algorithms

MDPI

Article

A Multi-Dimensional Matrix Product—A Natural Tool for Parameterized Graph Algorithms

Mirosław Kowaluk [1] **and Andrzej Lingas** [2,*]

[1] Institute of Informatics, University of Warsaw, 00-927 Warsaw, Poland
[2] Department of Computer Science, Lund University, Box 118, 22100 Lund, Sweden
* Correspondence: andrzej.lingas@cs.lth.se

Abstract: We introduce the concept of a k-dimensional matrix product D of k matrices $A_1, \ldots, A_k$ of sizes $n_1 \times n, \ldots, n_k \times n$, respectively, where $D[i_1, \ldots, i_k]$ is equal to $\sum_{\ell=1}^{n} A_1[i_1, \ell] \times \ldots \times A_k[i_k, \ell]$. We provide upper bounds on the time complexity of computing the product and solving related problems of computing witnesses and maximum witnesses of the Boolean version of the product in terms of the time complexity of rectangular matrix multiplication. The multi-dimensional matrix product framework is useful in the design of parameterized graph algorithms. First, we apply our results on the multi-dimensional matrix product to the fundamental problem of detecting/counting copies of a fixed pattern graph in a host graph. The recent progress on this problem has not included complete pattern graphs, i.e., cliques (and their complements, i.e., edge-free pattern graphs, in the induced setting). The fastest algorithms for the aforementioned patterns are based on a reduction to triangle detection/counting. We provide an alternative simple method of detection/counting copies of fixed size cliques based on the multi-dimensional matrix product. It is at least as time efficient as the triangle method in cases of K_4 and K_5. Next, we show an immediate reduction of the k-dominating set problem to the multi-dimensional matrix product. It implies the $W[2]$ hardness of the problem of computing the k-dimensional Boolean matrix product. Finally, we provide an efficient reduction of the problem of finding the lowest common ancestors for all k-tuples of vertices in a directed acyclic graph to the problem of finding witnesses of the Boolean variant of the multi-dimensional matrix product. Although the time complexities of the algorithms resulting from the aforementioned reductions solely match those of the known algorithms, the advantage of our algorithms is simplicity. Our algorithms also demonstrate the versatility of the multi-dimensional matrix product framework.

Keywords: subgraph isomorphism; clique; lowest common ancestor; time complexity

Citation: Kowaluk, M.; Lingas, A. A Multi-Dimensional Matrix Product—A Natural Tool for Parameterized Graph Algorithms. *Algorithms* **2022**, *15*, 448. https://doi.org/10.3390/a15120448

Academic Editor: Vangelis Th. Paschos

Received: 25 September 2022
Accepted: 24 November 2022
Published: 28 November 2022

1. Introduction

Matrix multiplication over various rings or semi-rings is a basic tool in science and engineering. By the definition, the matrix product of two $n \times n$ matrices can be easily computed using $O(n^3)$ additions and multiplications over the respective ring or semi-ring. This is optimal in the arithmetic case and the Boolean case if, respectively, only the operations of arithmetic addition and multiplication or the Boolean OR and AND are allowed [1,2]. Five decades ago, Strassen presented a divide-and-conquer algorithm for arithmetic matrix multiplication based on algebraic term cancellation, breaking out of the $O(n^3)$ method. It used only $O(n^{2.8074})$ multiplications, additions and subtractions [3]. The complexity of matrix multiplication is measured in terms of ω which is the smallest real number such that two $n \times n$ matrices can be multiplied using $O(n^{\omega+\epsilon})$ operations over the field of reals, for all $\epsilon > 0$ (i.e., the number of operations is $O(n^{\omega+o(1)})$ [4]. The series of improvements of the upper bounds on ω starting from $\omega < 2.8074$ culminates in the recent result of Alman and Vassilevska Williams showing $\omega < 2.3729$ [4]. By a straightforward reduction of the Boolean matrix product to the corresponding arithmetic one for $0-1$ matrices, the same asymptotic upper bounds hold for the Boolean matrix product.

We generalize the concept of matrix product of two matrices over a ring or semiring to include that of a k-dimensional matrix product D of k matrices $A_1, \ldots, A_k$, of sizes $n_1 \times n, \ldots, n_k \times n$, respectively, where $D[i_1, \ldots, i_k] = \sum_{\ell=1}^{n} A_1[i_1, \ell] \times \ldots \times A_k[i_k, \ell]$. Note that the *two*-dimensional product of the matrices A_1 and A_2 is equivalent to the standard matrix product of A_1 and the transpose $(A_2)^T$ of the matrix A_2. For $k \geq 2$, we define $\omega_k(1, r)$ as the smallest real number such that the k-dimensional arithmetic product of k $n \times n^r$ matrices can be computed using $O(n^{\omega_k(1,r)+\epsilon})$ operations over the field of reals, for all $\epsilon > 0$. We show that $\omega_k(1, r)$ does not exceed $\omega(\lceil k/2 \rceil, r, \lfloor k/2 \rfloor)$, where $\omega(p, q, r)$ stands for the smallest real number such that an $n^p \times n^q$ matrix can be multiplied by $n^q \times n^r$ matrix using $O(n^{\omega(p,q,r)+\epsilon})$ operations over the field of reals for all $\epsilon > 0$.

If A and B are two Boolean matrices and C is their Boolean matrix product, then for any entry $C[i, j] = 1$ of C, a witness is an index m such that $A[i, m] \wedge B[m, j] = 1$. The smallest (or, largest) possible witness is called the *minimum witness* (or, *maximum witness*, respectively).

The problems of finding "witnesses" have been studied for several decades, mostly within stringology (e.g., algorithms for symbol matches or mismatches [5,6]) and graph algorithms (e.g., algorithms for shortest paths [7,8]). More recently, the problems of finding extreme (i.e., minimum or maximum) witnesses have found numerous applications (e.g., algorithms for all-pairs lowest common ancestors [9] or all-pairs bottleneck weight path problem [10]). Alon and Naour showed that the witness problem for the Boolean matrix product of two Boolean matrices, i.e., the problem of reporting a witness for each non-zero entry of the product, to be solvable in time required to compute the product up to polylogarithmic multiplicative factors [11]. (By the *time* complexity of an algorithm, we mean the number of operations performed by the algorithm in the unit cost random access machine model, which is expressed as a function of an input size parameter. Note that when the inputs to the aforementioned operations are integers of size polynomial in the input size, then the number of required bit operations is larger than that of unit cost operations only by a polylogarithmic factor.) Czumaj et al. showed that the maximum witness problem, i.e., the problem of reporting the maximum witness for each non-zero entry of the Boolean matrix product of two $n \times n$ Boolean matrices, was solvable in $O(n^{2+\lambda+o(1)})$ time [9], where λ satisfies the equation $\omega(1, \lambda, 1) = 1 + 2\lambda$.

We also generalize the concept of a witness for a non-zero entry of the Boolean matrix product of two Boolean matrices to include a witness for a non-zero entry of the k-dimensional Boolean product of k Boolean matrices. For a non-zero entry $D[i_1, \ldots, i_k]$ of the k-dimensional Boolean product of k Boolean matrices $A_1, \ldots, A_k$, of sizes $n_1 \times n, \ldots, n_k \times n$, respectively, a witness is any index ℓ such that $\bigwedge_{j=1}^{k} A_j[i_j, \ell] = 1$. The witness problem for the k-dimensional Boolean matrix product D is to report a witness for each non-zero entry of D and analogously, the maximum witness problem for D is to report the maximum witness for each non-zero entry of D. We show that for $n \times n$ matrices $A_1, \ldots, A_k$, the witness problem for their k-dimensional Boolean product can be solved in $O(n^{\omega(\lceil k/2 \rceil, 1, \lfloor k/2 \rfloor)+o(1)})$ time, while the maximum witness problem for this product admits a solution in $O(n^{k+\lambda+o(1)}) = O(n^{k+\lambda'+o(1)})$ time, where λ satisfies the equation $\omega_k(1, \lambda) = k - 1 + 2\lambda$ and λ' satisfies the equation $\omega(\lceil k/2 \rceil, \lambda', \lfloor k/2 \rfloor) = k - 1 + 2\lambda'$. By the recent results on rectangular matrix multiplication, the latter bound for $k = 2$ is $O(n^{2.529})$.

In the second part of our paper, we provide an alternative simple method of detection/counting copies of fixed size cliques based on the multi-dimensional matrix product. We show also an immediate reduction of the k-dominating set problem to the multi-dimensional matrix product implying the $W[2]$ hardness of computing the product. Finally, we provide an efficient reduction of the problem of finding lowest common ancestors for all k-tuples of vertices in a directed acyclic graph to the problem of finding maximum witnesses of the Boolean variant of the multi-dimensional matrix product. These applications demonstrate the versatility of the multi-dimensional matrix product. Although the time

complexities of the algorithms resulting from the aforementioned reductions solely match those of the known algorithms, the advantage of our algorithms is simplicity.

1.1. Clique Detection

Our first application of the multi-dimensional matrix product provides a simple alternative method for clique detection.

The problems of detecting, finding, counting or listing subgraphs or induced subgraphs of a host graph that are isomorphic to a pattern graph are central in graph algorithms. They are generally termed as *subgraph isomorphism* and *induced subgraph isomorphism*, respectively. Several well-known NP-hard problems such as the independent set, clique, Hamiltonian cycle or Hamiltonian path can be regarded as their special cases.

Recent examples of applications of different variants of subgraph isomorphism include among other things [12,13]: a comparison of bio-molecular networks by their so-called motifs [14], an analysis of social networks by counting the number of copies of a small pattern graph [15], graph-matching constraints in the automatic design of processor systems [16], and the detection of communication patterns between intruders in network security [17]. In the aforementioned applications, the pattern graphs are typically of fixed size which allows for polynomial-time solutions.

At the beginning of the 1980s, Itai and Rodeh [18] presented the following straightforward reduction of not only triangle detection but also triangle counting to fast matrix multiplication. Let A be the $0-1$ adjacency matrix of the host graph G on n vertices (see Preliminaries). Consider the matrix product $C = A \times A$. Note that $C[i,j] = \sum_{l=1}^{n} A[i,l]A[l,j]$ is equal to the number of two-edge paths connecting the vertices i and j. Hence, if $\{i,j\}$ is an edge of G, then $C[i,j]$ is the number of triangles in G including the edge $\{i,j\}$. Consequently, the number of triangles in an n-vertex graph can be reported in $O(n^{\omega+o(1)})$ time.

A few years later, Necetril and Poljak [19] showed an efficient reduction of detecting and counting copies of any pattern graph both in the standard and induced case to the aforementioned method for triangle detection and counting. The idea is to divide the pattern graph into three almost equal parts and to build an auxiliary graph on copies of subgraphs isomorphic to one of three parts. Then, the triangle detection/counting method is run on the auxiliary graph. Two decades later, Eisenbrand and Grandoni [20] (cf. [21]) refined this general triangle method by using fast algorithms for rectangular matrix multiplication instead of those for square matrix multiplication. For a pattern graph on $r \geq 3$ vertices and a host graph on n vertices, the (refined) general triangle method runs in time $O(n^{\omega(\lfloor r/3 \rfloor, \lceil (r-1)/3 \rceil, \lceil r/3 \rceil)+o(1)})$ [19–21].

Up to now, the general triangle method remains the fastest known universal method for the detection and counting standard and induced copies of fixed pattern graphs. In the recent two decades, there has been a real progress in the design of efficient algorithms for the detection and even counting of fixed pattern graphs both in the standard [12,22] and induced case [12,13,23–25]. Among other things, the progress has been based on the use of equations between the numbers of copies of different fixed patterns in the host graph [13,21,22,24] and randomization [12,13,24]. Unfortunately, this progress has not included complete pattern graphs, i.e., K_r graphs (and their complements, i.e., edge-free pattern graphs in the induced setting). For the aforementioned pattern graphs, the generalized triangle method remains the fastest known one.

In this paper, we consider another universal and simple method that in fact can be viewed as another type of generalization of the classic algorithm for triangle detection and counting due to Itai and Rodeh. We can rephrase the description of their algorithm as follows. At the beginning, we form a list of subgraphs isomorphic to K_2 (i.e., edges), and then for each subgraph on the list, we count the number of vertices outside it that are adjacent to both vertices of the subgraph; in other words, we count the number of extensions of the subgraph to a clique on three vertices. The latter task can be completed efficiently by squaring the adjacency matrix of the host graph. We can generalize the algorithm to include detection/counting K_r copies, $r \geq 3$, by replacing K_2 with K_{r-1} and using the $(r-1)$-

dimensional matrix product of $r-1$ copies of the adjacency matrix instead of squaring the matrix. Listing the subgraphs of the host graph isomorphic to K_{r-1} can be completed by enumerating $(r-1)$ vertex subsets and checking if they induce K_{r-1}. For $r = O(1)$, it takes $O(n^{r-1})$ time, so the overall time required by this simple alternative method is $O(n^{r-1} + n^{\omega_{r-1}+o(1)})$, where ω_k is the smallest real number such that the k-dimensional matrix product of k $n \times n$ matrices can be computed using $O(n^{\omega_k+\epsilon})$ operations over the field of reals, for all $\epsilon > 0$ (see Definition 1 in Section 3). Recall that we show in particular $\omega_k \leq \omega(\lceil k/2 \rceil, 1, \lfloor k/2 \rfloor)$. Hence, our alternative method in particular computes the number of K_4 copies in an n-vertex graph in $O(n^{\omega(2,1,1)+o(1)})$ time and the number of K_5 copies in $O(n^{\omega(2,1,2)+o(1)})$ time. In addition, if the input graph contains a copy of K_4 or K_5, respectively, then a copy of K_4 can be found in the graph in $O(n^{\omega(2,1,1)+o(1)})$ time while that of K_5 can be found in the graph in $O(n^{\omega(2,1,2)+o(1)})$ time by a slightly modified alternative method. Thus, our upper time bounds for K_4 and K_5 at least match those for K_4 and K_5 yielded by the generalized triangle method [20]. If $\omega_k < \omega(\lceil k/2 \rceil, 1, \lfloor k/2 \rfloor)$ for k equal to 3 or 4, then we would achieve a breakthrough in the detection/counting of K_4 or K_5, respectively. For K_r, where $r \geq 6$, the generalized triangle method asymptotically subsumes our alternative method and for K_3, the methods coincide.

1.2. Small Dominating Sets

Our second application of the multi-dimensional matrix product is an immediate reduction of the k-dominating set problem to the k-dimensional matrix product. It rephrases the known algorithm for the aforementioned problem due to Eisenbrand and Grandoni [20] (cf. [26]). The k-dominating set problem for a graph G and a fixed natural number k is to determine if there is a set S of at most k vertices in G such that each vertex in G outside S is adjacent to at least one vertex in S. It is a basic $W[2]$ complete problem equivalent to the k-set cover problem asking if there is a set cover of cardinality at most k [27].

1.3. Finding Lowest Common Ancestors

Our third application of the multi-dimensional matrix product provides a simple generalization of known results on finding the lowest common ancestors for all pairs of vertices to include all k-tuples of vertices for fixed $k \geq 2$.

The problem of finding a *lowest common ancestor* (LCA) for sets of vertices in a tree, or more generally, in a *directed acyclic graph* (DAG) is a basic problem in algorithmic graph theory. It has several applications ranging from algorithm design through object-oriented programming languages [28] to phylogenetic networks [29]. An LCA of a set S of vertices in a DAG is an ancestor of all vertices in S that has no proper descendant which is an ancestor of all vertices in S [30]. The problem of preprocessing a DAG such that LCA queries can be answered quickly for any pair of vertices has been studied extensively in the literature [9,31–33]. The all-pairs LCA problem is to compute LCA for all pairs of vertices in the input tree or DAG. For trees, it can be solved in linear time [34]. For DAGs, Bender et al. were the first to provide a substantially subcubic algorithm for this problem [35]. Czumaj et al. improved their upper bound to $O(n^{2.575})$ by a reduction to the problem of finding maximum witnesses for the Boolean matrix product of two $n \times n$ Boolean matrices [9]. Taking into account the recent progress in fast rectangular matrix multiplication, the latter bound reduces to $O(n^{2.529})$. Very recently, Grandoni et al. have presented an $\tilde{O}(n^{2.447})$-time algorithm for the all pairs LCA problem [33].

The problem of finding LCAs for vertex sets of size greater than two is natural, e.g., in a genealogy search in phylogenetic networks [29]. Yuster generalized the all-pairs LCA problem in DAGs to include finding lowest common ancestors for all k-tuples of vertices [36] (cf. [37]). We term the generalized problem as the all k-tuples LCA problem.

We provide a simple reduction of the all k-tuples LCA problem to the maximum witness problem for the k-dimensional Boolean matrix product of k copies of the transitive closure matrix of the input DAG. As a result, we obtain upper time bounds for the all k-tuples LCA problem matching those established by Yuster in [36] and Kowaluk et al. in [37].

1.4. Paper Organization

In the next section, the basic matrix and graph notation used in the paper is presented. Section 3 is devoted to the k-dimensional matrix product of k matrices, in particular the upper time bounds on the product and the related problems of computing witnesses and maximum witnesses for the Boolean version of the product in terms of those for fast rectangular matrix multiplication. In Section 4, the alternative method for detection/counting copies of fixed cliques in a host graph relying on the multi-dimensional matrix product is presented and analyzed. In Section 5, the immediate reduction of the k-dominating set problem to the k-dimensional matrix product is given. In Section 6, the application of the results from Section 3 to the all k-tuples LCA problem is presented. We conclude with final remarks.

2. Preliminaries

For a positive integer r, we shall denote the set of positive integers not greater than r by $[r]$.

For a matrix D, D^T denotes its transpose. Recall that for positive real numbers p, q, s, $\omega(p,q,s)$ denotes the smallest real number such that an $n^p \times n^q$ matrix can be multiplied by $n^q \times n^s$ matrix using $O(n^{\omega(p,q,s)+\epsilon})$ operations over the field of reals for all $\epsilon > 0$. For convenience, ω stands for $\omega(1,1,1)$.

Let α stand for $sup\{0 \leq q \leq 1 : \omega(1,q,1) = 2 + o(1)\}$. The following recent lower bound on α is due to Le Gall and Urrutia [38].

Fact 1. *The inequality* $\alpha > 0.31389$ *holds [38].*

Recall also that a *witness* for a non-zero entry $C[i,j]$ of the Boolean matrix product C of a Boolean $p \times q$ matrix A and a Boolean $q \times s$ matrix B is any index $\ell \in [q]$ such that $A[i,\ell]$ and $B[\ell,j]$ are equal to 1. The *witness problem* is to report a witness for each non-zero entry of the Boolean matrix product of the two input Boolean matrices.

Alon and Naor provided a solution to the witness problem for the Boolean matrix product of two square Boolean matrices [11] which is almost equally fast as that for square matrix multiplication [4]. It can be easily generalized to include the Boolean product of two rectangular Boolean matrices of sizes $n^p \times n^q$ and $n^q \times n^s$, respectively. The asymptotic matrix multiplication time $n^{\omega+o(1)}$ is replaced by $n^{\omega(p,q,s)+o(1)}$ in the generalization.

Fact 2. *For positive* p,q,s*, the witness problem for the Boolean matrix product of an* $n^p \times n^q$ *Boolean matrix with an* $n^q \times n^s$ *Boolean matrix can be solved (deterministically) in* $O(n^{\omega(p,q,s)+o(1)})$ *time.*

A *directed acyclic graph* (DAG) is a directed graph not containing directed cycles. A vertex v is an *ancestor* (*proper ancestor*) of a vertex u in a DAG if there is a directed path (directed path of non-zero length, respectively) from u to v. A vertex u is a *descendant* (*proper descendant*) of a vertex v in a DAG if v is an ancestor (proper ancestor) of u in the DAG. Recall that a *lowest common ancestor* (LCA) of a set of vertices in a DAG is a vertex that is an ancestor of all vertices in the set but none of its proper descendants is also an ancestor of all vertices in the set.

We shall consider also simple undirected graphs.

A *subgraph* of the graph $G = (V,E)$ is a graph $H = (V_H, E_H)$ such that $V_H \subseteq V$ and $E_H \subseteq E$.

An *induced subgraph* of the graph $G = (V,E)$ is a graph $H = (V_H, E_H)$ such that $V_H \subseteq V$ and $E_H = E \cap (V_H \times V_H)$. A subgraph of G *induced by* $S \subseteq V$ is a graph $F = (V_F, E_F)$ such that $V_F = S$ and $E_F = E \cap (S \times S)$. It is denoted by $G[S]$.

For simplicity, we shall refer to a subgraph of a graph G that is isomorphic to K_r as a *copy* of K_r in G or just a K_r copy in G.

The *adjacency matrix* A of a graph $G = (V,E)$ is the $0-1$ $n \times n$ matrix such that $n = |V|$ and for $1 \leq i,j \leq n$, $A[i,j] = 1$ if and only if $\{i,j\} \in E$.

3. Multi-Dimensional Matrix Product and Its Witnesses

For the convenience of the reader, we gather all the definitions related to the multi-dimensional matrix product, repeating some of them, below.

Definition 1. *For k $n_q \times n$ matrices A_q, $q = 1, \ldots, k$, (arithmetic or Boolean, respectively), their k-dimensional (arithmetic or Boolean, respectively) matrix product D is defined by*

$$D[i_1, i_2, \ldots, i_k] = \sum_{\ell=1}^{n} A_1[i_1, \ell] \times A_2[i_2, \ell] \times \ldots \times A_k[i_k, \ell] \text{ or}$$

$$D[i_1, i_2, \ldots, i_k] = \bigvee_{\ell=1}^{n} A_1[i_1, \ell] \wedge A_2[i_2, \ell] \wedge \ldots \wedge A_k[i_k, \ell],$$

respectively, where $i_q \in [n_q]$ for $q = 1, \ldots, k$. In the arithmetic case, $\omega_k(r_1, r_2)$ denotes the smallest real number such that the k-dimensional arithmetic product of k $n^{r_1} \times n^{r_2}$ matrices can be computed using $O(n^{\omega_k(1,r)+\epsilon})$ operations over the field of reals, for all $\epsilon > 0$. For convenience, ω_k stands for $\omega_k(1, 1)$. In the Boolean case, the corresponding numbers are denoted by $\omega_k^B(r_1, r_2)$ and ω_k^B, respectively. Clearly, we have $\omega_k^B(r_1, r_2) \leq \omega_k(r_1, r_2)$ and $\omega_k^B \leq \omega_k$.
A witness for a non-zero entry $D[i_1, i_2, \ldots, i_k]$ of the k-dimensional Boolean matrix product is any index $\ell \in [n]$ such that $A_1[i_1, \ell] \wedge A_2[i_2, \ell] \wedge \ldots \wedge A_k[i_k, \ell]$ is equal to (Boolean) 1. The witness problem for the k-dimensional Boolean matrix product is to report a witness for each non-zero entry of the product. The maximum witness problem for the k-dimensional Boolean matrix product is to report the maximum witness for each non-zero entry of the product.

Note that in particular, the *two*-dimensional matrix product of the matrices A_1 and A_2 coincides with the standard matrix product of A_1 and $(A_2)^T$ which yields $\omega_2 = \omega$.

The following lemma provides an upper bound on the time complexity of the multi-dimensional matrix product in terms of those for rectangular matrix multiplication.

Lemma 1. *Let k, k_1, k_2 be three positive integers such that $k = k_1 + k_2$. Both in the arithmetic and Boolean case, the k-dimensional matrix product of k $n \times n^r$ matrices can be computed using $O(n^{\omega(k_1,r,k_2)+o(1)})$ arithmetic operations, consequently $\omega_k \leq \omega(k_1, 1, k_2)$. In addition, in the Boolean case, the witness problem for the k-dimensional matrix product can be solved in $O(n^{\omega(k_1,r,k_2)+o(1)})$ time.*

Proof. To prove the first part, it is sufficient to consider the arithmetic case as the Boolean one trivially reduces to it.

Let $A_1, \ldots, A_k$ be the input matrices. Form an $n^{k_1} \times n^r$ matrix A whose rows are indexed by k_1-tuples of indices in $[n]$ and whose columns are indexed by indices in $[n^r]$ such that $A[i_1,\ldots,i_{k_1}, \ell] = A_1[i_1, \ell] \times \ldots \times A_{k_1}[i_{k_1}, \ell]$. Similarly, form an $n^{k_2} \times n^r$ matrix B whose rows are indexed by k_2-tuples of indices in $[n]$ and whose columns are indexed by indices in $[n^r]$ such that $B[j_1,\ldots,j_{k_2}, \ell] = A_{k_1+1}[j_1, \ell] \times \ldots \times A_k[j_{k_2}, \ell]$. Compute the rectangular matrix product C of the matrix A with the matrix B^T. By the definitions, the $D[i_1, \ldots, i_{k_1}, i_{k_1+1}, \ldots, i_k]$ entry of the k-dimensional product of the input matrices $A_1, \ldots, A_k$ is equal to the entry $C[i_1,\ldots,i_{k_1}, i_{k_1+1},\ldots,i_k]$. The matrices A, B can be formed in $O(n^{k_1+r} + n^{k_2+r})$ time, i.e., $O(n^{\max(k_1+r,k_2+r)})$ time, while the product C can be computed in $O(n^{\omega(k_1,r,k_2)+o(1)})$ time.

To prove the second part of the lemma, it is sufficient to consider Boolean versions of the matrices A, B, C and use Fact 2. □

By combining Lemma 1 with Fact 1, we obtain the following corollary.

Corollary 1. *For even $k \geq 8$, $\omega_k = k + o(1)$.*

Proof. We obtain the following chain of equations on the asymptotic time required by the k-dimensional matrix product using Lemma 1 and Fact 1:

$$n^{\omega(k/2,1,k/2)+o(1)} = (n^{k/2})^{\omega(1,2/k,1)+o(1)} = (n^{k/2})^{2+o(1)} = n^{k+o(1)}.$$

□

Consider the Boolean case. By the definition, the k-dimensional Boolean matrix product can be computed in time proportional to the size of the product multiplied by the size of witness range. By generalizing the column–row method for the Boolean matrix product of two Boolean matrices, we obtain an output-sensitive upper bound on the time required to compute the k-dimensional Boolean matrix product.

Theorem 1. *Suppose $k \geq 2$ and $k = O(1)$. Let w be the total number of witnesses for the entries of the k-dimensional Boolean product of k $n \times n$ Boolean matrices. The non-zero entries of the product can be listed in $O(n^2 + w)$ time. Consequently, they can be listed in $O(n^2 + sn)$ time, where s is the number of non-zero entries in the product.*

Proof. Let D stand for the k-dimensional Boolean matrix product. For $m = 1, \ldots, k$, $\ell = 1, \ldots, n$, compute the set $S_m(\ell)$ of indices i such that $A_m[i, \ell] = 1$. Next, for $\ell = 1, \ldots, n$, compute the Cartesian product $S(\ell) = S_1(\ell) \times \ldots \times S_k(\ell)$. Note that $S(\ell)$ coincides with the set of k-tuples $i_1, \ldots, i_k$ such that ℓ is a witness for $D[i_1, \ldots, i_k]$.

Assuming $k = O(1)$, we can compute the sets $S_m(\ell)$ in $O(n^2)$ time. The sets $S(\ell)$ can be computed in $O(w + n)$ time. On their basis, the non-negative entries of the product D can be listed in $O(w)$ time. This proves the first part. The second part follows immediately from the fact that a non-zero entry of D can have at most n witnesses. □

The best known algorithm for the maximum witness problem for the two-dimensional Boolean matrix product from [9] relies on the multiplication of rectangular submatrices of the input matrices. We generalize this idea to include the maximum witness problem for the k-dimensional Boolean matrix product.

First, each of the input matrices is divided into vertical strip submatrices of an appropriate width $\leq \ell$. Then, the multi-dimensional Boolean products of the corresponding submatrices are computed in order to detect, for each positive entry of the multi-dimensional output matrix, the interval of length $\leq \ell$ containing the maximum witness. Finally, the maximum witnesses are found separately for each of the positive entries by verifying the respective $\leq \ell$ candidates. More details can be found in the following pseudo-code (in Algorithm 1).

Algorithm 1 Generalize the multiplication of rectangular submatrices of the input matrices to include the maximum witness problem for the k-dimensional Boolean matrix product.

Input: Boolean $n \times n$ matrices A_i, $i \in [k]$, and a parameter $\ell \in [n]$.
Output: maximum witnesses for all non-zero entries of the k-dimensional Boolean product of the matrices $A_1, \ldots, A_k$ and "No" for all zero entries of the product.

1. For $i \in [k]$, divide A_i into $\lceil n/\ell \rceil$ vertical strip submatrices $A_i^1, \ldots, A_i^{\lceil n/\ell \rceil}$ of width ℓ with the exception of the last one that can have width $\leq \ell$.
2. **for** $p \in [\lceil n/\ell \rceil]$ compute the k-dimensional Boolean product B_p of $A_1^p, \ldots, A_k^p$
3. **for** all $i_1, \ldots, i_k \in [n]$ **do**
   (a) Find the largest p such that $B_p[i_1, \ldots, i_k] = 1$ or set $p = 0$ if it does not exist.
   (b) **if** $p > 0$ **then** return $\ell(p-1) + \max\{r \in [\ell(p-1)+1, \ell p] | A_1^p[i_1, r] \wedge \ldots \wedge A_k^p[i_k, r] = 1\}$ **else** return "No"

The correctness of Algorithm 1 is obvious. By its time analysis and Lemma 1, we obtain the following theorem.

Theorem 2. *Suppose $k \geq 2$ and $k = O(1)$. For any $\ell \in [n]$, the maximum witness problem for the k-dimensional Boolean product of k Boolean $n \times n$ matrices can be solved by Algorithm 1 in time $O((n/\ell)n^{\omega_k(1,\log_n \ell)+o(1)} + n^{k+1}/\ell + n^k\ell) \leq O((n/\ell)n^{\omega(\lceil k/2\rceil,\log_n \ell,\lfloor k/2\rfloor)+o(1)} + n^{k+1}/\ell + n^k\ell)$.*

Proof. Step 1 takes $O(n^2)$ time. Step 2 requires $O((n/\ell)n^{\omega_k(1,\log_n \ell)+o(1)})$ time. Step 3(a) takes $O(n^k \times n/\ell)$ time totally. Finally, Step 3(b) requires $O(n^k\ell)$ time totally. □

By Theorem 2, the total time taken by Algorithm 1 for maximum witnesses is

$$O((n/\ell) \cdot n^{\omega_k(1,\log_n \ell)+o(1)} + n^{k+1}/\ell + n^k\,\ell).$$

By setting r to $\log_n \ell$, our upper bound transforms to $O(n^{1-r+\omega_k(1,r)+o(1)} + n^{k+1-r} + n^{k+r})$. Since a solution λ to the equation $1 - r + \omega_k(1,r) = k + r$ satisfies $\lambda \geq \frac{1}{2}$ by $\omega_k(1,\lambda) \geq k$, we can get rid of the additive n^{k+1-r} term. In addition, it follows from $\omega_k(1,\lambda) \leq \omega(\lceil k/2\rceil,\lambda,\lfloor k/2\rfloor)$ that $\lambda \leq \lambda'$, where λ' satisfies $\omega(\lceil k/2\rceil,\lambda',\lfloor k/2\rfloor) = k - 1 + 2\lambda'$. Hence, we obtain the following result.

Theorem 3. *Let λ satisfy $\omega_k(1,\lambda) = k - 1 + 2\lambda$ and let λ' satisfy $\omega(\lceil k/2\rceil,\lambda',\lfloor k/2\rfloor) = k - 1 + 2\lambda'$, respectively. Suppose $k = O(1)$. The maximum witnesses for all non-zero entries of the k-dimensional Boolean product of k $n \times n$ Boolean matrices can be computed by Algorithm 1 in $O(n^{k+\lambda+o(1)}) = O(n^{k+\lambda'+o(1)})$ time.*

4. Breaking the Hegemony of the Triangle Method in Clique Detection

The following Algorithm 2 is a straightforward generalization of that due to Itai and Rodeh for triangle counting [18] to include K_r counting, for $r \geq 3$.

Algorithm 2 A straightforward generalization of that due to Itai and Rodeh for triangle counting [18] to include K_r counting, for $r \geq 3$.

1. form a list L of all K_{r-1} copies in G
2. $t \leftarrow 0$
3. **for** each $C \in L$ **do**
   increase t by the number of vertices in G that are adjacent to all vertices of C
4. return t/r

The correctness of Algorithm 2 follows from the fact that the number of K_r copies including a given copy C of K_{r-1} in the host graph is equal to the number of vertices outside C in the graph that are adjacent to all vertices in C and that a copy of K_r includes exactly r distinct copies of K_{r-1} in the graph.

The first step of Algorithm 2 can be implemented in $O(n^{r-1})$ time. We can use the $(r-1)$-dimensional matrix product to implement the third step by using the next lemma. It immediately follows from the definition of the product.

Lemma 2. *Let D be the k-dimensional matrix product of k copies of the adjacency matrix of the input graph G on n vertices. Then, for any k tuple $i_1, i_2, \ldots, i_k$ of vertices of G, the number of vertices in G adjacent to each vertex in the k tuple is equal to $D[i_1, i_2, \ldots, i_k]$.*

By the discussion and Lemma 2, we obtain the following theorem.

Theorem 4. *The number of K_r copies in the input graph on n vertices can be computed (by Algorithm 2) in $O(n^{r-1} + n^{\omega_{r-1}+o(1)})$ time.*

By Lemma 1, we obtain the following corollary from Theorem 4, matching the upper time bounds on the detection/counting copies of K_4 and K_5 established in [20].

Corollary 2. *The number of K_4 copies in an n-vertex graph can be computed (by Algorithm 2) in $O(n^{\omega(2,1,1)+o(1)})$ time while the number of K_5 copies in an n-vertex graph can be computed (by Algorithm 2) in $O(n^{\omega(2,1,2)+o(1)})$ time. In addition, if the input graph contains a copy of K_4 or K_5, respectively, then a copy of K_4 can be found in the graph in $O(n^{\omega(2,1,1)+o(1)})$ time while that of K_5 ca n be found in the graph in $O(n^{\omega(2,1,2)+o(1)})$ time (by a modification of Algorithm 2).*

5. *k*-Dominating Set

The problem of the *k*-dominating set is to determine, for a graph G and a fixed natural number k, if there is a set S of k vertices in G such that each vertex in G either is adjacent to at least one vertex in S or belongs to S. It is a basic $W[2]$ complete problem equivalent to the *k*-set cover [27].

Eisenbrand and Grandoni improved on the straightforward enumeration algorithm by using fast rectangular matrix multiplication [20] (cf. [26]). Their Algorithm 3 can be simply rephrased in terms of a *k*-dimensional matrix product as follows.

Algorithm 3 Improved straightforward enumeration algorithm by using fast rectangular matrix multiplication.

Input: A graph G on n vertices and a natural number $k \geq 2$.
Output: A solution to the *k*-dominating set problem in G.

1. $\bar{A} \leftarrow$ the complement of the adjacency matrix of G
2. $D \leftarrow$ the k dimensional matrix product of k copies of $\bar{A}$
3. **if** if there is a zero entry $D[i_1, \ldots, i_k]$ **then** output "Yes" **else** output "No"

It is easy to see that for a set $S = \{v_{i_1}, \ldots, v_{i_k}\}$ of k vertices in G, $D[i_1, \ldots, i_k]$ is just the number of vertices that are not adjacent to (dominated by) any vertex in S. Thus, S is a *k*-dominating set if $D[i_1, \ldots, i_k] = 0$. Hence, assuming that n is the number of vertices in G, we obtain the following upper bound given in [20] by a simple application of Lemma 1.

Theorem 5. *Algorithm 3 solves the k-dominating set problem in $O(n^{\omega(\lceil k/2 \rceil, 1, \lfloor k/2 \rfloor)+o(1)})$ time.*

The reduction of the *k*-dominating set problem to the *k*-dimensional matrix product given in Algorithm 3 implies the $W[2]$ hardness of the problem of computing the *k*-dimensional matrix product. By Theorem 5.4 in [39], we obtain the following corollary.

Corollary 3. *The problem of computing the k-dimensional Boolean matrix product is not solvable in $f(k)n^{O(k)}$ time, for any function f, unless $FPT = W[1]$.*

6. All *k*-Tuples LCA

In this section, we consider the lowest common ancestors of $k \geq 2$ vertices [30]. For an example, see Figure 1.

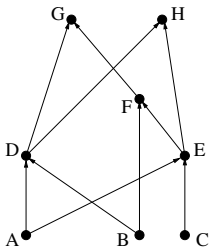

Figure 1. Both F and H are LCA for the 3-tuple A, B, C while G does not have this property.

Recall that an LCA of vertices $v_1, \ldots, v_k$ is an ancestor v of each vertex v_i, $i \in [k]$ such that v has no proper descendant that is also an ancestor of each vertex v_i, $i \in [k]$.

We can generalize the method of solving the all pairs LCA problem in DAGs based on maximum witnesses of Boolean matrix products from [9] to include the all *k*-tuples LCA

problem, i.e., reporting for each k-tuple of vertices in the input DAG an LCA, as follows (in Algorithm 4).

Algorithm 4 Reporting for each k-tuple of vertices in the input DAG an LCA.

Input: A DAG G on n vertices and a natural number $k \geq 2$.
Output: A solution to the all k-tuples LCA problem in the input DAG.

1. Topologically sort vertices of the input DAG G and number them in the sorted order starting from the source vertices.
2. Compute the $n \times n$ transitive closure matrix A of G such that $A[i, j] = 1$ if and only if the vertex j is an ancestor of the vertex i.
3. Compute maximum witnesses for non-zero entries of the k-dimensional Boolean matrix product D of k copies of the Boolean version of the matrix A.
4. For each k-tuple $i_1, i_2, \ldots, i_k$ of vertices, output the maximum witness of $D[i_1, i_2, \ldots, i_k]$ (if any) as the LCA of the k-tuple.

The correctness of this generalized method follows from the fact that by the definition of the matrix A and the k-dimensional Boolean matrix product D, the set of witnesses of $D[i_1, i_2, \ldots, i_k]$ coincides with the set of common ancestors of $i_1, i_2, \ldots, i_k$, and that the maximum witness of of $D[i_1, i_2, \ldots, i_k]$ cannot have any descendant in the aforementioned set by the vertex numbering.

The topological sort requires $O(n^2)$ time while computing the transitive closure takes $O(n^{\omega+o(1)})$ time. Hence, by Theorem 3, we obtain upper bounds basically matching those derived by Yuster in [36] and Kowaluk et al. in [37].

Theorem 6. *Let λ satisfy $\omega_k(1, \lambda) = k - 1 + 2\lambda$ and let λ' satisfy $\omega(\lceil k/2 \rceil, \lambda', \lfloor k/2 \rfloor) = k - 1 + 2\lambda'$, respectively. Suppose $k = O(1)$. The all k-tuples LCA problem can be solved by Algorithm 4 in $O(n^{k+\lambda+o(1)}) = O(n^{k+\lambda'+o(1)})$ time.*

Corollary 4. *Let λ, λ' and λ'' satisfy $\omega_3(1, \lambda) = 2 + 2\lambda$, $\omega(2, \lambda', 1) = 2 + 2\lambda'$, and $\omega(1, \lambda'', 1) = 1 + 2\lambda''$, respectively. The all 3-tuples LCA problem can be solved in $O(n^{3+\lambda+o(1)}) = O(n^{3+\lambda'+o(1)})$ $= (n^{3+\lambda''+o(1)}) = O(n^{3.529})$ time.*

Proof. By dividing an $n^2 \times n^r$ matrix into n $n \times n^r$ matrices, we obtain the inequality $1 + \omega(1, r, 1) \geq \omega(2, r, 1)$. The inequality $\lambda'' \geq \lambda'$ follows from $1 + \omega(1, r, 1) \geq \omega(2, r, 1)$ similarly as $\lambda' \geq \lambda$ follows from $\omega_3(1, r) \leq \omega(2, r, 1)$ in the proof of Theorem 3. By the recent results on fast rectangular multiplication, $\lambda'' < 0.529$ holds. □

7. Final Remarks

We have applied our results on the multi-dimensional matrix product to provide a simple alternative method for clique detection and to rephrase known algorithms for k-dominating set problem and the all k-tuples LCA problem in DAGs. Our direct reduction of the k-dominating set problem to the k-dimensional matrix product yields the $W[2]$ hardness of the latter problem. In the context of the reduction, note that the problem of computing the k-dimensional (even Boolean) matrix product is more general than the k-dominating set problem, since the input matrices do not have to be identical as in the reduction.

It is an intriguing open problem if the upper bounds in terms of rectangular matrix multiplication on the k-dimensional matrix product of k square matrices given in Lemma 1 are asymptotically tight. In other words, the question is if $\omega_k = \min_{k'=1}^{k-1} \omega(k', 1, k - k')$ holds or more specifically if $\omega_k = \omega(\lceil k/2 \rceil, 1, \lfloor k/2 \rfloor)$? If this was not the case for k equal to 3 or 4, then we would achieve a breakthrough in detection/counting of K_4 or K_5, respectively.

An argument for the inequality $\omega_k < \omega(k_1, 1, k_2)$, for positive integers k_1, k_2 satisfying $k = k_1 + k_2$, is that in the context of the efficient reduction in the proof of Lemma 1, the rectangular matrix product seems more general than the k-dimensional one. A reverse efficient reduction seems to be possible only under very special assumptions. However,

proving such an inequality would be extremely hard as it would imply $\omega(k_1, 1, k_2) > k$ and in consequence $\omega > 2$ by the straightforward reduction of the rectangular matrix product to the square one. On the other hand, this does not exclude the possibility of establishing better upper bounds on ω_k than those known on $\omega(k_1, 1, k_2)$.

A related question is whether or not λ satisfying the equation $\omega_k(1, \lambda) = k - 1 + 2\lambda$ is equal or less than λ' satisfying the equation $\omega(\lceil k/2 \rceil, \lambda, \lfloor k/2 \rfloor) = k - 1 + 2\lambda$? An inequality for $k = 2$ would yield the improvement of the longstanding upper time bound for the maximum witness problem of the Boolean product of two $n \times n$ Boolean matrices from $O(n^{2+\lambda'})$ to $O(n^{2+\lambda})$.

These two open problems and the quest for finding further applications of the multi-dimensional matrix product form the future directions of our research.

Author Contributions: Conceptualization, M.K. and A.L.; Formal analysis, M.K. and A.L.; Writing—original draft, M.K.; Writing—review & editing, A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by Swedish Research Council grant 621-2017-03750.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Paterson, M. Complexity of Monotone Networks for Boolean Matrix Product. *Theor. Comput. Sci.* **1975**, *1*, 13–20. [CrossRef]
2. Schnorr, C.-P. A Lower Bound on the Number of Additions in Monotone Computations. *Theor. Comput. Sci.* **1976**, *2*, 305–315. [CrossRef]
3. Strassen, V. Gaussian elimination is not optimal. *Numer. Math.* **1969**, *13*, 354–356. [CrossRef]
4. Alman, J.; Williams, V.V. A Refined Laser Method and Faster Matrix Multiplication. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), Virtual, 10–13 January 2021; pp. 522–539.
5. Aumman, N.; Levenstein, M.; Levenstein, N.; Tsur, D. Finding witnesses by peeling. In Proceedings of the Combinatorial Pattern Matching (CPM), London, ON, Canada, 9–11 July 2007; Volume 4580, pp. 28–39.
6. Muthukrshnan, S. New results and open problems related to non-standard stringology. In Proceedings of the 6th Combinatorial Pattern Matching (CPM), Espoo, Finland, 5–7 July 1995; pp. 298–317.
7. Alon, N.; Galil, Z.; Margalit, O.; Naor, M. Witnesses for Boolean matrix multiplication and for shortest paths. In Proceedings of the 33rd Symposium on Foundations of Computer Science (FOCS), Pittsburgh, PA, USA, 24–27 October 1992; pp. 417–426.
8. Zwick, U. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM* **2002**, *49*, 289–317. [CrossRef]
9. Czumaj, A.; Kowaluk, M.; Lingas, A. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theor. Comput. Sci.* **2007**, *380*, 37–46. [CrossRef]
10. Shapira, A.; Yuster, R.; Zwick, U. All-Pairs Bottleneck Paths in Vertex Weighted Graphs. *Algorithmica* **2011**, *59*, 621–633. [CrossRef]
11. Alon, N.; Naor, M. Derandomization, witnesses for Boolean matrix multiplication and construction of perfect hash functions. *Algorithmica* **1996**, *16*, 434–449. [CrossRef]
12. Floderus, P.; Kowaluk, M.; Lingas, A.; Lundell, E.-M. Detecting and Counting Small Pattern Graphs. *SIAM J. Discret. Math.* **2015**, *29*, 1322–1339. [CrossRef]
13. Williams, V.V.; Wang, J.R.; Williams, R.; Yu H. Finding Four-Node Subgraphs in Triangle Time. In Proceedings of the SODA, San Diego, CA, USA, 4–6 January 2015; pp. 1671–1680.
14. Alon, N.; Dao, P.; Hajirasouliha, I.; Hormozdiari, F.; Sahinalp, S.C. Biomolecular network motif counting and discovery by color coding. *Bioinformatics* **2008**, *24*, 241–249. [CrossRef]
15. Schank, T.; Wagner, D. Finding, Counting and Listing All Triangles in Large Graphs, an Experimental Study. In *Experimental and Efficient Algorithms, WEA 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 606–609.
16. Wolinski, C.; Kuchcinski, K.; Raffin, E. Automatic Design of Application-Specific Reconfigurable Processor Extensions with UPaK Synthesis Kernel. *ACM Trans. Des. Autom. Electron. Syst.* **2009**, *15*, 1–36. [CrossRef]
17. Sekar, V.; Xie, Y.; Maltz, D.A.; Reiter, M.K.; Zhang, H. Toward a framework for internet forensic analysis. In Proceedings of the Third Workshop on Hot Topics in Networking (HotNets-HI), San Diego, CA USA, 15–16 November 2004.
18. Itai, A.; Rodeh, M. Finding a minimum circuit in a graph. *SIAM J. Comput.* **1978**, *7*, 413–423. [CrossRef]
19. Nešetřil, J.; Poljak, S. On the complexity of the subgraph problem. *Comment. Math. Univ. Carol.* **1985**, *26*, 415–419.
20. Eisenbrand, F.; Grandoni, F. On the complexity of fixed parameter clique and dominating set. *Theor. Comput. Sci.* **2004**, *326*, 57–67. [CrossRef]
21. Kloks, T.; Kratsch, D.; Müller, H. Finding and counting small induced subgraphs efficiently. *Inf. Process. Lett.* **2000**, *74*, 115–121. [CrossRef]

22. Kowaluk, M.; Lingas, A.; Lundell, E.-M. Counting and detecting small subgraphs via equations and matrix multiplication. *SIAM J. Discret. Math.* **2013**, *27*, 892–909. [CrossRef]
23. Bläser, M.; Komarath, B.; Sreenivasaiah, K. Graph Pattern Polynomials. *arXiv* **2018**, arXiv:1809.08858.
24. Dalirrooyfard, M.; Vuong, T.D.; Williams, V.V. Graph pattern detection: Hardness for all induced patterns and faster non-induced cycles. In Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, Phoenix, AZ, USA, 23–26 June 2019; pp. 1167–1178.
25. Floderus, P.; Kowaluk, M.; Lingas, Ã.; Lundell, E.-M. Induced subgraph isomorphism: Are some patterns substantially easier than others? *Theor. Comput. Sci.* **2015**, *605*, 119–128. [CrossRef]
26. Patrascu, M.; Williams, R. On the Possibility of Faster SAT Algorithms. In Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2010), Austin, TX, USA, 17–19 January 2010; pp. 1065–1075.
27. Downey, R.G.; Fellows, M.R. *Parameterized Complexity*; Springer: Berlin/Heidelberg, Germany, 1999.
28. Ducournau, R.; Habib, M. On some algorithms for multiple inheritance in object-oriented programming. In Proceedings of the ECOOP' 87 European Conference on Object-Oriented Programming, Paris, France, 15–17 June 1987; Volume 276, pp. 243–252.
29. Fischer, J.; Huson, D.H. New common ancestor problems in trees and directed acyclic graphs. *Inf. Process. Lett.* **2010**, *110*, 331–335. [CrossRef]
30. Aho, A.V.; Hopcroft, J.E.; Ullman, J.D. On finding lowest common ancestors in trees. *SIAM J. Comput.* **1976**, *5*, 115–132. [CrossRef]
31. Bender, M.A.; Farach-Colton, M. The LCA problem revisited. In Proceedings of the 4th Latin American Symposium on Theoretical Informatics (LATIN'00), Punta del Este, Uruguay, 10–14 April 2000; pp. 88–93.
32. Eckhardt, S.; Mühling, A.M.; Nowak, J. Fast lowest common ancestor computations in dags. In Proceedings of the 15th Annual European Symposium on Algorithms (ESA), Eilat, Israel, 8–10 October 2007; pp. 705–716.
33. Grandoni, F.; Italiano, G.F.; Lukasiewicz, A.; Parotsidis, N.; Uznanski, P. All-Pairs LCA in DAGs: Breaking through the $O(n^{2.5})$ barrier. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), Virtual, 10–13 January 2021; pp. 273–289.
34. Harel, D.; Tarjan, R.E. Fast algorithms for finding nearest common ancestors. *SIAM J. Comput.* **1984**, *13*, 338–355. [CrossRef]
35. Bender, M.A.; Farach-Colton, M.; Pemmasani, G.; Skiena, S.; Sumazin, P. Lowest common ancestors in trees and directed acyclic graphs. *J. Algorithms* **2005**, *57*, 75–94. [CrossRef]
36. Yuster, R. All-Pairs Disjoint Paths from a Common Ancestor in $\tilde{0}(n^{\omega})$ Time. *Theor. Comput. Sci.* **2008**, *396*, 145–150. [CrossRef]
37. Kowaluk, M.; Lingas, A.; Nowak, J. A Path Cover Technique for LCAs in Dags. In Proceedings of the 11th Scandinavian Workshop on Algorithm Theory (SWAT 2008), Gothenburg, Sweden, 2–4 July 2008; pp. 222–233.
38. Le Gall, F.; Urrutia, F. Improved Rectangular Matrix Multiplication using Powers of the Coppersmith-Winograd Tensor. In Proceedings of the SODA, New Orleans, LA, USA, 7–10 January 2018; pp. 1029–1046.
39. Chen, J.; Huang, X.; Kanj, I.A.; Xia, G. Strong computational lower bounds via parameterized complexity. *J. Comput. Syst. Sci.* **2006**, *72*, 1346–1367. [CrossRef]