Multiplication of 0-1 Matrices via Clustering

Jesper Jansson[1], Mirosław Kowaluk[2], Andrzej Lingas[3(✉)], and Mia Persson[4]

[1] Graduate School of Informatics, Kyoto University, Kyoto, Japan
jj@i.kyoto-u.ac.jp
[2] Institute of Informatics, University of Warsaw, Warsaw, Poland
kowaluk@mimuw.edu.pl
[3] Department of Computer Science, Lund University, Lund, Sweden
Andrzej.Lingas@cs.lth.se
[4] Department of Computer Science and Media Technology, Malmö University, Malmö, Sweden
Mia.Persson@mau.se

Abstract. We study applications of clustering (in particular the k-center clustering problem) in the design of 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. Let λ_A and λ_B denote the minimum maximum radius of a cluster in an ℓ-center clustering of the rows of A and in a k-center clustering of the columns of B, respectively. 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 $2\lambda_A$ in $O(n^2\ell)$ time or at most $2\lambda_B$ in $O(n^2k)$ time.
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 $O(\lambda_A)$ time or $O(\lambda_B)$ time, respectively.
3. A simple deterministic algorithm for the exact arithmetic matrix product of A and B running in time $O(n^2(\ell + k + \min\{\lambda_A, \lambda_B\}))$.

Keywords: arithmetic matrix multiplication · clustering · Hamming space · minimum spanning tree

1 Introduction

The arithmetic matrix product of two 0-1 matrices is closely related to the Boolean one of the corresponding Boolean matrices. For square $n \times n$ matrices, both can be computed in $O(n^{2.372})$ time [1,23]. Both are basic tools in science and engineering. Unfortunately, no truly subcubic practical algorithms for any of them are known. Therefore, many researchers studied the complexity of these

V. Chau et al. (Eds.): IJTCS-FAW 2025, LNCS 15828, pp. 92–102, 2025.
https://doi.org/10.1007/978-981-96-8312-3_7

products for special input matrices, e.g., sparse or structured matrices [3,4,11, 15,21,24], providing faster and often more practical algorithms.

The method of multiplying matrices with clustered rows or columns, proposed for Boolean matrix product in [4] and subsequently generalized in [11,15] and used in [2], relies on the construction of an approximate 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 randomized dimension reduction.

In case of the arithmetic matrix product of 0-1 matrices, in some cases, a faster approximate arithmetic matrix multiplication can be more useful [8,21]. Among other things, it can enable to identify largest entries in the product matrix and it can be also used to provide a fast estimation of the number of: the so called witnesses for the Boolean product of two Boolean matrices [14], triangles in a graph, or more generally, subgraphs isomorphic to a small pattern graph [12] 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) [8,21].

Cohen and Lewis [8] and Drineas *et al.* [9] used random sampling to approximate arithmetic matrix product. Their papers 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 [21]). The approximation algorithm in [9] runs in $O(n^2c)$ time. Drineas et al. [9] also derived bounds on the entrywise differences between the exact matrix product and its approximation. Unfortunately, the best of these bounds is $\Omega(M^2n/\sqrt{c})$, where M is the maximum value of an entry in A and B. By using a sketch technique, Sarlós [22] obtained the same Frobenius norm guarantees, also in $O(n^2c)$ 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 [21] 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 cor-

rect 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 [20] developed analogous deterministic algorithms employing different techniques. For approximation results related to sparse arithmetic matrix products, see [19,21].

1.1 Our Contributions

In this paper, we exploit the possibility of applying the classic simple 2-approximation algorithm for the k center clustering problem [16] 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.

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, (the number of coordinate positions they differ from each other). Each center in T induces a cluster consisting of all points in P for which it is the nearest center.

Let λ_A and λ_B denote the minimum maximum radius of a cluster in an ℓ-center clustering of the rows of A or in a k-center clustering of the columns of B, respectively. Assuming that A and B are of sizes $p\times q$ and $q\times r$, respectively, 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 $2\lambda_A$ in $O(pq\ell + pr)$ time if $p \geq r$ or at most $2\lambda_B$ in $O(qrk + pr)$ time if $p \leq r$.
2. A simple deterministic preprocessing of the matrices A and B in $O(pq\ell + pr)$ time if $p \geq r$ or $O(qrk + pr)$ time if $p \leq r$ 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 $O(\lambda_A)$ time if $p \geq r$ or $O(\lambda_B)$ time if $p \leq r$.
3. A simple deterministic algorithm for the exact arithmetic matrix product of A and B running in time $O(pq\ell + rqk + \min\{pr\lambda_A + rq\ell, pr\lambda_B + pqk\})$.

1.2 Techniques

All our main results rely on the classical, simple 2-approximation algorithm for the k-center clustering problem (farthest-point clustering) due to Gonzalez [16] (see also Fact 1). Two of them rely also 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 $\{0,1\}^q$, where $a = a'$ or $b = b'$, in time roughly proportional to $\mathrm{ham}(a,a') + \mathrm{ham}(b,b')$. The idea has been used in [4,11,15]. As in the aforementioned papers, 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 also Lemma 6).

1.3 Paper Organization

The next section contains basic definitions. Section 3 presents our approximation algorithm for the arithmetic product of two 0-1 matrices and the preprocessing enabling efficient answers to queries asking for the value of an arbitrary entry of the arithmetic product matrix. Section 4 is devoted to our algorithm for the exact arithmetic matrix product of two 0-1 matrices. We conclude with a short discussion on possible extensions of our results.

2 Preliminaries

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

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 the 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 $\mathrm{ham}(a,b)$.

The k-center clustering problem in a Hamming space $\{0,1\}^d$ is as follows: given a set P of n points in $\{0,1\}^d$, find a set T of k points in $\{0,1\}^d$ that minimize $\max_{v\in P} \min_{u\in T} \mathrm{ham}(v,u)$.

The minimum-diameter k-clustering problem in a Hamming space $\{0,1\}^d$ is as follows: given a set P of n points in $\{0,1\}^d$, find a partition of P into k subsets $P_1, P_2, \ldots, P_k$ that minimize $\max_{i\in[k]} \max_{v,u\in P_i} \mathrm{ham}(v,u)$. Note that the k-center clustering problem could be also termed as the minimum-radius k-clustering problem. It is known to be NP-hard and even NP-hard to approximate within $2-\epsilon$ for any constant $\epsilon > 0$ [10,13].

Fact 1. *[16] Let P be a set of n points in $\{0,1\}^d$, and let $k \in [n]$. There is a simple deterministic 2-approximation algorithm for the k-center clustering and minimum-diameter k-clustering problems running in $O(ndk)$ time.*

3 An Approximate Arithmetic Matrix Product of 0-1 Matrices

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

procedure $APPROXMMCLUS(A,B,\ell)$
Input: Two 0-1 matrices A and B of sizes $p\times q$ and $q\times r$, respectively, where $p \geq r$, 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 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 the center of the cluster in the ℓ-clustering to which A_{i*} belongs.
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 $cen_\ell(A_{i*})$.

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 center of a cluster and a member of the cluster. 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 center of a cluster and a member of the cluster.

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

Proof. Recall that $p \geq r$ is assumed in the input to $APPROXMMCLUS(A, B, \ell)$. For $1 \leq i \leq p$ and $1 \leq j \leq r$, D_{ij} is the inner product of $cen_\ell(A_{i*})$, where $\mathrm{ham}(A_{i*}, cen_\ell(A_{i*})) \leq 2\lambda(A, \ell, row)$, 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 2. *$APPROXMMCLUS(A, B, \ell)$ can be implemented in $O(pq\ell + pr + T(\ell, q, r))$ time.*

Proof. Recall that $p \geq r$. 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, i.e., the classic algorithm of Gonzalez [16]. 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 $T(\ell, q, r) = O(\ell q\ \mathrm{r}) = O(\ell qp)$ if $p \geq r$, Lemmata 1 and 2 yield the first part (1) of our first main result (Theorem 1 below), for $p \geq r$. Its second part (2) for $p \leq r$ follows from the first part by $(AB)^\top = B^\top A^\top$. Note that then the number of rows in $B^\top$, which is r, is not less than the number of columns in $A^\top$, which is p. Simply, we run $APPROXMMCLUS(B^\top, A^\top, k)$ in order 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 that $\lambda(B^\top, k, row) = \lambda(B, k, col)$.

Theorem 1. *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* $O(pq\ell + pr)$ *if* $p \geq r$,
2. $2\lambda(B, k, col)$ *in time* $O(rqk + pr)$ *if* $p \leq r$.

We slightly extend $APPROXMMCLUS(A, B, \ell)$ in order to obtain a preprocessing for answering queries about single entries of the arithmetic matrix product of A and B.

procedure $PREPROCMMCLUS(A, B, \ell)$
Input: Two 0-1 matrices A and B of sizes $p \times q$ and $q \times r$, respectively, where $p \geq r$, and a positive integer ℓ not exceeding p.
Output: The $p \times r$ matrix D returned by $APPROXMMCLUS(A, B, \ell)$, and for $1 \leq i \leq p$, the set of coordinate indices $ind(A, i)$ on which A_{i*} differs from its cluster center.

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

Lemma 3. $PREPROCMMCLUS(A, B, \ell)$ *can be implemented in* $O(pq\ell + pr + T(\ell, q, r))$ *time.*

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

Our procedure for answering a query about a single entry of the matrix product of A and B is as follows.

procedure $QUERYMMCLUS(A, B, \ell, i, j)$
Input: The preprocessing done by $PREPROCMMCLUS(A, B, \ell)$ for 0-1 matrices A and B of sizes $p \times q$ and $q \times r$, respectively, where $p \geq r$, $\ell \in [p]$, 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 $APPROXMMCLUS(A, B, \ell)$ in $PREPROCMMCLUS(A, B, \ell, k)$.
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 4. $QUERYMMCLUS(A, B, \ell, i, j)$ *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 5. *$QUERYMMCLUS(A, B, \ell, k, i, j)$ takes $O(\lambda(A, \ell, row))$ time.*

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

By putting Lemmata 3, 4, and 5 together, and using the straightforward $O(sqt)$-time algorithm to multiply matrices of size $s \times q$ and $q \times t$, we obtain our next main result for $p \geq r$. The case $p \leq r$ reduces to the case $p \geq r$ by $(AB)^\top = B^\top A^\top$. Recall that then the number of rows in $B^\top$, which is r, is not less than the number of columns in $A^\top$, which is p. Also, we have $\lambda(B^\top, k, row) = \lambda(B, k, col)$. We simply run $PREPROCMMCLUS(B^\top, A^\top, k)$ and $QUERYMMCLUS(B^\top, A^\top, k, j, i)$ instead.

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 matrices can be preprocessed by a simple deterministic algorithm in $O(pq\ell + pr)$ time if $p \geq r$ or $O(rqk + pr)$ time if $p \leq r$ 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 $O(\lambda(A, \ell, row))$ time if $p \geq r$ or $O(\lambda(B, k, col))$ time if $p \leq r$.*

4 The Exact Arithmetic Matrix Product of 0-1 Matrices

Theorem 2 yields the following corollary.

Corollary 1. *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 $O(pq\ell + pr\lambda(A, \ell, row))$ time if $p \geq r$ or $O(rqk + pr\lambda(B, k, col))$ time if $p \leq r$.*

There is however a slightly better way of obtaining a simple deterministic algorithm for the arithmetic matrix product of two 0-1 matrices via ℓ-center clustering of the rows of the first matrix or k-center clustering of the columns of the second matrix. The idea is to use 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 [4,11,15]. The technique easily generalizes to 0-1 rectangular matrices. We shall use the following procedure and lemma in the spirit of [4,11,15].

procedure $MMST(A, B, T)$
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 any pair A_{m*}, A_{i*}, 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 $j = 1, \ldots, r$, iterate the following steps:
   (a) Compute C_{sj} where A_{s*} is the row of A from which the traversal U of T starts.
   (b) While following U, iterate the following steps:
      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 6. *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 and a spanning tree T_B of the columns of B in the Hamming space $\{0,1\}^q$, the arithmetic matrix product of A and B can be computed in time $O(pq + qr + pr + \min\{r \times \mathrm{ham}(T_A), p \times \mathrm{ham}(T_B)\})$.*

Proof. First, we shall prove that $MMST(A, B, T_A)$ computes the arithmetic matrix product of A and B in time $O(pq + qr + r \times \mathrm{ham}(T_A))$. The correctness of the procedure $MMST$ 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 $MMST(A, B, T_A)$ 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, $MMST(A, B, T_A)$ can be implemented in time $O(pq + qr + rp + r \times \mathrm{ham}(T_A))$.

Similarly, we can run $MMST(B^\top, A^\top, T_B)$ to obtain the transpose of the arithmetic matrix product of A and B. So, to obtain the lemma, we can alternate the steps of $MMST(A, B, T_A)$ and $MMST(B^\top, A^\top, T_B)$, and stop whenever any of the calls is completed. □

Theorem 3. *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 $O(pq\ell + rqk + \min\{pr\lambda(A, \ell, row) + rq\ell,\ pr\lambda(B, k, col) + pqk\})$.*

Proof. We 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. Similarly, we 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 specification of the method in [16]. 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 path of length $\ell - 1$. The Hamming cost of T_A is at most $(p-\ell)2\lambda(A, \ell, row) + (\ell - 1)q$. Similarly, we obtain a spanning tree T_B of the columns of B having the Hamming cost not exceeding $(r-k)2\lambda(B, k, col) + (k-1)q$. The theorem follows from Lemma 6 by straightforward calculations. □

5 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 [16] as a basic tool in our 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 and provides a solution within 2 of the optimum. 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 [10,17],

One could easily generalize our main results by replacing Gonzalez's algorithm with a simple and efficient approximation algorithm for the more general problem of k-center clustering with outliers [6]. 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 the approximation ratio achievable in polynomial time (e.g., 3 in [6] and 2 in [18]) not 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 [5,7].

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 could 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.

Acknowledgments. J.J. was partially supported by KAKENHI grant 24K22294.

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 (2025)
2. Alves, J., Moustafa, S., Benkner, S., Francisco, A.: Accelerating graph neural networks with a novel matrix compression format (2024). https://doi.org/10.48550/arXiv.2409.02208
3. Anand, E., van den Brand, J., McCarthy, R.: The structural complexity of matrix-vector multiplication. arXiv:2502.21240 (2025)
4. Björklund, A., Lingas, A.: Fast boolean matrix multiplication for highly clustered data. In: Dehne, F., Sack, J.-R., Tamassia, R. (eds.) WADS 2001. LNCS, vol. 2125, pp. 258–263. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44634-6_24
5. Charikar, M., Guha, S., Tardos, E., Shmoys, D.: 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 (1999)
6. Charikar, M., Khuller, S., Mount, D., 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 (2001)
7. Chen, K.: On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. SIAM J. Comput. **39**, 923–947 (2009)
8. Cohen, E., Lewis, D.D.: Approximating matrix multiplication for pattern recognition tasks. J. Algorithms **30**(2), 211–252 (1999)
9. Drineas, P., Kannan, R., Mahoney, M.: Fast monte carlo algorithms for matrices I: approximating matrix multiplication. SIAM J. Comput. **36**(1), 132–157 (2006)
10. Feder, T., Greene, D.: Optimal algorithms for approximate clustering. In: Proceedings of ACM Symposium on Theory of Computing (STOC 1988), pp. 434–444. ACM (1988)
11. Floderus, P., Jansson, J., Levcopoulos, C., Lingas, A., Sledneu, D.: 3D rectangulations and geometric matrix multiplication. Algorithmica **80**(1), 136–154 (2018)
12. Floderus, P., Kowaluk, M., Lingas, A., Lundell, E.: Detecting and counting small pattern graphs. SIAM J. Discret. Math. **29**(3), 1322–1339 (2015)

13. Gąsieniec, L., Jansson, J., Lingas, A.: Approximation algorithms for hamming clustering problems. J. Discrete Algorithms **2**(2), 289–301 (2004)
14. Gąsieniec, L., Kowaluk, M., Lingas, A.: Faster multi-witnesses for Boolean matrix multiplication. Inf. Process. Lett. **109**(4), 242–247 (2009)
15. Gąsieniec, L., Lingas, A.: An improved bound on boolean matrix multiplication for highly clustered data. In: Dehne, F., Sack, J.-R., Smid, M. (eds.) WADS 2003. LNCS, vol. 2748, pp. 329–339. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45078-8_29
16. Gonzalez, T.: Clustering to minimize the maximum intercluster distance. Theor. Comput. Scirnce **38**, 293–306 (1985)
17. Har-Peled, S., Mendel, M.: Fast construction of nets in low-dimensional metrics and their applications. SIAM J. Comput. **35**(5), 1148–1184 (2006)
18. Harris, D., Pensyl, T., Srinivasan, A., Trinh, K.: A lottery model for center-type problems with outliers. In: Proceedings of the International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2017), pp. 10:1–10:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2017)
19. Iven, M., Spencer, C.: A note on compressed sensing and the complexity of matrix multiplication. Inf. Process. Lett. **109**(10), 468–471 (2009)
20. Kurzkov, K.: Deterministic algorithms for skewed matrix products. In: Proceedings of the International Symposium on Theoretical Aspects of Computer Science (STACS 2013), pp. 466–477. Schloss Dagstuhl- Leibniz-Zentrum fuer Informatik, vol. 20 (2013)
21. Pagh, R.: Compressed matrix multiplication. ACM Trans. Comput. Theory (TOCT) **5**(3), 1–17 (2013)
22. 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 (2006)
23. 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). ACM-SIAM (2024)
24. Yuster, R., Zwick, U.: Fast sparse matrix multiplication. ACM Trans. Algorithms **1**, 2–13 (2005)