Journal of Computer and System Sciences 118 (2021) 108–118

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Pushing the Online Boolean Matrix-vector Multiplication conjecture off-line and identifying its easy cases ☆

Leszek Gąsieniec [a], Jesper Jansson [b], Christos Levcopoulos [c], Andrzej Lingas [c,*], Mia Persson [d]

[a] Department of Computer Science, University of Liverpool, Ashton Street, L69 38X, UK
[b] Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong
[c] Department of Computer Science, Lund University, 22100 Lund, Sweden
[d] Department of Computer Science and Media Technology, Malmö University, 20506 Malmö, Sweden

ARTICLE INFO

Article history:
Received 13 August 2019
Received in revised form 17 December 2020
Accepted 26 December 2020
Available online 8 January 2021

Keywords:
Boolean matrix
Product of matrix and vector
Dynamic graph problems
Online computation
Time complexity

ABSTRACT

Henzinger et al. posed the so-called Online Boolean Matrix-vector Multiplication (OMv) conjecture and showed that it implies tight hardness results for several basic dynamic or partially dynamic problems [STOC'15]. We first show that the OMv conjecture is implied by a simple *off-line* conjecture that we call the MvP conjecture. We then show that if the definition of the OMv conjecture is generalized to allow individual (i.e., it might be different for different matrices) polynomial-time preprocessing of the input matrix, then we obtain another conjecture (called the OMvP conjecture) that is in fact equivalent to our MvP conjecture. On the other hand, we demonstrate that the OMv conjecture does not hold in restricted cases where the rows of the matrix or the input vectors are clustered, and develop new efficient randomized algorithms for such cases. Finally, we present applications of our algorithms to answering graph queries.

1. Introduction

Henzinger et al. considered the following *Online Boolean Matrix-vector Multiplication* (OMv) problem in [8]. Initially, an $(n \times n)$-Boolean matrix M is given. Then, for $i = 1, \ldots, n$, in the i-th round an n-dimensional Boolean column vector v_i is given and the task is to compute the product between M and v_i before the next round. The objective is to design a (possibly randomized) algorithm that solves the OMv problem, i.e., that computes all the n products as quickly as possible. In [8], Henzinger et al. provided efficient reductions of the OMv problem to several basic dynamic or partially dynamic problems including subgraph connectivity, Pagh's problem, d-failure connectivity, decremental single-source shortest paths, and decremental transitive closure.

The *OMv conjecture* was also introduced in [8]. To express it (as well as certain other conjectures below), we shall refer to an $O(n^{3-\epsilon})$ bound or an $O(n^{2-\epsilon})$ bound, where ϵ is a positive constant, as *substantially sub-cubic* or *substantially sub-quadratic*, respectively.

☆ A preliminary version of this article has appeared in Proceedings of the 13th International Workshop on Frontiers in Algorithmics (FAW 2019), *Lecture Notes in Computer Science*, Vol. 11458, pp. 156–169, Springer Nature Switzerland AG, 2019.

* Corresponding author.
E-mail addresses: L.A.Gasieniec@liverpool.ac.uk (L. Gąsieniec), jesper.jansson@polyu.edu.hk (J. Jansson), Christos.Levcopoulos@cs.lth.se (C. Levcopoulos), Andrzej.Lingas@cs.lth.se (A. Lingas), mia.persson@mau.se (M. Persson).

https://doi.org/10.1016/j.jcss.2020.12.004
0022-0000/

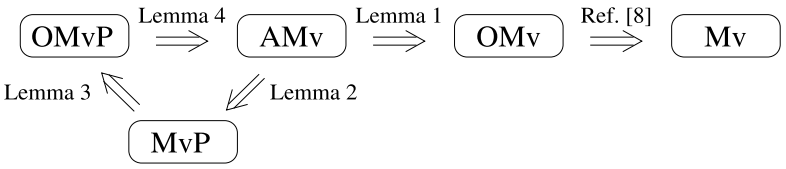

Fig. 1. The relationships between the conjectures.

Conjecture 1. ***OMv conjecture*** *(Henzinger et al. [8]) There is no randomized algorithm that solves the OMv problem in substantially sub-cubic time with an error probability of at most* $1/3$.

Their conjecture implies tight hardness results for the aforementioned dynamic or partially dynamic problems [8].
As shown in [8], the OMv conjecture also implies the following conjecture called the *Mv conjecture*.

Conjecture 2. ***Mv conjecture*** *(Henzinger et al. [8]) There is no randomized algorithm that after a polynomial-time universal preprocessing of any* $(n \times n)$*-Boolean matrix* M *computes the Boolean product of* M *with an arbitrary Boolean* n*-dimensional column vector in substantially sub-quadratic time with an error probability of at most* $1/3$.

The fastest known algorithm for the OMv problem is due to Green Larsen and Williams [7]. Their recent (non-combinatorial) randomized algorithm runs in $O(n^3/2^{\Omega(\sqrt{\log n})})$ time. Williams [11] has also shown that any $(n \times n)$-Boolean matrix can be preprocessed in $O(n^{2+\epsilon})$ time so the Boolean product of the matrix with an arbitrary n-dimensional Boolean vector can be computed in $O(n^2/\log^2 n)$ time. This implies that the Mv problem corresponding to the Mv conjecture admits an $O(n^2/\log^2 n)$-time solution. In another line of research, Chakraborty et al. [4] have recently established tight cell probe bounds for succinct Boolean matrix-vector multiplication.

We shall refer to a problem considered in a conjecture and consequently the conjecture as *off-line* or *on-line* depending on whether the whole input to the problem is given at the beginning or it is given in series of pieces, respectively. According to this convention, the OMv conjecture is on-line while the Mv conjecture is off-line. Still an off-line problem can have some on-line flavor, e.g., in the form of preprocessing.

1.1. Our contributions

In this paper, we introduce and study a number of related conjectures, whose relationships to each other and to previous work are summarized in Fig. 1. Formulating the conjectures, we shall distinguish between a *universal preprocessing* that can be applied to any Boolean square matrix and an *individual preprocessing* that can be applied only to a given Boolean square matrix. In other words, in the universal case there exists a uniform preprocessing algorithm applicable to all Boolean square matrices while in the individual case for each Boolean square matrix there exists a (not necessarily uniform) preprocessing algorithm.

We first prove that the OMv conjecture is implied by the following simple off-line conjecture that we call the MvP conjecture: For any constant $\epsilon > 0$ and any polynomial p there is an $(n \times n)$ Boolean matrix M that cannot be (individually) preprocessed in $p(n)$ time such that the Boolean product of M with an arbitrary n-dimensional column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$. To help us prove the implication, we introduce a variant called the AMv conjecture that turns out to be equivalent to the MvP conjecture. We also prove that if the OMv problem is relaxed by allowing for an individual polynomial-time preprocessing of the matrix M then the corresponding online conjecture (named OMvP) becomes equivalent to our MvP conjecture.

There is a subtle but important difference between our MvP conjecture and the Mv conjecture mentioned above: In our conjecture, the preprocessing is individual (not uniform) with respect to the matrices, while in the Mv conjecture, one considers a universal (uniform) preprocessing. It follows that the difficulty of proving/disproving the OMv conjecture lies between the two aforementioned off-line conjectures; namely, OMv is not more difficult than MvP and not easier than Mv.

We also observe that one of the techniques used above can be applied to the Combinatorial Boolean Matrix Multiplication (CBMM) conjecture. This conjecture states that there is no combinatorial (randomized) algorithm for the Boolean product of two $(n \times n)$-Boolean matrices that runs in substantially subcubic time [2,8]. We show that if the CBMM conjecture is modified to allow for a polynomial-time universal preprocessing of one of the matrices, the resulting conjecture will still be equivalent to the original CBMM conjecture.

Next, by adapting known algorithms for Boolean matrix product of matrices with clustered data [3,5,6], we obtain a combinatorial randomized algorithm for the product of an $(n \times n)$-Boolean matrix M and an arbitrary n-dimensional Boolean column vector v running in $\tilde{O}(n + ST_M)$ time after an $O(n^2)$-time preprocessing of M, where ST_M stands for the cost of a minimum spanning tree of the rows of M under the extended Hamming distance (never exceeding the Hamming distance). We present also a deterministic algorithm for the Mv problem running in $\tilde{O}(n + ST_M)$ time after an $O(n^3)$-time preprocessing of M. Consequently, we obtain a combinatorial randomized algorithm for the OMv problem running in $\tilde{O}(n(n + ST_M))$ time. We also show that OMv admits a combinatorial randomized algorithm running in $\tilde{O}(n \max\{ST(V), n^{1+o(1)}\})$ time,

where $ST(V)$ stands for the cost of a minimum spanning tree of the column vectors $v_1, ..., v_n$ under the extended Hamming distance. The time analysis of the latter algorithm relies in part on our analysis of an approximate nearest-neighbor online heuristic for the aforementioned minimum spanning tree.

The overwhelming majority of the reductions of the OMv problem to other dynamic or partially dynamic problems in [8] are unfortunately one-way reductions that do not yield applications of our algorithms for the OMv and Mv problems. Following the applications of the Mv problem given in [2,11], we provide analogous applications of our algorithms to vertex subset queries (e.g., for a given graph, such a query asks if a given subset of vertices is independent) and triangle membership queries.

1.2. Organization of the Paper

Section 2 introduces three new conjectures and shows implications and equivalences between them and the OMv conjecture. Its final subsection discusses the combinatorial Boolean matrix product. In Section 3, we develop new randomized algorithms for the OMv and Mv problems whose time complexity depends on the minimum cost of a spanning tree of the rows of the matrix or the input column vectors under the extended Hamming distance, and Section 4 presents applications of our algorithms to answering graph queries. Section 5 concludes with some final remarks.

2. Off-line conjectures

In this section, we introduce several new conjectures related to the OMv conjecture and study relationships between them and the OMv conjecture (see Fig. 1).

By the *auxiliary Boolean Matrix-vector multiplication problem* (AMv) we shall mean the problem of computing the product of a fixed $(n \times n)$-Boolean matrix M, that can be (individually) preprocessed in $O(n^{3-\epsilon})$ time for some fixed $\epsilon > 0$, with an arbitrary n-dimensional Boolean column vector v. We first state the following conjecture corresponding to the AMv problem.

Conjecture 3. ***AMv conjecture*** *There is no randomized algorithm that after an individual preprocessing of an $(n \times n)$-Boolean matrix M in substantially sub-cubic time computes the Boolean product of M with an arbitrary n-dimensional Boolean column vector v in substantially sub-quadratic time with an error probability of at most $1/3$.*

Note that the AMv conjecture is an offline conjecture.

2.1. Relationship between the AMv conjecture and the OMv conjecture

Our first result states that the AMv conjecture implies the OMv conjecture.

Lemma 1. *Let ϵ be a positive constant and let M be an $(n \times n)$-Boolean matrix. If the OMv problem for M can be solved in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$ then the matrix M can be (individually) preprocessed in $O(n^{3-\epsilon})$ time such that the Boolean product of M with an arbitrary input n-dimensional Boolean column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$. Consequently, the AMv conjecture implies the OMv conjecture.*

Proof. Construct a sequence of n-dimensional Boolean vectors $v_1,....v_n$ iteratively by picking as v_i a vector that jointly with the preceding vectors maximizes the total time of the assumed OMv solution for $v_1, ..., v_i$. Since the assumed OMv solution for the whole sequence takes $O(n^{3-\epsilon})$ time, there must be $i \in \{1, ..., n\}$ such that the product of M with v_i is computed in $O(n^{2-\epsilon})$ time after computing the products of M with the preceding vectors in the sequence. The computation of all the products clearly takes $O(n^{3-\epsilon})$ time and it has an error probability of at most $1/3$. By the definition of v_i, if we compute instead of the product of M with v_i, the product of M with an arbitrary n-dimensional input vector v, the computation will take only $O(n^{2-\epsilon})$ time after the products with the preceding vectors have been computed. Again, the computation of all the products, and hence in particular that of M with v, will have an error probability of at most $1/3$. Since the vectors $v_1...v_{i-1}$ are fixed, the computation of the products of M with the preceding vectors can be regarded as an $O(n^{3-\epsilon})$-time preprocessing. □

Unfortunately, we cannot show the reverse implication, i.e., that the OMv conjecture implies the AMv one like it implies the Mv conjecture [8]. The reason is that in the definition of the AMv problem, we do not require a universal preprocessing that could work for any matrix M of size $n \times n$; we only require the existence of an individual preprocessing for a given M.

In the next lemma, we demonstrate that allowing for an arbitrary (individual) polynomial-time preprocessing instead of the substantially subcubic one yields a new problem that is still equivalent to the AMv one. This lemma and its proof idea of dividing the matrix and the vector into appropriate submatrices and subvectors are similar to Lemma 2.3 in [8] and its proof. The proof technique can be also credited to [10].

Lemma 2. *Let δ and ϵ be positive constants. If for any $(n \times n)$-Boolean matrix M there is an $O(n^{3+\delta})$-time individual preprocessing such that the product of M with an arbitrary n-dimensional Boolean column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$ then there is a positive constant ϵ' such that after an $O(n^{3-\epsilon'})$-time individual preprocessing the product of M with such a vector v can be computed in $O(n^{2-\epsilon'})$ time with an error probability of at most $1/3$.*

Proof. Divide M into $n^{2\alpha}$ quadratic submatrices $M_{i,j}$ of size $n^{1-\alpha} \times n^{1-\alpha}$, where $i, j \in \{1, ..., n^{\alpha}\}$. Preprocess all the submatrices in $O(n^{2\alpha} \times (n^{1-\alpha})^{3+\delta})$ time. Then, the product of M with the vector v can be computed in $O(n^{2\alpha} \times (n^{1-\alpha})^{2-\epsilon} + n^{1+\alpha})$ time. The last term in the expression represents the cost of summing the results of the products of the submatrices with respective subvectors of v of length $n^{1-\alpha}$. In order to obtain an exponent of the total preprocessing time in the form $3-\epsilon'$ and the exponent of computing the product in the form $2-\epsilon'$, it is sufficient to solve the inequalities $2\alpha + (1-\alpha)(3+\delta) < 3$, $2\alpha + (1-\alpha)(2-\epsilon) < 2$ and $1+\alpha < 2$ with respect to α. Any α in the open interval $(\frac{\delta}{1+\delta}, 1)$ satisfies these inequalities.

Following the proof of Lemma 2.3 in [8], we can keep the error probability below $1/3$ by repeating the computation of each of the products of a submatrix of M with a respective vector $O(\log n)$ times, and picking the most frequent answer. In order to tackle the additional logarithmic factor in the time complexity, we can slightly decrease our ϵ'. □

We shall call the problem and the conjecture resulting from the AMv problem and the AMv conjecture by replacing an $O(n^{3-\epsilon})$ (individual) preprocessing time with a polynomial (individual) preprocessing time, the *Boolean Matrix-vector multiplication with polynomial-time (individual) preprocessing problem* and the *Boolean Matrix-vector multiplication with polynomial-time (individual) preprocessing conjecture* (MvP for short), respectively.

Conjecture 4. *MvP conjecture* *There is no randomized algorithm that after an individual polynomial-time preprocessing of an $(n \times n)$-Boolean matrix M computes the Boolean product of M with an arbitrary n-dimensional Boolean column vector v in substantially sub-quadratic time with an error probability of at most $1/3$.*

Since the MvP conjecture trivially implies the AMv conjecture, Lemmas 1 and 2 give us the following theorem.

Theorem 1. *The AMv and MvP conjectures are equivalent and they imply the OMv conjecture.*

2.2. Relaxing the OMv problem

In this subsection, we consider generalized versions of the OMv problem and the OMv conjecture that allow for individual polynomial-time preprocessing of the matrix. We shall term them the *OMvP problem* and the *OMvP conjecture*, respectively. Our goal is to establish how the OMvP conjecture is related to the MvP and AMv conjectures.

The proof of the following lemma is analogous to that of Lemma 1.

Lemma 3. *Let ϵ be a positive constant, and let M be an $(n \times n)$-Boolean matrix. If the OMvP problem for M and any positive natural number n, after a polynomial-time (individual) preprocessing of M can be solved in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$ then the matrix M can be (individually) preprocessed in polynomial time such that the Boolean product of M with an arbitrary input n-dimensional Boolean column vector v can be computed in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$. Consequently, the MvP conjecture implies the OMvP conjecture.*

Proof. First, individually preprocess M in polynomial time following the lemma assumptions on OMvP. Then, construct a sequence of n-dimensional Boolean vectors v_1,....v_n iteratively by picking as v_i a vector that jointly with the preceding vectors maximizes the total time of the assumed OMvP solution for $v_1, ..., v_i$. Since the assumed OMvP solution for the whole sequence takes $O(n^{3-\epsilon})$ time, there must be $i \in \{1, ..., n\}$ such that the product of M with v_i is computed in $O(n^{2-\epsilon})$ time after computing the products of M with the preceding vectors in the sequence. The computation of the latter products clearly takes $O(n^{3-\epsilon})$ time and it has an error probability of at most $1/3$. By the definition of v_i, if we compute instead of the product of M with v_i, the product of M with an arbitrary n-dimensional input vector v, the computation will take only $O(n^{2-\epsilon})$ time after the products with the preceding vectors are computed. Again, the computation of all the products, and hence in particular that of M with v, will have an error probability of at most $1/3$. Since the vectors $v_1 ... v_{i-1}$ are fixed, the computation of the products of M with the preceding vectors jointly with the initial individual preprocessing of M forms a polynomial-time individual preprocessing of M. □

Now we are ready to show that the OMvP conjecture implies the AMv conjecture.

Lemma 4. *Let ϵ be a positive constant, and let M be an $(n \times n)$-Boolean matrix. If the AMv problem for M can be solved in $O(n^{2-\epsilon})$ time with an error probability of at most $1/3$ after an $O(n^{3-\epsilon})$ individual preprocessing of M then the OMvP problem for the matrix M and n Boolean column vectors can be solved in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$. Consequently, the OMvP conjecture implies the AMv conjecture.*

Proof. Before computing the product of M with the first vector, perform the appropriate individual $O(n^{3-\epsilon})$ time preprocessing of M. After that the product of M with each consecutive vector can be computed in $O(n^{2-\epsilon})$ time, so the total time for n vectors becomes $O(n^{3-\epsilon})$. We can keep the error probability below $1/3$ for the whole sequence of input vectors similarly as in the proof of Lemma 2. □

According to Theorem 1, the AMv and MvP conjectures are equivalent. By applying Lemmas 3 and 4, we immediately obtain the following extension of Theorem 1. (See also the summary in Fig. 1.)

Theorem 2. *The MvP, AMv, and OMvP conjectures are equivalent.*

2.3. Combinatorial Boolean matrix product

Henzinger et al. showed in Lemma 2.3 in [8] that the possibility of universal bounded polynomial-time preprocessing in the OMv conjecture is not essential. By using a similar technique, we can also obtain a result of similar flavor for the Combinatorial Boolean Matrix Multiplication conjecture (CBMM). (For a definition of "combinatorial" in this setting, see, e.g., [2,8].) This well-known conjecture can be expressed as follows.

Conjecture 5. ***CBMM conjecture*** *There is no randomized combinatorial algorithm that computes the Boolean product of two $(n \times n)$-Boolean matrices A and B in substantially subcubic time with an error probability of at most $1/3$.*

The proof of the following lemma is similar to that of Lemma 2.

Lemma 5. *Let δ and ϵ be positive constants. If there is a combinatorial $O(n^{3+\delta})$-time universal preprocessing of any $(n \times n)$-Boolean matrix A such that the product of A with an arbitrary $(n \times n)$-Boolean matrix B can be combinatorially computed in $O(n^{3-\epsilon})$ time with an error probability of at most $1/3$ then there is an $\epsilon' > 0$ such that the product of A with such a matrix B can be combinatorially computed in $O(n^{3-\epsilon'})$ time with an error probability of at most $1/3$.*

Proof. Following the proof of Lemma 2, divide A into $n^{2\alpha}$ quadratic submatrices $A_{i,j}$ of size $n^{1-\alpha} \times n^{1-\alpha}$, where $i, j \in \{1, ..., n^{\alpha}\}$. Preprocess all the submatrices in $O(n^{2\alpha} \times (n^{1-\alpha})^{3+\delta})$ time. Next, similarly divide B into $n^{2\alpha}$ quadratic submatrices $B_{i,j}$ of size $n^{1-\alpha} \times n^{1-\alpha}$, where $i, j \in \{1, ..., n^{\alpha}\}$. Then, the product of A with the matrix B can be computed in $O(n^{3\alpha} \times (n^{1-\alpha})^{3-\epsilon} + n^{2\alpha} n^{1-\alpha} n^{\alpha})$ time. The last term in the expression represents the cost of summing the results of the $n^{3\alpha}$ products of the submatrices. In order to obtain an exponent of the total preprocessing time in the form $3 - \epsilon'$ and the exponent of computing the product in the form $3 - \epsilon'$ it is sufficient to solve the inequalities $2\alpha + (1-\alpha)(3+\delta) < 3$, $3\alpha + (1-\alpha)(3-\epsilon) < 3$ and $1 + 2\alpha < 3$ with respect to α. Any α in the open interval $(\frac{\delta}{1+\delta}, 1)$ satisfies these inequalities.

We can keep the error probability below $1/3$ analogously as in the proof of Lemma 2. □

We shall call the conjecture resulting from the CBMM conjecture by allowing a combinatorial universal polynomial-time preprocessing of one of the input matrices the *CBMM with (polynomial-time) universal preprocessing conjecture*, or CBMMUP for short. By Lemma 5 and the equality $AB = (B^t A^t)^t$ in case the second matrix is preprocessed, we obtain the following theorem.

Theorem 3. *The CBMM and CBMMUP conjectures are equivalent.*

3. Easy cases of matrices and vectors for the conjectures

In this section, we demonstrate that the OMv conjecture does not hold in restricted cases where the rows of the matrix or the input vectors are clustered, and develop some new efficient randomized algorithms for such cases.

Björklund et al. [3] proposed a method for multiplying two Boolean matrices by using a close approximation of the minimum spanning tree of the rows or columns of one of the matrices under the Hamming distance. Subsequently, the method has been generalized to include the so-called *extended Hamming distance* [6] and integer matrix multiplication [5]. In the first warming-up subsection, we present an explicit adaptation of the aforementioned generalizations to the case of the product of an $(n \times n)$-Boolean (or $0-1$) matrix M and an n-dimensional Boolean (or $0-1$) column vector v in the context of the OMv conjecture. Several results presented in the first subsection can be regarded as implicit in [5,6]. This is not the case in the second subsection handling the online scenario where the input column vectors are clustered. Here, we have to develop a novel online approach involving among other things an analysis of an approximate nearest-neighbor online heuristic for minimum spanning tree of the vectors under the extended Hamming distance. We shall use the following concepts in both subsections below.

Definition 1. *For two $0-1$ strings $s = s_1 s_2 ... s_m$ and $u = u_1 u_2 ... u_m$, their Hamming distance, i.e., the number of $k \in \{1, ..., m\}$, s.t., $s_k \neq u_k$, is denoted by $H(s, u)$. The extended Hamming distance, $EH(s, u)$, between the strings is defined by a recursive equation*

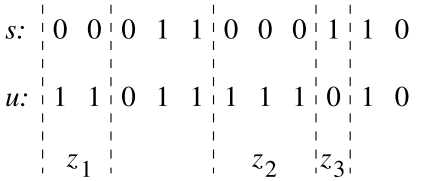

Fig. 2. An illustration of the extended Hamming distance between two strings s and u of equal length. Here, $H(s,u)=6$ while $EH(s,u)=3$. The three differentiating blocks are z_1, z_2, z_3, with $h(z_1)=1$, $h(z_2)=1$, and $h(z_3)=-1$.

$EH(s,u) = EH(s_{l+1}...s_m, u_{l+1}...u_m) + (s_1 + u_1 \mod 2)$, *where l is the maximum number such that $s_j = s_1$ and $u_j = u_1$ for $j = 1, ..., l$.*

Note that the extended Hamming distance between two strings never exceeds the Hamming one and it better reflects the real cost of turning one of the strings into the other.

Definition 2. *For two $0-1$ strings $s = s_1 s_2 ... s_m$ and $u = u_1 u_2 ... u_m$, a* differentiating block for s and u *is a maximal consecutive subsequence z of $1, 2, ...m$, such that either for each $i \in z$, $s_i = 1$ and $u_i = 0$ hold, or for each $i \in z$, $s_i = 0$ and $u_i = 1$ hold. In the first case, we denote $h(z) = -1$ and in the second case, $h(z) = 1$.*

See Fig. 2 for an example.

3.1. Small spanning tree of the rows of the matrix (warming up)

For $c \geq 1$ and a finite set S of points in a metric space, a *c-approximate minimum spanning tree for S* is a spanning tree in the complete weighted graph on S, with edge weights equal to the distances between the endpoints, whose total weight is at most c times the minimum.

Fact 4. *(Lemma 3 in [6]) For $\epsilon > 0$, a $(2+\epsilon)$-approximate minimum spanning tree for a set of n $0-1$ strings of length d under the extended Hamming metric can be computed by a Monte Carlo algorithm in time $O(dn^{1+1/(1+\epsilon/2)})$.*

By selecting $\epsilon = 2 \log n$, we obtain the following lemma.

Lemma 6. *Let M be an $(n \times n)$-Boolean matrix. An $O(\log n)$-approximation minimum spanning tree for the set of rows of M under the extended Hamming distance can be constructed by a Monte Carlo algorithm in $O(n^2)$ time.*

We shall also use the following data structure, easily obtained by computing all prefix sums:

Fact 5. *(e.g., see [5]) For a sequence of integers $a_1, a_2, \ldots, a_n$, one can construct a data structure that supports a query asking for reporting the sum $\sum_{k=i}^{j} a_k$ for $1 \leq i \leq j \leq n$ in $O(1)$ time. The construction takes $O(n)$ time.*

By using Lemma 6 and Fact 5, we obtain the following algorithm (Algorithm 1) for computing the arithmetic product of the input Boolean matrix M and Boolean vector v interpreted as $0-1$ ones. Observe that the aforementioned arithmetic product immediately yields the corresponding Boolean one.

Definition 3. *For an $(n \times n)$-Boolean matrix A, let ST_A stand for the minimum cost of a spanning tree of $row_i(A)$, $i \in \{1, ..., n\}$, under the extended Hamming distance.*

Lemma 7. *Algorithm 1 runs in $\tilde{O}(n^2 + ST_M)$ time with high probability. If Steps 1, 2, 3 are treated as a preprocessing of the matrix M then it runs in $\tilde{O}(n + ST_M)$ time with high probability after an $O(n^2)$-time preprocessing.*

Proof. The approximate minimum spanning tree T in Step 1 can be constructed by a Monte Carlo algorithm in $O(n^2)$ time by Lemma 6. Its traversal can be found in $O(n)$ time. Since the length of the traversal is linear in n, Step 2 can be easily implemented in $O(n^2)$ time. Step 3 takes $O(n)$ time by Fact 5. Finally, based on Step 2, Step 4 (b)-iii takes $\tilde{O}(1 + EH(row_i(M), row_l(M)))$ time. Let U stand for the set of directed edges forming the traversal of the spanning tree T. It follows that Step 4 (b) can be implemented in $\tilde{O}(n + \sum_{(i,l)\in U} EH(row_i(M), row_l(M)))$ time, i.e., in $\tilde{O}(n + ST_M)$ time by Lemma 6. Consequently, Step 4 takes $\tilde{O}(n + ST_M)$ time. □

By Lemma 7, we obtain:

Algorithm 1

Input: An $(n \times n)$-Boolean matrix M and an n-dimensional Boolean column vector v.

Output: The arithmetic product $c = (c_1, ..., c_n)$ of M and v interpreted as a $0-1$ matrix and a $0-1$ vector, respectively.

1. Find an $O(\log n)$-approximate spanning tree T for the rows $row_i(M)$, $i = 1, \ldots, n$, of M under the extended Hamming distance and a traversal (i.e., a not necessarily simple path visiting all vertices) of T.
2. For each pair $(row_i(M), row_l(M))$, where the latter row follows the former in the traversal, find a set S of the differentiating blocks for $row_i(M)$ and $row_l(M)$ as well as the differences $h(s)$ (1 or -1) between the common value of each entry in $M_{l,\min s}, \ldots, M_{l,\max s}$ and the common value of each entry in $M_{i,\min s}, \ldots, M_{i,\max s}$ for each $s \in S$.
3. Initialize a data structure D for counting partial sums of the values of coordinates on continuous fragments of the vector v.
4. Iterate the following steps:
   (a) Compute c_q where q is the index of the row from which the traversal of T starts.
   (b) While following the traversal of T, iterate the following steps:
      i. Set i, l to the indices of the previously traversed row and the currently traversed row, respectively.
      ii. Set c_l to c_i.
      iii. For each differentiating block s for $row_i(M)$ and $row_l(M)$, compute $\sum_{k \in s} v_k$ using D and set c_l to $c_l + h(s) \sum_{k \in s} v_k$.
5. Output the vector $(c_1, c_2, ..., c_n)$

Theorem 6. *The Boolean product c of an $(n \times n)$-Boolean matrix M and an n-dimensional Boolean column vector v can be computed by a randomized algorithm in $\tilde{O}(n + ST_M)$ time with high probability after $O(n^2)$-time preprocessing.*

Proof. The correctness of Algorithm 1 follows from the observation that a differentiating block s for $row_i(M)$ and $row_l(M)$ yields the difference $h(s) \sum_{k \in s} v_k$ between c_l and c_i just on the fragment corresponding to $M_{i,\min s}, \ldots, M_{i,\max s}$ and $M_{l,\min s}, \ldots, M_{l,\max}$, respectively. Lemma 7 yields the upper bounds in terms of ST_M. □

We can compute the exact minimum spanning tree of the rows of the matrix M under the extended Hamming distance by computing extended Hamming distances between all pairs of rows of M in $O(n^3)$ time. Hence by replacing the randomized approximate computation of the minimum spanning tree with the deterministic exact one in Algorithm 1, we obtain the following theorem.

Theorem 7. *The Boolean product c of an $(n \times n)$-Boolean matrix M and an n-dimensional Boolean column vector v can be computed deterministically in $\tilde{O}(n + ST_M)$ time after $O(n^3)$-time preprocessing.*

Corollary 1. *The OMv problem for an $(n \times n)$-Boolean matrix can be solved by a randomized algorithm in $\tilde{O}(n(n + ST_M))$ time with high probability while the Mv problem can be solved by a randomized algorithm in $\tilde{O}(n + ST_M)$ time with high probability after $O(n^2)$-time preprocessing or it can be solved deterministically in $\tilde{O}(n + ST_M)$ time after $O(n^3)$-time preprocessing.*

3.2. Small spanning tree of the input vectors

In this subsection, we assume an online scenario where besides the Boolean matrix there is given a sequence of n-dimensional Boolean vectors received one at a time. In order to specify and analyze our algorithm (Algorithm 2), we need the following concepts and facts related to them.

Definition 4. *For a metric space P and a point $q \in P$, a* c-approximate nearest neighbor *of q in P is a point $p \in P$ different from q such that for all $p' \in P$, $p' \neq q$, $dist(p, q) \leq c \times dist(p', q)$. The ϵ-approximate nearest neighbor search problem (ϵ-NNS) in P is to find for a query point $q \in P$ a $(1 + \epsilon)$-approximate nearest neighbor of q in P.*

Fact 8. *(See the third row in Table 4.3.1.1 in [1]) For $\epsilon > 0$, there is a Monte Carlo algorithm for the dynamic (i.e., supporting point insertions and deletions) ϵ-NNS in $\{0, 1\}^d$ under the Hamming metric which requires $O(d\ell^{\frac{1}{1+2\epsilon}+o(1)})$ query time and $O(d\ell^{\frac{1}{1+2\epsilon}+o(1)})$ update time, where ℓ is the maximum number of stored vectors in $\{0, 1\}^d$.*

Fact 9. *[6] There is a simple, linear-time, transformation of any $0-1$ string w into the string $t(w)$ such that for any two $0-1$ strings s and u, $EH(s, u) = \lceil \frac{H(t(s), t(u))}{2} \rceil$.*

Note that the size of $t(w)$ is linear in that of w by the linear-time complexity of the transformation. By combining Facts 8 and 9, we obtain the following corollary.

Corollary 2. *There is a randomized Monte Carlo algorithm for a dynamic $O(\log \ell)$-NNS in $\{0,1\}^d$ under the extended Hamming metric which requires $O(d\ell^{o(1)})$ query time and $O(d\ell^{o(1)})$ update time.*

Our online algorithm is as follows.

Algorithm 2

Input: Given a priori an $(n \times n)$-Boolean matrix M and an online sequence of n-dimensional Boolean vectors $v_1, v_2, ..., v_\ell$, received one at a time.

Output: For $i = 1, ..., \ell$, the arithmetic product $c^i = (c^i_1, ..., c^i_n) = Mv_i$ of M and v_i, treated as a 0-1 matrix and a 0-1 column vector, is output before receiving v_{i+1}.

1. For $j = 2, \ldots, n$, initialize a data structure D_j that for any interval $u \subseteq \{1, ..., n\}$ reports $\sum_{k \in u} M[j,k]$ using Fact 5.
2. Receive the first vector v_1 and compute the arithmetic product $c^1 = (c^1_1, ..., c^1_n)$ of M with v_1 by the definition of matrix-vector multiplication.
3. For $i = 2, \ldots, \ell$, receive the i-th vector $v_i = (v^i_i, ..., v^i_n)$ and iterate the following steps:
   (a) Find an $O(\log \ell)$-approximate nearest neighbor v_m of v_i in the set $\{v_1, ..., v_{i-1}\}$.
   (b) Determine the differentiating blocks s and the differences $h(s)$ for v_m and v_i.
   (c) For $j = 1, \ldots, n$ iterate the following steps.
      i. Set c^i_j to c^m_j.
      ii. For each differentiating block s of v_m and v_i iterate the following steps.
         A. Compute $\sum_{k \in s} M[j,k]$ using D_j.
         B. Set c^i_j to $c^i_j + h(s) \sum_{k \in s} M[j,k]$.
   (d) Output $c^i = (c^i_1, ..., c^i_n)$

The following lemmas analyze the time complexity of Algorithm 2. The first lemma is a direct consequence of Corollary 2.

Lemma 8. *There is a randomized Monte Carlo algorithm for a dynamic $O(\log \ell)$-NNS in $\{0,1\}^d$ under the extended Hamming metric such that:*

- *The insertions of the vectors v_1 through v_ℓ in Algorithm 2 can be implemented in $O(n\ell^{1+o(1)})$ total time.*
- *The $O(\log \ell)$-approximate nearest neighbors of v_i, $i = 2, ..., \ell$, in $\{v_1, ..., v_{i-1}\}$, in Step 3 (a) of Algorithm 2 can be found with high probability in $O(n\ell^{1+o(1)})$ total time.*

Proof. By Corollary 2, the $\ell - 1$ updates and $\ell - 2$ $O(\log \ell)$-approximate nearest neighbor queries take $O(n\ell^{1+o(1)})$ total time. □

Lemma 9. *Algorithm 2 can be implemented in time $\tilde{O}(n(\ell^{1+o(1)} + \sum_{i=2}^{\ell} \min\{dist(v_i, v_j) | j < i\}))$.*

Proof. Step 1 can be implemented in $O(n^2)$ time by Fact 5 while Step 2 can be trivially done in $O(n^2)$ time by the definition. Step 3 (a) takes $t(i)$ time, where $t(i)$ is the time taken by finding an $O(\log \ell)$-approximate neighbor of v_i in $\{v_1, v_2, ..., v_{i-1}\}$ and inserting v_i in the dynamic data structure supporting the $O(\log \ell)$-approximate neighbor queries, with high probability. The differentiating blocks s and the differences $h(s)$ for v_m and v_i can be easily determined in $O(n)$ time in Step 3 (b). Since the number of the aforementioned blocks is within a polylogarithmic factor of $\min\{dist(v_i, v_j) | j < i\}$, the whole update of c^m to c^i in Step 3 (c) takes $\tilde{O}(n(1 + \min\{dist(v_i, v_j) | j < i\}))$ time. Finally, it follows from Lemma 8 that $\sum_{i=2}^{\ell} t(i) = O(n\ell^{1+o(1)})$. □

To pursue our time analysis of Algorithm 2, we need to show $\sum_{i=2}^{\ell} \min\{dist(v_i, v_j) | j < i\} = \tilde{O}(ST(V))$, where $ST(V)$ is the minimum cost of the spanning tree of the vectors in $V = \{v_1, v_2, ..., v_\ell\}$ under the extended Hamming distance. For this purpose, we shall analyze the following simple heuristic for an online variant of the minimum spanning tree problem (MST).

Approximate Nearest-Neighbor Heuristic for MST

Input: an online sequence V of points (in particular, vectors) $v_1, v_2, \ldots$ received one at a time.

Output: a sequence of spanning trees T_i of the points v_1 through v_i constructed before receiving v_{i+1} for all $i > 1$.

1. Set T_1 to the singleton tree $\{v_1\}$;
2. **for** each received point v_i, $i > 1$ **do**
   (a) find an $f(i)$-approximate nearest neighbor u of v_i in the set of points received so far;
   (b) let T_i be the spanning tree T_{i-1} built for points received before v_i and expand it by $\{u, v_i\}$;
   (c) output T_i.

Theorem 10. *Assume that the function f is not decreasing and the input points to the approximate nearest-neighbor heuristic for MST are drawn from a metric space. The spanning tree constructed by the heuristic for the first t points has cost not exceeding $\lceil \log_2 t \rceil f(t)$ times the minimum.*

Proof. Assume first that t is a power of two. Let $V = \{v_1, ..., v_t\}$ be the sequence of t points received, where v_i is the i-th point received.

Consider a minimum cost perfect matching P of V (recall that V is even). For each edge $\{v_i, v_j\}$ in P, where $i < j$, the cost of connecting v_j to the current spanning tree T_{j-1} of v_1 through v_{j-1} does not exceed $f(t) \times dist(v_i, v_j)$. Thus, for $t/2$ points v_l in V, the accumulated cost of connecting them to the current spanning tree T_{l-1} does not exceed the total cost of P times $f(t)$. Note that the total cost of P is not greater than half the minimum cost $TSP(V)$ of the traveling salesperson tour of V. Simply, the tour can be decomposed into two perfect matchings of V.

In order to estimate from above the cost of connecting the remaining $t/2$ points to the current spanning trees, we iterate our argument.

Thus, let V_1 denote the remaining set of points and let P_1 be their minimum-cost perfect matching. We can again estimate the cost of connecting half of the $t/2$ points in V_1 to the current spanning trees by the cost of P_1 times $f(t)$. On the other hand, we can estimate the cost of P_1 by $\frac{1}{2}TSP(V_1) \leq \frac{1}{2}TSP(V)$. We handle analogously the remaining $t/4$ points and so on. After $\log_2 t$ iterations, we are left with the first point, and can estimate the total cost of connecting all other points to the current spanning trees by $\log_2 t \times f(t)TSP(V)/2$. On the other hand, by the doubling MST heuristic, we know that $TSP(V)$ is at most twice the cost $ST(V)$ of minimum-cost spanning tree of V. We conclude that the cost of the spanning tree of V constructed by the approximate nearest-neighbor heuristic does not exceed $\log_2 t \times f(t)ST(V)$.

If t is not a power of two, we have to consider minimum-cost maximum cardinality matchings instead of minimum-cost perfect matchings. Let $t' = 2^{\lceil \log_2 t \rceil}$. Observe that the number of the remaining points after each iteration when we start with a sequence S of t points will be not greater than that when we start with a sequence S' of t' points, where S' is an extension of the sequence S. This completes the proof of the $\lceil \log_2 t \rceil f(t)ST(V)$ upper bound. □

In the special case when $f(\,) \equiv 1$, our online heuristic for MST in a way coincides with the greedy one for incremental minimum Steiner tree from [9], which on weighted graphs satisfying the triangle inequality could fairly easily be adapted to consider received vertices only. Hence, in this case a logarithmic upper bound on approximation factor could be also deduced from Theorem 3.2 in [9]. By combining Lemma 9 with Theorem 10, we obtain our main result in this section.

Theorem 11. *Let M be an $(n \times n)$-Boolean matrix. For an online sequence V of n-dimensional Boolean vectors $v_1, v_2, ..., v_\ell$ received one at a time, the Boolean products Mv_i of M and v_i can be computed before receiving v_{i+1} in total time $\tilde{O}(n(\ell^{1+o(1)} + ST(V)))$ with high probability by a randomized algorithm, where $ST(V)$ is the minimum cost of the spanning tree of the vectors in V under the extended Hamming distance.*

Proof. The correctness of Algorithm 2 follows from the observation that a differentiating block s for v_m and v_i yields the difference $h(s)\sum_{k\in s} M[j,k]$ between c_j^m and c_j^u just on the fragments $v^m_{\min s}, \ldots, v^m_{i,\max s}$ and $v^i_{\min s}, \ldots, v^i_{\max s}$, respectively. By Theorem 10, we have $\sum_{i=2}^{\ell} \min\{dist(v_i, v_j) | j < i\} = \tilde{O}(ST(V))$. Now it is sufficient to plug the latter estimation in the upper time bound of Lemma 9 to complete the proof. □

4. Applications to graph queries

Suppose that we are given a graph $G = (V, E)$ on n vertices and a subset S of V. In [11] Williams observed that the questions if S is a dominating set, an independent set, or a vertex cover in G, can be easily answered by computing the Boolean product of the adjacency matrix of G with appropriate Boolean vectors. Hence, he could conclude (Corollary 3.1 in [11]) that these questions can be answered in $O(n^2/(\epsilon \log n)^2)$ time after an $O(n^{2+\epsilon})$ preprocessing of G by using his method of multiplying an $(n \times n)$-Boolean matrix with an n-dimensional column vector in $O(n^2/(\epsilon \log n)^2)$ time after an

$O(n^{2+\epsilon})$-time preprocessing of the matrix. By plugging in our method of Boolean matrix-vector multiplication (Theorem 6) instead, we obtain the following result.

Corollary 3. *A graph G on n vertices can be preprocessed in $O(n^2)$ time such that one can determine if a given subset of vertices in G is a dominating set, an independent set, or a vertex cover of G in $\tilde{O}(n + ST_G)$ time with high probability, where ST_G is the minimum cost of a spanning tree of the rows of the adjacency matrix of G under the extended Hamming distance. Using the same preprocessing, one can determine if a query vertex belongs to a triangle in G in $\tilde{O}(n + ST_G)$ time with high probability.*

Proof. A subset S of vertices in G can be represented by an n-dimensional Boolean column vector w with 1 on the j-th coordinate iff the j-th vertex belongs to S. Then, as Williams observed in [11], S is independent in G iff the vector u resulting from multiplying the adjacency matrix of G with w has zeros on the coordinates corresponding to the vertices in S. Next, S is a dominating set of G iff each vertex in $V \setminus S$ has a neighbor in S, i.e., iff u has ones on the coordinates corresponding to vertices in $V \setminus S$. Furthermore, S is a vertex cover of G iff $V \setminus S$ is an independent set of G, i.e., iff the vector resulting from multiplying the adjacency matrix of G with the complement of w has zeros on the coordinates corresponding to the vertices in $V \setminus S$. Finally, Williams also observed that the problem of determining if a query vertex v belongs to a triangle in a given graph reduces to checking the set of neighbors of v for independence (see Corollary 3.2 in [11]).

Hence, it is sufficient to plug in our solution to matrix-vector multiplication given in Theorem 6 to obtain the corollary. The preprocessing of G consists only of the construction of its adjacency matrix and a logarithmic approximation of ST_G in $O(n^2)$ time. Note also that the extended Hamming distance between two 0-1 strings is equal to the extended Hamming distance between the complements of these two strings. Thus, the upper bound in terms of ST_G is also valid in case of vertex cover. □

To obtain corresponding applications of the results from subsection 3.2, we need to consider the online versions of the graph subset queries. Thus, we are given a graph G on n vertices and an online sequence of subsets S_1,...,S_ℓ of vertices in G. The task is to preprocess G first and then to determine for $i = 1, ..., \ell$, if S_i is a dominating set, an independent set, or a vertex cover of G, respectively, before S_{i+1} has been received. Analogously, we obtain the following online version of Corollary 3 by plugging in Theorem 11 instead of Theorem 6.

Corollary 4. *A graph G on n vertices can be preprocessed in $O(n^2)$ time such that for an online sequence S of subsets S_1,...,S_ℓ of vertices in G, for $i = 1, ..., \ell$, one can determine if S_i is a dominating set, an independent set, or a vertex cover of G before receiving S_{i+1} (in $i < \ell$ case) in $\tilde{O}(n(\ell^{1+o(1)} + ST_S))$ total time with high probability, where ST_S is the minimum cost of a spanning tree of the characteristic vectors representing the subsets in S under the extended Hamming distance. Using the same preprocessing, for an online sequence $v_1, ..., v_\ell$ of query vertices, for $i = 1, ..., \ell$, one can determine if v_i belongs to a triangle in G before receiving v_{i+1} (in case $i < \ell$) in $\tilde{O}(n(\ell^{1+o(1)} + ST_\ell))$ total time with high probability, where ST_ℓ is the minimum cost of a spanning tree of the ℓ characteristic vectors representing the sets of neighbors of the vertices under the extended Hamming distance.*

5. Final remarks

Our results in Section 3 imply that to prove the OMv, AMv, and MvP conjectures, it suffices to consider $(n \times n)$-Boolean matrices where ST_M is almost quadratic in n.

Interestingly enough, our approximate nearest-neighbor heuristic for MST combined with the standard MST doubling and shortcutting techniques immediately yields a corresponding online heuristic for TSP in metric spaces. By Theorem 10, it provides TSP tours TSP_s of length at most $2\lceil \log_2 s \rceil f(s)$ times larger than the optimum, where s is the number of input vectors and $f(s)$ is an upper bound on the approximation factor in the approximate nearest neighbor subroutine. The resulting TSP heuristic for $i = 2, ...$ simply finds an $f(i)$-nearest neighbor u of the new vector v_i and replaces the edge between u and its predecessor w by the path $\{w, v_i\}$, $\{v_i, u\}$ in TSP_{i-1} in order to obtain TSP_i.

CRediT authorship contribution statement

All the co-authors have contributed to this paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Thanks go to the anonymous reviewers for their valuable comments. The authors were supported in part by Swedish Research Council grant 621-2017-03750. JJ was also supported by PolyU Fund 1-ZE8L.

References

[1] A. Andoni, P. Indyk, Nearest neighbors in high-dimensional spaces, in: J.E. Goodman, J. O'Rourke, C.D. Toth (Eds.), 3rd edition, Handbook of Discrete and Computational Geometry, CRC Press, Boca Raton, FL, 2017, 43rd chapter.
[2] N. Bansal, R. Williams, Regularity lemmas and combinatorial algorithms, Theory Comput. 8 (1) (2012) 69–94.
[3] A. Björklund, A. Lingas, Fast Boolean matrix multiplication for highly clustered data, in: Proc. of WADS 2001, in: LNCS, vol. 2125, 2001, pp. 258–263.
[4] D. Chakraborty, L. Kamma, K. Green Larsen, Tight cell probe bounds for succinct Boolean matrix-vector multiplication, in: Proc. of STOC 2018, 2018, pp. 1297–1306.
[5] P. Floderus, J. Jansson, C. Levcopoulos, A. Lingas, D. Sledneu, 3D rectangulations and geometric matrix multiplication, Algorithmica 80 (1) (2018) 136–154.
[6] L. Gąsieniec, A. Lingas, An improved bound on Boolean matrix multiplication for highly clustered data, in: Proc. of WADS 2003, in: LNCS, vol. 2748, 2003, pp. 329–339.
[7] K. Green Larsen, R.R. Williams, Faster online matrix-vector multiplication, in: Proc. of SODA 2017, 2017, pp. 2182–2189.
[8] M. Henzinger, S. Krinninger, D. Nanongkai, T. Saranurak, Unifying and strengthening hardness for dynamic problems via the online matrix-vector multiplication conjecture, in: Proc. of STOC 2015, 2015, pp. 21–30.
[9] M. Imase, B.M. Waxman, Dynamic Steiner tree problem, SIAM J. Discrete Math. 4 (3) (1991) 369–384.
[10] V. Vassilevska Williams, R. Williams, Subcubic equivalences between path, matrix, and triangle problems, J. ACM 65 (5) (September 2018) 27:1–27:38 (preliminary version FOCS 2010).
[11] R. Williams, Matrix-vector multiplication in sub-quadratic time (some preprocessing required), in: Proc. of SODA 2007, 2007, pp. 995–2001.