Further Algebraic Algorithms in the Congested Clique Model and Applications to Graph-Theoretic Problems

François Le Gall(✉)

Graduate School of Informatics, Kyoto University, Kyoto, Japan
legall@i.u-kyoto.ac.jp

Abstract. Censor-Hillel et al. [PODC'15] recently showed how to efficiently implement centralized algebraic algorithms for matrix multiplication in the congested clique model, a model of distributed computing that has received increasing attention in the past few years. This paper develops further algebraic techniques for designing algorithms in this model. We present deterministic and randomized algorithms, in the congested clique model, for efficiently computing multiple independent instances of matrix products, computing the determinant, the rank and the inverse of a matrix, and solving systems of linear equations. As applications of these techniques, we obtain more efficient algorithms for the computation, again in the congested clique model, of the all-pairs shortest paths and the diameter in directed and undirected graphs with small weights, improving over Censor-Hillel et al.'s work. We also obtain algorithms for several other graph-theoretic problems such as computing the number of edges in a maximum matching and the Gallai-Edmonds decomposition of a simple graph, and computing a minimum vertex cover of a bipartite graph.

1 Introduction

Background. The congested clique model is a model in distributed computing that has recently received increasing attention [2,6–11,17–19,22,23]. In this model n nodes communicate with each other over a fully-connected network (i.e., a clique) by exchanging messages of size $O(\log n)$ in synchronous rounds. Compared with the more traditional congested model [24], the congested clique model removes the effect of distances in the computation and thus focuses solely on understanding the role of congestion in distributed computing.

Typical computational tasks studied in the congested clique model are graph-theoretic problems [2,6–8,11,22], where a graph G on n vertices is initially distributed among the n nodes of the network (the ℓ-th node of the network knows the set of vertices adjacent to the ℓ-th vertex of the graph, and the weights of the corresponding edges if the graph is weighted) and the nodes want to compute properties of G. Besides their theoretical interest and potential applications, such problems have the following natural interpretation in the congested clique

C. Gavoille and D. Ilcinkas (Eds.): DISC 2016, LNCS 9888, pp. 57–70, 2016.
DOI: 10.1007/978-3-662-53426-7_5

model: the graph G represents the actual topology of the network, each node knows only its neighbors but can communicate to all the nodes of the network, and the nodes want to learn information about the topology of the network.

Censor-Hillel et al. [2] recently developed algorithms for several graph-theoretic problems in the congested clique model by showing how to implement centralized algebraic algorithms for matrix multiplication in this model. More precisely, they constructed a $O(n^{1-2/\omega})$-round algorithm for matrix multiplication, where ω denotes the exponent of matrix multiplication (the best known upper bound on ω is $\omega < 2.3729$, obtained in [16,29], which gives exponent $1-2/\omega < 0.1572$ in the congested clique model), improving over the $O(n^{2-\omega})$ algorithm mentioned in [7], in the following setting: given two $n \times n$ matrices A and B over a field, the ℓ-th node of the network initially owns the ℓ-th row of A and the ℓ-column of B, and needs to output the ℓ-th row and the ℓ-column of the product AB. Censor-Hillel et al. consequently obtained $O(n^{1-2/\omega})$-round algorithms for several graph-theoretic tasks that reduce to computing the powers of (some variant of) the adjacency matrix of the graph, such as counting the number of triangles in a graph (which lead to an improvement over the prior best algorithms for this task [6,7]), detecting the existence of a constant-length cycle and approximating the all-pairs shortest paths in the input graph (improving the round complexity obtained in [22]). One of the main advantages of such an algebraic approach in the congested clique model is its versatility: it makes possible to construct fast algorithms for graph-theoretic problems, and especially for problems for which the best non-algebraic centralized algorithm is highly sequential and does not seem to be implementable efficiently in the congested clique model, simply by showing a reduction to matrix multiplication (and naturally also showing that this reduction can be implemented efficiently in the congested clique model).

Our results. In this paper we develop additional algebraic tools for the congested clique model.

We first consider the task of computing in the congested clique model not only one matrix product, but multiple independent matrix products. More precisely, given k matrices $A_1, \ldots, A_k$ each of size $n \times m$ and k matrices $B_1, \ldots, B_k$ each of size $m \times m$, initially evenly distributed among the n nodes of the network, the nodes want to compute the k matrix products $A_1B_1, \ldots, A_kB_k$. Prior works [2,7] considered only the case $k = 1$ and $m = n$, i.e., one product of two square matrices. Our contribution is thus twofold: we consider the rectangular case, and the case of several matrix products as well. Let us first discuss our results for square matrices ($m = n$). By using sequentially k times the matrix multiplication algorithm from [2], k matrix products can naturally be computed in $O(kn^{1-2/\omega})$ rounds. In this work we show that we can actually do better.

Theorem 1 (Simplified version). *In the congested clique model k independent products of pairs of $n \times n$ matrices can be computed with round complexity*

$$\begin{cases} O(k^{2/\omega}n^{1-2/\omega}) & if \ \ 1 \leq k < n, \\ O(k) & if \ \ k \geq n. \end{cases}$$

This generalization of the results from [2] follows from a simple strategy: divide the n nodes of the network into k blocks (when $k \leq n$), each containing roughly n/k nodes, compute one of the k matrix products per block by using an approach similar to [2] (i.e., a distributed version of the best centralized algorithm computing one instance of square matrix multiplication), and finally distribute the relevant part of the k output matrices to all the nodes of the network. Analyzing the resulting protocol shows that the dependence in k in the overall round complexity is reduced to $k^{2/\omega}$. This sublinear dependence in k has a significant number of implications (see below).

The complete version of Theorem 1, given in Sect. 3, also considers the general case where the matrices may not be square (i.e., the case $m \neq n$), which will be crucial for some of our applications to the All-Pairs Shortest Path problem. The proof becomes more technical than for the square case, but is conceptually very similar: the main modification is simply to now implement a distributed version of the best centralized algorithm for rectangular matrix multiplication. The upper bounds obtained on the round complexity depend on the complexity of the best centralized algorithms for rectangular matrix multiplication (in particular the upper bounds given in [15]). While the major open problem is still whether the product of two square matrices can be computed in a constant (or nearly constant) number of rounds, our results show that for $m = O(n^{0.651\cdots})$, the product of an $n \times m$ matrix by an $m \times n$ matrix can indeed be computed in $O(n^{\epsilon})$ rounds for any $\epsilon > 0$. We also show lower bounds on the round complexity of the general case (Proposition 1 in Sect. 3), which are tight for most values of k and m, based on simple arguments from communication complexity.

We then study the following basic problems in linear algebra: computing the determinant, the rank or the inverse of an $n \times n$ matrix over a finite field $\mathbb{F}$ of order upper bounded by a polynomial of n, and solving a system of n linear equations and n variables. We call these problems $\mathsf{DET}(n, \mathbb{F})$, $\mathsf{Rank}(n, \mathbb{F})$, $\mathsf{INV}(n, \mathbb{F})$ and $\mathsf{SYS}(n, \mathbb{F})$, respectively (the formal definitions are given in Sect. 2). While it is known that in the centralized setting these problems can be solved with essentially the same time complexity as matrix multiplication [1], these reductions are typically sequential and do not work in a parallel setting. In this paper we design fast deterministic and randomized algorithm for these four basis tasks, and obtain the following results.

Theorem 2. *Assume that $\mathbb{F}$ has characteristic greater than n. In the congested clique model, the deterministic round complexity of $\mathsf{DET}(n, \mathbb{F})$ and $\mathsf{INV}(n, \mathbb{F})$ is $O(n^{1-1/\omega})$.*

Theorem 3. *Assume that $\mathbb{F}$ has order $|\mathbb{F}| = \Omega(n^2 \log n)$. In the congested clique model, the randomized round complexity of $\mathsf{DET}(n, \mathbb{F})$, $\mathsf{SYS}(n, \mathbb{F})$ and $\mathsf{Rank}(n, \mathbb{F})$ is $O(n^{1-2/\omega} \log n)$.*

The upper bounds of Theorems 2 and 3 are $O(n^{0.5786})$ and $O(n^{0.1572})$, respectively, by basing our implementation on the asymptotically fastest (but impractical) centralized algorithm for matrix multiplication corresponding to the upper

bound $\omega < 2.3729$. These bounds are $O(n^{2/3})$ and $O(n^{1/3} \log n)$, respectively, by basing our implementation on the trivial (but practical) centralized algorithm for matrix multiplication (corresponding to the bound $\omega \leq 3$). These algorithms are obtained by carefully adapting to the congested clique model the relevant known parallel algorithms [5,12–14,25] for linear algebra, and using our efficient algorithm for computing multiple matrix products (Theorem 1) as a subroutine. An interesting open question is whether $\mathsf{INV}(n, \mathbb{F})$ can be solved with the same (randomized) round complexity as the other tasks. This problem may very well be more difficult; in the parallel setting in particular, to the best of our knowledge, whether matrix inversion can be done with the same complexity as these other tasks is also an open problem.

Applications of our results. The above results give new algorithms for many graph-theoretic problems in the congested clique model, as described below and summarized in Table 1.

Table 1. Summary of the applications of our algebraic techniques to graph-theoretic problems in the congested clique model. Here n both represents the number of vertices in the input graph and the number of nodes in the network.

Problem	Round complexity	Previously
APSP (undirected, weights in $\{0, 1, \ldots, M\}$)	$\tilde{O}\left(M^{\frac{2}{\omega}} n^{1-\frac{2}{\omega}}\right)$	$\tilde{O}\left(Mn^{1-\frac{2}{\omega}}\right)$
APSP (directed, constant weights)	$O(n^{0.2096})$	$\tilde{O}(n^{1/3})$
Diameter (undirected, weights in $\{0, 1, \ldots, M\}$)	$\tilde{O}\left(M^{\frac{2}{\omega}} n^{1-\frac{2}{\omega}}\right)$	$\tilde{O}\left(Mn^{1-\frac{2}{\omega}}\right)$
Computing the size of a maximum matching	$O\left(n^{1-\frac{2}{\omega}} \log n\right)$	—
Computing allowed edges in a perfect matching	$O(n^{1-1/\omega})$	—
Gallai-Edmonds decomposition	$O(n^{1-1/\omega})$	—
Minimum vertex cover in bipartite graphs	$O(n^{1-1/\omega})$	—

Our main key tool to derive these applications is Theorem 7 in Sect. 3, which gives an algorithm computing efficiently the distance product (defined in Sect. 2) of two matrices with small integer entries based on our algorithm for multiple matrix multiplication of Theorem 1. Computing the distance product is a fundamental graph-theoretic task deeply related to the All-Pairs Shortest Path (APSP) problem [27,28,30]. Combining this result with techniques from [28], and observing that these techniques can be implemented efficiently in the congested clique model, we then almost immediately obtain the following result.

Theorem 4. *In the congested clique model, the deterministic round complexity of the all-pairs shortest paths problem in an undirected graph of n vertices with integer weights in $\{0, \ldots, M\}$, where M is an integer such that $M \leq n$, is $\tilde{O}(M^{2/\omega} n^{1-2/\omega})$.*

Since computing the diameter of a graph reduces to solving the all-pairs shortest paths, we obtain the same round complexity for diameter computation in the same class of graphs. This improves over the $\tilde{O}(Mn^{1-2/\omega})$-round algorithm for these tasks (implicitly) given in [2]. The main application of our results nevertheless concerns the all-pair shortest paths problem over directed graphs (for which the approach based on [28] does not work) with constant weights. We obtain the following result by combining our algorithm for distance product computation with Zwick's approach [30].

Theorem 5. *In the congested clique model, the randomized round complexity of the all-pairs shortest paths problem in a directed graph of n vertices with integer weights in $\{-M, \ldots, 0, \ldots, M\}$, where $M = O(1)$, is $O(n^{0.2096})$.*

Prior to this work, the upper bound for the round complexity of this problem was $\tilde{O}(n^{1/3})$, obtained by directly computing the distance product (as done in [2]) in the congested clique model. Again, Theorem 5 follows easily from Theorem 7 and the observation that the reduction to distance product computation given in [30] can be implemented efficiently in the congested clique model. The exponent 0.2096 in the statement of Theorem 5 is derived from the current best upper bounds on the complexity of rectangular matrix multiplication in the centralized setting [15].

Theorems 2 and 3 also enable us to solve a multitude of graph-theoretic problems in the congested clique model with a sublinear number of rounds. Examples described in this paper are computing the number of edges in a maximum matching of a simple graph with $O(n^{1-2/\omega} \log n)$ rounds, computing the set of allowed edges in a perfect matching, the Gallai-Edmonds decomposition of a simple graph, and a minimum vertex cover in a bipartite graph with $O(n^{1-1/\omega})$ rounds. These results are obtained almost immediately from the appropriate reductions to matrix inversion and similar problems known the centralized setting [3,20,26] — indeed it is not hard to adapt all these reductions so that they can be implemented efficiently in the congested clique model. Note that while non-algebraic centralized algorithms solving these problems also exist (see, e.g., [21]), they are typically sequential and do not appear to be efficiently implementable in the congested clique model. The algebraic approach developed in this paper, made possible by our algorithms for the computation of the determinant, the rank and the inverse of matrix, appears to be currently the only way of obtaining fast algorithms for these problems in the congested clique model.

Remarks on the organization of the paper. Due to space constraints, most of the technical proofs are not included, but can be found in the full version of the present paper. The discussion of randomized algorithms for the determinant (Theorem 3) is also omitted from this version. The whole discussion detailing the applications of our algebraic methods is omitted as well, with the exception of the statement of Theorem 7 given in Sect. 3.

2 Preliminaries

Notations. Through this paper we will use n to denote the number of nodes in the network. The n nodes will be denoted $1, 2, \ldots, n$. The symbol $\mathbb{F}$ will always denote a finite field of order upper bounded by a polynomial in n (which means that each field element can be encoded with $O(\log n)$ bits and thus sent using one message in the congested clique model). Given any positive integer p, we use the notation $[p]$ to represent the set $\{1, 2, \ldots, p\}$. Given any $p \times p'$ matrix A, we will write its entries as $A[i, j]$ for $(i, j) \in [p] \times [p']$, and use the notation $A[i, *]$ to represent its i-th row and $A[*, j]$ to represent its j-th column.

Graph-theoretic problems in the congested clique model. As mentioned in the introduction, typically the main tasks that we want to solve in the congested clique model are graph-theoretical problems. In all the applications given in this paper the number of vertices of the graph will be n, the same as the number of nodes of the network. The input will be given as follows: initially each node $\ell \in [n]$ has the ℓ-th row and the ℓ-th column of the adjacency matrix of the graph. Note that this distribution of the input, while being the most natural, is not essential; the only important assumption is that the entries are evenly distributed among the n nodes since they can then be redistributed in a constant number of rounds as shown in the following Lemma by Dolev et al. [6], which we will use many times in this paper.

Lemma 1. *[6] In the congested clique model a set of messages in which no node is the source of more than n messages and no node is the destination of more than n messages can be delivered within two rounds if the source and destination of each message is known in advance to all nodes.*

Algebraic problems in the congested clique model. The five main algebraic problems that we consider in this paper are defined as follows.

$\mathsf{MM}(n, m, k, \mathbb{F})$ — Multiple Rectangular Matrix Multiplications
Input: matrices $A_1, \ldots, A_k \in \mathbb{F}^{n \times m}$ and $B_1, \ldots, B_k \in \mathbb{F}^{m \times n}$ distributed among the n nodes
(Node $\ell \in [n]$ has $A_1[\ell, *], \ldots, A_k[\ell, *]$ and $B_1[*, \ell], \ldots, B_k[*, \ell]$)
Output: the matrices $A_1B_1, \ldots, A_kB_k$ distributed among the n nodes
(Node $\ell \in [n]$ has $A_1B_1[\ell, *], \ldots, A_kB_k[\ell, *]$ and $A_1B_1[*, \ell], \ldots, A_kB_k[*, \ell]$)

$\mathsf{DET}(n, \mathbb{F})$ — Determinant
Input: matrix $A \in \mathbb{F}^{n \times n}$ distributed among the n nodes
(Node $\ell \in [n]$ has $A[\ell, *]$ and $A[*, \ell]$)
Output: $\det(A)$ (Each node of the network has $\det(A)$)

$\mathsf{Rank}(n, \mathbb{F})$ — Rank
Input: matrix $A \in \mathbb{F}^{n \times n}$ distributed among the n nodes
(Node $\ell \in [n]$ has $A[\ell, *]$ and $A[*, \ell]$)

Output: rank(A) (Each node of the network has rank(A))

$\mathsf{INV}(n, \mathbb{F})$ — Inversion

Input: invertible matrix $A \in \mathbb{F}^{n \times n}$ distributed among the n nodes (Node $\ell \in [n]$ has $A[\ell, *]$ and $A[*, \ell]$)

Output: matrix A^{-1} distributed among the n nodes (Node $\ell \in [n]$ has $A^{-1}[\ell, *]$ and $A^{-1}[*, \ell]$)

$\mathsf{SYS}(n, \mathbb{F})$ — Solution of a linear system

Input: invertible matrix $A \in \mathbb{F}^{n \times n}$ and vector $b \in \mathbb{F}^{n \times 1}$, distributed among the n nodes (Node $\ell \in [n]$ has $A[\ell, *]$, $A[*, \ell]$ and b)

Output: the vector $x \in \mathbb{F}^{n \times 1}$ such that $Ax = b$ (Node $\ell \in [n]$ has $x[\ell]$)

Note that the distribution of the inputs and the outputs assumed in the above five problems is mostly chosen for convenience. For instance, if needed the whole vector x in the output of $\mathsf{SYS}(n, \mathbb{F})$ can be sent to all the nodes of the network in two rounds using Lemma 1. The only important assumption is that when dealing with matrices, the entries of the matrices must be evenly distributed among the n nodes.

We will also in this paper consider the distance product of two matrices, defined as follows.

Definition 1. *Let m and n be two positive integers. Let A be an $n \times m$ matrix and B be an $m \times n$ matrix, both with entries in $\mathbb{R} \cup \{\infty\}$. The distance product of A and B, denoted $A * B$, is the $n \times n$ matrix C such that $C[i, j] = \min_{s \in [m]} \{A[i, s] + B[s, j]\}$ for all $(i, j) \in [n] \times [n]$.*

We will be mainly interested in the case when the matrices have integer entries. More precisely, we will consider the following problem.

$\mathsf{DIST}(n, m, M)$ — Computation of the distance product

Input: an $n \times m$ matrix A and an $m \times n$ matrix B, with entries in $\{-M, \ldots, -1, 0, 1, \ldots, M\} \cup \{\infty\}$ (Node $\ell \in [n]$ has $A[\ell, *]$ and $B[*, \ell]$)

Output: the matrix $C = A * B$ distributed among the n nodes (Node $\ell \in [n]$ has $C[\ell, *]$ and $C[*, \ell]$)

Centralized algebraic algorithms for matrix multiplication. We now briefly describe algebraic algorithms for matrix multiplication and known results about the complexity of rectangular matrix multiplication. We refer to [1] for a detailed exposition of these concepts.

Let $\mathbb{F}$ be a field and m, n be two positive integer. Consider the problem of computing the product of an $n \times m$ matrix by an $m \times n$ matrix over $\mathbb{F}$. An algebraic algorithm for this problem is described by three sets $\{\alpha_{ij\mu}\}$, $\{\beta_{ij\mu}\}$ and $\{\lambda_{ij\mu}\}$ of coefficients from $\mathbb{F}$ such that, for any $n \times m$ matrix A and any $m \times n$ matrix B, the equality

$$C[i,j] = \sum_{\mu=1}^{t} \lambda_{ij\mu} S^{(\mu)} T^{(\mu)}$$

holds for all $(i,j) \in [n] \times [n]$, where $C = AB$ and

$$S^{(\mu)} = \sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{ij\mu} A[i,j], \qquad T^{(\mu)} = \sum_{i=1}^{n} \sum_{j=1}^{m} \beta_{ij\mu} B[j,i],$$

for each $s \in [t]$. Note that each $S^{(\mu)}$ and each $T^{(\mu)}$ is an element of $\mathbb{F}$. The integer t is called the rank of the algorithm, and corresponds to the complexity of the algorithm.

For instance, consider the trivial algorithm computing this matrix product using the formula

$$C[i,j] = \sum_{s=1}^{m} A[i,s] B[s,j].$$

This algorithm can be described in the above formalism by taking $t = n^2 m$, writing each $\mu \in [n^2 m]$ as a triple $\mu = (i', j', s') \in [n] \times [n] \times [m]$, and choosing

$$\lambda_{ij(i',j',s')} = \begin{cases} 1 & \text{if } i = i' \text{and } j = j', \\ 0 & \text{otherwise,} \end{cases}$$

$$\alpha_{ij(i',j',s')} = \begin{cases} 1 & \text{if } i = i' \text{and } j = s', \\ 0 & \text{otherwise,} \end{cases} \qquad \beta_{ij(i',j',s')} = \begin{cases} 1 & \text{if } i = j' \text{and } j = s', \\ 0 & \text{otherwise.} \end{cases}$$

Note that this trivial algorithm, and the description we just gave, also works over any semiring.

The exponent of matrix multiplication. For any non-negative real number γ, let $\omega(\gamma)$ denote the minimal value τ such that the product of an $n \times \lceil n^\gamma \rceil$ matrix over $\mathbb{F}$ by an $\lceil n^\gamma \rceil \times n$ matrix over $\mathbb{F}$ can be computed by an algebraic algorithm of rank $n^{\tau+o(1)}$ (i.e., can be computed with complexity $O(n^{\tau+\epsilon})$ for any $\epsilon > 0$). As usual in the literature, we typically abuse notation and simply write that such a product can be done with complexity $O(n^{\omega(\gamma)})$, i.e., ignoring the $o(1)$ in the exponent. The value $\omega(1)$ is denoted by ω, and often called the exponent of square matrix multiplication. Another important quantity is the value $\alpha = \sup\{\gamma \mid \omega(\gamma) = 2\}$.

The trivial algorithm for matrix multiplication gives the upper bound $\omega(\gamma) \leq 2 + \gamma$, and thus $\omega \leq 3$ and $\alpha \geq 0$. The current best upper bound on ω is $\omega < 2.3729$, see [16,29]. The current best bound on α is $\alpha > 0.3029$, see [15]. The best bounds on $\omega(\gamma)$ for $\gamma > \alpha$ can also be found in [15].

3 Matrix Multiplication in the Congested Clique Model

In this section we discuss the round complexity of Problems $\mathsf{MM}(n,m,k,\mathbb{F})$ and $\mathsf{DIST}(n,m,M)$.

Our first result is the following theorem.

Theorem 6 (Complete version). *For any positive integer $k \leq n$, the deterministic round complexity of* $\mathsf{MM}(n, m, k, \mathbb{F})$ *is*

$$\begin{cases} O(k) & if\ \ 0 \leq m \leq \sqrt{kn}, \\ O(k^{2/\omega(\gamma)} n^{1-2/\omega(\gamma)}) & if\ \ \sqrt{kn} \leq m < n^2/k, \\ O(km/n) & if\ \ m \geq n^2/k, \end{cases}$$

where γ is the solution of the equation

$$\left(1 - \frac{\log k}{\log n}\right)\gamma = 1 - \frac{\log k}{\log n} + \left(\frac{\log m}{\log n} - 1\right)\omega(\gamma). \tag{1}$$

For any $k \geq n$, the deterministic round complexity of $\mathsf{MM}(n, m, k, \mathbb{F})$ *is*

$$\begin{cases} O(k) & if\ \ 1 \leq m \leq n, \\ O(km/n) & if\ \ m \geq n. \end{cases}$$

The proof of Theorem 1, which will also show that Eq. (1) always has a solution when $k \leq n$ and $\sqrt{kn} \leq m < n^2/k$, can be found in the full version of the paper (a short discussion of the proof ideas was presented in the introduction). As briefly mentioned in the introduction, the round complexity is constant for any $k \leq \sqrt{n}$, and we further have round complexity $O(n^\epsilon)$, for any $\epsilon > 0$, for all values $k \leq n^{(1+\alpha)/2}$ (the bound $\alpha > 0.3029$ implies $(1+\alpha)/2 > 0.6514$). For the case $m = n$ the solution of Eq. (1) is $\gamma = 1$, which gives the bounds of the simplified version of Theorem 1 presented in the introduction.

We now give lower bounds on the round complexity of $\mathsf{MM}(n, m, k, \mathbb{F})$ that show that the upper bounds of Theorem 1 are tight, except possibly in the case $\sqrt{kn} \leq m < n^2/k$ when $k \leq n$.

Proposition 1. *The randomized round complexity of* $\mathsf{MM}(n, m, k, \mathbb{F})$ *is*

$$\begin{cases} \Omega(k) & if\ \ 1 \leq m \leq n, \\ \Omega(km/n) & if\ \ m \geq n. \end{cases}$$

Proof. We first prove the lower bound $\Omega(km/n)$ for any $m \geq n$. Let us consider instances of $\mathsf{MM}(n, m, k, \mathbb{F})$ of the following form: for each $s \in [k]$ all the rows of A_s are zero except the first row; for each $s \in [k]$ all the columns of B_s are zero except the second column. Let us write $C_s = A_s B_s$ for each $s \in [k]$. We prove the lower bound by partitioning the n nodes of the network into the two sets $\{1\}$ and $\{2, \ldots, n\}$, and considering the following two-party communication problem. Alice (corresponding to the set $\{1\}$) has for input $A_s[1, j]$ for all $j \in [m]$ and all $s \in [k]$. Bob (corresponding to the set $\{2, \ldots, n\}$) has for input $B_s[i, 2]$ for all $i \in [m]$ and all $s \in [k]$. The goal is for Alice to output $C_s[1, 2]$ for all $s \in [k]$. Note that $C_s[1, 2]$ is the inner product (over $\mathbb{F}$) of the first row of A_s and the second column of B_s. Thus $\sum_{s=1}^{k} C_s[1, 2]$ is the inner product of two vectors of size km. Alice and Bob must exchange $\Omega(km \log |\mathbb{F}|)$ bits to compute this value [4], which requires $\Omega(km/n)$ rounds in the original congested clique model.

We now prove the lower bound $\Omega(k)$ for any $m \geq 1$. Let us consider instances of $\mathsf{MM}(n, m, k, \mathbb{F})$ of the following form: for each $s \in [k]$, all entries of A_s are zero except the entry $A_s[1,1]$ which is one; for each $s \in [k]$, $B_s[i,j] = 0$ for all $(i,j) \notin \{(1,j) \mid j \in \{2,\ldots,n\}\}$ (the other $n-1$ entries are arbitrary). Again, let us write $C_s = A_s B_s$ for each $s \in [k]$. We prove the lower bound by again partitioning the n nodes of the network into the two sets $\{1\}$ and $\{2,\ldots,n\}$, and considering the following two-party communication problem. Alice has no input. Bob has for input $B_s[1,j]$ for all $j \in \{2,\ldots,n\}$ and all $s \in [k]$. The goal is for Alice to output $C_s[1,j]$ for all $j \in \{2,\ldots,n\}$ and all $s \in [k]$. Since the output reveals Bob's whole input to Alice, Alice must receive $\Omega(k(n-1)\log|\mathbb{F}|)$ bits, which gives round complexity $\Omega(k)$ in the original congested clique model. □

One of the main applications of Theorem 1 is the following result, which will imply all our results on the all-pairs shortest paths and diameter computation, including Theorems 4 and 5.

Theorem 7. *For any $M \leq n$ and $m \leq n$, the deterministic round complexity of $\mathsf{DIST}(n,m,M)$ is*

$$\begin{cases} O(M \log m) & \text{if } 0 \leq m \leq \sqrt{Mn \log m}, \\ O\left((M \log m)^{2/\omega(\gamma)} n^{1-2/\omega(\gamma)}\right) & \text{if } \sqrt{Mn \log m} \leq m \leq n^2/(M \log m), \\ O\left(mM \log m/n\right) & \text{if } n^2/(M \log m) \leq m \leq n, \end{cases}$$

where γ is the solution of the equation $\left(1 - \frac{\log M}{\log n}\right)\gamma = 1 - \frac{\log M}{\log n} + \left(\frac{\log m}{\log n} - 1\right)\omega(\gamma)$.

The proof of Theorem 7 is omitted, but can be found in the full version of the paper. The idea is to show that $\mathsf{DIST}(n,m,M)$ reduces to $\mathsf{MM}(n,m,k,\mathbb{F})$ for $k \approx M \log m$ and a well-chosen finite field $\mathbb{F}$, and then use Theorem 1 to get a factor $(M \log m)^{2/\omega(\gamma)}$, instead of the factor M obtained in a straightforward implementation of the distance product, in the complexity. This reduction is done by first applying a standard encoding of the distance product into a usual matrix product of matrices with integer entries of absolute value $\exp(M)$, and then using Fourier transforms to split this latter matrix product into roughly $M \log m$ independent matrix products over a small field.

4 Deterministic Computation of Determinant and Inverse Matrix

In this section we present deterministic algorithms for computing the determinant of a matrix and the inverse of a matrix in the congested clique model, and prove Theorem 2. Our algorithms can be seen as efficient implementations of the parallel algorithm by Preparata and Sarwate [25] based on the Faddeev-Leverrier method.

Let A be an $n \times n$ matrix over a field $\mathbb{F}$. Let $\det(\lambda I - A) = \lambda^n + c_1\lambda^{n-1} + \cdots + c_{n-1}\lambda + c_n$ be its characteristic polynomial. The determinant of A is $(-1)^n c_n$ and, if $c_n \neq 0$, its inverse is

$$A^{-1} = -\frac{A^{n-1} + c_1 A^{n-2} + \cdots + c_{n-2}A + c_{n-1}I}{c_n}.$$

Define the vector $\boldsymbol{c} = (c_1, \ldots, c_n)^T \in \mathbb{F}^{n\times 1}$. For any $k \in [n]$ let s_k denote the trace of the matrix A^k, and define the vector $\boldsymbol{s} = (s_1, \ldots, s_n)^T \in \mathbb{F}^{n\times 1}$. Define the $n \times n$ matrix

$$S = \begin{pmatrix} 1 & & & & \\ s_1 & 2 & & & \\ s_2 & s_1 & 3 & & \\ \vdots & \vdots & \vdots & \ddots & \\ s_{n-1} & s_{n-2} & s_{n-3} & \ldots & s_1 \; n \end{pmatrix}.$$

It can be easily shown (see, e.g., [5,25]) that $S\boldsymbol{c} = -\boldsymbol{s}$, which enables us to recover $\boldsymbol{c}$ from $\boldsymbol{s}$ if S is invertible. The matrix S is invertible whenever $n! \neq 0$, which is true in any field of characteristic zero or in any finite field of characteristic strictly larger than n. The following proposition shows that the inverse of an invertible triangular matrix can be computed efficiently in the congested clique model.

Proposition 2. *Let $\mathbb{F}$ be any field. There is a deterministic algorithm with round complexity $O(n^{1-2/\omega})$ that solves $\mathsf{INV}(n, \mathbb{F})$ when the input A is an invertible lower triangular matrix.*

We are now ready to give the proof of Theorem 2.

Proof (of Theorem 2). For convenience we assume that n is a square, and write $p = \sqrt{n}$. If n is not a square we can easily adapt the proof by taking $p = \lceil \sqrt{n} \rceil$. Observe that any integer $a \in \{0, 1, \ldots, n-1\}$ can be written in a unique way as $a = (a_1 - 1)p + (a_2 - 1)$ with $a_1, a_2 \in [p]$. Below when we write $a = (a_1, a_2) \in [n]$, we mean that a_1 and a_2 are the two elements in $[p]$ such that $a = (a_1-1)p+(a_2-1)$.

For any $\ell \in [n]$, let R_ℓ be the $p \times n$ matrix such that the i-th row of R_ℓ is the ℓ-th row of $A^{(i-1)p}$, for each $i \in [p]$. Similarly, for any $\ell \in [n]$, let C_ℓ be the $n \times p$ matrix such that the j-th column of C_ℓ is the ℓ-th column of A^{j-1}, for each $j \in [p]$. For each $\ell \in [n]$ define $U_\ell = R_\ell C_\ell$, which is a $p \times p$ matrix. Observe that, for any $k = (k_1, k_2) \in [n]$, the identity

$$s_k = \sum_{\ell=1}^{n} U_\ell[k_1, k_2] \tag{2}$$

holds. We will use this expression, together with the equation $\boldsymbol{c} = -S^{-1}\boldsymbol{s}$ to compute the determinant in the congested clique model.

In order to compute the inverse of A we then use the following approach. For any $(a_1, a_2) \in [p] \times [p]$, define the coefficient $c_{a_1,a_2} \in \mathbb{F}$ as follows:

$$c_{a_1,a_2} = \begin{cases} c_{n-1-(a_1-1)p-(a_2-1)} & \text{if } (a_1, a_2) \neq (p, p), \\ 1 & \text{if } (a_1, a_2) = (p, p). \end{cases}$$

For any $a_2 \in [p]$, define the $n \times n$ matrix E_{a_2} as follows:

$$E_{a_2} = \sum_{a_1=1}^{p} c_{a_1,a_2} A^{(a_1-1)p}.$$

Note that the following holds whenever $c_n \neq 0$:

$$A^{-1} = -\frac{\sum_{a=0}^{n-1} c_{n-1-a} A^a}{c_n} = -\frac{\sum_{a_1=1}^{p} \sum_{a_2=1}^{p} c_{a_2,a_1} A^{(a_1-1)p+(a_2-1)}}{c_n},$$

which gives

$$A^{-1} = -\frac{\sum_{a_2=1}^{p} E_{a_2} A^{a_2-1}}{c_n}. \tag{3}$$

The algorithm for $\mathsf{DET}(n, \mathbb{F})$ and $\mathsf{INV}(n, \mathbb{F})$ is described in Fig. 1. Steps 1 and 7.2 can be implemented in $O(p^{2/\omega} n^{1-2/\omega})$ rounds from Theorem 1 (or its simplified version in the introduction). Step 5 can be implemented in $O(n^{1-2/\omega})$ rounds, again from Theorem 1. At Steps 2, 3 and 6 each node receives n elements from the field $\mathbb{F}$, so each of these three steps can be implemented in

1. The matrices $A^{(a_1-1)p}$ and A^{a_2-1} are computed for all $a_1, a_2 \in [p]$ using the distributed algorithm of Theorem 1. At the end of this step node $\ell \in [n]$ has the whole $p \times n$ matrix R_ℓ and the whole $n \times p$ matrix C_ℓ.
2. Node $\ell \in [n]$ locally computes U_ℓ, and sends $U_\ell[k_1, k_2]$ to each node $k = (k_1, k_2) \in [n]$.
3. Node $k = (k_1, k_2) \in [n]$, who received $U_\ell[k_1, k_2]$ for all $\ell \in [n]$ at the previous step, locally computes s_k using Equation (2). Node k then sends s_k to all the nodes.
4. Node $\ell \in [n]$, who received $\boldsymbol{s}$ at Step 3, locally constructs $S[\ell, *]$ and $S[*, \ell]$.
5. The matrix S^{-1} is computed using the algorithm of Proposition 2. At the end of this step, node $\ell \in [n]$ has $S^{-1}[\ell, *]$ and $S^{-1}[*, \ell]$.
6. Node $\ell \in [n]$ locally computes c_ℓ from $S^{-1}[\ell, *]$ and $\boldsymbol{s}$, and sends c_ℓ to all nodes.
7. The determinant of A is $(-1)^n c_n$. If $c_n = 0$ the matrix A is not invertible. Otherwise the nodes compute A^{-1} as follows:
   - 7.1 Node $\ell \in [n]$ computes $E_{a_2}[\ell, *]$ for each $a_2 \in [p]$ (this can be done locally since $\boldsymbol{c}$ and each row $A^{(a_1-1)p}[\ell, *]$ are known from Steps 6 and 1, respectively).
   - 7.2 The matrices $E_{a_2} A^{a_2-1}$ are computed for all $a_2 \in [p]$ using the distributed algorithm of Theorem 1 (since, besides $E_{a_2}[\ell, *]$ obtained at the previous step, each node $\ell \in [n]$ knows $A^{a_2-1}[*, \ell]$ from the result of the computation of Step 1). At the end of this step, node $\ell \in [n]$ has the ℓ-th row and the ℓ-th column of the matrix $E_{a_2} A^{a_2-1}$ for all $a_2 \in [p]$.
   - 7.3 Node $\ell \in [n]$ computes locally $A^{-1}[\ell, *]$ and $A^{-1}[*, \ell]$ using Equation (3).

Fig. 1. Distributed algorithm for computing the determinant of an $n \times n$ matrix A and computing A^{-1} if $\det(A) \neq 0$. Initially each node $\ell \in [n]$ has as input $A[\ell, *]$ and $A[*, \ell]$.

two rounds from Lemma 1. The other steps (Steps 4, 7.1 and 7.3) do not require any communication. The total round complexity of the algorithm is thus $O\left(p^{2/\omega}n^{1-2/\omega}\right) = O\left(n^{1-1/\omega}\right)$, as claimed. □

Acknowledgments. The author is grateful to Arne Storjohann for precious help concerning the computation of the determinant and to anonymous reviewers for their comments. This work is supported by the Grant-in-Aid for Young Scientists (A) No. 16H05853, the Grant-in-Aid for Scientific Research (A) No. 16H01705, and the Grant-in-Aid for Scientific Research on Innovative Areas No. 24106009 of the Japan Society for the Promotion of Science and the Ministry of Education, Culture, Sports, Science and Technology in Japan.

References

1. Bürgisser, P., Clausen, M., Shokrollahi, M.A.: Algebraic Complexity Theory. Springer, Heidelberg (1997)
2. Censor-Hillel, K., Kaski, P., Korhonen, J.H., Lenzen, C., Paz, A., Suomela, J.: Algebraic methods in the congested clique. In: Proceedings of the 34th Symposium on Principles of Distributed Computing, pp. 143–152 (2015)
3. Cheriyan, J.: Randomized $\tilde{O}(M(|V|))$ algorithms for problems in matching theory. SIAM J. Comput. **26**(6), 1635–1669 (1997)
4. Chu, J.I., Schnitger, G.: Communication complexity of matrix computation over finite fields. Math. Syst. Theor. **28**(3), 215–228 (1995)
5. Csanky, L.: Fast parallel matrix inversion algorithms. In: Proceedings of the 16th Annual Symposium on Foundations of Computer Science, pp. 11–12 (1975)
6. Dolev, D., Lenzen, C., Peled, S.: "Tri, tri again": finding triangles and small subgraphs in a distributed setting. In: Aguilera, M.K. (ed.) DISC 2012. LNCS, vol. 7611, pp. 195–209. Springer, Heidelberg (2012)
7. Drucker, A., Kuhn, F., Oshman, R.: On the power of the congested clique model. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 367–376 (2014)
8. Hegeman, J.W., Pandurangan, G., Pemmaraju, S.V., Sardeshmukh, V.B., Scquizzato, M.: Toward optimal bounds in the congested clique: Graph connectivity and MST. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 91–100 (2015)
9. Hegeman, J.W., Pemmaraju, S.V.: Lessons from the congested clique applied to mapreduce. In: Halldórsson, M.M. (ed.) SIROCCO 2014. LNCS, vol. 8576, pp. 149–164. Springer, Heidelberg (2014)
10. Hegeman, J.W., Pemmaraju, S.V., Sardeshmukh, V.B.: Near-constant-time distributed algorithms on a congested clique. In: Kuhn, F. (ed.) DISC 2014. LNCS, vol. 8784, pp. 514–530. Springer, Heidelberg (2014)
11. Henzinger, M., Krinninger, S., Nanongkai, D.: A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In: Proceedings of the 48th Annual ACM Symposium on Theory of Computing, pp. 489–498 (2016)
12. Kaltofen, E., Pan, V.Y.: Processor efficient parallel solution of linear systems over an abstract field. In: Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 180–191 (1991)

13. Kaltofen, E., Pan, V.Y.: Processor-efficient parallel solution of linear systems II: the positive characteristic and singular cases (extended abstract). In: Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, pp. 714–723 (1992)
14. Kaltofen, E., Saunders, B.D.: On Wiedemann's method of solving sparse linear systems. In: Proceedings of the 9th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, pp. 29–38 (1991)
15. Le Gall, F.: Faster algorithms for rectangular matrix multiplication. In: Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, pp. 514–523 (2012)
16. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, pp. 296–303 (2014)
17. Lenzen, C.: Optimal deterministic routing and sorting on the congested clique. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 42–50 (2013)
18. Lenzen, C., Wattenhofer, R.: Tight bounds for parallel randomized load balancing: extended abstract. In: Proceedings of the 43rd ACM Symposium on Theory of Computing, pp. 11–20 (2011)
19. Lotker, Z., Pavlov, E., Patt-Shamir, B., Peleg, D.: MST construction in $o(\log \log n)$ communication rounds. In: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 94–100 (2003)
20. Lovász, L.: On determinants, matchings, and random algorithms. In: Fundamentals of Computation Theory, pp. 565–574 (1979)
21. Lovász, L., Plummer, M.D.: Matching Theory. American Mathematical Society (2009)
22. Nanongkai, D.: Distributed approximation algorithms for weighted shortest paths. In: Proceedings of the 46th Symposium on Theory of Computing, pp. 565–573 (2014)
23. Patt-Shamir, B., Teplitsky, M.: The round complexity of distributed sorting: extended abstract. In: Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, pp. 249–256 (2011)
24. Peleg, D.: Distributed computing: a locality-sensitive approach. Society for Industrial and Applied Mathematics (2000)
25. Preparata, F.P., Sarwate, D.V.: An improved parallel processor bound in fast matrix inversion. Inf. Process. Lett. **7**(3), 148–150 (1978)
26. Rabin, M.O., Vazirani, V.V.: Maximum matchings in general graphs through randomization. J. Algorithms **10**(4), 557–567 (1989)
27. Seidel, R.: On the all-pairs-shortest-path problem in unweighted undirected graphs. J. Comput. Syst. Sci. **51**(3), 400–403 (1995)
28. Shoshan, A., Zwick, U.: All pairs shortest paths in undirected graphs with integer weights. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, pp. 605–615 (1999)
29. Vassilevska Williams, V.: Multiplying matrices faster than coppersmith-winograd. In: Proceedings of the 44th Symposium on Theory of Computing, pp. 887–898 (2012)
30. Zwick, U.: All pairs shortest paths using bridging sets and rectangular matrix multiplication. J. ACM **49**(3), 289–317 (2002)