The bounded membership problem of the monoid $SL_2(N)$

Jin-Yi Cai* Zicheng Liu†

Abstract

$SL_2(N)$ is the set of all 2×2 matrices with nonnegative integer entries and determinant 1. The bounded membership problem BMN of $SL_2(N)$ is that given a subset S of $SL_2(N)$, a matrix $A \in SL_2(N)$, and an integer n, whether A can be represented as a product of at most n matrices (repetitions are allowed) in S. We prove that BMN is NP-complete, but its randomized version under natural distribution is solvable in average polynomial time. Furthermore, it is proved that if the number of elements of S is a constant, then BMN is polynomial time computable.

1 Introduction

In 1990, Gurevich [5] proved the first algebraic average case complete problem, the matrix decomposition problem. He also raised the question whether a simpler version of the problem, which is in some sense a more natural problem, is complete in the average case as well, or is solvable in average polynomial time. This is the motivation for our current paper. [1] Here, we consider the simple version of another similar average case complete problem, the matrix correspondence problem [5]. The two sets of problems are closely related—the simple version of the matrix decomposition problem becomes the bounded membership problem of $SL_2(Z)$, the simple version of the matrix correspondence problem becomes the bounded membership problem of $SL_2(N)$.

We first give the definitions of matrix decomposition problem and matrix correspondence problem.

$SL_2(Z)$ is the multiplicative group of two-by-two integer matrices of determinant 1. $SL_2(N)$ is the subset of $SL_2(Z)$ that consists of only positive matrices (all the entries are nonnegative integers). It is easy to see that $SL_2(N)$ is a monoid under matrix multiplication. In the following, for any matrix A we use $|A|$ to denote the logarithm base 2 of the maximum absolute value of all its entries.

Definition 1.1 *Matrix decomposition problem* MD is the following decision problem:

*Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Email: `cai@cs.buffalo.edu`. Supported in part by NSF grant CCR 9057486 and CCR 9319093, and an Alfred T. Sloan Fellowship.

†Department of Computer Science, Princeton University, Princeton, NJ 08544. Supported in part by NSF grant CCR 9057486.

[1] The question raised by Gurevich was recently solved by Cai et. al. [2]

Instance: A matrix $X \in SL_2(Z)$, a finite set S of pairs of matrices in $SL_2(Z)$, say $S = \{(A_1, B_1), (A_2, B_2), ..., (A_m, B_m)\}$, and a natural number n.

Question: Are there indices $i_1, i_2, ..., i_t$, where $t \leq n$, such that

$$X = A_{i_1} A_{i_2} ... A_{i_t} B_{i_t} ... B_{i_2} B_{i_1}?$$

Size: $|X| + \sum_{i=1}^{m} (|A_i| + |B_i|) + n$.

Definition 1.2 *Matrix correspondence problem* MC is the following decision problem:

Instance: A matrix $X \in SL_2(N)$, a finite set S of pairs of matrices in $SL_2(N)$, say $S = \{(A_1, B_1), (A_2, B_2), ..., (A_m, B_m)\}$, and a natural number n.

Question: Are there indices $i_1, i_2, ..., i_t$, where $t \leq n$, such that

$$X B_{i_1} B_{i_2} ... B_{i_t} = A_{i_1} A_{i_2} ... A_{i_t}?$$

Size: $|X| + \sum_{i=1}^{m} (|A_i| + |B_i|) + n$.

For any given integer σ, we use MD(σ) to denote the problem MD with the restriction that the cardinality of S in the definition of MD is bounded by σ. Similarly, we use MC(σ) to denote the problem MC with the restriction that the cardinality of S in the definition of MC is bounded by σ. For large σ, MC(σ) and MD(σ) under natural distributions have been proved to be complete for the average case [5][1].

Suppose we let all B_i be the unit matrix I in the above, then we obtain a natural variant of the problems which we call the bounded membership problem for the matrix group $SL_2(Z)$ and the bounded membership problem for the matrix monoid $SL_2(N)$. The unbounded version of the problem is known as the Magnus problem for the unimodular group and monoid [10] [9]. In terms of complexity of the problems we expect that they become somewhat easier by letting all $B_i = I$, and we call them simple versions of the MD and MC problems. More precisely, we define:

Definition 1.3 *Bounded membership problem* for $SL_2(Z)$, denoted by BMZ, is the following decision problem:

Instance: A matrix $X \in SL_2(Z)$, a subset S of $SL_2(Z)$, and a natural number n.

Question: Are there matrices $A_1, A_2, ..., A_t$ in S, (repetitions are allowed), $t \leq n$, such that $X = A_1 A_2 ... A_t$?

Size: $|X| + \sum_{A \in S} |A| + n$.

Definition 1.4 *Bounded membership problem* for $SL_2(N)$, denoted by BMN, is the following decision problem:

Instance: A matrix $X \in SL_2(N)$, a subset S of $SL_2(N)$, and a natural number n.

Question: Are there matrices $A_1, A_2, ..., A_t$ in S, (repetitions are allowed), $t \leq n$, such that $X = A_1 A_2 ... A_t$?

Size: $|X| + \sum_{A \in S} |A| + n$.

For any integer σ, BMZ(σ)(respectively, BMN(σ)) is the problem BMZ (respectively, BMN) with the restriction that the cardinality of S is bounded by σ. In this paper we will also discuss search problems associated with these decision problems.

Blass and Gurevich [1] proved that BMZ is NP-complete. The motivation for the current paper was to resolve the question raised by Blass and Gurevich whether the randomized version of BMZ under natural distribution is complete for the average case or solvable in average polynomial time. In this paper, we study the problem BMN. We prove that BMN is NP-complete but its randomized version under natural distribution is solvable in average polynomial time. Furthermore, we prove that BMN(σ) is solvable in polynomial time for any constant σ, which shows exactly where the hard part of BMN is.

Recently, while the current paper was under review by referees, the original problem raised by Blass and Gurevich with respect to BMZ was resolved by Cai et. al. [2]. They showed that indeed as conjectured [1] BMZ under natural distribution is solvable in average polynomial time.

2 Notations

Throughout this paper, unless otherwise specified, the size $|s|$ of a binary string s is its length. The size $|x|$ of an integer x is the length of its binary representation. The size $|S|$ of a set S is the sum of the sizes of all its elements. The size of a matrix in $SL_2(N)$ is the maximum size of all its elements. For any set S, we use $\#S$ to denote the number of elements in S. If x and y are both strings, xy is the concatenation of the two strings. If x and y are both integers or matrices, xy denotes the product of them.

The following definitions on average case complexity theory are first given by Levin [8]. Our presentation follows that given by Gurevich [6] [5].

Definition 2.1 A *domain* X consists of:

1. An underlying set, called the universe of X, often denoted by X as well, comprising strings in some alphabet Σ_X;
2. A size function, assigning to each $x \in X$ a positive integer $|x| = |x|_X$, called the size of x; and
3. A probability distribution P_X on X.

Definition 2.2 Let f be a function from a domain X to nonnegative reals. f is *polynomial on average*, abbreviated AP, if there is an $\epsilon > 0$, such that $\sum_{|x|>0} \frac{f^\epsilon(x)}{|x|} P_X(x) < \infty$.

Lemma 2.3 [1] The collection of AP functions on a domain is closed under addition and multiplication.

Definition 2.4 *A randomized decision problem* is a pair (D, μ) where D is a decision problem and μ is a probability distribution on the instances of D.

Definition 2.5 A randomized decision problem (D, μ) is *solvable in average polynomial time* if there exists an algorithm for D with running time $T(x)$ on instance x such that $T(x)$ is polynomial on average with respect to domain $(\Sigma_D^*, |\cdot|_D, \mu)$ where Σ_D is the alphabet of D, $|\cdot|_D$ is the size function.

For the definitions of randomized reductions and completeness for the average case, the reader is referred to [6].

In the following, we define the standard uniform probability distribution of the instances of BMN. For a discussion of various definitions of natural distributions, the reader is referred to [1].

We use PI to denote the customary distribution on the set of positive integers: PI(n)=$1/(n(n+1))$. We use PM to denote the uniform distribution on $SL_2(N)$: PM(X)=PI($|X|$)$\frac{1}{\psi(|X|)}$, where $\psi(n)$ is the number of elements in $SL_2(N)$ with size n.

We will use ν to denote the natural distribution on the instances (X, S, n) of BMN: $\nu(X, S, n) = \mathrm{PI}(n) * \mathrm{PM}(X) * \mathrm{PI}(\#S) * \Pi_{A \in S} \mathrm{PM}(A)$.

3 BMN is NP-complete

First we prove that BMN is NP-complete by adapting a proof in [1] for BMZ.

Definition 3.1 *Simple Integer Sum* is the following decision problem :

Instance: A positive integer K, a finite set S of positive integers, and an integer n.

Question: Can K be represented as $\sum_{i=1}^{m} b_i$ where $m \leq n$, and numbers b_i are (not necessarily distinct) elements of S ?

Size: $|K| + \sum_{x \in S} |x| + n$.

Lemma 3.2 Simple Integer Sum is NP-complete.

Proof We reduce X3C to Simple Integer Sum, where X3C is the problem of *Exact Cover by 3-Sets* [4]:

Instance: A positive integer q and a collection C of 3-element subsets of the set $X = \{1, 2, 3, ..., 3q\}$.

Question: Is there an exact cover $C' \subseteq C$ for X (so that each element of X belongs to exactly one member of C')?

Size: $q +$ the size of C.

Given an instance (q, C) of X3C, let $l = |q|$, the binary length of the integer q, and let B be the collection of all binary strings of length $3ql$. Consider a string in B as a sequence of $3q$ blocks of length l. For each i, $1 \le i \le 3q$, let a_i be the integer represented by a string in B with exactly one 1 which appears at the rightmost position of the i-th block counted from right to left.

Claim 3.3 [1] If $\sum_{i=1}^{3q} \alpha_i a_i = \sum_{i=1}^{3q} \beta_i a_i$ and $0 \le \alpha_i, \beta_i \le q$ for each i, then $\alpha_i = \beta_i$ for each i.

We define the reduction f from X3C to Simple Integer Sum as follows:

$$f(q, C) = (K, \{y(T) : T \in C\}, q),$$

where $K = \sum_{i=1}^{3q} a_i$ and $y(T) = \sum_{i \in T} a_i$. If $C' \subseteq C$ is an exact cover for X , then $\#C'$ is q and K is represented as the sum of the q numbers $y(T)$ where $T \in C'$. Conversely, suppose $K = \sum_{j=1}^{m} y(T_j)$ where $m \le q$ and each $T_j \in C$. For each $T \in C$, let $\chi_T(i)$ be 1 if $i \in T$, and 0 otherwise. Then

$$K = \sum_{j=1}^{m} y(T_j) = \sum_{j=1}^{m} \sum_{i \in T_j} a_i = \sum_{j=1}^{m} \sum_{i=1}^{3q} \chi_{T_j}(i) a_i = \sum_{i=1}^{3q} \left(\sum_{j=1}^{m} \chi_{T_j}(i) \right) a_i.$$

Since $\sum_{j=1}^{m} \chi_{T_j}(i) \le m \le q$ for each i, by Claim 3.3 we have $\sum_{j=1}^{m} \chi_{T_j}(i) = 1$ for each i. Thus the set of all the T_j's forms an exact cover for X. □

Theorem 3.4 BMN is NP-complete.

Proof We reduce Simple Integer Sum to BMN. For each integer y, let $g(y) = \begin{pmatrix} 1 & y \\ 0 & 1 \end{pmatrix}$. If $g(y) = Y$ and $g(z) = Z$ then $g(y + z) = YZ$. It is easy to see that the following reduction reduces Simple Integer Sum to BMN:

$$f(K, S, n) = (g(K), \{g(y) | y \in S\}, n)$$

□

4 BMN is solvable in average polynomial time

Now we prove that, while NP-complete, BMN is solvable in average polynomial time.

4.1 The monoid BS

We need to introduce the monoid BS which is the set of all binary strings under the operation of concatenation. BS is a free monoid generated by the letters 0 and 1.

Lemma 4.1 [1] The monoid $SL_2(N)$ is isomorphic to the monoid BS.

Let $A_0 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, and $B_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$. There are exactly two isomorphisms of $SL_2(N)$ onto BS. One of them takes A_0 to 0 and B_0 to 1 while the other takes A_0 to 1 and B_0 to 0. Let I be the isomorphism that takes A_0 to 0. We can transform BMN to a bounded membership problem for BS using this isomorphism. More precisely, we define

Definition 4.2 BMS is the decision problem :

Instance: A binary string x, a finite set S of binary strings, say $S = \{w_1, w_2, ..., w_m\}$, and a natural number n.

Question: Are there indices $i_1, i_2, ..., i_t$, where $t \leq n$, such that $x = w_{i_1} w_{i_2} ... w_{i_t}$?

Size: $|x| + \sum_{i=1}^{m} |w_i| + n$.

We prove

Lemma 4.3 BMS is solvable in polynomial time.

Proof Let x be a binary string and let $S = \{w_1, w_2, ..., w_m\}$ be a set of binary strings. Without loss of generality, we can assume that none of the strings x and $w_1, w_2, \ldots, w_m$ is the empty string ϵ. Assume $x = b_1 b_2 ... b_{|x|}$ where each b_i is 0 or 1. Let $x_0 = \epsilon$. For each $1 \leq j \leq |x|$, let $x_j = b_1 ... b_j$.

Let $D_0 = \{0\}$. For each $1 \leq k \leq n$, let $D_k = \{j \mid \text{ there exist } i_1, i_2, ..., i_k \text{ such that } x_j = w_{i_1} w_{i_2} ... w_{i_k}\}$. Then $\# D_k$ is at most $|x|$. The algorithm works by computing D_k, $0 \leq k \leq n$, inductively. Once all the D_k's are computed, to decide whether x can be represented as a concatenation of at most n strings in S, we just check whether $|x|$ is in $\bigcup_{k=0}^{n} D_k$.

To compute D_{k+1} from D_k, we do the following. For each $i \in D_k$ and $w \in S$ we check whether $x_i w = x_{i+|w|}$. If the answer is yes, we let D_{k+1} contain $i + |w|$. Since $\# D_k \leq |x|$ for all k, the time to compute D_{k+1} from D_k is polynomial in $|x| + \sum_{i=1}^{m} |w_i|$. Therefore the total time is polynomial in $n + |x| + \sum_{i=1}^{m} |w_i|$. □

As a corollary, we have the following theorem:

Theorem 4.4 BMN is solvable in time polynomial in $n + |I(X)| + \sum_{Y \in S} |I(Y)|$.

Proof Use the isomorphism I. □

Notice that $|I(X)|$ is not necessarily polynomial in the size of $|X|$. What we need next is to bound the size $|I(X)|$ in terms of $|X|$ on the average. The fact that $|I(X)|$ is polynomial on the average was proved in [1]; our proof here leads to a precise bound, and some results in the process of the proof will be used in the next section to give a polynomial time algorithm for BMN(σ).

4.2 $|I(X)|$ on the average

We first collect some preliminary results on unimodular matrices.

If $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$, then we have $\gcd(a, b) = 1$ and $\gcd(c, d) = 1$.

Lemma 4.5 For any positive integers a, b that are relatively prime $\gcd(a, b) = 1$, there exists a unique pair of integers c, d, where $0 \le c < a$ and $0 < d \le b$, such that $ad - bc = 1$.

Proof From the Euclidean algorithm, there are integers b, d such that $ad - bc = 1$. We have $a, b \ge 1$. By replacing c and d with $c + ka$ and $d + kb$ respectively, we can assume that $0 \le c < a$. Then, $d = \frac{1+bc}{a} > 0$, and

$$d < \frac{1}{a} + b \le b + 1.$$

Hence, being an integer, $d \le b$.

For uniqueness, suppose we have $ad - bc = 1$ and $ad' - bc' = 1$, satisfying the constraints. Then $a(d - d') = b(c - c')$, thus $a | c - c'$ and $b | d - d'$, since $\gcd(a, b) = 1$. Hence $c = c'$ and $d = d'$. $\square$

If $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$, and one of a or b is 0, then the only possibility is the matrix $\begin{pmatrix} 1 & 0 \\ c & 1 \end{pmatrix}$, (note that all entries are nonnegative).

We define a notion of domination. We say that $\begin{pmatrix} a \\ c \end{pmatrix} \ge \begin{pmatrix} b \\ d \end{pmatrix}$ if $a \ge b$ and $c \ge d$. Similarly, $(a \;\; b) \ge (c \;\; d)$ if $a \ge c$ and $b \ge d$.

Lemma 4.6 If $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$, then either $\begin{pmatrix} a \\ c \end{pmatrix} \ge \begin{pmatrix} b \\ d \end{pmatrix}$ or $\begin{pmatrix} b \\ d \end{pmatrix} \ge \begin{pmatrix} a \\ c \end{pmatrix}$ or $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = I$.

Proof We have $ad - bc = 1$. First suppose $b = 0$. Then $a = d = 1$. Thus either $\begin{pmatrix} a \\ c \end{pmatrix} \ge \begin{pmatrix} b \\ d \end{pmatrix}$, or $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. The case of $c = 0$ is similar.

Now suppose $bc \neq 0$. By $ad - bc = 1$, either $a \ge b$ or $d \ge c$. Suppose $a \ge b$. If also $c \ge d$, then $\begin{pmatrix} a \\ c \end{pmatrix} \ge \begin{pmatrix} b \\ d \end{pmatrix}$. If $a \ge b$ and $d > c$, then $d \ge c + 1$ and $1 + bc = ad \ge b(c + 1)$. It follows that $b = 1$ and all inequalities $\ge$ above are really equalities. i.e., $a = b = 1$ and $d = c + 1$. Thus, $\begin{pmatrix} a \\ c \end{pmatrix} \le \begin{pmatrix} b \\ d \end{pmatrix}$.

Symmetrically, suppose $d \ge c$. If also $b \ge a$, then $\begin{pmatrix} a \\ c \end{pmatrix} \le \begin{pmatrix} b \\ d \end{pmatrix}$. If however $d \ge c$ and $a > b$, then as above $d = c = 1$ and $a = b + 1$, thus, $\begin{pmatrix} a \\ c \end{pmatrix} \ge \begin{pmatrix} b \\ d \end{pmatrix}$. We have covered all cases. $\square$

Apply the lemma to the transpose of a matrix in $SL_2(N)$, we get

Corollary 4.7 If $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$, then either $(a \;\; b) \ge (c \;\; d)$, or $(c \;\; d) \ge (a \;\; b)$, or $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = I$.

We show next that it follows from Lemma 4.6 that every $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$ is expressible uniquely as a product of $A_0 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, and $B_0 = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$. This was shown in [1]; however in order to prove our precise bound in Theorem 4.10 we will give the following proof. Suppose $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \neq I$, and $\begin{pmatrix} a \\ c \end{pmatrix} \geq \begin{pmatrix} b \\ d \end{pmatrix}$. Then $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a-b & b \\ c-d & d \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$. Furthermore, one can not use $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, as $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} a & b-a \\ c & d-c \end{pmatrix}$ does not have nonnegative entries. Similarly, if $\begin{pmatrix} b \\ d \end{pmatrix} \geq \begin{pmatrix} a \\ c \end{pmatrix}$, then uniquely, $\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b-a \\ c & d-c \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ in $SL_2(N)$. In either case, one can apply the same step to the matrix $\begin{pmatrix} a-b & b \\ c-d & d \end{pmatrix} \in SL_2(N)$, or $\begin{pmatrix} a & b-a \\ c & d-c \end{pmatrix} \in SL_2(N)$, to continue the process, provided they are not already identity I.

This is really a version of the Euclidean algorithm. It is easy to see that if $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$ is a product of A_0's and B_0's, then if the first term is A_0 then $(c \;\; d) \geq (a \;\; b)$, and if the first term is B_0 then $(a \;\; b) \geq (c \;\; d)$. Similarly, if the last term is A_0 then $\begin{pmatrix} a \\ c \end{pmatrix} \geq \begin{pmatrix} b \\ d \end{pmatrix}$, and if the last term is B_0 then $\begin{pmatrix} b \\ d \end{pmatrix} \geq \begin{pmatrix} a \\ c \end{pmatrix}$.

Lemma 4.8 Any $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$ is uniquely expressible as a product of A_0 and B_0. The first term and last term being A_0 or B_0 correspond to exactly four cases of whether $(a \;\; b) \geq (c \;\; d)$ and whether $\begin{pmatrix} a \\ c \end{pmatrix} \geq \begin{pmatrix} b \\ d \end{pmatrix}$.

Suppose $(a \;\; b) \geq (c \;\; d)$, we know from Lemma 4.5 that the pair $(c \;\; d)$ which makes up $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SL_2(N)$ satisfying the constraint $(a \;\; b) \geq (c \;\; d)$ is unique. The process of obtaining the unique expression as a product of A_0's and B_0's is exactly the same as a subtractive Euclidean algorithm performed on the entries a and b, and $|I(X)|$ is precisely the number of steps of this subtractive Euclidean algorithm. The other case of $(c \;\; d) \geq (a \;\; b)$ is symmetric. We note that one can obtain a more succinct expression for $I(X)$ using exponents for A_0 and B_0, $X = X_1^{i_1} X_2^{i_2} \ldots X_r^{i_r}$, where for any $1 \leq i \leq r$, X_i is either A_0 or B_0 and $X_i \neq X_{i+1}$ for $1 \leq i \leq r-1$. This expression is computable in polynomial time, using the ordinary Euclidean algorithm (with divisions). In fact, suppose we are in the case $(a \;\; b) \geq (c \;\; d)$ and $\begin{pmatrix} a \\ c \end{pmatrix} \geq \begin{pmatrix} b \\ d \end{pmatrix}$, then $b \neq 0$ otherwise $b \geq d = 0$. We claim $a \neq b$, for otherwise, by $\gcd(a, b) = 1$, $a = b = 1$. Then by $ad - bc = 1$, $d = c + 1$, and we violate the domination condition $\begin{pmatrix} a \\ c \end{pmatrix} \geq \begin{pmatrix} b \\ d \end{pmatrix}$. So $a \neq b$. Let $k = \lceil a/b \rceil - 1$, then $0 < a - kb \leq b$.

Moreover for all $i < k$, $a - ib > b$. Hence by Lemma 4.6, for all $i < k$, $\begin{pmatrix} a-ib \\ c-id \end{pmatrix} \geq \begin{pmatrix} b \\ d \end{pmatrix}$. If $b \not| a$ then $a - kb < b$, and by the same lemma $\begin{pmatrix} b \\ d \end{pmatrix} \geq \begin{pmatrix} a-kb \\ c-kd \end{pmatrix}$; if $b|a$, then $b = 1$ by $\gcd(a, b) = 1$. Then $a - kb = b = 1$, and since $b \geq d$, and $ad - bc = 1$, $d = 1$. It follows that it must be that $\begin{pmatrix} a-kb & b \\ c-kd & d \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = B_0$, since it is in $SL_2(N)$. In either case the Euclidean algorithm must switch direction at this point, and we have $X_r = A_0$ and $i_r = k$. The other cases are similar. We then successively compute all other X_j and i_j.

Let $s(m, n)$ be one less than the sum of all partial quotients in the continued fraction representation of m/n, for $1 \leq m \leq n$. Then $s(m, n)$ is the number of subtractive steps in the following algorithm for finding the greatest common divisor of m and n by mutual subtraction: Replace the larger number by the difference of the two numbers until both are equal; then the common value is the gcd.

Yao and Knuth proved the following beautiful result[11]:

$$\sum_{m \leq n} s(m, n) = \frac{6}{\pi^2} n (\log n)^2 + O(n \log n (\log \log n)^2).$$

This implies that for any n, the *average* number of steps the above subtractive algorithm takes for any pair of integers $m \leq n$ is asymptotically $\frac{6}{\pi^2}(\log n)^2$. We want to show that the above estimate is also valid for pairs of integers that are known to be *relatively prime*, as this is the case for both rows of any unimodular matrix.

Let

$$Y(n) = \sum_{m \leq n} s(m, n),$$

and

$$Y'(n) = \sum_{m \leq n,\ (m,n)=1} s(m, n).$$

We are interested in the sum $\sum_{n \leq N} Y'(n) = \sum_{m \leq n \leq N,\ (m,n)=1} s(m, n)$, which is the sum of all costs of the subtractive algorithm on *relatively prime* pairs of integers up to N.

Theorem 4.9

$$\sum_{n \leq N} Y'(n) = \frac{18}{\pi^4} N^2 (\log N)^2 + O(N^2 \log N (\log \log N)^2).$$

Proof Clearly,

$$\begin{aligned} Y(n) &= \sum_{d|n} \left(\sum_{m \leq n,\ (m,n)=d} s(m, n) \right) \\ &= \sum_{d|n} Y'(n/d), \end{aligned}$$

since $s(m, n) = s(m/d, n/d)$ when $d|m$ and $d|n$. Thus, by the Möbius inversion formula

$$Y'(n) = \sum_{d|n} \mu(d) Y(n/d),$$

where $\mu(n)$ is the Möbius function of number theory.

Now

$$\begin{aligned}
\sum_{n \le N} Y'(n) &= \sum_{n \le N} \sum_{d|n} \mu(d) Y(n/d) \\
&= \sum_{i,j \le N \ i \cdot j \le N} \mu(i) Y(j) \\
&= \sum_{i \le N} \mu(i) \sum_{j \le \lfloor N/i \rfloor} Y(j).
\end{aligned}$$

Here the second step is justified by the fact that the sum is really over all lattice points (i, j) under the hyperbola $i \cdot j \le N$ in the first quadrant $1 \le i, j \le N$.

Applying the asymptotic formula of Yao and Knuth, we get

$$\sum_{j \le \lfloor N/i \rfloor} Y(j) = \int_1^{N/i} \frac{6}{\pi^2} x (\log x)^2 dx + E,$$

where the error term

$$\begin{aligned}
E &= O(\frac{N}{i}(\log N)^2 + \sum_{j \le \lfloor N/i \rfloor} j \log j (\log \log j)^2) \\
&= O(\left(\frac{N}{i}\right)^2 \log N (\log \log N)^2 + \frac{N}{i}(\log N)^2).
\end{aligned}$$

Since

$$\int_1^{N/i} x(\log x)^2 dx = \frac{N^2}{2i^2} \left(\left(\log \frac{N}{i} \right)^2 - \log \frac{N}{i} + \frac{1}{2} \right) - \frac{1}{4},$$

$$\begin{aligned}
\sum_{n \le N} Y'(n) &= \sum_{i \le N} \mu(i) \left[\int_1^{N/i} \frac{6}{\pi^2} x (\log x)^2 dx + E \right] \\
&= \sum_{i \le N} \mu(i) \left[\frac{3}{\pi^2} \frac{N^2}{i^2} \left(\left(\log \frac{N}{i} \right)^2 - \log \frac{N}{i} + \frac{1}{2} \right) - O(1) + E \right] \\
&= \frac{3}{\pi^2} \sum_{i \le N} \mu(i) \left[\frac{N^2}{i^2} \left((\log N)^2 + O(\log N \log i) \right) + E \right].
\end{aligned}$$

Now $\sum_{i=1}^{\infty} \frac{\mu(i)}{i^2} = \left(\sum_{i=1}^{\infty} \frac{1}{i^2} \right)^{-1} = \frac{6}{\pi^2}$, and

$$\left| \sum_{i=1}^{\infty} \frac{\mu(i)}{i^2} \log i \right| \le \sum_{i=1}^{\infty} \frac{\log i}{i^2} < \infty,$$

$$\left| \sum_{i \le N} \frac{\mu(i)}{i} \right| \le \log N + O(1),$$

we get the main term $\frac{18}{\pi^4}N^2(\log N)^2$, with an error term bounded by

$$O(N^2 \log N) + O(N^2 \log N(\log\log N)^2) + O(N(\log N)^3) = O(N^2 \log N(\log\log N)^2). \quad \Box$$

Since

$$\sum_{n \le N} \phi(n) = \sum_{m \le n \le N, (m,n)=1} 1 = \frac{3}{\pi^2}N^2 + O(N \log N),$$

where ϕ is the Euler totient function [7], we have the theorem:

Theorem 4.10 The average number of steps the subtractive gcd algorithm takes on a pair of *relatively prime* integers up to N is

$$\frac{6}{\pi^2}(\log N)^2 + O(\log N(\log\log N)^2).$$

Therefore we have the following theorem

Theorem 4.11 (BMN, ν) is solvable in average polynomial time.

Proof By Theorem 4.4, Theorem 4.10 and Lemma 2.3. $\Box$

5 BMN(σ)

We proved BMN to be NP-complete by reducing Simple Integer Sum to it. From the reduction, we can see that actually the Simple Integer Sum problem is a special case of BMN. Notice that the form of Simple Integer Sum looks simpler than that of BMN. In particular, due to the commutativity of addition, the Simple Integer Sum problem is solvable in polynomial time if the cardinality of S is restricted to be a constant. But the multiplication of matrices doesn't have this commutativity, so we would like to ask: Is BMN also solvable in polynomial time if the cardinality of S is restricted to be a constant? In the following, we prove that this is true, which shows that the difficult part of BMN lies exactly in the Simple Integer Sum problem.

We have seen that every matrix $X \in SL_2(N)$ can be uniquely represented as $X_1^{i_1}X_2^{i_2}...X_r^{i_r}$, where for any $1 \le i \le r$, X_i is either A_0 or B_0 and $X_i \ne X_{i+1}$ for $1 \le i \le r-1$. The uniqueness means that if X has another representation, say, $X = Y_1^{j_1}Y_2^{j_2}...Y_t^{j_t}$ satisfying the same condition, then $t = r$ and $X_k = Y_k$ and $i_k = j_k$ for $1 \le k \le r$.

For any $X \in SL_2(N)$, we will denote $h(X) = ((X_1, i_1), (X_2, i_2), ...(X_r, i_r))$ if $X = X_1^{i_1}X_2^{i_2}...X_r^{i_r}$. This representation is polynomial time computable; in fact, as seen above, the computation in obtaining this representation is exactly the same process as when we apply the Euclidean algorithm on the columns.

To describe the algorithm for BMZ(σ), we need to introduce the optimization problem associated with BMN(σ), denoted by OBMN(σ): Given an instance (X, S, n) where $X \in SL_2(N)$ and $S \subset SL_2(N)$, find the minimum integer $y \le n$, such that (X, S, y) is a positive instance of BMN(σ), i.e., X can be represented as a product of no more than y matrices in S. If there exists no such $y \le n$, the minimum value is defined to be infinity.

A matrix $X \in SL_2(N)$ is called an A_0- (respectively, B_0-) simple matrix if X is a power of A_0 (respectively, B_0). A simple matrix is either an A_0- or a B_0-simple matrix. If a matrix is not a simple matrix, we call it a nonsimple matrix.

Lemma 5.1 If S does not contain any nonsimple matrices, then OBMN(σ) is solvable in polynomial time for any constant σ.

Proof We will need the following version of the Simple Integer Sum problem OI(β): An instance is (x, S, n), where x is a positive integer, S is a set of positive integers and the number of elements in S is no bigger than β. The problem is to

$$\begin{array}{ll} minimize & \sum_{k \in S} t_k \\ s.t. & \sum_{k \in S} k t_k = x \\ & 0 \leq t_k \leq n, \forall k \in S \end{array}$$

For any constant β, OI(β) is solvable in polynomial time, since the number of possible feasible solutions is at most $(n+1)^{\beta}$.

Suppose $h(X) = ((X_1, i_1), ..., (X_m, i_m))$. Then to solve OBMN($\sigma$) on instance (X, S, n), it suffices to solve m separate instances of OBMN(σ), on $(X_k^{i_k}, S, n)$ for $1 \leq k \leq m$. Suppose v_k is the optimal value of OBMN(σ) on instance $(X_k^{i_k}, S, n)$. If $\sum_{k=1}^m v_k$ is bigger than n, then the optimal value of OBMN(σ) on instance (X, S, n) is infinity. Otherwise, it is equal to $\sum_{k=1}^m v_k$.

For any given k, the problem of OBMN(σ) on instance $(X_k^{i_k}, S, n)$ is equivalent to that of OI(σ) on instance (i_k, S_k, n) where $S_k = \{j | X_k^j \in S\}$. Since OI($\sigma$) is solvable in polynomial time, OBMN(σ) on instance $(X_k^{i_k}, S, n)$ is solvable in polynomial time. Therefore OBMN(σ) on instance (X, S, n) is solvable in polynomial time. $\Box$

The following notations are needed to describe our algorithm.

Given a matrix $X \in SL_2(N)$, there is a unique decomposition of X into a product of A_0's and B_0's. We will denote the ith factor (counted from left to right) by $X[i]$ where $i \geq 1$. For any integer $i \leq j$, denote $X[i,j] = \Pi_{k=i}^j X[k]$. We denote $[i,j] = \{k \mid i \leq k \leq j, \text{and } k \text{ is an integer}\}$.

Theorem 5.2 BMN(σ) is solvable in polynomial time.

Proof Given an instance (X, S, n) of BMN(σ). Let S^* be the set of all those nonsimple matrices in S. Consider the following algorithm for BMN(σ).

Algorithm

Step 1: We will construct a directed graph $G = (V, E)$. For each edge $e \in E$, we will assign a weight $w(e)$ to it. We will not construct E explicitly, rather, an edge exists in E only when a finite weight is assigned to the edge. All other edges are implicitly assumed not to exist. Let $l = |I(X)|$, the length of $I(X)$.

$V := \{(0,T), (l+1, H)\} \cup \{(i,H), (j,T) | \exists Y \in S^*, \text{such that } X[i,j] = Y, 1 \leq i \leq j \leq l\}$. (We will explain how to compute V later. Here H and T stand for head and tail respectively.)

$w((i,H),(j,T)) := 1$ if there exists a $Y \in S^*$, such that $X[i,j] = Y$, where $1 \leq i \leq j \leq l$.

For $(i,T), (j,H) \in V$, $i < j$: If $j = i+1$, let the weight of $((i,T),(j,H))$ be 0. Otherwise, solve OBMN(σ) on instance $(X[i+1, j-1], S \setminus S^*, n)$. Notice that this instance consists of only simple matrices. Let the weight of $((i,T),(j,H))$ be the optimal value.

Step 2. Find the shortest path from $(0,T)$ to $(l+1, H)$ in the directed weighted graph defined above. If the length of the shortest path is no more than n, then we assert that

X can be represented as a product of at most n matrices of S. Otherwise the algorithm answers NO.

End of Algorithm.

We want to prove that the algorithm works correctly. It is easy to see that if there is a path with length m, then there must be a representation of X as m matrices in S. Now we prove the converse. Suppose there is a representation of X as a product of at most n matrices, say, $X = C_1C_2...C_m$ with $m \leq n$. Let all the nonsimple matrices among C_i's be $C_{i_1}, C_{i_2}, ..., C_{i_p}$, where $1 \leq i_1 < i_2 < ... < i_p \leq m$. For each k, let $l_k = \sum_{1\leq j<i_k} |I(C_j)|$, let $d_k = |I(C_{i_k})|$, Then $X[l_k+1, l_k+d_k] = C_{i_k}$. Therefore $(l_k+1, H) \in V$, $(l_k+d_k, T) \in V$, and $w((l_k+1, H), (l_k+d_k, T)) = 1$. We want to show that there is a path $(0, T) \rightarrow (l_1+1, H) \rightarrow (l_1+d_1, T) \rightarrow (l_2+1, H) \rightarrow ....\ (l_p+1, H) \rightarrow (l_p+d_p, T) \rightarrow (l+1, H)$ in the graph with total length no more than m.

We need to prove that

1. there is an edge from (l_k+d_k, T) to $(l_{k+1}+1, H)$ with weight at most $i_{k+1} - i_k - 1$ for $1 \leq k \leq p-1$.

2. there is an edge from $(0, T)$ to (l_1+1, H) with weight at most $i_1 - 1$.

3.there is an edge from (l_p+d_p, T) to $(l+1, H)$ with weight at most $m - i_p$.

Once we have proved the above three claims, then the path from $(0, T)$ to $(l+1, H)$ has total weight no more than

$$i_1 - 1 + 1 + (i_2 - i_1 - 1) + 1 + (i_3 - i_2 - 1) + 1 + ... + (i_p - i_{p-1} - 1) + 1 + m - i_p = m.$$

Now let's prove the above three claims. We will only prove the first one. The others are similar.

We know that for all $i_k < j < i_{k+1}$, C_j is a simple matrix. Since $X[l_k+d_k+1, l_{k+1}] = \Pi_{i_k<j<i_{k+1}} C_j$, $X[l_k+d_k+1, l_{k+1}]$ can be represented as a product of $i_{k+1} - i_k - 1$ simple matrices. Therefore the minimum value of the OBMN(σ) problem on instance $(X[l_k + d_k + 1, l_{k+1}], S \setminus S^*, n)$ is no more than $i_{k+1} - i_k - 1$. Therefore the weight of the edge $((l_k+d_k, T), (l_{k+1}+1, H))$ is at most $i_{k+1} - i_k - 1$.

Finally, we prove that this algorithm runs in polynomial time. It is well known that finding the shortest path between two vertices in a directed weighted graph (with nonnegative weights) can be done in polynomial time (for example, see [3]). So it suffices to show that the construction of the graph can be done in polynomial time.

The only step that is not immediately seen to be computable in polynomial time is the construction of V. We note that $l = |I(X)|$ could be exponentially large. However, the key observation is that the number of "alternations" between A_0 and B_0 in the product representation of X is polynomially bounded in the binary length of X.

More precisely, in order to show that V can be computed in polynomial time, it suffices to show that, for each $Y \in S^*$, the computation of $V(Y) = \{(i,j) | 1 \leq i \leq j \leq l, X[i,j] = Y\}$ can be done in polynomial time. Assume $h(Y) = ((Y_1, j_1), (Y_2, j_2), ..., (Y_u, j_u))$ and $h(X) = ((Z_1, t_1), (Z_2, t_2), ..., (Z_v, t_v))$. Notice that if $X[i,j] = Y$ and $t_1 + t_2 + .. + t_{k-1} < i \leq t_1 + t_2 + ... + t_k$, then $i = t_1 + t_2 + ... + t_k - j_1 + 1$ since Y is a nonsimple matrix. Therefore

$$V(Y) = \bigcup_{1\leq k\leq v} \{(\sum_{s=1}^{k} t_s - j_1 + 1, \sum_{s=1}^{k} t_s + j_2 + ... + j_u) | X[\sum_{s=1}^{k} t_s - j_1 + 1, \sum_{s=1}^{k} t_s + j_2 + ... + j_u)] = Y\}.$$

Thus $V(Y)$ is computable in polynomial time. □

We can extend Theorem 5.2 a little bit. The proof still works if we allow the size of S to be variable but the number of simple matrices in S to be bounded by a constant. Thus we have the following result.

Corollary 5.3 If the number of simple matrices in S is a constant, then BMN is solvable in polynomial time.

Acknowledgement: We wish to thank Professors Gurevich and Blass for sending us their pre-publication manuscript.

References

[1] A. Blass and Y. Gurevich, "Matrix Decomposition is Complete for the Average Case", To appear in SIAM J. Computing.

[2] J. Cai, W. H. J. Fuchs, D. Kozen and Z. Liu, "Efficient Average-Case Algorithms for the Modular Group," In the Proceedings of *The 35th Annual Symposium on Foundations of Computer Science*, (1994) 143–152.

[3] T. H. Cormen, C. E. Leiserson and R. L. Rivest, "Introduction to Algorithms", The MIT Press, 1990.

[4] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, New York, 1979.

[5] Y. Gurevich, "Matrix Decomposition Problem is Complete for the average case", In Proc. of the 31st Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, 1990, 802-811.

[6] Y. Gurevich, "Average Case Completeness", J. Computer and System Sciences 42:3, June 1991, 346-398.

[7] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University Press, 5th edition, 1988 printing.

[8] L. Levin, "Average Case Complete Problems", SIAM Journal of Computing 15 (1986), 285–286.

[9] R. Lyndon, P. Schupp, *Combinatorial group theory.* Springer-Verlag, 1977.

[10] W. Magnus, A. Karrass and D. Solitar, *Combinatorial group theory.* Interscience Publishers, 1966.

[11] A. C. Yao and D. E. Knuth, "Analysis of the subtractive algorithm for greatest common divisors", Proc. Nat. Acad. Sci USA 72:12 (1975), 4720–4722.