S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru

Том 11, стр. 144–144 (2014) УДК 510.652
MSC 11U99

ГЕНЕРИЧЕСКИЕ ПОЛИНОМИАЛЬНЫЕ АЛГОРИТМЫ ДЛЯ ПРОБЛЕМЫ О РЮКЗАКЕ В НЕКОТОРЫХ МАТРИЧНЫХ ПОЛУГРУППАХ

А.Н. РЫБАЛОВ

Abstract. In this paper, we propose generic polynomial algorithms for the knapsack problems over semigroups of non-negative integer matrices of arbitrary order and semigroup of non-negative second-order integer matrices with determinant 1. The research was funded in accordance with the state task of the IM SB RAS, project FWNF-2022-0003.

Keywords: generic complexity, knapsack problem, integer matrices.

1. Введение

Проблема о рюкзаке является классической проблемой комбинаторной оптимизации, изучаемой многие десятилетия. Современное состояние исследований по этой проблеме отражено в обзорах Качиани, Иори, Локателли и Мартелло [2, 3], а также в статье Шперлинга и Кочетова [16]. Формулировка оптимизационной проблемы о рюкзаке состоит в следующем. Пусть имеется набор предметов, каждый из которых имеет два параметра – объем и ценность. Также имеется рюкзак определённого объема. Задача заключается в том, чтобы собрать рюкзак с максимальной ценностью предметов внутри, соблюдая при этом ограничение рюкзака на суммарный объем. Кроме того, рассматриваются еще и распознавательные варианты этой проблемы. Например, одна из таких проблем формулируется следующим образом. Дано множество натуральных чисел $A = \{a_1, \ldots, a_n\}$ и натуральное число S. Все числа записаны в двоичной системе счисления. Необходимо определить, можно ли в множестве A выбрать

Rybalov, A.N., Generic polynomial algorithms for the knapsack problem in some matrix semigroups.

Работа выполнена в рамках государственного задания ИМ СО РАН, проект FWNF-2022-0003.

Поступила 5 июля 2022 г., опубликована 1 сентября 2022 г.

подмножество чисел (каждое число можно повторять несколько раз), которые в сумме дают число S. Отметим, что она (при условии, когда повторы запрещены) под именем KNAPSACK содержится в списке NP-полных проблем в знаменитой статье Карпа [8]. Сейчас этот вариант распознавательной проблемы о рюкзаке называется проблемой о сумме подмножества.

Мясников, Николаев и Ушаков [10] ввели аналог распознавательной проблемы о рюкзаке для произвольных групп (полугрупп). Ими была изучена вычислительная сложность этих проблем для различных групп, в частности была доказана полиномиальная разрешимость для гиперболических групп и доказана NP-полнота для групп Баумслага-Солитера. Для полугрупп целочисленных матриц эта проблема также является NP-полной, а потому для них, при условии $\mathrm{P} \neq \mathrm{NP}$, нет полиномиальных алгоритмов, получающих решение для всех входов.

Генерические алгоритмы [7] решают проблемы на множестве почти всех входов, а на редких оставшихся входах выдают неопределенный ответ. Рыбалов [14, 15] построил полиномиальные генерические алгоритмы для проблемы о сумме подмножеств в полугруппах неотрицательных целочисленных матриц произвольного порядка и полугруппе неотрицательных целочисленных матриц второго порядка с определителем 1. В данной работе предлагаются полиномиальные генерические алгоритмы для распознавательной проблемы о рюкзаке в этих полугруппах. Также для этих полугрупп рассматриваются оптимизационные проблемы о рюкзаке, в которых необходимо минимизировать число матричных множителей, входящих в произведение в ненулевой степени, а также минимизировать сумму степеней матриц в произведении. Для этих проблем тоже доказывается их полиномиальная генерическая разрешимость.

Предлагаемые алгоритмы основаны на методе динамического программирования, который состоит в следующем. В процессе работы алгоритма задача разбивается на подзадачи, которые подаются на вход этого же алгоритма. Этот механизм называется *рекурсивным вызовом алгоритма*. Полученные подзадачи, в свою очередь, разбиваются еще на подзадачи и так далее. В конце концов доходим до простых задач, которые легко решаются. Всевозможных подзадач, вообще говоря, экспоненциально много. Но среди них очень много одинаковых, а уникальных подзадач полиномиально ограниченное число. Таким образом, отслеживая появление одинаковых подзадач и вовремя «отсекая» соответствующие рекурсивные вызовы, мы добиваемся того, что алгоритм работает за полиномиальное время. Конечно же, эта схема работает не всегда: например, для всех входов какой-либо NP-полной проблемы она не работает, иначе было бы P=NP. Однако полученные результаты показывают, что для рассматриваемых NP-полных проблем она работает для «почти всех» входов.

2. Генерические алгоритмы

Пусть I – некоторое множество входов. Для подмножества $S \subseteq I$ определим последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где I_n – множество входов размера n, а $S_n = S \cap I_n$ – множество входов из S размера n. *Асимптотической плотностью* S назовем предел (если он существует)

$$\rho(S) = \lim_{n\to\infty} \rho_n(S).$$

Множество S называется *генерическим*, если $\rho(S) = 1$ и *пренебрежимым*, если $\rho(S) = 0$.

Алгоритм $\mathcal{A}$ с множеством входов I называется *генерическим*, если множество $\{x \in I : \mathcal{A}(x)$ останавливается$\}$ генерическое. Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если

$$\forall x \in I\ \mathcal{A}(x) \text{ останавливается } \Rightarrow\ f(x) = \mathcal{A}(x).$$

Генерический алгоритм $\mathcal{A}$ работает за полиномиальное время, если существует полином $p(n)$ такой, что

$$\forall x \in I\ \mathcal{A}(x) \text{ останавливается } \Rightarrow t_{\mathcal{A}}(x) < p(size(x)),$$

где $size(x)$ – размер входа x, а $t_{\mathcal{A}}(x)$ – время работы алгоритма $\mathcal{A}$ на входе x. Еще такие алгоритмы мы будем называть полиномиальными генерическими.

С практической точки зрения, когда требуется построить алгоритм, решающий конкретную алгоритмическую проблему для почти всех входов, удобнее рассматривать алгоритмы следующего типа [5]. Каждый такой алгоритм останавливается на всех входах, на входах из некоторого генерического множества выдает правильный ответ, а на пренебрежимом множестве остальных входов выдает специальный ответ «?» – «Не знаю».

Алгоритм $\mathcal{A}$ с множеством входов I и множеством выходов $J \cup \{?\}$ $(? \notin J)$ называется *эффективно генерическим*, если

(1) $\mathcal{A}$ останавливается на всех входах из I,
(2) множество $\{x \in I : \mathcal{A}(x) =?\}$ пренебрежимо.

Эффективно генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если

$$\forall x \in I\ \mathcal{A}(x) \neq ?\ \Rightarrow\ f(x) = \mathcal{A}(x).$$

Множество $S \subseteq I$ и соответствующая проблема распознавания (S, I) *(эффективно) генерически разрешимы*, если существует (эффективно) генерический алгоритм, вычисляющий характеристическую функцию S.

Легко видеть, что из эффективной генерической разрешимости следует генерическая разрешимость. Действительно, любой эффективный генерический алгоритм можно легко переделать в генерический заменив выдачу ответа «?» на бесконечное зацикливание. В обратную сторону это неверно – см., например, теорему 2.22 и следствие 2.24 в [6]. Однако для полиномиальной сложности верно и обратное: из полиномиальной генерической разрешимости следует полиномиальная эффективная разрешимость. Действительно, если имеется полиномиальная оценка $p(n)$ на время работы генерического алгоритма в случае, когда он останавливается, то можно завести счетчик T числа шагов, и, в случае, если $T > p(n)$, можно обрывать вычисление и выдавать ответ «?», – в этом случае генерический алгоритм уже не остановится. Таким образом получается эффективно генерический полиномиальный алгоритм, решающий ту же проблему.

3. Полугруппы целочисленных неотрицательных матриц

Обозначим через ω множество натуральных чисел с нулем, а через $\mathrm{M}_k(\omega)$ полугруппу матриц порядка k с целыми неотрицательными элементами с обычной операцией умножения матриц. Порядок матриц k будет фиксированным на протяжении всего подраздела. Элементы $\mathrm{M}_k(\omega)$ будем представлять матрицами из целых неотрицательных чисел. Размер целого положительного числа a, обозначаемый $size(a)$, – это длина его записи в двоичной системе счисления. Под размером матрицы $M = ||a_{ij}||$ будем понимать

$$size(M) = \max_{i,j=1,\ldots,k}\{size(a_{ij})\}.$$

Для любого подмножества $S \subseteq \mathrm{M}_k(\omega)$ обозначим через $S_{\leq n}$ множество всех матриц из S размера $\leq n$.

Сформулируем теперь проблему о рюкзаке для полугруппы $\mathrm{M}_k(\omega)$. Пусть даны произвольные матрицы $(M_1, M_2, \ldots, M_n, M)$ из $\mathrm{M}_k(\omega)$, каждая размером не более n. Определить, существуют ли степени $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n \in \omega$ такие, что имеет место

$$M_1^{\varepsilon_1} M_2^{\varepsilon_2} \ldots M_n^{\varepsilon_n} = M.$$

Размер входа $(M_1, M_2, \ldots, M_n, M)$ полагаем равным n. Условимся, что если матрица $M = E$ – единичная, то M есть произведение пустого подмножества матриц, а потому проблема суммы подмножеств с такой матрицей M разрешима. Это допущение послужит для удобства описания генерического алгоритма в дальнейшем.

Следующее утверждение говорит о том, что для этой проблемы, при условии $\mathrm{P} \neq \mathrm{NP}$, не существует полиномиального алгоритма, который решал бы ее для всех входов.

Лемма 1. *Проблема о рюкзаке для $M_k(\omega)$ NP-полна.*

Доказательство. Докажем, что к ней полиномиально сводится классическая проблема о рюкзаке для натуральных чисел. Определим функцию $F : \omega \to \mathrm{M}_k(\omega)$ следующим образом. Для любого $a \in \omega$ положим

$$F(a) = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 & a \\ 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \ldots & & & & & \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{pmatrix},$$

Полиномиальная сводимость сопоставляет входу $(a_1, a_2, \ldots, a_n, S)$ классической проблемы о рюкзаке для ω вход $(F(a_1), F(a_2), \ldots, F(a_n), F(S))$ проблемы о рюкзаке для $\mathrm{M}_k(\omega)$. Легко проверить, что для любого набора чисел $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n \in \omega$ равенство

$$a_1\varepsilon_1 + a_2\varepsilon_2 + \ldots + a_n\varepsilon_n = S$$

выполняется тогда и только тогда, когда

$$F(a_1)^{\varepsilon_1} F(a_2)^{\varepsilon_2} \ldots F(a_n)^{\varepsilon_n} = F(S).$$

Заметим, что ограничения на размер матриц

$$F(a_1), F(a_2), \ldots, F(a_n), F(S),$$

которые фигурируют в формулировке проблемы о сумме подмножеств для матриц, обходятся добавлением к матрицам необходимого количества единичных матриц $E_1, \ldots, E_k$. При этом k будет ограничено полиномом от n. □

Нам потребуются следующие леммы, которые были доказаны в [14].

Лемма 2. *Пусть $\mathcal{Z}$ есть множество матриц из $M_k(\omega)$ с определителем 0. Тогда*

$$\lim_{n\to\infty} \frac{|\mathcal{Z}_{\leq n}|}{|M_k(\omega)_{\leq n}|} = 0.$$

Лемма 3. *Пусть $S_{\leq n}$ – множество матриц M из $M_k(\omega)$ размера $\leq n$ таких, что матричное уравнение $M = XY$ имеет более $p(n) = (2(n+1))^{k^2+1}$ различных решений $X, Y \in M_k(\omega)$ таких, что $size(X), size(Y) \leq n$. Тогда*

$$\frac{|S_{\leq n}|}{|M_k(\omega)_{\leq n}|} < \frac{1}{2(n+1)}.$$

Будем использовать эти утверждения для доказательства следующего результата.

Теорема 1. *Проблема о рюкзаке для $M_k(\omega)$ является генерически разрешимой за полиномиальное время.*

Доказательство. Опишем как работает эффективно генерический полиномиальный алгоритм $\mathcal{A}$ на входе $(M_1, \ldots, M_n, M)$ размера n.

(1) Вычисляем определитель матрицы M. Если $det(M) = 0$, выдаем ответ «?». Из леммы 2 следует, что множество входов $(M_1, \ldots, M_n, M)$, у которых $det(M) = 0$, пренебрежимо.
(2) Вычисляем определители матриц $M_1, \ldots, M_n$. Выбрасывам те матрицы, у которых определитель равен 0 – они не могут участвовать в произведении, которое равно M, так как $det(M) \neq 0$. На последующих шагах считаем, что все матрицы $M_1, \ldots, M_n, M$ невырождены.
(3) В последующей работе алгоритм $\mathcal{A}$ будет осуществлять рекурсивные вызовы алгоритма $\mathcal{A}$ на других входных данных. Будем контролировать число таких вызовов, для чего заведем счетчик вызовов R, который в самом начале будет равен 0.
(4) Если счетчик числа рекурсивных вызовов R стал равен $np(n)$, где p – полином из леммы 3, то останавливаем все рекурсивные вызовы и выдаем ответ «?».
(5) Если $M = E$ – единичная матрица, то останавливаем все запущенные рекурсивные вызовы алгоритма $\mathcal{A}$ и выдаем ответ «ДА».
(6) Для каждой матрицы M_i, $i = 1, \ldots, n$ решаем матричное уравнение $M_i X = M$. Оно сводится к системе линейных уравнений, которое решаем методом Гаусса в рациональных числах. Если решение получается в натуральных числах, то запускаем рекурсивно алгоритм $\mathcal{A}$ для всех входов $(M_{i+1}, \ldots, M_n, X)$ и $(M_i, \ldots, M_n, X)$ таких, для которых алгоритм $\mathcal{A}$ не был запущен ранее. При этом каждый раз увеличиваем счетчик рекурсивных вызовов $R := R + 1$.
(7) Если ни одна из этих систем неразрешима в натуральных числах, то останавливаем текущий рекурсивный вызов алгоритма $\mathcal{A}$ и выдаем ответ «Нет решения на данной подзадаче».

(8) Если все запущенные рекурсивные вызовы в какой-то момент остановились и выдали ответ «Нет решения на данной подзадаче», то выдаем ответ «НЕТ».

Докажем полиномиальность алгоритма $\mathcal{A}$. Каждая вычислительная процедура (метод Гаусса, вычисление определителя), используемая внутри алгоритма работает за полиномиальное время. Кроме того, число рекурсивных вызовов алгоритма $\mathcal{A}$ самим собой ограничено полиномом – оно не превосходит $np(n)$, где p – полином из леммы 3.

Докажем теперь пренебрежимость множества входов, на которых алгоритм $\mathcal{A}$ выдает ответ «?». Из леммы 3 следует, что множество входов $(M_1, \ldots, M_n, M)$ проблемы суммы подмножеств размера n, в которых число решений матричного уравнения $M = XY$ в матрицах из $\mathrm{M}_k(\omega)$ не превосходит $p(n)$, является генерическим. Заметим, что для любого такого входа $(M_1, \ldots, M_n, M)$ алгоритм $\mathcal{A}$ выдаст ответ, отличный от ответа «?». Для того, чтобы убедиться в этом, оценим число возможных подзадач $(M_i, \ldots, M_n, M')$, для которых происходят рекурсивные вызовы алгоритма $\mathcal{A}$. Для всех матриц, кроме последней имеется не более n вариантов выбора. Для матрицы M' всегда найдется матрица X такая, что $M = XM'$. Значит число вариантов выбора матрицы M' не может быть больше числа решений матричного уравнения $M = XY$ в матрицах из $\mathrm{M}_k(\omega)$, то есть, согласно лемме 3, $p(n)$. Итого получаем не более $np(n)$ вариантов для возможных подзадач $(M_i, \ldots, M_n, M')$. А так как в алгоритме счетчик числа рекурсивных вызовов ограничен как раз значением $np(n)$, то все эти рекурсивные вызовы происходят и соответствующие подзадачи решаются. Поэтому на таких входах алгоритм обязательно выдает ответ, отличный от «?». □

Отметим, что описанный алгоритм можно несколько ускорить, используя процедуру вычисления определителя, близкую по сложности к умножению матриц [11]. Для умножения матриц можно использовать алгоритм Штрассена-Винограда [17, 18], который эффективнее обычного алгоритма умножения матриц порядка большего нескольких сотен. Обзоры современных алгоритмов быстрого матричного умножения есть в [9, 13]. Также для проверки невырожденности целочисленных матриц можно использовать алгоритмы приведения к нормальной форме Смита [1].

Сформулируем теперь две оптимизационные проблемы о рюкзаке для полугруппы $\mathrm{M}_k(\omega)$. Пусть даны произвольные матрицы $(M_1, M_2, \ldots, M_n, M)$ из $\mathrm{M}_k(\omega)$, каждая размером не более n. Найти степени $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n \in \omega$ такие, что

$$M_1^{\varepsilon_1} M_2^{\varepsilon_2} \ldots M_n^{\varepsilon_n} = M$$

и такие, что

(1) число ненулевых чисел среди $\varepsilon_1, \ldots, \varepsilon_n$ минимально (*проблема минимизации числа множителей рюкзака*),
(2) сумма $\varepsilon_1 + \ldots + \varepsilon_n$ минимальна (*проблема минимизации суммы степеней множителей рюкзака*).

Теорема 2. *Проблемы минимизации числа и суммы степеней множителей рюкзака для $M_k(\omega)$ являются генерически разрешимыми за полиномиальное время.*

Доказательство. Рассмотрим алгоритм из доказательства теоремы 1. Если в каждой подзадаче, помимо матриц хранить еще текущие показатели степеней матриц, которые могут входить в будущие решения, то в результате его работы находятся показатели $\varepsilon_1, \ldots, \varepsilon_n$ для всех возможных решений матричного уравнения

$$M_1^{\varepsilon_1} M_2^{\varepsilon_2} \ldots M_n^{\varepsilon_n} = M.$$

При этом на шаге 5 нужно, выяснив, что решение есть, не обрывать вычисления, а добавить найденное решение в список решений и оборвать только текущий рекурсивный вызов. В итоге, учитывая, что число этих решений ограничено полиномом от размера входа, можно за полиномиальное время найти то, которое минимизирует нужный нам параметр. □

4. Полугруппа целочисленных неотрицательных матриц второго порядка с определителем 1

Через $SL_2(\omega)$ обозначим полугруппу матриц второго порядка с целыми неотрицательными целыми элементами, определитель которых равен 1, с обычной операцией умножения матриц. Элементы $SL_2(\omega)$ будем представлять матрицами, а под размером матрицы

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

будем понимать максимум из размеров натуральных чисел a, b, c, d, записанных в двоичной системе счисления. Проблема о рюкзаке и проблемы минимизации числа и суммы степеней множителей рюкзака для $SL_2(\omega)$ формулируются аналогично тому, как это делалось в предыдущем разделе. Согласно лемме 1, проблема о рюкзаке для $SL_2(\omega)$ является NP-полной. Для любого подмножества $S \subseteq SL_2(\omega)$ обозначим через $S_{\leq n}$ множество всех матриц из S размера $\leq n$.

Нильсен в [12] доказал, что полугруппа $SL_2(\omega)$ является свободной полугруппой с двумя порождающими

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \; B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Таким образом, любая матрица из $SL_2(\omega)$ единственным образом представляется в виде произведения матриц, каждая из которых есть либо A, либо B. Будем называть такое произведение *нормальной формой* элемента из $SL_2(\omega)$.

Опишем алгоритм Ψ приведения матриц из $SL_2(\omega)$ к нормальной форме. Этот алгоритм работает на матрице

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = M_0 = \begin{pmatrix} a_0 & b_0 \\ c_0 & d_0 \end{pmatrix}$$

размера n следующим образом.

(1) Вначале инициализируем счетчик числа раундов $k = 0$.
(2) Вначале выходная строка w_0 пустая.
(3) Если $M_k = E$, выдаем ответ w_k. Через M_k обозначается матрица, а через w_k выходная строка на текущем раунде k.
(4) Если число раундов $k = n^5$, то выдаем ответ «?». Такая оценка необходима для получения нужной верхней границы на долю тех входов, для которых алгоритм не может найти нормальную форму (более подробные разъяснения см. в [15]).

(5) Пусть матрица $M_k = \begin{pmatrix} a_k & b_k \\ c_k & d_k \end{pmatrix}$ на данном раунде. Если $a_k \leq c_k$, то полагаем

$$M_{k+1} = B^{-1}M_k = \begin{pmatrix} a_k & b_k \\ c_k - a_k & d_k - b_k \end{pmatrix}$$

и $w_{k+1} = Bw_k$. Иначе (если $a_k > c_k$), полагаем

$$M_{k+1} = A^{-1}M_k = \begin{pmatrix} a_k - c_k & b_k - d_k \\ c_k & d_k \end{pmatrix}$$

и $w_{k+1} = Aw_k$.

(6) Увеличиваем счетчик числа раундов $k := k + 1$.

(7) Возвращаемся на шаг 3.

Следующая лемма была доказана в [15].

Лемма 4. *Алгоритм Ψ является полиномиальным и эффективно генерическим. Более того, пусть $S = \{M \in SL_2(\omega) : \Psi(M) =?\}$. Тогда существует константа $C > 0$ такая, что для любого n выполнено*

$$\frac{|S_{\leq n}|}{|SL_2(\omega)_{\leq n}|} < \frac{C}{n^2}.$$

Будем использовать это утверждение для доказательства следующего результата.

Теорема 3. *Проблема о рюкзаке в полугруппе $SL_2(\omega)$ генерически полиномиально разрешима.*

Доказательство. Рассмотрим эффективно генерический полиномиальный алгоритм Σ, который будет работать на входе $(M_1, \ldots, M_n, M)$ в два этапа.

Сначала с помощью алгоритма Ψ все матрицы $M_1, \ldots, M_n, M$ приводятся к нормальной форме в виде слов $(w_1, \ldots, w_n, w)$ над алфавитом $\{A, B\}$ как элементы свободной полугруппы. Если при этом алгоритм Ψ хотя бы раз выдал ответ «?», то и алгоритм Σ выдает ответ «?». Иначе слова $(w_1, \ldots, w_n, w)$ передаются на второй этап. Заметим, что множество входов G, на котором алгоритм Σ не выдает ответ «?», является генерическим. Действительно

$$G_n = \{(M_1, \ldots, M_n, M) : \Psi(M_1) \neq?, \ldots, \Psi(M_n) \neq?, \Psi(M) \neq?\}.$$

Откуда по лемме 4

$$\rho_n(G) > \left(1 - \frac{C}{n^2}\right)^n \left(1 - \frac{C}{n^2}\right) = \left(\left(1 - \frac{C}{n^2}\right)^{n^2}\right)^{1/n} \left(1 - \frac{C}{n^2}\right)$$

и значит $\lim_{n\to\infty} \rho_n(G) = 1$.

На втором этапе решается проблема о рюкзаке в свободной полугруппе $\{A, B\}^*$. Имеются слова $(w_1, \ldots, w_n, w)$ над алфавитом $\{A, B\}$, причем их длины ограничены n^5 – это следует из описания алгоритма Ψ. Нужно выяснить, существуют ли степени $\varepsilon_1, \ldots, \varepsilon_n \in \omega$ такие, что $w_1^{\varepsilon_1} \ldots w_n^{\varepsilon_n} = w$. Для этого применим следующий алгоритм Ω. Для удобства описания алгоритма будем считать, что при пустом слове w проблема разрешима (ответ ДА).

(1) Если слово w пустое, останавливается (и останавливает все запущенные рекурсивные вызовы) и выдает ответ «ДА».

(2) Среди слов $w_1, \ldots, w_n$ ищет все слова $w_{j_1}, \ldots, w_{j_k}$, которые являются префиксами слова w. Если таких слов нет, то останавливает текущий рекурсивный вызов и выдает ответ «Нет решения на данной подзадаче».

(3) Запускает рекурсивно алгоритм Ω для всех входов

$$(w_{j_m+1}, w_{j_m+2}, \ldots, w_n, \overline{w}_{j_m} w),$$

и

$$(w_{j_m}, w_{j_m+1}, \ldots, w_n, \overline{w}_{j_m} w),$$

таких, для которых алгоритм Ω не был запущен ранее. Здесь через $\overline{w}_{j_m} w$ обозначено слово w без начального куска w_{j_m}.

(4) Если все запущенные рекурсивные вызовы в какой-то момент остановились и выдали ответ «Нет решения на данной подзадаче», выдает ответ «НЕТ».

Для того, чтобы доказать полиномиальность алгоритма Ω, заметим, что число возможных рекурсивных вызовов Ω не превосходит числа возможных подзадач вида $(w_m, w_{m+1}, \ldots, w_n, w')$, где w' – слово w без некоторого начального куска. Число таких подзадач не превосходит $n \cdot |w| < n \cdot n^5 = n^6$ – ограничено полиномом от n. □

Доказательство следующей теоремы аналогично доказательству теоремы 2.

Теорема 4. *Проблемы минимизации числа и суммы степеней множителей рюкзака для $SL_2(\omega)$ являются генерически разрешимыми за полиномиальное время.*

Автор выражает благодарность рецензенту за полезные замечания и предложения по улучшению текста статьи.

References

[1] S. Birmpilis, G. Labahn, A. Storjohann. *A fast algorithm for computing the Smith normal form with multipliers for a nonsingular integer matrix*, Journal of Symbolic Computation, **116** (2023), 146–182. *https://doi.org/10.1016/j.jsc.2022.09.002*

[2] V. Cacchiani, M. Iori, A. Locatelli, S. Martello. *Knapsack problems – An overview of recent advances. Part I: Single knapsack problems*, Computers and Operations Research, **143:105692** (2022), 1–13. *https://doi.org/10.1016/j.cor.2021.105692*

[3] V. Cacchiani, M. Iori, A. Locatelli, S. Martello. *Knapsack problems – An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems*, Computers and Operations Research, **143:105693** (2022), 1–14. *https://doi.org/10.1016/j.cor.2021.105693*

[4] M. Garey, D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979. 340p. *https://bohr.wlu.ca/hfan/cp412/references/ChapterOne.pdf*

[5] D. Hirschfeldt. *Some Questions in Computable Mathematics*, Computability and Complexity, (2017), 22–55. *https://math.uchicago.edu/ drh/Papers/Papers/open.pdf*

[6] C. Jockusch, P. Schupp. *Generic computability, Turing degrees, and asymptotic density*, Journal of the London Mathematical Society, **85:2** (2012), 472–490. *https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.225.1953&rep=rep1&type=pdf*

[7] I. Kapovich, A. Myasnikov, P. Schupp, V. Shpilrain. *Generic-case complexity and decision problems in group theory*, Journal of Algebra, **264** (2003), 665–694. *https://arxiv.org/pdf/math/0203239*

[8] R. Karp. *Reducibility among combinatorial problems*, Complexity of Computer Computations. The IBM Research Symposia Series, (1972), 85–103. *https://link.springer.com/chapter/10.1007/978-1-4684-2001-2_9*

[9] E. Karstadt, O. Schwartz. *Matrix multiplication, a little faster*, Journal of ACM, **67:1** (2020), article no. 1, 1–31. *https://doi.org/10.1145/3364504*

[10] A. Miasnikov, A. Nikolaev, A. Ushakov. *Knapsack problems in groups*, Mathematics of Computation, **84** (2015), 987–1016. *https://arxiv.org/pdf/1302.5671*

[11] V. Neiger, C. Pernet. *Deterministic computation of the characteristic polynomial in the time of matrix multiplication*, Journal of Complexity, **67:101572** (2021), 1–35. *https://doi.org/10.1016/j.jco.2021.101572*

[12] J. Nielsen. *Die Gruppe der dreidimensionalen Gittertransformationen*, Kgl. Danske Vid. Selsk., Math.-fys. Medd., **5:12** (1924), 3–29. (in German) *http://gymarkiv.sdu.dk/MFM/kdvs/mfm%201-9/mfm-5-12.pdf*

[13] A. Rosowski. *Fast commutative matrix algorithms*, Journal of Symbolic Computation, **114** (2023), 302–321. *https://doi.org/10.1016/j.jsc.2022.05.002*

[14] A. Rybalov. *On generic complexity of the subset sum problem for semigroups of integer matrices*, Applied Discrete Mathematics, **50** (2020), 118–126. (in Russian) *http://www.mathnet.ru/php/archive.phtml?wshow=paper & jrnid=pdm & paperid=727 & option_lang=rus*

[15] A. Rybalov. *On generic complexity of the subset sum problem in monoids and groups of integer matrix of order two*, Herald of Omsk State University, **25:4** (2020), 10–15. (in Russian) *https://elib.omsu.ru/issues/183/7188.pdf*

[16] S.M. Shperling, Y.A. Kochetov. *A knapsack problem for rectangles under center-of-gravity constraints*, Journal of Applied and Industrial Mathematics. **16:3** (2022), 563–571. *https://doi.org/10.1134/S199047892203019X*

[17] V. Strassen. *Gaussian elimination is not optimal*, Numerische Mathematik, **13:4** (1969), 354–356. *https://doi.org/10.1007/BF02165411*

[18] S. Winograd. *On multiplication of 2x2 matrices*, Linear Algebra and its Applications, **4:4** (1971), 381–388. *https://doi.org/10.1016/0024-3795(71)90009-7*

Alexander Nikolaevich Rybalov
Sobolev Institute of Mathematics,
prospekt Koptyuga 4,
Novosibirsk, 630090, Russia.
Email address: alexander.rybalov@gmail.com