Theory of Computing Systems (2026) 70:41
https://doi.org/10.1007/s00224-026-10286-7

Multiplication of 0-1 Matrices via Clustering

Jesper Jansson[1] · Miroslaw Kowaluk[2] · Andrzej Lingas[3] · Mia Persson[4]

Received: 22 December 2025 / Accepted: 19 June 2026

Abstract

We study applications of clustering (in particular, the Hamming k-center clustering problem) in the design of efficient and practical algorithms for computing an approximate and the exact arithmetic matrix product of two 0-1 rectangular matrices with clustered rows or columns, respectively. Our results in part can be regarded as an extension of the clustering-based approach to Boolean square matrix multiplication due to Arslan and Chidri (CSC 2011). We provide a simple and efficient deterministic algorithm for approximate matrix product of 0-1 matrices, where the additive error is proportional to the minimum maximum radius in an ℓ-center clustering of the rows of the first matrix or an k-center clustering of the columns of the second matrix. We use the approximation algorithm as a preprocessing after which a query asking for the exact value of an arbitrary entry in the product matrix can be answered in time proportional to the additive error. As a consequence, we obtain a simple deterministic algorithm for the exact matrix product of 0-1 matrices. We also present an alternative simple deterministic algorithm for the exact product and in addition, faster analogous randomized algorithms for an approximate and the exact matrix products of 0-1 matrices based on randomized ℓ- and k-center clustering.

A preliminary version of this article appeared in *Proceedings of Frontiers of Algorithmics - the 19th International Joint Conference* (IJTCS-FAW 2025), Lecture Notes in Computer Science, Vol. 15828, pp. 92–102, Springer Nature, 2025.

✉ Andrzej Lingas
Andrzej.Lingas@cs.lth.se

Jesper Jansson
jj@i.kyoto-u.ac.jp

Miroslaw Kowaluk
kowaluk@mimuw.edu.pl

Mia Persson
Mia.Persson@mau.se

1 Graduate School of Informatics, Kyoto University, Kyoto, Japan

2 Institute of Informatics, University of Warsaw, Warsaw, Poland

3 Department of Computer Science, Lund University, Lund, Sweden

4 Department of Computer Science and Media Technology, Malmö University, Malmö, Sweden

Published online: 29 June 2026

Keywords Arithmetic matrix multiplication · k-center clustering · Hamming space

1 Introduction

The arithmetic matrix product of two 0-1 matrices is closely related to the Boolean one of the corresponding Boolean matrices. Both are basic tools in science and engineering. For square $n \times n$ matrices, both can be computed in $O(n^{2.372})$ time by using fast matrix multiplication algorithms based on algebra [1, 2]. Unfortunately, the latter algorithms suffer from huge overheads and no truly subcubic practical algorithms for any of these two problems is known. Therefore, many researchers studied the complexity of these products for special input matrices, e.g., sparse or structured matrices [3–9], providing faster and often more practical algorithms.

The method of multiplying matrices with clustered rows or columns, proposed for Boolean matrix product in [5] and subsequently generalized in [6, 7] and used in [10], relies on the construction of an approximate minimum spanning tree of the rows of the first input matrix or the columns of the second input matrix in a Hamming space. Then, each column or each row of the product matrix is computed with the help of a traversal of the tree in time proportional to the total Hamming cost of the tree up to a logarithmic factor. Simply, the next entry in a column or a row in the product matrix can be obtained from the previous one in time roughly proportional to the Hamming distance between the consecutive (in the tree traversal) corresponding rows or columns of the first or the second input matrix, respectively. Thus, in case the entire tree cost is substantially subquadratic in n, the total running time of this method becomes substantially subcubic provided that a good approximation of a minimum spanning tree of the rows of the first input matrix or the columns of the second one can be constructed in substantially subcubic time. As for simplicity and practicality, a weak point of this method is that in order to construct such an approximation relatively quickly, it employs a rather involved randomized algorithm.

Arslan and Chidri presented a simple algorithm for the Boolean matrix product of two input Boolean matrices with rows (in the first matrix) and columns (in the other matrix) within a small diameter d (in terms of the Hamming distance) in [11]. The basic idea is to compute the inner Boolean product of a representative of the rows of the first matrix and a representative of columns of the second matrix and then relatively cheaply recover the entries of the Boolean product matrix from the inner product. Their algorithm runs in $O(dn^2)$ time for $n \times n$ matrices. They also presented a generalization of their algorithm to include grouping the rows of the first matrix as well as the columns of the second matrix into k clusters of maximum radius within 2 of the minimum s. They incorporate Gonzalez's 2-approximation algorithm for the k-center problem [12] to achieve such a k-clustering. Their general algorithm runs in $O((k+s)n^2)$ time[1].

[1] They also claim that one can speed up their general algorithm by using the fast 2-approximation algorithm for the k-center problem due to Feder and Greene [13] instead of Gonzalez's one. Unfortunately, the former algorithm seems to have rather an exponential hidden dependence on the dimension (in this case n) in view of Lemmata 4.2 and 4.3 in [13] than a linear one as it is assumed in [11].

In case of the arithmetic matrix product of 0-1 matrices, in some cases, a faster approximate arithmetic matrix multiplication can be more useful [8, 14]. Among other things, it can enable to identify largest entries in the product matrix and it can also be used to provide a fast estimation of the number of: the so called witnesses for the Boolean product of two Boolean matrices [15], triangles in a graph, or more generally, subgraphs isomorphic to a small pattern graph [16] etc. There is a number of results on approximate arithmetic matrix multiplication, where the quality of approximation is expressed in terms of the Frobenius matrix norm $||\ ||_F$ (i.e., the square root of the sum of the squares of the entries of the matrix) [8, 14].

Cohen and Lewis [14] and Drineas et al. [17] used random sampling to approximate arithmetic matrix product. Their articles provide an approximation D of the matrix product AB of two $n \times n$ matrices A and B such that $||AB - D||_F = O(||AB||_F/\sqrt{c})$, for a parameter $c > 1$ (see also [8]). The approximation algorithm in [17] runs in $O(n^2 c)$ time. Drineas et al. [17] also derived bounds on the entrywise differences between the exact matrix product and its approximation. Unfortunately, the best of these bounds is $\Omega(M^2 n/\sqrt{c})$, where M is the maximum value of an entry in A and B. By using a sketch technique, Sarlós [18] obtained the same Frobenius norm guarantees, also in $O(n^2 c)$ time. However, he derived stronger individual upper bounds on the additive error of each entry D_{ij} of the approximation matrix D. They are of the form $O(||A_{i*}||_2 ||B_{*j}||_2/\sqrt{c})$, where A_{i*} and B_{*j} stands for the i-th row of A and the j-th column of B, respectively, that hold with high probability. More recently, Pagh [8] presented a randomized approximation $\tilde{O}(n(n + c))$-time algorithm for the arithmetic product of $n \times n$ matrices A and B such that each entry of the approximate matrix product differs at most by $||AB||_F/\sqrt{c}$ from the correct one. His algorithm first compresses the matrix product to a product of two polynomials and then uses the fast Fourier transform to multiply the polynomials. Subsequently, Kutzkov [19] developed analogous deterministic algorithms employing different techniques. For approximation results related to sparse arithmetic matrix products, see [8, 20].

1.1 Our Contributions

In this article, we exploit the possibility of applying the classic simple 2-approximation algorithm for the k center clustering problem [12] in order to derive efficient and practical deterministic algorithms for computing an approximate and the exact arithmetic matrix product of two 0-1 rectangular matrices A and B with clustered rows or columns, respectively. In addition, we consider the possibility of applying the recent faster randomized algorithm for k center clustering from [21] instead of that from [12].

Most of our results are based on the idea of using cluster representatives/centers to compute faster a smaller matrix product and then relatively cheaply updating the small product to the target one. This natural idea has been used already in [11][2]. Our approach conceptually extends that in [11] in several ways. We consider the more general problem of computing the arithmetic matrix product of 0-1 rectangular matrices while [11] is concerned with the Boolean matrix product of square Boolean

[2] We came across this idea independently without knowing [11] before writing the preliminary conference version of this article.

matrices. We also provide clustering-based approximation and preprocessing for a query on the value of single entry of the aforementioned arithmetic product; no prior clustering-based explicit results of this kind seem to be known in the literature. In [11], the rows of the first matrix are grouped in the same number of clusters as the columns of the second matrix, while in our case the numbers are in general different. Also, in [11] representatives/centers of both row clusters and column clusters are used to compute the smaller product while we use in the basic deterministic version either the row ones or the column ones. We also provide a faster randomized cluster-based method of multiplying 0-1 matrices relying on a fast randomized clustering.

The k-center clustering problem in a Hamming space $\{0,1\}^d$ is for a set P of n points in $\{0,1\}^d$ to find a set T of k points in $\{0,1\}^d$ that minimize $\max_{v\in P}\min_{u\in T}\mathrm{ham}(v,u)$, where $\mathrm{ham}(v,u)$ stands for the Hamming distance between v and u (i.e., the number of coordinates where v and u differ). Each center in T induces a cluster consisting of all points in P for which it is the nearest center.

Let $\lambda(A,\ell,row)$ and $\lambda(B,k,col)$ denote the minimum maximum Hamming distance between a row of A or a column of B and the closest center in an ℓ-center clustering of the rows of A or in a k-center clustering of the columns of B, respectively. (Note that in extreme cases, $\lambda(A,\ell,row)$ or $\lambda(B,k,col)$ can even be equal to 0.) In particular, when A and B are square matrices of size $n\times n$, we obtain the following results.

1. A simple deterministic algorithm that approximates each entry of the arithmetic matrix product of A and B within an additive error of at most $2\lambda(A,\ell,row)$ in $O(n^2\ell)$ time or at most $2\lambda(B,k,col)$ in $O(n^2k)$ time. Similarly, a simple randomized algorithm that for any $\epsilon\in(0,\frac{1}{2})$ approximates each entry of the product within an additive error of with high probability (w.h.p.) at most $(2+\epsilon)(\lambda(A,\ell,row)+\lambda(B,k,col))$ in time $O(n^2\log n/\epsilon^2+\ell nk)$.
2. A simple deterministic preprocessing of the matrices A and B in $O(n^2\ell)$ time or $O(n^2k)$ time after which every query asking for the exact value of an arbitrary entry of the arithmetic matrix product of A and B can be answered in $O(\lambda(A,\ell,row)+1)$ time or $O(\lambda(B,k,col)+1)$ time, respectively. Similarly, a simple randomized preprocessing of the matrices in time $O(n^2\log n+\ell nk)$ after which every query asking for the exact value of an arbitrary entry of the product of the matrices can be answered in $O(\lambda(A,\ell,row)+\lambda(B,k,col)+1)$ time w.h.p.
3. Simple deterministic algorithms for the exact arithmetic matrix product of A and B running in time $O(n^2\min\{\ell+\lambda(A,\ell,row),k+\lambda(B,k,col)\})$. Alternatively, a simple randomized algorithm for the exact product running in time $O(n^2\log n+\ell nk+n^2(\lambda(A,\ell,row)+\lambda(B,k,col))$ w.h.p.

In the fully symmetric case, i.e., when the matrices A, B are square, $\ell=k$, and $\lambda(A,\ell,row)=\lambda(B,k,col)$, the asymptotic running times of our deterministic algorithms for the arithmetic matrix product of 0-1 matrices A, B coincide with the asymptotic running time of the algorithm for the Boolean matrix product presented in [11].

1.2 Techniques

All our main deterministic results rely on the classical, simple 2-approximation algorithm for the k-center clustering problem (farthest-point clustering) due to Gonzalez [12], whose properties are summarized in Fact 1 below. Our results also rely on the idea of updating the inner product of two vectors a and b in $\{0,1\}^q$ over the Boolean or an arithmetic semi-ring to that of two vectors a' and b' in $\{0,1\}^q$, in time roughly proportional to $\mathrm{ham}(a,a')+\mathrm{ham}(b,b')$. The idea has been used in [5–7, 11]. As in some of the aforementioned articles, in case of our alternative deterministic algorithm for 0-1 matrix multiplication, we combine it with a traversal of an approximate minimum spanning tree of the rows or columns of an input matrix in the Hamming space $\{0,1\}^q$, where q is the length of the rows or columns (see Lemma 1).

1.3 Article Organization

The next section contains basic definitions. Section 3 presents a hybrid algorithm for the arithmetic matrix product of 0-1 matrices combing clustering of the rows or columns of the input matrices with the technique of traversing an approximate minimum spanning tree of the rows or columns, respectively. Section 4 is devoted to our approximation algorithms for the arithmetic product of two 0-1 matrices. Section 5 presents preprocessing enabling efficient answers to queries asking for the value of an arbitrary entry of the arithmetic product matrix. As a corollary, an upper time bound on the construction of the exact arithmetic matrix product of 0-1 matrices follows, it matches that yielded by the hybrid algorithm from Section 3. We conclude with a short discussion on potential extensions of our results.

2 Preliminaries

For a positive integer r, $[r]$ stands for the set of positive integers not exceeding r.

For two vectors $(a_1,\ldots,a_d)$ and $(b_1,\ldots,b_d)$ in $\mathbb{R}^d$, their *inner product* is equal to $\sum_{\ell=1}^{d} a_\ell b_\ell$. The transpose of a matrix D is denoted by $D^\top$. If the entries of D are in $\{0,1\}$ then D is a 0-1 matrix.

The *Hamming distance* between two points a, b (vectors) in $\{0,1\}^d$ is the number of coordinates in which the two points differ. Alternatively, it can be defined as the distance between a and b in the L_1 metric over $\{0,1\}^d$. It is denoted by $\mathrm{ham}(a,b)$.

An event is said to hold *with high probability* (w.h.p. for short) in terms of a parameter N related to the input size if it holds with probability at least $1-\frac{1}{N^\alpha}$, for any constant α not less than 1.

The *k-center clustering problem in a Hamming space* is as follows: Given a set P of n points in a Hamming space $\{0,1\}^d$, find a set T of k points in $\{0,1\}^d$ that minimize $\max_{v\in P}\min_{u\in T}\mathrm{ham}(v,u)$. It is known to be NP-hard already for $k=1$ [22] and also NP-hard to approximate within a ratio of $2-\epsilon$ for any constant $\epsilon>0$ when k is part of

the input [13, 23]. Note that according to the problem definition, the points in T do not have to belong to the set P. The *minimum-diameter k-clustering problem in a Hamming space* is: Given a set P of n points in a Hamming space $\{0, 1\}^d$, find a partition of P into k subsets $P_1, P_2, \ldots, P_k$ that minimize $\max_{i\in[k]} \max_{v,u\in P_i} \mathrm{ham}(v, u)$. The k-center clustering problem can also be termed the minimum-radius k-clustering problem.

Fact 1 [12] (Gonzalez's algorithm) There is a simple deterministic 2-approximation algorithm for the k-center clustering and minimum-diameter k-clustering problems in a d-dimensional Hamming space running in $O(ndk)$ time. Moreover, the approximate solution returned by Gonzalez's algorithm consists entirely of points from the input set P.

We will use the last property given in Fact 1 (that is, the approximate solutions found by Gonzalez's algorithm consist of points from the input only) in the proof of Theorem 2.

A combination of a variant of randomized dimension reduction in Euclidean spaces in the framework of Johnson and Lindenstrauss [24] given by Achlioptas in [25] and the observation that the Hamming distance between two 0-1 vectors is equal to their squared L_2 distance makes it possible to decrease the dimension to a logarithmic one in the number of input points, at the cost of worsening the approximation guarantee by some arbitrarily small ϵ. Taking into account the time needed to compute the images of the input points in the subspace of logarithmic dimension, this yields the following fact shown in [21].

Fact 2 [21] For any fixed $\epsilon \in (0, 1/2)$, there is a randomized algorithm for the k-center clustering problem in a Hamming space running in $O(n \log n(d + k)/\epsilon^2)$ time that computes a $(2 + \epsilon)$-approximation of an optimal k-center clustering w.h.p.

3 A Hybrid Algorithm for Multiplication of 0-1 Matrices

In this section, we present a simple deterministic algorithm for the arithmetic matrix product of two 0-1 matrices. It combines an approximate ℓ-center clustering of the rows of the first matrix or an approximate k-center clustering of the columns of the second matrix with the aforementioned technique of traversing an approximate minimum spanning tree of the rows of the first matrix or the columns of the second matrix in an appropriate Hamming space in order to compute a row or column of the product matrix [5–7]. We generalize the technique to include rectangular matrices. The clustering is used for a fast deterministic construction of an approximate minimum spanning tree. The prior algorithms based on the traversing technique use a fast randomized construction of an appropriate approximate minimum spanning tree [5–7].

We shall use the following procedure and lemma in the spirit of [5–7].

procedure $MMCLUS\text{-}ST(A, B, T)$

Input: Two matrices A and B of sizes $p \times q$ and $q \times r$, respectively, and a spanning tree T of the rows of A in the Hamming space $\{0, 1\}^q$.

Output: The arithmetic matrix product C of A and B.

1. Construct a traversal (i.e., a non-necessarily simple path visiting all vertices) U of T.
2. For each pair composed of m-th row A_{m*} of A and i-th row A_{i*} of A,, where the latter row follows the former in the traversal U, compute the set $diff(m, i)$ of indices $h \in [q]$ where $A_{ih} \neq A_{mh}$.
3. For $j = 1, \ldots, r$ do:
   (a) Compute C_{sj} where A_{s*} is the row of A from which the traversal U of T starts.
   (b) While following U, do:
      (i) Set m, i to the indices of the previously traversed row of A and the currently traversed row of A, respectively.
      (ii) Set C_{ij} to C_{mj}.
      (iii) For each $h \in diff(m, i)$, if $A_{ih}B_{hj} = 1$ then set C_{ij} to $C_{ij} + 1$ and if $A_{mh}B_{hj} = 1$ then set C_{ij} to $C_{ij} - 1$.

Define the Hamming cost $\mathrm{ham}(S)$ of a spanning tree S of a point set $P \subset \{0, 1\}^d$ by $\mathrm{ham}(S) = \sum_{(v,u)\in S} \mathrm{ham}(v, u)$.

Lemma 1 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. Given a spanning tree T_A of the rows of A in the Hamming space $\{0, 1\}^q$, $MMCLUS\text{-}ST(A, B, T_A)$ computes the arithmetic matrix product of A and B in time $O(pq + qr + pr + r \times \mathrm{ham}(T_A))$.*

Proof The correctness of $MMCLUS\text{-}ST(A, B, T_A)$ follows from the correctness of the updates of C_{ij} in the block of the inner loop, i.e., in Step 3(b). Step 1 of $MMCLUS\text{-}ST(A, B, T_A)$ can be done in $O(p)$ time while Step 2 requires $O(pq)$ time. The first step in the block under the outer loop, i.e., computing C_{sj} in Step 3(a), takes $O(q)$ time. The crucial observation is that the second step in this block, i.e., Step 3(b), requires $O(p + \mathrm{ham}(T_A))$ time. Simply, the substeps (i), (ii) take $O(1)$ time while the substep (iii) requires $O(|diff(m, i)| + 1)$ time. Since the block is iterated r times, the whole outer loop, i.e., Step 3, requires $O(qr + pr + r\mathrm{ham}(T_A))$ time. Thus, $MMCLUS\text{-}ST(A, B, T_A)$ can be implemented in time $O(pq + qr + rp + r \times \mathrm{ham}(T_A))$. □

Theorem 2 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. Given parameters $\ell \in [p]$ and $k \in [r]$, the arithmetic matrix product of A and B can be computed by a simple deterministic algorithm in time $O(pr + \min\{pq\ell + rq\ell + pr \times \lambda(A, \ell, row), rqk + pqk + pr \times \lambda(B, k, col)\})$.*

Proof We can determine an ℓ-center clustering of the rows of A in $\{0, 1\}^q$ of maximum cluster radius not exceeding $2\lambda(A, \ell, row)$ in $O(pq\ell)$ time by employing Fact 1.

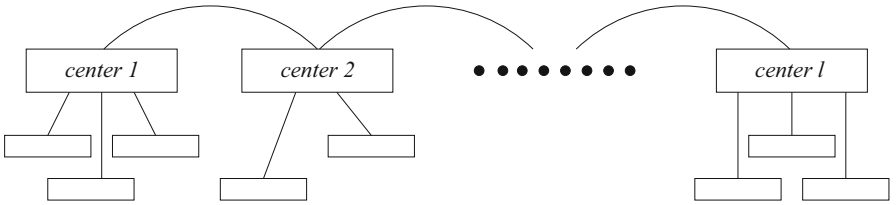

Fig. 1 An example of the spanning tree T_A of the rows of A

Similarly, we can construct a k-center clustering of the columns of B in $\{0,1\}^q$ of maximum cluster radius not exceeding $2\lambda(B,k,col)$ in $O(rqk)$ time. The centers in both aforementioned clusterings are some rows of A and some columns of B, respectively, by the last part of Fact 1. Hence, the ℓ-center clustering gives rise to a spanning tree T_A of the rows of A with all members of a cluster being pendants of their cluster center and the centers connected by a spanning tree of the centers, e.g., a path of length $\ell-1$, see Fig. 1. The Hamming cost of T_A is at most $(p-\ell)2\lambda(A,\ell,row)+(\ell-1)q$. [3] Similarly, we can obtain a spanning tree T_B of the columns of B having the Hamming cost not exceeding $(r-k)2\lambda(B,k,col)+(k-1)q$. Note that the arithmetic matrix product of $B^\top$ and $A^\top$ is equal to the transpose of the arithmetic matrix product of A and B. Hence, to obtain the theorem, we can alternate the steps of $MMCLUS\text{-}ST(A,B,T_A)$ with the steps of $MMCLUS\text{-}ST(B^\top,A^\top,T_B)$ and stop whenever any of the calls is completed. Note also that a k-clustering of columns of B is equivalent to a k-clustering of the rows of $B^\top$ and hence $\lambda(B^\top,k,row)=\lambda(B,k,col)$. The theorem follows from Lemma 1 by substituting the upper bounds on the Hamming cost of T_A and T_B for $\mathrm{ham}(T_A)$ and $\mathrm{ham}(T_B)$ and adding the asymptotic running time of Gonzalez's algorithm for ℓ-center and k-center clusterings, respectively. □

4 An Approximate Arithmetic Matrix Product of 0-1 Matrices

In this section we present two algorithms for approximate arithmetic matrix product of 0-1 matrices. The first is deterministic while the second is randomized. The general idea is to find ℓ approximate centers of the rows of the first matrix or/and k approximate centers of the columns of the second matrix and then use the centers instead of the rows or/and the columns while computing a matrix product. The latter product is a base for the approximate matrix product. A similar idea has been used in [11] to compute the exact Boolean matrix product of square Boolean matrices.

Our deterministic approximation algorithm for the arithmetic matrix product of two 0-1 matrices is specified by the following procedure.

[3] In case the minimum Hamming cost of a spanning tree of the centers was substantially smaller than $(\ell-1)q$, one could reduce the term $rq\ell$ in the upper bound,, at the cost of increasing the running time of the algorithm by that needed for the construction of a nearly optimal spanning tree of the centers.

procedure $MMCLUS\text{-}Approx(A, B, \ell)$

Input: Two 0-1 matrices A and B of sizes $p \times q$ and $q \times r$, respectively, and a positive integer ℓ not exceeding p.

Output: A $p \times r$ matrix D, where for $1 \leq i \leq p$ and $1 \leq j \leq r$, D_{ij} is an approximation of the inner product C_{ij} of the i-th row A_{i*} of A and the j-th column B_{*j} of B.

1. Determine an approximate solution to the ℓ-center clustering of the rows of the matrix A in $\{0, 1\}^q$. For each row A_{i*} of A, set $cen_\ell(A_{i*})$ to its closest center, with ties broken arbitrarily.
2. Form the $\ell \times q$ matrix A', where the i'-th row is the i'-th center in the approximate ℓ-center clustering of the rows of A.
3. Compute the arithmetic $\ell \times r$ matrix product C' of A' and B.
4. For $1 \leq i \leq p$ and $1 \leq j \leq r$, set D_{ij} to $C'_{i'j}$, where the i'-th row $A'_{i'*}$ of A' is $cen_\ell(A_{i*})$.

The idea of the procedure is very simple. First, it runs an approximate algorithm for the ℓ-center problem on the rows of A in step 1. Next, it forms a smaller matrix A' whose ℓ rows are the computed centers of the rows of A in step 2. Then, it computes the arithmetic matrix product C' of the matrices A' and B in step 3, see Fig. 2. After having computed C', the procedure sets each entry D_{ij} of the approximate matrix D to the entry of C' holding the inner product of the center closest to the i-th row with the j-th column of B in step 4.

For a 0-1 $p \times q$ matrix A, let $\lambda(A, \ell, row)$ be the minimum, over all ℓ-center clusterings of the rows of A in the Hamming space $\{0, 1\}^q$, of the maximum Hamming distance between a row of A and the closest center. Similarly, for a 0-1 $q \times r$ matrix B, let $\lambda(B, k, col)$ be the minimum, over all k-center clusterings of the columns of B in the Hamming space $\{0, 1\}^q$, of the maximum Hamming distance between a column of B and the closest center.

Lemma 3 *Suppose a 2-approximation algorithm for the ℓ-center clustering is used in $MMCLUS\text{-}Approx(A, B, \ell)$ and let C stand for the arithmetic product of A and B. Then, for $1 \leq i \leq p$ and $1 \leq j \leq r$, $|C_{ij} - D_{ij}| \leq 2\lambda(A, \ell, row)$.*

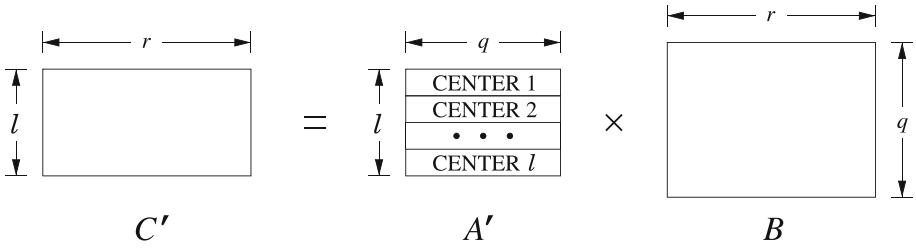

Fig. 2 Computing the matrix C' in step 3 of $MMCLUS\text{-}Approx(A, B, \ell)$

Proof For $1 \leq i \leq p$ and $1 \leq j \leq r$, D_{ij} is the inner product of $cen_\ell(A_{i*})$, where $\text{ham}(A_{i*}, cen_\ell(A_{i*})) \leq 2\lambda(A, \ell, row)$, with B_{*j}. Hence, C_{ij}, which is the inner product of A_{i*} with B_{*j}, can differ at most by $2\lambda(A, \ell, row)$ from D_{ij}. □

By $T(s, q, t)$, we shall denote the worst-case time taken by the multiplication of two 0-1 matrices of sizes $s \times q$ and $q \times t$, respectively.

Lemma 4 *$MMCLUS\text{-}Approx(A, B, \ell)$ runs in $O(pq\ell + pr + T(\ell, q, r))$ time.*

Proof Step 1, which includes the assignment of the closest center to each row of A, can be done in $O(pq\ell)$ time by using Fact 1. Step 2 takes $O(\ell q)$ time, which is $O(T(\ell, q, r))$ time. Finally, Step 3 takes $T(\ell, q, r)$ time while Step 4 can be done in $O(pr)$ time. Thus, the overall time is $O(pq\ell + pr + T(\ell, q, r))$. □

We can use the straightforward $O(sqt)$-time algorithm for the multiplication of two matrices of sizes $s \times q$ and $q \times t$, respectively. Since $T(\ell, q, r) = O(\ell qr)$, Lemmata 3 and 4 yield the first variant (1) of our first main result (Theorem 5 below). The second variant (2) follows from the first one by $(AB)^\top = B^\top A^\top$. Simply, we run $MMCLUS\text{-}Approx(B^\top, A^\top, k)$ to compute an approximation of the transpose of the arithmetic matrix product of A and B. Note also that a k-clustering of columns of B is equivalent to a k-clustering of the rows of $B^\top$ and hence $\lambda(B^\top, k, row) = \lambda(B, k, col)$.

Theorem 5 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. There is a simple deterministic algorithm which provides an approximation of all entries of the arithmetic matrix product of A and B within an additive error of at most:*

1. *$2\lambda(A, \ell, row)$ in time $O(pq\ell + \ell qr + pr)$, or*
2. *$2\lambda(B, k, col)$ in time $O(rqk + pqk + pr)$, respectively.*

The heavy three-linear terms $pq\ell$, ℓqr, rqk, and pqk in the upper time bounds in Theorem 5 originate from the upper time bound on the center clustering problem in Fact 1 and the time taken by matrix multiplication. We can get rid of these terms by using a faster randomized algorithm for approximate center clustering according to Fact 2, both on the rows of the first matrix A and the columns of the second matrix B, in a way similar to [11]. Then, only the small matrices induced by the centers of the rows of A and the columns of B, respectively, need to be multiplied. In this way, the terms $pq\ell$ and rqk are replaced by the presumably smaller term ℓqk in the upper time bound at the cost of increasing the additive error.

procedure $MMCLUS\text{-}R\text{-}Approx(A, B, \ell, k, \epsilon)$

Input: Two 0-1 matrices A and B of sizes $p \times q$ and $q \times r$, respectively, positive integers ℓ, k not exceeding p, r, respectively, and a real number $\epsilon \in (0, 1/2)$.

Output: A $p \times r$ matrix D', where for $1 \leq i \leq p$ and $1 \leq j \leq r$, D'_{ij} is a $(2+\epsilon)$-approximation of the inner product C_{ij} of the i-th row A_{i*} of A and the j-th column B_{*j} of B.

1. Determine an approximate ℓ-center clustering of the rows of the matrix A in $\{0,1\}^q$ by using Fact 2. For each row A_{i*} of A, set $cen_\ell(A_{i*})$ to its closest center, with ties broken arbitrarily.
2. Determine an approximate k-center clustering of the columns of the matrix B in $\{0,1\}^q$ by using Fact 2. For each column B_{*j} of B, set $cen_k(B_{*j})$ to its closest center, with ties broken arbitrarily.
3. Form the $\ell \times q$ matrix A', where the i'-th row is the i'-th center in the approximate ℓ-center clustering of the rows of A.
4. Form the $q \times k$ matrix B', where the j'-th column is the j'-th center in the approximate k-center clustering of the columns of B.
5. Compute the arithmetic $\ell \times k$ matrix product C'' of A' and B'.
6. For $1 \leq i \leq p$ and $1 \leq j \leq r$, set D'_{ij} to $C''_{i'j'}$, where the i'-th row $A'_{i'*}$ of A' is $cen_\ell(A_{i*})$ and the j'-th column $B'_{*j'}$ of B' is $cen_k(B_{*j})$.

Lemma 6 *Let C stand for the arithmetic product of A and B. For $1 \leq i \leq p$ and $1 \leq j \leq r$, $|C_{ij} - D'_{ij}| \leq (2+\epsilon)(\lambda(A, \ell, row) + \lambda(B, k, col))$ holds w.h.p.*

Proof For $1 \leq i \leq p$ and $1 \leq j \leq r$, D'_{ij} is the inner product of $cen_\ell(A_{i*})$, where $\text{ham}(A_{i*}, cen_\ell(A_{i*})) \leq (2+\epsilon)\lambda(A, \ell, row)$ w.h.p., with $cen_k(B_{*j})$, where $\text{ham}(B_{*j}, cen_k(B_{*j})) \leq (2+\epsilon)\lambda(B, k, col)$ w.h.p. by Fact 2. Hence, C_{ij}, which is the inner product of A_{i*} with B_{*j}, can differ at most by $(2+\epsilon)(\lambda(A, \ell, row) + \lambda(B, k, col))$ from D'_{ij} w.h.p. □

Lemma 7 *$MMCLUS\text{-}R\text{-}Approx(A, B, \ell, k, \epsilon)$ can be implemented in time $O(p \log p(q+\ell)/\epsilon^2 + r \log r(q+k)/\epsilon^2 + pr + T(\ell, q, k))$.*

Proof In Step 1, the ℓ centers of the rows of A can be found in $O(p \log p(q+\ell)/\epsilon^2)$ time by Fact 2. Also, the ℓ clusters induced by the centers can be formed in $O(p \log p(q+\ell)/\epsilon^2)$ time in order to approximate minimum-diameter ℓ-clustering of the rows of A; see the proof of Fact 2 in [21]. Importantly, each member of such a cluster is within a Hamming distance not exceeding $(2+\epsilon)\lambda(A, \ell, row)$ from its center. Symmetrically, Step 2 takes $O(r \log r(q+k)/\epsilon^2)$ time. Steps 3 and 4 can be done $O(\ell q)$ time and $O(kq)$ time, respectively, which is $O(T(\ell, q, k))$ time. Finally, Step 5 takes $T(\ell, q, k)$ time while Step 6 can be done in $O(pr)$ time. Thus, the overall time is $O(p \log p(q+\ell)/\epsilon^2 + r \log r(q+k)/\epsilon^2 + pr + T(\ell, q, k))$. □

Lemmata 6 and 7 yield our second theorem.

Theorem 8 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. There is a simple randomized algorithm which for any $\epsilon \in (0, 1/2)$ provides an approximation of all entries of the arithmetic matrix product of A and B within an additive error of at most $(2+\epsilon)(\lambda(A, \ell, row) + \lambda(B, k, col))$ w.h.p. in time $O(p \log p(q+\ell)/\epsilon^2 + r \log r(q+k)/\epsilon^2 + \ell qk + pr)$.*

5 Preprocessing for Answering Single-entry Queries

In this section, we use our algorithms for an approximate matrix product of 0-1 matrices for the construction of a preprocessing of the input matrices enabling fast answers to queries asking for the exact value of a single entry in the product matrix. As corollaries, we obtain upper time bounds on computing the exact matrix product, one of them matching that of Theorem 2.

We can apply $MMCLUS\text{-}Approx(A, B, \ell)$ to obtain a preprocessing for answering queries about single entries of the arithmetic matrix product of A and B. In the first step of the preprocessing, $MMCLUS\text{-}Approx(A, B, \ell)$ is run. In the second step, for each row of A, the set of indices of coordinates on which it differs from its closest center is computed.

procedure $MMCLUS\text{-}Preproc(A, B, \ell)$

Input: Two 0-1 matrices A and B of sizes $p \times q$ and $q \times r$, respectively, and a positive integer ℓ not exceeding p.

Output: The $p \times r$ matrix D returned by $MMCLUS\text{-}Approx(A, B, \ell)$, and for $1 \le i \le p$, the set of coordinate indices $ind(A, i)$ on which A_{i*} differs from the closest center.

1. Run $MMCLUS\text{-}Approx(A, B, \ell)$.
2. For $1 \le i \le p$, determine the set $ind(A, i)$ of coordinate indices on which A_{i*} differs from $cen_\ell(A_{i*})$.

Lemma 9 *$MMCLUS\text{-}Preproc(A, B, \ell)$ runs in $O(pq\ell + pr + T(\ell, q, r))$ time.*

Proof Step 1 can be done in $O(pq\ell + pr + T(\ell, q, r))$ time by Lemma 4. Step 2 can easily be implemented in $O(pq)$ time. □

The idea of our procedure for answering a query about a single entry C_{ij} of the matrix product of A and B is to recover the exact value of the entry from its approximation. Recall that the approximation is the inner product of the center assigned to the i-th row of A with the j-th column of B. The correction takes time proportional to the size of the set of indices of coordinates on which the i-th row and the center differ. See the procedure $MMCLUS\text{-}Query(A, B, \ell, i, j)$ for details.

procedure $MMCLUS\text{-}Query(A, B, \ell, i, j)$

Input: The preprocessing done by $MMCLUS\text{-}Preproc(A, B, \ell)$ for 0-1 matrices A and B of sizes $p \times q$ and $q \times r$, respectively, $\ell \in [p]$, and two query indices $i \in [p]$ and $j \in [r]$.

Output: The inner product C_{ij} of the i-th row A_{i*} of A and the j-th column B_{*j} of B.

1. Set C_{ij} to the entry D_{ij} of the matrix D computed by $MMCLUS\text{-}Approx(A, B, \ell)$ in $MMCLUS\text{-}Preproc(A, B, \ell)$.
2. For $m \in ind(A, i)$ do
   (a) If the m-th coordinate of the center assigned to A_{i*} is 0 and $B_{mj} = 1$ then $C_{ij} \leftarrow C_{ij} + 1$.
   (b) If the m-th coordinate of the center assigned to A_{i*} is 1 and B_{mj} is also 1 then $C_{ij} \leftarrow C_{ij} - 1$.

Lemma 10 *$MMCLUS\text{-}Query(A, B, \ell, i, j)$ is correct, i.e., the final value of C_{ij} is the inner product of the i-th row A_{i*} of A and the j-th column B_{*j} of B.*

Proof C_{ij} is initially set to D_{ij}, which is the inner product of the center assigned to A_{i*} and B_{*j}. Then, C_{ij} is appropriately corrected by increasing or decreasing with 1 for each coordinate index $m \in ind(A, i)$ which contributes 1 to the inner product of A_{i*} and B_{*j} and 0 to the inner product of the center of A_{i*} and B_{*j} or *vice versa*. □

Lemma 11 *$MMCLUS\text{-}Query(A, B, \ell, i, j)$ takes $O(\lambda(A, \ell, row)+1)$ time.*

Proof Recall that $2\lambda(A, \ell, row)$ is an upper bound on the maximum Hamming distance between a row of A and the closest center in the ℓ-center clustering computed by $MMCLUS\text{-}Approx(A, B, \ell)$ in $MMCLUS\text{-}Preproc(A, B, \ell)$. Step 1 takes $O(1)$ time. Since the m-th coordinate in the centers can be accessed in the matrix A' computed by $MMCLUS\text{-}Approx(A, B, \ell)$, each of the two substeps in the block of the loop in Step 2 can be done in $O(1)$ time. Finally, since $|ind(A, i)| \leq 2\lambda(A, \ell, row)$, the block is iterated at most $2\lambda(A, \ell, row)$ times. Consequently, the whole Step 2 takes $O(\lambda(A, \ell, row)+1)$ time. □

By putting Lemmata 9, 10, and 11 together, and using the straightforward $O(sqt)$-time algorithm to multiply matrices of size $s \times q$ and $q \times t$, we obtain the first variant of our next main result. The second variant reduces to the first one by $(AB)^\top = B^\top A^\top$. We simply run $MMCLUS\text{-}Preproc(B^\top, A^\top, k)$ and $MMCLUS\text{-}Query(B^\top, A^\top, k, j, i)$ instead.

Theorem 12 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. Given parameters $\ell \in [p]$ and $k \in [r]$, the matrices can be preprocessed by a simple deterministic algorithm in $O(pq\ell + \ell qr + pr)$ time or $O(rqk + pqk + pr)$ time such that a query asking for the exact value of a single entry C_{ij} of the arithmetic matrix product C of A and B can be answered in $O(\lambda(A, \ell, row)+1)$ time or $O(\lambda(B, k, col)+1)$ time, respectively.*

Analogously, by using Lemmata 6 and 7, we obtain an alternative, randomized variant of Theorem 12.

To begin with, we modify $MMCLUS\text{-}Preproc(A, B, \ell)$ to $MMCLUS\text{-}R\text{-}Preproc(A, B, \ell, k, \epsilon)$. The modified procedure runs $MMCLUS\text{-}R\text{-}Approx(A, B, \ell, k, \epsilon)$ instead of $MMCLUS\text{-}Approx(A, B, \ell)$ in its first step. Its second step is the same as in $MMCLUS\text{-}Preproc(A, B, \ell)$. In an additional third step, the set $ind(B, j)$ of coordinate indices on which B_{*j} differs from $cen_k(B_{*j})$ is computed for $1 \leq j \leq r$. By Lemma 7 and the fact that the second and third steps can be done in $O(pq + rq)$ tine, we infer that $MMCLUS\text{-}R\text{-}Preproc(A, B, \ell, k, \epsilon)$ can be implemented in time $O(p \log p(q + \ell)/\epsilon^2 + r \log r(q + k)/\epsilon^2 + pr + T(\ell, q, k))$.

Next, we modify $MMCLUS\text{-}Query(A, B, \ell, i, j)$ to $MMCLUS\text{-}R\text{-}Query(A, B, \ell, k, \epsilon, i, j)$, replacing $MMCLUS\text{-}Preproc(A, B, \ell)$ with $MMCLUS\text{-}R\text{-}Preproc(A, B, \ell, k, \epsilon)$ in the input. In the first analogous step of the modified procedure, C_{ij} is set to the entry D'_{ij} of the matrix D' computed by the call of $MMCLUS\text{-}R\text{-}Approx(A, B, \ell, k, \epsilon)$ in $MMCLUS\text{-}R\text{-}Preproc(A, B, \ell, k, \epsilon)$. The second step is the same as in $MMCLUS\text{-}Query(A, B, \ell, i, j)$. In an additional third step, C_{ij} is symmetrically updated with respect to $ind(B, j)$ as follows:

For $m \in ind(B, j)$, if the m-th coordinate of the center assigned to B_{*j} is 0 and $A_{im} = 1$ then C_{ij} is increased by 1. Also, if the m-th coordinate of the center assigned to B_{*j} is 1 and A_{im} is also 1 then C_{ij} decreased by 1.

The correctness of $MMCLUS\text{-}R\text{-}Query(A, B, \ell, k, \epsilon, i, j)$ is obvious as that of $MMCLUS\text{-}Query(A, B, \ell, i, j)$. By Lemma 6, $MMCLUS\text{-}R\text{-}Query(A, B, \ell, k, \epsilon, i, j)$ takes $O(\lambda(A, \ell, row) + \lambda(B, k, col) + 1)$ time w.h.p.

By the correctness of $MMCLUS\text{-}R\text{-}Query(A, B, \ell, k, \epsilon, i, j)$ and $MMCLUS\text{-}R\text{-}Preproc(A, B, \ell, k, \epsilon)$, and the upper time bounds on the running times of these procedures, we obtain the following theorem.

Theorem 13 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. Given parameters $\ell \in [p]$ and $k \in [r]$, the matrices can be preprocessed by a simple randomized algorithm in time $O(p \log p(q + \ell) + r \log r(q + k) + \ell qk + pr)$ such that a query asking for the exact value of a single entry C_{ij} of the arithmetic matrix product C of A and B can be answered in $O(\lambda(A, \ell, row) + \lambda(B, k, col)+1)$ time w.h.p.*

Theorems 12 and 13 can be especially useful in situations where some moderate number of entries in the arithmetic product matrix of the input 0-1 matrices are of interest. Then, the overall time taken by the preprocessing and the queries might be substantially smaller than that when computing each of the entries separately as well as that needed to compute the whole matrix product.

Theorem 12 yields the following corollary.

Corollary 14 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. Given parameters $\ell \in [p]$ and $k \in [r]$, the arithmetic matrix product of A and B can be computed by a simple deterministic algorithm in time $O(pr + \min\{pq\ell + \ell qr + pr\lambda(A, \ell, row), rqk + pqk + pr\lambda(B, k, col)\})$.*

Proof It follows from Theorem 12 that by querying for the value of each entry in the arithmetic matrix product A and B we can compute the product in time

1. $O(pq\ell + \ell qr + pr(\lambda(A, \ell, row)) + 1)$ or
2. $O(rqk + pqk + pr(\lambda(B, k, col) + 1))$.

To obtain the upper time bound stated in the theorem, we can just alternate the steps of the algorithm yielding the first upper bound with the steps of the algorithm yielding the second upper bound and stop whenever any of these algorithms stops. □

Similarly, we obtain the following corollary from Theorem 13.

Corollary 15 *Let A and B be two 0-1 matrices of sizes $p \times q$ and $q \times r$, respectively. Given parameters $\ell \in [p]$ and $k \in [r]$, the arithmetic matrix product of A and B can be computed by a simple randomized algorithm in time $O(p \log p(q+\ell)+r \log r(q+k) + \ell qk + pr(\lambda(A, \ell, row) + \lambda(B, k, col)+1))$ w.h.p.*

Note that the upper time bound from Corollary 14 coincides with that from Theorem 2. However, the origins of the terms ℓqr and pqk in these two upper bounds are quite different. In case of Corollary 14, the terms are caused by the use of the straightforward matrix multiplication algorithm. They clearly could be improved if we used fast arithmetic multiplication, but then we would lose the simplicity of our algorithm. In case of Theorem 14, these terms reflect the asymptotic worst-case Hamming cost of connecting the ℓ or k centers by an easily construable spanning tree. The cost might be much lower but it depends on the centers[4]. Finally, in the fully symmetric case, where $p = q = r = n$, $\ell = k$, and $\lambda(A, \ell, row) = \lambda(B, k, col)$, the upper time bound from Corollary 14 and Theorem 2 coincides with that for the Boolean matrix product established in [11].

6 Potential Extensions

The rows or columns in the input 0-1 matrices can be very long. Also, a large number of clusters might be needed in order to obtain a low upper bound on their radius. Among other things, for these reasons, we have picked Gonzalez's classical algorithm for the k-center clustering problem [12] as a basic tool in our deterministic approach to the arithmetic matrix product of two 0-1 matrices with clustered rows or columns. The running time of his algorithm is linear not only in the number of input points but also in their dimension, and in the parameter k. Importantly, it is very simple, deterministic, provides a solution within 2 of the optimum, and can be applied in Hamming spaces. For instance, there exist faster (in terms of n and k) 2-approximation algorithms for k-center clustering with hidden exponential dependence on the dimension in their running time, see [13, 26]. Among several newer works on speeding approximation algorithms for k-center clustering (e.g., [21, 27–29]) only [21] explicitly includes Hamming spaces. The randomized approximation method for the k-center clustering problem in Hamming spaces from [21] is summarized in Fact 2. It provides approximation guarantees arbitrarily close to 2, is substantially faster than

[4] One could also connect the centers to their 1-median whose i-th coordinate is 1 if the majority of centers have 1 on this coordinate otherwise it is 0. In this way the total Hamming cost of connecting the centers reduces to $\ell q/2$ or $kq/2$, respectively, but this does not yield any asymptotic improvement.

Gonzalez's algorithm when the dimension and k are superlogarithmic, and is relatively simple. For these reasons, our improved randomized algorithms for the approximate and exact 0-1 matrix multiplication are based on this method.

One could easily generalize our main results by replacing approximation algorithms for k-center clustering with approximation algorithms for the more general problem of k-center clustering with outliers [30]. In the latter problem, a given number z of input points could be discarded as outliers when trying to minimize the maximum cluster radius. Unfortunately, the algorithms for this more general problem tend to be more complicated and the focus seems to be on the approximation ratio achievable in polynomial time (e.g., 3 in [30] and 2 in [31]) and not on the time complexity.

There are many other variants of clustering than k-center clustering, and plenty of methods have been developed for them in the literature. In fact, in the design of efficient algorithms for the exact arithmetic matrix product of 0-1 matrices with clustered rows or columns, using the k-median clustering could seem more natural. The objective in the latter problem is to minimize the sum of distances between the input points and their nearest centers. Unfortunately, no simple deterministic $O(1)$-approximation algorithms for the latter problem that are efficient in case the dimension and k parameters are large seem be to available [32, 33].

Our approximate and exact algorithms for the matrix product of 0-1 matrices as well as the preprocessing of the matrices can be categorized as supervised since they assume that the user has some knowledge on the input matrices and can choose reasonable values of the parameters ℓ and k guaranteeing relatively low overall time complexity. Otherwise, one can try the ℓ-center and k-center clustering subroutines for a number of combinations of different values of ℓ and k in order to pick the combination yielding the lowest upper bound on the overall time complexity of the algorithm or preprocessing. The situation is somewhat easier to handle when the input matrices are square, as shown in the following example.

Example Let A and B be two 0-1 $n \times n$ matrices. The upper bound on the time needed to compute their arithmetic product of A and B in Theorem 2 and Corollary 14 simplifies to $O(n^2 \min\{\ell + \lambda(A, \ell, row), k + \lambda(B, k, col\}))$. Suppose that we would like to compute the product in time not exceeding roughly $ct(n)$, where t is some nondecreasing positive-integer function satisfying $t(n) \geq n^2$ and c is a small positive constant. This implies that both ℓ and k should not exceed $n^2/t(n)$ too much. On the other hand, both $\lambda(A, \ell, row)$ and $\lambda(B, k, col)$ are nonincreasing with respect to ℓ or k, respectively. Hence, we should pick ℓ and k as large as possible, e.g., $\lceil t(n)/n^2 \rceil$. Note that we can compute 2-approximations of $\lambda(A, \lceil t(n)/n^2 \rceil, row)$ and $\lambda(B, \lceil t(n)/n^2 \rceil, col)$ by running Gonzalez's algorithm in $O(t(n))$ time. If neither $\lambda(A, \lceil t(n)/n^2 \rceil, row)$ nor $\lambda(B, \lceil t(n)/n^2 \rceil, col)$ is at most $dt(n)/n^2$, where d is some low constant not exceeding c, then we need to increase, e.g., double, the threshold function $t(n)$ and repeat the procedure.

Acknowledgements The authors are grateful to the anonymous reviewers for their valuable comments and suggestions.

Author Contributions All coauthors contributed to various parts of the manuscript in ideas and writing and reviewed the whole manuscript. J.J. and M.K. also prepared the figures.

Funding Open access funding provided by Lund University. Jesper Jansson was partially supported by Japan Society for the Promotion of Science - KAKENHI grant 24K22294.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Alman, J., Duan, R., Vassilevska Williams, V., Xu, Y., Xu, Z., Zhou, R.: More asymmetry yields faster matrix multiplication. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2025), pp. 2005–2039. ACM-SIAM, Philadelphia, PA (2025)
2. Vassilevska Williams, V., Xu, Y., Xu, Z., Zhou, R.: New bounds for matrix multiplication: from alpha to omega. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2024), pp. 3792–3835. ACM-SIAM, Philadelphia, PA (2024)
3. Arslan, A.N., Chidri, A.: An efficient multiplication algorithm for thin matrices and for matrices with similar rows and columns. In: Proceedings of the 2010 International Conference on Scientific Computing (CSC 2010), pp. 147–152. CSREA Press, U.S.A. (2010)
4. Anand, E., Brand, J., McCarthy, R.: The Structural Complexity of Matrix-Vector Multiplication. arXiv:2502.21240 (2025)
5. Björklund, A., Lingas, A.: Fast Boolean matrix multiplication for highly clustered data. In: Proceedings of the Algorithms and Data Structures Symposium (WADS 2001), pp. 258–263. Springer, Berlin Heidelberg (2001)
6. Floderus, P., Jansson, J., Levcopoulos, C., Lingas, A., Sledneu, D.: 3D rectangulations and geometric matrix multiplication. Algorithmica **80**(1), 136–154 (2018)
7. Gąsieniec, L., Lingas, A.: An improved bound on Boolean matrix multiplication for highly clustered data. In: Proceedings of the Algorithms and Data Structures Symposium (WADS 2003), pp. 329–339. Springer, Berlin Heidelberg (2003)
8. Pagh, R.: Compressed matrix multiplication. ACM Trans. Computat. Theor. (TOCT) **5**(3), 1–17 (2013)
9. Yuster, R., Zwick, U.: Fast sparse matrix multiplication. ACM Trans. Algorithms **1**, 2–13 (2005)
10. Alves, J.N.F., Moustafa, S., Benkner, S., Francisco, A.P., Gansterer, W.N., Russo, L.M.S.: Accelerating graph neural networks using a novel computation-friendly matrix compression format. In: Proceedings of the 2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS 2025), pp. 1091–1103. IEEE, Piscataway, NJ (2025)
11. Arslan, A.N., Chidri, A.: A clustering-based matrix multiplication algorithm. In: Proceedings of the 2011 International Conference on Scientific Computing (CSC 2011), pp. 303–307. CSREA Press, U.S.A. (2011)
12. Gonzalez, T.: Clustering to minimize the maximum intercluster distance. Theoret. Comput. Scirnce. **38**, 293–306 (1985)
13. Feder, T., Greene, D.: Optimal algorithms for approximate clustering. In: Proceedings of ACM Symposium on Theory of Computing (STOC 1988), pp. 434–444. ACM, Chicago, IL (1988)
14. Cohen, E., Lewis, D.D.: Approximating matrix multiplication for pattern recognition tasks. J. Algorithms **30**(2), 211–252 (1999)

15. Gąsieniec, L., Kowaluk, M., Lingas, A.: Faster multi-witnesses for Boolean matrix multiplication. Inf. Process. Lett. **109**(4), 242–247 (2009)
16. Floderus, P., Kowaluk, M., Lingas, A., Lundell, E.: Detecting and counting small pattern graphs. SIAM J. Discret. Math. **29**(3), 1322–1339 (2015)
17. Drineas, P., Kannan, R., Mahoney, M.W.: Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. SIAM J. Comput. **36**(1), 132–157 (2006)
18. Sarlós, T.: Improved approximation algorithms for large matrices via random projections. In: Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS 2006), pp. 143–152. IEEE Computer Society, Piscataway, NJ (2006)
19. Kurzkov, K.: Deterministic algorithms for skewed matrix products. In: Proceedings of the International Symposium on Theoretical Aspects of Computer Science (STACS 2013). LIPIcs, vol. 20, pp. 466–477. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany (2013)
20. Iven, M.A., Spencer, C.V.: A note on compressed sensing and the complexity of matrix multiplication. Inf. Process. Lett. **109**(10), 468–471 (2009)
21. Kowaluk, M., Lingas, A., Persson, M.: Fast approximate ℓ-center clustering in high dimensional space. Algo. MDPI. **19**(3), 243 (2026)
22. Frances, M., Litman, A.: On covering problems of codes. Theor. Comput. Syst. **30**(2), 113–119 (1997)
23. Gąsieniec, L., Jansson, J., Lingas, A.: Approximation algorithms for Hamming clustering problems. J. Discrete Algo. **2**(2), 289–301 (2004)
24. Johnson, W.B., Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. In: Conference in Modern Analysis and Probability. Contemporary Mathematics, vol. 26, pp. 189–206. American Mathematical Society, Providence, RI (1984)
25. Achlioptas, D.: Database-friendly random projections: Johnson-Lindenstrauss with binary coins. J. Comput. Syst. Sci. **66**(4), 671–687 (2003)
26. Har-Peled, S., Mendel, M.: Fast construction of nets in low-dimensional metrics and their applications. SIAM J. Comput. **35**(5), 1148–1184 (2006)
27. Eppstein, D., Har-Peled, S., Sidiropoulos, A.: Approximate greedy clustering and distance selection for graph metrics. J. Comput. Geom. **11**(1), 629–652 (2020)
28. Filtser, A., Jiang, S.H.C., Li, Y., Naredla, A.M., Psarros, I., Yang, Q., Zhang, Q.: Faster approximation algorithms for k-center via data reduction. In: Proceedings of the 42nd International Conference on Machine Learning. PMLR, vol. 267, pp. 17189–17202 (2025)
29. Jiang, S.H.C., Krauthgamer, R., Sapir, S.: Moderate dimension reduction for k-center clustering. In: Proceedings of the International Symposium on Computational Geometry (SoCG 2024). LIPIcs, vol. 293, pp. 64–16416. Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany (2024)
30. Charikar, M., Khuller, S., Mount, D.M., Narasimhan, G.: Algorithms for facility location problems with outliers. In: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), pp. 642–651. ACM-SIAM, Philadelphia, PA (2001)
31. Harris, D.G., Pensyl, T., Srinivasan, A., Trinh, K.: A lottery model for center-type problems with outliers. ACM Trans. Algo. **15**(3), 1–25 (2019). Article No. 36
32. Charikar, M., Guha, S., Tardos, E., Shmoys, D.B.: A constant-factor approximation algorithm for the k-median problem. In: Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC 1999), pp. 1–10. ACM, New York, NY (1999)
33. Chen, K.: On coresets for k-median and k-means clustering in metric and Euclidean spaces and their applications. SIAM J. Comput. **39**(3), 923–947 (2009)