for LCA-queries takes $O(nr)$ time since the size of the trees is bounded by $O(n)$. Finally, since the resulting data structure supports representative LCA queries in time $O(r)$, computing the solutions for all pairs takes time $O(n^2r)$. □

In [?] Kowaluk and Lingas show that the ALL-PAIRS REPRESENTATIVE LCA problem can be solved in time $\widetilde{O}(n^{\omega} + \mathrm{w(G)n^2})$. Furthermore, this bound is improved to $\widetilde{O}(\mathrm{w(G)n^2})$ [?] by using randomization. In the following we show that our path cover technique can be applied to obtain a simple algorithm for ALL-PAIRS REPRESENTATIVE LCA running in time $\widetilde{O}(\mathrm{w(G)n^2})$.

For a given minimum path cover of size $r = \mathrm{w(G)}$, the above lemma immediately yields the claimed time bounds. However, it is not known how to compute a minimum path cover within time $O(n^2\mathrm{w(G)})$. Nonetheless, we show how to compute a path cover $\mathcal{P}$ of G_{clo} of size $O(\mathrm{w(G)}\log \mathrm{n})$ in time $\widetilde{O}(n^2\mathrm{w(G)})$, i.e., $\mathcal{P}$ is minimal up to a logarithmic factor. We apply the technique to improve the upper bound of the ALL-PAIRS REPRESENTATIVE LCA and ALL-PAIRS ALL LCA problems on dags of small width.

The standard solution for computing a minimum chain cover of a dag reduces the problem to finding a maximum matching in a bipartite graph [?]. The number of edges in the bipartite graph corresponds to the number of edges in G_{clo}. In the deterministic setting, the best algorithm for solving the bipartite matching problem is still the $O(m\sqrt{n})$ solution given by Hopcroft and Karp [?]. Recently, Mucha et al. [?] have shown that a maximum matching in a bipartite graph can be found in $O(n^{\omega})$ using randomization. On the other hand, Felsner et al. [?] showed that it is possible to recognize a poset P of width at most k in time $O(n^2k)$. In their approach, the parameter k is given beforehand and – in the case that the width of P is bounded by k – their algorithm can be extended to output a chain cover of size k.

Definition 9 (Greedy Path Cover) *Let $G = (V,E)$ be a dag. A greedy path cover of G is obtained by recursively finding paths $P_1,\ldots,P_r$ such that for all P_i, the value $|P_i \setminus \bigcup_{k\le i-1} P_k|$, i.e., the number of vertices on P_i that are not covered by any of the paths $P_1,\ldots,P_{i-1}$, is maximized.*

That is, we reduce the dag into paths by recursively finding paths containing as much uncovered vertices as possible. In Felsner et al. [?], a similar approach is used to decompose a partial order P in a greedy manner. In their approach, all covered vertices are removed from the partial order. Since we do not want to construct the poset G_{clo}, we cannot discard vertices. However, the following establishes a one-to-one correspondence between decomposing G and G_{clo}

Lemma 10 *$P_1,\ldots,P_r$ is a greedy path cover of G if and only if $C_1,\ldots,C_r$ is a greedy chain partition of G_{clo}, where $C_i = P_i \setminus \bigcup_{k\le i-1} P_k$.*

Proof. We prove the lemma by induction on i. For $i = 1$, we obviously have $P_1 = C_1$. Suppose now the claim is true for $k = 1,\ldots i-1$. Let C_i be a chain of maximum size in $G_{\mathrm{clo}} \setminus \bigcup_{k\le i-1} P_k$. Then, by definition, there exists a path P_i in G such that the number of uncovered vertices on P_i is equal to $|C_i|$. On the other hand, suppose that P_i is a path in G such that the number of uncovered vertices is strictly larger than $|C_i|$. Then, again, it is easy to see that this implies an existence of a chain of size strictly larger than $|C_i|$ in $G_{\mathrm{clo}} \setminus \bigcup_{k\le i-1} P_k$ contradicting our assumption. □

The above lemma implies that a greedy path cover is optimal up to a logarithmic factor.

Corollary 11 *For a greedy path cover $\mathcal{P} = \{P_1,\ldots,P_r\}$, $r \le \mathrm{w(G)}\log \mathrm{n}$ holds.*

Proof. In Felsner et al. [?], it is shown that the size of a greedy chain cover of a partial order P is at most $\mathrm{w(P)} \log \mathrm{n}$. The corollary follows from Lemma 10 and the fact that $\mathrm{w(P)} = \mathrm{w(G)}$. □

Lemma 12 *A greedy path cover $P_1,\ldots,P_r$ of G can be computed in time $O(mr)$.*

Proof. The (weighted) single source longest path problem can be solved in time $O(n+m)$ by using standard algorithms for solving the single source shortest path problem in dags, see, e.g., [?]. Observe that arbitrary edge weights are possible. We assume without loss of generality that G is equipped with a single source s, otherwise we simply add a super source. We initialize each edge in G with weight 1. Then, we recursively solve the single source longest path problem for the source s. After the ith iteration we set the weight of each edge $e = (v,w)$ such that $w \in P_i$ to 0. It is easy to check that the weight of P_i is equal to the number of uncovered vertices on P_i. Hence, we obtain a greedy path cover. It is also easy to see that each step can be implemented in time $O(m)$. □

Lemma 8, Corollary 11, and Lemma 12 imply the following theorem.

Theorem 13 ALL-PAIRS REPRESENTATIVE LCA *can be solved in time $\widetilde{O}(\mathrm{w(G)n^2})$ and* ALL-PAIRS ALL LCA *can be solved in time $\widetilde{O}(\mathrm{w(G)^2n^2})$.*

This result improves the upper bound of ALL-PAIRS REPRESENTATIVE LCA ([?]) for dags with $\mathrm{w(G)} = O(n^{\omega-2-\delta})$ and the general upper time bound for ALL-PAIRS ALL LCA ([?]) for dags with $\mathrm{w(G)} = O(n^{\frac{\omega(2,1,1)-2}{2}-\delta})$ for a constant $\delta > 0$. Using results on the expected value of the width of $G_{n,p}$ random dags [?] we get the following corollary. The respective average case complexities match results previously established in [?].

Corollary 14 *Using a path-cover-based-approach,* ALL-PAIRS REPRESENTATIVE LCA *and* ALL-PAIRS ALL LCA *can be solved in time $\widetilde{O}(n^2)$ on a dag with n vertices in the average case under the assumption that the input space is distributed according to the $G_{n,p}$ model for random dags with edge probability $p = O(1)$.*

4 Combining Small Width and Low Depth

The path cover technique described in the previous chapter can be naturally used together with the solution for dags of low depth given in [?]. The *depth* of a vertex v denoted by $\mathrm{dp(v)}$ is defined as the length of longest path to v in G. The depth of G, $\mathrm{dp(G)}$ is given by $\mathrm{dp(G)} = \max_{\mathrm{v \in V}}\{\mathrm{dp(v)}\}$.

Theorem 15 *For a dag G with depth $\mathrm{dp(G)} = \mathrm{n^q}$,* ALL-PAIRS REPRESENTATIVE LCA *can be solved in $\widetilde{O}(n^{q+\omega(1,1-q,1)})$.*

This result is achieved by exploiting the fact that common ancestor of maximum depth in a dag G is a lowest common ancestor. Observe that a possible witness would have greater depth by definition. The following lemma can be derived from the above proposition.

Lemma 16 *For a dag $G = (V,E)$ with n vertices and depth $\mathrm{dp(G)} = \mathrm{n^q}$, the* ALL-PAIRS REPRESENTATIVE LCA *problem can be solved in time $\widetilde{O}(n^{2+\mu-\delta})$ if $q \leq 1-\mu-\delta$ for an arbitrary small constant $\delta > 0$.*

Proof. ALL-PAIRS REPRESENTATIVE LCA in G can be solved in time $\widetilde{O}(n^{q+\omega(1,1-q,1)})$ (Thm. 15). We make use of Proposition **??**. Let $\beta = \frac{\omega-2}{1-\alpha}$. Then we want to solve the following inequality $q+\omega(1,1-q,1) \leq 2+\mu-\delta$ (i) for q. Recall that μ satisfies $\omega(1,\mu,1) = 1+2\mu$. By Proposition **??** we get: $\mu = \frac{1-\beta\alpha}{2-\beta}$ (ii). Now we plug (ii) into (i) and solve for q to get $q \leq 1-\mu-\delta$.

□

Assume first that we have already computed G_{clo}. On a high level, the combination of the above result with our path cover technique works as follows:

1. Construct a partial chain cover $C_1,\ldots,C_r$ of G_{clo} greedily. That is, search for chains of maximum size until a termination criterion (to be specified below) is satisfied. Note that we do not require that the chains cover all vertices of G_{clo}. Note further that in general $r \neq \text{w(G)}$.
2. Construct the special DFS trees associated with the chains as described in the previous chapter and prepare them for constant time LCA queries.
3. Reduce G_{clo} along the chains. That is, let $V_C = \bigcup_{1\leq i\leq r} C_i$. Then we remove all edges (v,w) such that $v \in V_C$. Observe that all edges (v,w) where $v \notin V_C$ are retained. The resulting graph is called the *reduced dag* denoted by $G^R = (V,E^R)$.
4. For all $x,y \in V$, compute maximum depth common ancestors in G^R. The maximum depth CAs are either lowest common ancestors in G or all of their witnesses are in the set V_C. Thus, in a second step we search for possible witnesses by querying the special DFS trees for vertices in V_C. If witnesses exist, i.e., common ancestors that are successors of the maximum depth CAs, we output the witness with highest label which is a lowest common ancestor.

The reasoning behind this approach is as follows. By decomposing the graph along the longest chains we reduce successively the depth of G_{clo}. As soon as we reach a certain threshold depth we can apply the algorithm given in [**?**] to efficiently compute maximum depth CAs in the reduced dag. Moreover, if a maximum depth CA in the reduced dag G^R is not a lowest common ancestor in the original dag, we know that all its witnesses are covered by the chains. Observe that outgoing edges of non-covered vertices are not removed. Hence, if z' is a witness of z and $\{x,y\}$ and we suppose $z' \notin V_C$, we have $z \rightsquigarrow z'$ and $z' \rightsquigarrow x,y$ in the reduced dag contradicting the fact that z is a maximum depth CA in G^R.

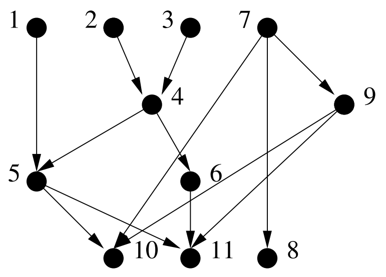
In the following we refer to the algorithm described in this section as the *combined algorithm*. Before proving the key properties of the approach we give a formal description of the decomposition:

Lemma 17 *Let z_h be a maximum depth common ancestor in G^R of a pair $\{x,y\}$. Then either $z_h \in \text{LCA}_G\{x,y\}$ or all witnesses of z_h and $\{x,y\}$ in G are in the set V_C.*

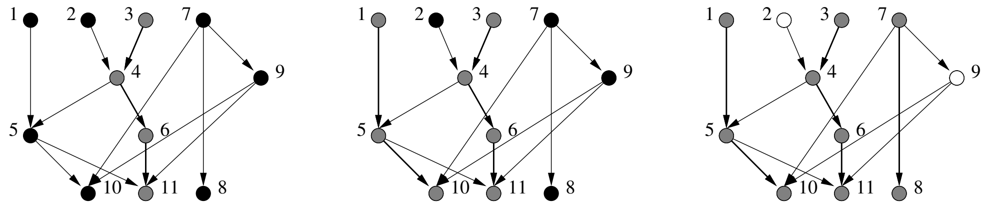
Proof. Suppose that z_h is not an LCA of $\{x,y\}$ in G. In that case there exists a witness z' such that $z' \in \text{LCA}_G\{x,y\}$ and z' is a successor of z_h. This immediately implies $z' \in V_C$. Indeed, suppose that $z' \notin V_C$. This implies (i) $\text{dp}(z') > \text{dp}(z_h)$ since z' is a successor of z and (z_h,z') is not removed by the decomposition and (ii) $z' \in \text{CA}_{G^R}\{x,y\}$. But, (i) and (ii) contradict the fact that z_h is the maximum depth CA of $\{x,y\}$ in G^R. □

Observe that we have constructed special DFS trees for the chains in G_{clo}. However, in general it is also possible to use an approach that allows constructing the trees in G. This involves computing a greedy path cover of G instead of the greedy chain cover of G_{clo}. In any case, the reduced dag G^R results from G_{clo}.

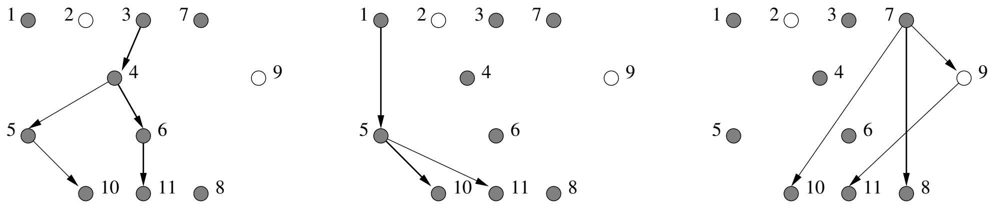
We specify the full algorithm for finding representative LCAs for each vertex pair with respect to G.



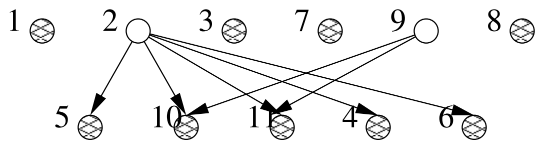
(a) input dag G



(b) computing a partial chain cover $\mathcal{C}$ in a greedy manner, $\mathcal{C} = \{(3,4,6,11),(1,5,10),(7,8)\}$



(c) the special DFS trees corresponding to the partial chain cover, the result set Z for queries for LCA$\{10,11\}$ in the trees is given by $Z = \{4,5,7\}$



(d) the reduced dag G^R, both 2 and 9 correspond to maximum depth CAs of $\{10,11\}$. However, only one of them is returned, e.g., 2.

Figure 2: Example for the combined algorithm on a dag with 11 vertices. Suppose that the maximum depth CA of $\{10,11\}$ in G^R that is returned is 2. Then, the maximum vertex in Z (7) is not an LCA. However, restricting the candidate vertices to successors of 2 excludes 7. In consequence 5 is returned and $5 \in$ LCA$\{10,11\}$.

Algorithm 1: Algorithm for representative LCA

```
Input: The reduced dag $G^R = (V, E^R)$ and special DFS trees $T_{C_1}, \dots, T_{C_r}$ that are prepared for constant
       LCA queries (output of the preprocessing algorithm)
Output: All-pairs representative LCA matrix
begin
    Compute maximum depth CAs on for all vertex pairs in $G^R$ using the algorithm described in [?].
    foreach Vertex pair $\{x, y\}$ do
        Let $z_h$ be the maximum depth CA of $\{x, y\}$ with respect to $G^R$.
        Initialize an empty result set $Z$.
        foreach chain $C_i$, $1 \leq i \leq r$ do
            Query LCA$\{x, y\}$ in $T_{C_i}$ and add the result to $Z$.
        end
        Remove all vertices from $Z$ that are not successors of $z_h$.
        Return the maximum vertex in $Z$ and $z_h$ if $Z$ is empty.
    end
end
```

An example of the combined algorithm is given in Figure **??**.

The correctness of Algorithm **??** is a consequence of Lemma **??** and the results of Section 3. We turn our attention to the complexity of this approach. Obviously, the running time depends on (i) the cardinality r of the partial chain cover and (ii) $\mathrm{dp}(\mathrm{G}^\mathrm{R})$. Our goal is an algorithm that does not exceed the worst case complexity for the general problem of $O(n^{2+\mu})$ but performs better in as many cases as possible. To this end, one can specify an implementation of the termination criterion for the preprocessing algorithm as follows: Pick threshold parameters W and H and terminate whenever $r \geq W$ or $\mathrm{dp}(\mathrm{G}^\mathrm{R}) \leq \mathrm{H}$. Any reasonable choice of H should satisfy $H \leq n^{1-\mu}$ according to Lemma 16. Observe that any choice of W such that $W \leq n^{\mu}$ is sufficient to guarantee a worst case upper bound of $O(n^{2+\mu})$. Simply modify the combined algorithm such that it uses the general solution whenever $r \geq W$ (which implies $\mathrm{dp}(\mathrm{G}^\mathrm{R}) > \mathrm{H}$). However, there is a more elegant way to find the optimal decomposition. The idea is to look for the intersection point of the functions $n^{q+\omega(1,1-q,1)}$ and rn^2, i.e., the functions that describe the asymptotic behavior of the two approaches. The respective termination criterion becomes (neglecting polylogarithmic factors)

$$rn^2 > n^{\log_n(\mathrm{dp}(\mathrm{G}^\mathrm{R}))+\omega(1,1-\log_\mathrm{n}(\mathrm{dp}(\mathrm{G}^\mathrm{R})),1)}. \tag{1}$$

In order to prove the following theorem, we make use of the following proposition which is due to Huang and Pan [?]. Recall that $\omega(a,b,c)$ denotes the exponent of the multiplication of an $n^a \times n^b$ by an $n^b \times n^c$ matrix.

Proposition 18 (Rectangular Matrix Multiplication) *Let* $\omega = \omega(1,1,1) < 2.376$ *and let* $\alpha = \sup\{0 \leq r \leq 1 : \omega(1,r,1) = 2 + o(1)\}$. *Then*

$$\omega(1,r,1) \leq \begin{cases} 2+o(1) & \text{if } 0 \leq r \leq \alpha \\ 2+\frac{\omega-2}{1-\alpha}(r-\alpha)+o(1) & \text{if } \alpha \leq r \leq 1. \end{cases} \tag{2}$$

The current best bound for α *is* $\alpha \leq 0.294$ *and is due to Coppersmith [?].*

Theorem 19 *The time complexity of the combined algorithm can be bounded by* $O(n^{2+\mu})$ *where* μ *satisfies* $\omega(1,\mu,1) = 1+2\mu$.

Proof. Let in the following $\beta = \frac{\omega-2}{1-\alpha}$. Let r be the cardinality of the path cover produced by the preprocessing algorithm and let $\mathrm{dp}(\mathrm{G}^\mathrm{R}) = \mathrm{n}^\mathrm{q}$ be the depth of the reduced graph G^R. By Equation (**??**) we have $n^{q+\omega(1,1-q,1)} < rn^2 < n^{q+\omega(1,1-q,1)} + 1$ and from this we can conclude $r \le n^{q+\omega(1,1-q,1)-2+o(1)}$. Moreover, $\widetilde{O}(n^2 r)$ is obviously an upper time bound for the combined algorithm. We can safely assume that $q \le 1/2$. To see this, suppose first that $q > 1/2$. Since every path covers at least n^q vertices, we have $r \le n^{1-q}$. On the other, $r > n^{q+o(1)}$ by Equation (**??**) and thus $n^{1-q} > n^{q+o(1)}$, a contradiction for $q > 1/2$. We apply Proposition **??** and get $r \le n^{q+2+\beta(1-q-\alpha)-2+o(1)}$. This simplifies to $r \le n^{q(1-\beta)+\beta(1-\alpha)+o(1)}$.Since we know that each path in the path cover covers at least h vertices we have $q \le \log_n \frac{n}{r}$. Combining the above two inequalities yields $r \le n^{\log_n \frac{n}{r}(1-\beta)+\beta(1-\alpha)+o(1)}$. Since we have $n^{\log_n \frac{n}{r}(1-\beta)+\beta(1-\alpha)} = \left(\frac{n}{r}\right)^{1-\beta} \cdot n^{\beta(1-\alpha)}$, this simplifies to $r \le n^{\frac{1-\beta\alpha}{2-\beta}+o(1)}$. On the other hand, recall that μ satisfies $\omega(1,\mu,1) = 1+2\mu$. Again, we use Proposition **??** to obtain $\mu = \frac{1-\beta\alpha}{2-\beta}$, which concludes the proof.

□

As an immediate consequence of the above lemma and the current best bounds for ω and α we get the following corollary.

Corollary 20 *The time complexity of the algorithm for solving* ALL-PAIRS REPRESENTATIVE LCA *is $O(n^{2.575})$.*

Observe that the average case bound of $\widetilde{O}(n^2)$ is still valid under the assumption that the input space is distributed according to the $G_{n,p}$ model with constant edge probability p. To see this, observe that the transitive closure of a random dag in the $G_{n,p}$ model for arbitrary values of p can be computed in average case time $\widetilde{O}(n^2)$ using the algorithm given by [**?**].

The combination of these two techniques narrows down the classes of dags for which no solution faster than $O(n^{2+\mu})$ is known considerably. Indeed, recall that for dags G of depth $\mathrm{dp(G)} \le \mathrm{n}^{1-\mu-\delta}$ the ALL-PAIRS REPRESENTATIVE LCA problem can be solved in time $\widetilde{O}(n^{2+\mu-\delta})$ by Lemma 16.

Theorem 21 *For a dag G with n vertices and an arbitrary constant $\mu \ge \delta > 0$* ALL-PAIRS REPRESENTATIVE LCA *can be solved in time $\widetilde{O}(n^{2+\mu-\delta})$ if G does not contain a subgraph H that contains at least $n^{\mu-\delta}$ (vertex-disjoint) maximum size (w.r.t. the greedy approach) chains of length at least $n^{1-\mu-\delta}$.*

Observe that this is a significant restriction for bad dag classes, namely an almost linear-sized, i.e., $n^{1-2\delta}$ for an arbitrary small constant δ, subdag of extremely regular structure.

5 Applications

ALL-K-SUBSETS REPRESENTATIVE LCA: The ALL-K-SUBSETS REPRESENTATIVE LCA problem is an extension of the ALL-PAIRS REPRESENTATIVE LCA problem. Given a constant $k \ge 2$, compute a representative LCA for each k-subset of vertices in G. The problem has been considered recently by Yuster [**?**]. He shows that the ALL-K-SUBSETS REPRESENTATIVE LCA problem can be solved in time $O(n^{3.575})$ for $k = 3$ and $O(n^{k+1/2})$ for $k \ge 4$. We improve slightly upon the bound for $k = 3$ by using our combined approach.

Theorem 22 *The* ALL-K-SUBSETS REPRESENTATIVE LCA *problem can be solved in time $O(n^{3.5214})$ for $k = 3$ and $\widetilde{O}(n^{k+\frac{1}{2}})$ for $k \ge 4$.*

Proof. Let $G = (V,E)$ be a dag of depth $\mathrm{dp(G)} = \mathrm{n^q}$. Combining the ideas in [?] and [?] for computing LCAs in dags of small depth, ALL-K-SUBSETS REPRESENTATIVE LCA can be solved by computing (arbitrary) witnesses for Boolean matrix products MM^T for each level of the dag. More specifically, let l_i be the number of vertices on level L_i, then the matrix M corresponding to level L_i is an $n^{\binom{n}{k/2}} \times n^{l_i}$ matrix. Further, by Jensen's inequality, the time complexity of the algorithm is maximized if the levels of G are of equal size. That is, the running time of this approach can be bounded by $\widetilde{O}(n^{q+\omega(1,\frac{2(1-q)}{k},1)\frac{k}{2}})$. The additional polylogarithmic factor results from the witness computations.

On the other hand, we note that the LCA of a k-subset of vertices in a preprocessed tree can be computed in $O(1)$ since we assume that k is a constant. This implies that ALL-K-SUBSETS REPRESENTATIVE LCA can be solved in time $O(n^k \mathrm{w(G)})$ by extending the ideas in Section 3.

Now we combine these two approaches in an analogous way as described above. We get an algorithm with time complexity $\widetilde{O}(rn^k + n^{q+\omega(1,\frac{2(1-q)}{k},1)\frac{k}{2}})$ for $\frac{2(1-q)}{k} > 0.294$ and $\widetilde{O}(rn^k + n^{q+k})$ for $\frac{2(1-q)}{k} \leq 0.294$, where $q \leq \log_n \frac{n}{r}$. Observe first that for $k \geq 4$ the respective function is minimized for $q = \frac{1}{2}$, which implies a time bound of $\widetilde{O}(n^{k+\frac{1}{2}})$.

Let now $k = 3$ and assume first that $\frac{2(1-q)}{3} \leq 0.294$. This implies $q \geq 0.559$ and hence an upper bound of $\widetilde{O}(n^{k+0.559})$. Suppose now that $\frac{2(1-q)}{3} > 0.294$. We solve the equality $rn^k = n^{q+\omega(1,\frac{2(1-q)}{k},1)\frac{k}{2}}$. By using $q \leq \log_n \frac{n}{r}$ and $\beta = \frac{\omega-2}{1-\alpha}$ we find the bound for $r \leq n^{1-\frac{3}{2}\beta\alpha}$ in a similar way as in the proof of Theorem 13. The claim follows now since $0.5214 < 1 - \frac{3}{2}\beta\alpha$. □

Note that Theorem **??** slightly subsumes Yuster's upper time bound for $k = 3$ and matches his remaining upper bounds for $k \geq 4$ [?] ignoring polylogarithmic factors. Again, we observe that if the input dags are distributed according to the $G_{n,p}$ model for random dags such that p is a constant, ALL-K-SUBSETS REPRESENTATIVE LCA can be solved in time $\widetilde{O}(n^k)$ in the average case. Nonetheless, we can even state the following corollary which is valid for all choices of the edge probability p.

Corollary 23 ALL-K-SUBSETS REPRESENTATIVE LCA *can be solved in time $\widetilde{O}(n^k)$ in the average case.*

Proof. The following claim is a straightforward generalization of results described in [?].

Claim 24 *Let $G = (V,E)$ be a dag and let $\{y_1,\dots,y_k\}$ be a k-subset of vertices. Furthermore, let z be the maximum CA (MCA) of $\{y_1,\dots,y_k\}$.*

1. *If y_1 is an ancestor of $y_2,\dots,y_k$, then $z = y_1$.*
2. *If y_1 is not an ancestor of $y_2,\dots,y_k$, then the following holds: Let $\{x_1,\dots,x_l\}$ be the parents of y_1. Let $Z = \bigcup_{1\leq i\leq l} MCA\{x_i,y_2,\dots,y_k\}$. Then $MCA\{y_1,\dots,y_k\}$ is the maximum vertex in Z.*

The above claim yields a dynamic programming algorithm that is analogous to algorithms described in [?]. We can bound the running time by $O(m_{\mathrm{red}}\, n^{k-1})$ which implies an average case complexity of $O(n^k \log n)$. □

Space-Efficient LCA Computations: With respect to possible applications, the consideration of *all-pairs* LCA problems are fairly specific. Typically, one is not interested in LCA information associated with every possible vertex pair in a dag. Rather, LCAs are only queried on a subset of the vertex pairs. Precomputing the answers to all possible queries and storing these answers in a matrix is brute force in this context. Considering the huge memory requirements of this approach it is desirable to design more space-efficient solutions. More specifically, we give a more general formulation of LCA problems as follows: Preprocess G such that a query of the form LCA$\{x,y\}$ can be answered as quickly as possible. Again, the query LCA$\{x,y\}$ can be any variant of LCAs in dags. The quality of solutions to the above problems is measured in terms of preprocessing time, space requirements, and query time. We refer to these parameters by a triple $\langle p(n), s(n), q(n)\rangle$, i.e., using an ALL-PAIRS REPRESENTATIVE LCA solution we get $\langle O(n^{2.575}), O(n^2), O(1)\rangle$. This point of view is naturally taken in the context of LCA queries in trees, i.e., LCAs in trees can be solved in time $\langle O(n), O(n), O(1)\rangle$ [**?**]. Our path cover technique yields space efficient solutions for dags with moderately sized width.

Theorem 25 *The representative LCA problem in dags can be solved within* $\langle \widetilde{O}(\mathrm{w(G)n^2}), \widetilde{\mathrm{O}}(\mathrm{w(G)n}), \widetilde{\mathrm{O}}(\mathrm{w(G)})\rangle$.

Again, we note that results on the expected width of random $G_{n,p}$ dags imply significantly improved average case complexities in many cases. More specifically, let $p = f(n)/n$, such that $f(n)$ is an arbitrary positive function with $f(n) \leq n$. Then, it is possible to show that the representative LCA problem can be solved with $o(n^2)$ space and $o(n)$ if $f(n) = \omega(1)$. On the other hand there exist solutions for sparse random which are space-efficient in the average case such that we can bound the average case complexity by $\langle o(n^3), o(n^2), o(n)\rangle$ for any choice of p. For more details we refer to the full version of this paper.

References

[1] H. Aït-Kaci, R. Boyer, P. Lincoln and R. Nasr. Efficient Implementation of Lattice Operations. *ACM Transactions on Programming Languages*, 11:1, 115–146, 1989.

[2] A. Aho, J. Hopcroft and J. Ullman. On Finding Lowest Common Ancestors in Trees. *SIAM Journal on Computing*, 5:1, 115–132, 1976.

[3] A. Barak and P. Erdös. On the maximal number of strongly independent vertices in a random acyclic directed graph. *SIAM J. Algebraic Discrete Methods*,vol. 5,pp. 508–514,1984.

[4] M. Baumgart, S. Eckhardt, J. Griebsch, S. Kosub and J. Nowak. All-Pairs Common Ancestor Problems in Weighted Directed Acyclic Graphs *Proc. ESCAPE'07*, pp.282–293.

[5] 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*, vol. 57, no. 2, pp. 75-94, 2005 (a preliminary version in *Proc. SODA 2001*, pp. 845–853).

[6] D. Coppersmith. Rectangular matrix multiplication revisited. *Journal of Symbolic Computation*, 13: 42–49, 1997.

[7] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progression. *Journal of Symbolic Computation*, 9: 251–290, 1990.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* 2nd edition, McGraw-Hill Book Company, Boston, MA, 2001.

[9] A. Czumaj, M .Kowaluk and A. Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. In *the special ICALP 2005 issue of Theoretical Computer Science*, 380 (1-2), pp. 37–46, 2007.

[10] R. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics* 51:1, 161–166, 1950.

[11] S. Eckhardt, A. Mühling and J. Nowak. Fast Lowest Common Ancestor Computations in Dags. *Proc. of ESA'07*, pp. 705–716, 2007.

[12] S. Felsner, V. Raghavan and J. Spinrad. Recognition Algorithms for Orders of Small Width and Graphs of Small Dilworth Number. *Order*, vol. 20 (4), pp. 351–364, 2003.

[13] L. R. Ford and D. R. Fulkerson. Flows in Networks. *Princeton University Press*, 1962.

[14] D. Harel and R. E. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13(2): 338–355, 1984.

[15] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J.Comput.*, vol. 2 (4), pp. 225–231, 1973.

[16] X. Huang and V.Y. Pan. Fast rectangular matrix multiplications and applications. *Journal of Complexity*, 14: 257–299, 1998.

[17] M. Kowaluk and A. Lingas. LCA queries in directed acyclic graphs. *Proc. 32nd International Colloquium on Automata, Languages and Programming (ICALP'05)*, pp. 241–248, 2005.

[18] M. Kowaluk and A. Lingas. Unique Lowest Common Ancestors in Dags Are Almost as Easy as Matrix Multiplication. *Proc. 15th Annual European Symposium on Algorithms (ESA'07)*, pp. 265–274, 2007.

[19] M. Mucha and P. Sankowski. Maximum Matchings via Gaussian Elimination. *Proc. FOCS'04*, pp. 248–255, 2004.

[20] M. Nykänen and E. Ukkonen. Finding lowest common ancestors in arbitrarily directed trees. *Information Processing Letters*, 50(6): 307–310, 1994.

[21] B. Schieber and U. Vishkin. On Finding Lowest Common Ancestors: Simplification and Parallelization. *SIAM Journal on Computing*, 17:6 1253–1262, 1988.

[22] A. A. Shäffer, S. K. Gupta, K. Shriram, and R. W. Cottingham Jr. Avoiding recomputation in linkage analysis. *Human Heredity*, 44: 225–237, 1994.

[23] K. Simon. An Improved Algorithm for Transitive Closure on Acyclic Digraphs. *Theor. Comput. Sci.*, vol. 58, pp. 325–346, 1988.

[24] R.E. Tarjan. Applications of path compression on balanced trees. *Journal of the ACM*, 26(4): 690–715, 1979.

[25] R. Yuster. All-pairs disjoint paths from a common ancestor in $\widetilde{O}(n^{\omega})$ time. Accepted for publication in Theor. Comput. Sci.