ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ В ДИСКРЕТНОЙ МАТЕМАТИКЕ

DOI 10.17223/20710410/20/7

УДК 631.391:519.2

БЫСТРЫЙ АЛГОРИТМ ВОССТАНОВЛЕНИЯ ИСТИННОГО РЕШЕНИЯ ФИКСИРОВАННОГО ВЕСА СИСТЕМЫ ЛИНЕЙНЫХ БУЛЕВЫХ УРАВНЕНИЙ С ИСКАЖЁННОЙ ПРАВОЙ ЧАСТЬЮ

А. Н. Алексейчук, А. Ю. Грязнухин

Институт специальной связи и защиты информации Национального технического университета Украины «Киевский политехнический институт», г. Киев, Украина

E-mail: alex-crypto@mail.ru, x67mail@gmail.com

Рассматривается система линейных булевых уравнений с искажённой правой частью, истинное решение которой имеет заданный вес, не зависящий от числа неизвестных в системе. Предлагается вероятностный алгоритм нахождения этого решения, имеющий меньшую временную сложность по сравнению с методом максимума правдоподобия. В отличие от известных алгоритмов, обладающих указанным свойством, предложенный алгоритм использует только операции сравнения и (поразрядного и арифметического) сложения двоичных целых чисел, что позволяет применять его на практике в случае, когда другие алгоритмы оказываются менее эффективными.

Ключевые слова: *система булевых уравнений с искаженной правой частью, вероятностный алгоритм.*

Введение

Рассмотрим систему булевых уравнений

$$Ax = b = Ax_0 \oplus \xi, \tag{1}$$

где A — случайная равновероятная матрица размера $m \times k$ над полем из двух элементов; x_0 — неизвестный k-мерный вектор; $\xi = (\xi_1, \ldots, \xi_m)^{\mathrm{T}}$ — случайный вектор с независимыми координатами, распределёнными по закону

$$\Pr\{\xi_i = 1\} = 1 - \Pr\{\xi_i = 0\} = p,\ i = 1, 2, \ldots, m,\ p \in (0, 1/2). \tag{2}$$

Система уравнений (СУ) (1) называется *системой линейных булевых уравнений с искажённой правой частью*, а вектор x_0 — её *истинным решением* [1].

Предположим, что вес (количество ненулевых координат) вектора x_0 равен фиксированному числу $\rho \geqslant 3$. Требуется восстановить этот вектор по известным значениям A, b, p и ρ.

Как правило, данную задачу решают в предположении, что число неизвестных k в системе (1) принимает любые достаточно большие натуральные значения, а вероятность искажения p не зависит от k. При этом требуется разработать алгоритм, позволяющий для любых наперёд заданных $p \in (0, 1/2)$ и $\rho \in \{3, 4, \ldots\}$ находить истинное

решение этой системы уравнений с любой наперёд заданной надёжностью при всех достаточно больших значениях k и m [1–3].

Назовём алгоритм A восстановления истинных решений СУ (1) *состоятельным*, если существует функция $m_0 : \mathbb{N}\times(0,1/2)\times(0,1/2)\times\{3,4,\ldots\} \longrightarrow \mathbb{N}$, удовлетворяющая следующему условию: для любых $p, \delta \in (0,1/2)$, $\rho \in \{3,4,\ldots\}$ существует число $k_0 \in \mathbb{N}$, такое, что для любого натурального $k \geqslant k_0$ и произвольного вектора $x_0 \in \{0,1\}^k$ веса ρ вероятность правильного восстановления этого вектора с помощью алгоритма A по известной реализации случайной системы (1), состоящей из $m = m_0(k,p,\delta,\rho)$ уравнений, больше либо равна $1-\delta$.

В дальнейшем рассматриваются только состоятельные алгоритмы. Трудоёмкость (или временная сложность в худшем случае) каждого из них является функцией четырёх параметров: $T = T(k,p,\delta,\rho)$, где $m = m_0(k,p,\delta,\rho)$; $k \geqslant k_0$; $p,\delta \in (0,1/2)$; $\rho \in \{3,4,\ldots\}$. Обычно интересуются асимптотическим поведением этой функции при $k \to \infty$ для каждого набора значений (p,δ,ρ). Наибольший интерес представляют алгоритмы, трудоёмкость которых полиномиально зависит от k, $(1-2p)^{-1}$ и δ^{-1}.

Уточним понятие элементарной операции, необходимое для оценки трудоёмкости алгоритмов. Отметим, что в большинстве работ, посвящённых алгоритмам решения систем булевых уравнений с искажёнными правыми частями, тип используемых операций не оговаривается явно. Как правило, это операции над целыми числами или двоичными векторами произвольной длины. В настоящей работе *элементарной* называется любая двоичная операция (булева функция двух переменных), а также операция $i \mapsto i+1$, где i — произвольное целое число. Любые (не обязательно элементарные) операции называются *условными*.

Остановимся подробнее на известных алгоритмах восстановления истинных решений систем уравнений (1). Наиболее естественный из них состоит в применении метода максимума правдоподобия и сводится к перебору всех k-мерных векторов веса ρ. Известно [1, теорема 5.1], что для любых $p \in (0,1/2)$, $\rho \in \{3,4,\ldots\}$ и $m = (1+\theta_k)\log\binom{k}{\rho}/(1-h(p))$, где $h(p) = -p\log p - (1-p)\log(1-p)$, $\lim\limits_{k\to\infty}\theta_k > 0$, метод максимума правдоподобия позволяет восстанавливать истинное решение СУ (1) с вероятностью, стремящейся к 1 при $k \to \infty$. Нетрудно убедиться, что в этом случае трудоёмкость метода составляет $\mathrm{O}((1-2p)^{-2}k^{\rho}\log k)$ элементарных операций, где O зависит только от ρ.

В [2] предложен асимптотически более эффективный алгоритм, основная идея которого заключается в том, чтобы построить таблицы D и D', состоящие из векторов Ay и $Az \oplus b$ соответственно, где y и z пробегают все k-мерные булевы векторы веса $\lfloor\rho/2\rfloor$ и $\lceil\rho/2\rceil$ соответственно, и применить к этим таблицам алгоритм решения задачи о ближайшей окрестности (approximate nearest neighbor problem), описанный в [4]. Последний позволяет с некоторой высокой вероятностью найти в таблицах векторы Ay_0 и $Az_0 \oplus b$, расположенные на определённом (не слишком большом) расстоянии Хэмминга. В [2] показано, что для любой последовательности положительных чисел $\omega_1, \omega_2, \ldots$, сходящейся к $+\infty$, и произвольных p, $\delta \in (0,1/2)$, $\rho \in \{3,4,\ldots\}$ при всех достаточно больших k и $m = \lceil\rho(1-2p)^{-2}\log(k\delta^{-1})\omega_k\rceil$ истинное решение СУ (1) равно $y_0 \oplus z_0$ с вероятностью не менее $1-\delta$. При этом для нахождения этого решения требуется $\mathrm{O}((1-2p)^{-2}k^{\lceil\rho/2\rceil(1+4p^2+\alpha_k(p))}\log(k\delta^{-1})\omega_k)$ условных операций, где $\lim\limits_{k\to\infty}\alpha_k(p) = 0$ для любого $p \in (0,1/2)$, а O зависит только от ρ (отметим, что в формулировках этого результата, приведённых в [2] (см. теорему 5 и следствие 1), имеются неточности и опечатки).

Обратим внимание, что в [4] отсутствуют явные оценки надёжности алгоритма решения задачи о ближайшей окрестности, а в [2] ошибочно предполагается, что этот алгоритм всегда завершается успешно. Поэтому вопрос о том, с какой вероятностью алгоритм из [2] восстановит вектор x_0, выполняя указанное выше количество операций, остаётся открытым.

Отметим также, что алгоритм из [4] предназначен для решения задачи о ближайшей окрестности в евклидовом пространстве, а не в пространстве Хэмминга. В результате (при стандартном вложении второго пространства в первое) основная доля вычислений приходится на операции над вещественными числами. Последнее обстоятельство существенно усложняет программную реализацию алгоритма, который становится труднопримени́мым на практике, если число неизвестных k заключено в пределах от нескольких сотен до нескольких тысяч, а вероятность p является величиной порядка 10^{-3}. (Отметим, что к решению систем уравнений с такими параметрами приводит задача восстановления некоторых линейных кодов по наборам случайных кодовых слов, искажённых в двоичном симметричном канале связи [5].)

В [3] предложен другой алгоритм восстановления вектора x_0, основанный на быстром умножении некоторых целочисленных матриц, трудоёмкость которого для любых $p \in (0, 1/2)$, $\rho \in \{3, 4, \ldots\}$ и всех достаточно больших k, m (зависящих от p и ρ) ограничена сверху величиной $poly((1-2p)^{-1})k^{0,8\rho}$ условных операций, где $poly$ — некоторый полином, не зависящий от параметров. Этот алгоритм интересен тем, что в оценке его трудоёмкости отсутствует зависимость от p в показателе степени числа k. Однако при указанных выше значениях k и p алгоритм из [3] практически не превосходит по эффективности метод максимума правдоподобия. При этом для его применения требуется существенно больший объём памяти.

Целью настоящей работы является разработка алгоритма, имеющего меньшую трудоёмкость по сравнению с методом максимума правдоподобия, допускающего простую программную реализацию и позволяющего на практике восстанавливать истинные решения систем уравнений (1) от нескольких сотен или тысяч неизвестных, по крайней мере, при малых значениях p и ρ. Представленный ниже алгоритм базируется на той же идее, что и алгоритм из [2], но использует только операции сравнения и (поразрядного и арифметического) сложения двоичных целых чисел. Вместо решения задачи о ближайшей окрестности предложено проводить быстрый поиск пары близких векторов в таблицах D и D' по критерию совпадения их фрагментов в случайно выбранном множестве координат. Отметим, что эта идея используется в [6–8] и ряде других работ по теории информационного поиска. Получены оценки надёжности и трудоёмкости предложенного алгоритма. Приведены результаты вычислительных экспериментов, показывающие, что средняя трудоёмкость алгоритма заметно меньше её верхней границы в худшем случае.

1. Теоретические результаты

В дальнейшем, если не оговорено противное, символ O обозначает некоторую положительную постоянную, не зависящую от каких-либо параметров. В частности, для любых функций φ и ψ, определённых на некотором множестве X и принимающих вещественные значения, формула $\varphi(x) = \mathrm{O}(\psi(x))$, $x \in X$, означает, что существует такое положительное число C, не зависящее от x, что $|\varphi(x)| \leqslant C|\psi(x)|$ для всех $x \in X$ (см. [9, с. 12]). Если при этом множество X не указано явно, то по умолчанию оно совпадает с пересечением областей определения функций φ и ψ.

Введём следующие обозначения: V_k — множество двоичных векторов длины k; $\|x\|$ — вес вектора $x \in V_k$; $\mathrm{d}(x, y)$ — расстояние Хэмминга между векторами $x, y \in V_k$; $\overline{a, b} = \{u \in \mathbb{Z} : a \leqslant u \leqslant b\}, a, b \in \mathbb{Z}$. Положим

$$\rho_1 = \lfloor \rho/2 \rfloor, \rho_2 = \lceil \rho/2 \rceil, N = \binom{k}{\rho_1}, \widetilde{N} = \binom{k}{\rho_2},$$
$$V^{(1)} = \{y \in V_k : \|y\| = \rho_1\}, V^{(2)} = \{z \in V_k : \|z\| = \rho_2\}.$$

Для любого множества $M \subseteq \overline{1, m}$ и произвольной матрицы U, строки которой занумерованы числами $1, 2, \ldots, m$, обозначим U_M подматрицу матрицы U, содержащуюся в её строках с номерами из M.

Предлагаемый *алгоритм восстановления вектора x_0 по указанным выше исходным данным* A, b, p, ρ зависит от параметров $l \in \mathbb{N}$, $\varepsilon \in (p, 1/2)$, $t \in \mathbb{N}$, где $t < (1-\varepsilon)m$, и имеет следующий вид.

Положить Out = «НЕТ».

Для каждого $i \in \overline{1, l}$ выполнить следующие действия.

Процедура 1. Сгенерировать случайное равновероятное t-множество $\mathcal{M}_i \subseteq \overline{1, m}$; построить таблицу D_i, состоящую из упорядоченных пар $(y, A_{\mathcal{M}_i} y)$, где $y \in V^{(1)}$; с помощью одного из известных быстрых алгоритмов сортировки построить новую таблицу $D_i{}^*$, расположив пары $(y, A_{\mathcal{M}_i} y)$ в порядке неубывания t-разрядных целых чисел, соответствующих векторам $A_{\mathcal{M}_i} y$, $y \in V^{(1)}$.

Процедура 2. Для каждого $z \in V^{(2)}$

— положить $b(z) = Az \oplus b$;

— используя алгоритм бинарного поиска, проверить по таблице $D_i{}^*$, существует ли вектор $y \in V^{(1)}$, такой, что $A_{\mathcal{M}_i} y = b(z)_{\mathcal{M}_i}$; если да, то

  — найти указанный вектор y;

  — если $\mathrm{d}(Ay, b(z)) \leqslant m\varepsilon$, то положить Out $= y \oplus z$ и закончить работу.

Прокомментируем описанный алгоритм. Он состоит из l шагов, на каждом из которых выполняются процедуры 1 и 2. Результатом работы алгоритма является значение Out, равное некоторому k-мерному двоичному вектору, рассматриваемому в качестве оценки истинного решения СУ (1), либо слову «НЕТ», если эту СУ решить не удалось (хотя бы даже ошибочно).

Процедура 1 базируется на идее, предложенной в [2], и аналогична этапу предвычислений в атаках «баланс время — данные — память» [10, 11]. Множества $\mathcal{M}_i$, $i \in \overline{1, l}$, следует выбирать независимо друг от друга, случайно и равновероятно из совокупности всех подмножеств мощности t множества $\overline{1, m}$. Для их генерации можно использовать алгоритм, описанный в [12, с. 212]. Для сортировки таблицы D_i следует применять быстрые алгоритмы, трудоёмкость которых составляет $\mathrm{O}(N \log N)$ операций сравнения в худшем случае [13].

Процедура 2 состоит в поиске векторов $y \in V^{(1)}$ и $z \in V^{(2)}$, для которых расстояние Хэмминга между векторами Ay и $b(z)$ не превосходит $m\varepsilon$. Поиск ведётся до первого успеха: в случае нахождения указанных векторов алгоритм завершает работу. Для ускорения поиска используется следующий приём [6–8]: рассмотрим фрагмент $b(z)_{\mathcal{M}_i}$ вектора $b(z)$ и проверим, не встречается ли он среди вторых компонент пар векторов, записанных в таблице D_i. Алгоритм бинарного поиска (см., например, [12, с. 236]) выполняет эту проверку за время $\mathrm{O}(\log N)$ вместо $\mathrm{O}(N)$ операций сравнения, если таблица D_i отсортирована по второй компоненте. Отметим также, что алгоритм бинарного

поиска находит ровно один вектор $y \in V^{(1)}$ со свойством $A_{\mathcal{M}_i} y = b(z)_{\mathcal{M}_i}$, если такой существует. Поэтому описанный алгоритм решения СУ (1) может совершить ошибку в случае, когда вектор $b(z)_{\mathcal{M}_i}$ встречается более одного раза в наборе $(A_{\mathcal{M}_i} y : y \in V^{(1)})$, $i \in \overline{1,l}$.

Обозначим $P_{\text{ош}}$ вероятность ошибки описанного алгоритма (подчеркнём, что она определяется относительно совместного распределения независимых случайных элементов A, ξ, $\mathcal{M}_1, \ldots, \mathcal{M}_l$, распределённых по указанным выше законам).

Теорема 1. Для любых $k, l, m, t, \rho \in \mathbb{N}$, $p \in (0, 1/2)$ и $\varepsilon \in (p, 1/2)$, таких, что $3 \leqslant \rho \leqslant k-1$, $t < (1-\varepsilon)m$, описанный алгоритм выполняет

$$T = \mathrm{O}(l\widetilde{N}(t \log N + \rho m)) \tag{3}$$

элементарных операций и использует память размера

$$S = \mathrm{O}(tN \log N + km) \ \text{ бит.} \tag{4}$$

При этом справедливо неравенство

$$P_{\text{ош}} \leqslant 2^{-t} N + \exp\{-2m(\varepsilon - p)^2\} + \widetilde{N}^2 \exp\{-2m(1/2-\varepsilon)^2\} + \exp\left\{-l(1-\varepsilon-tm^{-1})^t\right\}. \tag{5}$$

Доказательство. На i-м шаге алгоритма ($i \in \overline{1,l}$) для построения таблицы D_i достаточно выполнить не более $Nt\rho_1$ элементарных операций, а для формирования таблицы D_i^* — $\mathrm{O}(Nt \log N)$ операций. Кроме того, для каждого $z \in V^{(2)}$ вычисление вектора $b(z)$, поиск в таблице D_i^* и (возможная) проверка условия $\mathrm{d}(Ay, b(z)) \leqslant m\varepsilon$ потребуют не более $t\rho_2$, $\mathrm{O}(t \log N)$ и $\mathrm{O}(\rho m)$ элементарных операций соответственно. Следовательно, $T = \mathrm{O}(l(Nt\rho_1 + Nt \log N + \widetilde{N}(t\rho_2 + t \log N + \rho m)))$, откуда вытекает справедливость формулы (3).

Далее, для хранения матрицы A и вектора b требуется $\mathrm{O}(km)$ бит памяти, а для хранения таблиц D_i и D_i^* — $\mathrm{O}(tN \log N)$ бит (включая дополнительную память, необходимую для быстрой сортировки). Следовательно, выполняется равенство (4).

Убедимся в справедливости формулы (5). Обозначим

$$\mathcal{M} = (\mathcal{M}_1, ..., \mathcal{M}_i), \mathcal{N}(\xi) = \{i \in \overline{1,l} : \xi_i = 0\}, S(x_0) = \{(y,z) \in V^{(1)} \times V^{(2)} : y \oplus z \neq x_0\}$$

и рассмотрим следующие события:

$$\Omega_1 = \bigcup_{(y,z) \in S(x_0)} \{d(Ay, Az \oplus b) \leqslant m\varepsilon\}, \Omega_2 = \{\|\xi\| \geqslant m\varepsilon\}, \Omega_3 = \bigcap_{i=1}^{l} \{\mathcal{M}_i \nsubseteq \mathcal{N}(\xi)\} \backslash \Omega_2,$$

$$\Omega_4 = \{\text{Out} = \text{«НЕТ»}\} \backslash (\Omega_1 \cup \Omega_2 \cup \Omega_3).$$

Заметим, что если алгоритм совершает ошибку, то либо наступает событие Ω_1, либо выполняется равенство Out = «НЕТ». Следовательно,

$$P_{\text{ош}} \leqslant \Pr(\Omega_1) + \Pr(\Omega_2) + \Pr(\Omega_3) + \Pr(\Omega_4). \tag{6}$$

Оценим вероятность события Ω_1. Поскольку A — случайная равновероятная матрица, а вектор ξ не зависит от A, то в силу равенства (1) для любого $(y,z) \in S(x_0)$ случайная величина $\mathrm{d}(Ay, Az \oplus b) = \|A(y \oplus z \oplus x_0) \oplus \xi\|$ распределена по закону Бернулли с параметрами $(m, 1/2)$. Отсюда на основании неравенства для вероятностей больших уклонений (см., например, [14, с. 31]) следует, что

$$\Pr\{\mathrm{d}(Ay, Az \oplus b) \leqslant m\varepsilon\} \leqslant \exp\{-2m(1/2 - \varepsilon)^2\}$$

и, значит,

$$\Pr(\Omega_1) \leqslant |V^{(1)}| \cdot |V^{(2)}| \exp\{-2m(1/2-\varepsilon)^2\} \leqslant \widetilde{N}^2 \exp\{-2m(1/2-\varepsilon)^2\}. \tag{7}$$

Аналогично, используя неравенство для вероятностей больших уклонений, получим, что

$$\Pr(\Omega_2) \leqslant \exp\{-2m(\varepsilon-p)^2\}. \tag{8}$$

Далее, в силу независимости случайных равновероятных t-множеств $\mathcal{M}_1, \ldots, \mathcal{M}_l$ и вектора ξ, а также неравенства $t \leqslant (1-\varepsilon)m$ вероятность события $\Omega_3 = \bigcap\limits_{i=l}^{l}\{\mathcal{M}_i \nsubseteq \nsubseteq \mathcal{N}(\xi)\} \cap \{|\mathcal{N}(\xi)| > (1-\varepsilon)m\}$ не превосходит величины

$$N = \left(1 - \frac{\binom{\lceil (1-\varepsilon)m \rceil}{t}}{\binom{m}{t}}\right)^l \leqslant \left(1 - \frac{((1-\varepsilon)m)_t}{(m)_t}\right)^l,$$

где $(u)_t = u(u-1)\ldots(u-t+1)$ для любого $u \geqslant t$. Используя неравенства $(u-t)^t \leqslant (u)_t \leqslant u^t$, $u > t \geqslant 1$, получим отсюда, что

$$\begin{aligned}\Pr(\Omega_3) &\leqslant \left(1 - \left(\frac{(1-\varepsilon)m - t}{m}\right)^t\right)^l \leqslant \exp\left\{-l\left(\frac{(1-\varepsilon)m-t}{m}\right)^t\right\} = \\ &= \exp\left\{-l(1-\varepsilon-tm^{-1})^t\right\}.\end{aligned} \tag{9}$$

Покажем, наконец, что

$$\Pr(\Omega_4) \leqslant 2^{-t}N. \tag{10}$$

Зафиксируем векторы $y_0 \in V^{(1)}$, $z_0 \in V^{(2)}$, такие, что $x_0 = y_0 \oplus z_0$. Обозначим $U_i(\xi, \mathcal{M})$ событие, состоящее в том, что i — наименьшее натуральное число от 1 до l со свойством $\mathcal{M}_i \subseteq \mathcal{N}(\xi)$; $V(A, \mathcal{M}_i)$ — событие, состоящее в том, что случайный вектор $A_{\mathcal{M}_i}y_0$ встречается в наборе $(A_{\mathcal{M}_i}y : y \in V^{(1)})$ не менее двух раз. Справедливо равенство

$$\Pr(\Omega_4) = \sum_{i=1}^{l} \Pr(\Omega_4 \cap U_i(\xi, \mathcal{M})). \tag{11}$$

Пусть происходит событие $\Omega_4 \cap U_i(\xi, \mathcal{M})$. Тогда $\xi_{\mathcal{M}_i} = 0$ и вектор $A_{\mathcal{M}_i}z_0 \oplus b_{\mathcal{M}_i} = A_{\mathcal{M}_i}y_0$ присутствует среди вторых компонент пар, записанных в таблице D_i. С другой стороны, поскольку алгоритм возвращает значение Out = «НЕТ», на i-м шаге в результате выполнения процедуры 2 определяется вектор $y \in V^{(1)}$, отличный от y_0. Следовательно, происходит событие $V(A, \mathcal{M}_i)$. Итак,

$$\Pr(\Omega_4 \cap U_i(\xi, \mathcal{M})) \leqslant \Pr(\Omega_4 \cap V(A, \mathcal{M}_i)), i \in \overline{1,l}. \tag{12}$$

Далее, для любого фиксированного t-множества $M \subseteq \overline{1,m}$ вероятность события $V(A, M) = V(A, \mathcal{M}_i) \cap \{\mathcal{M}_i = M\}$ не превосходит среднего числа появлений нулевого вектора в наборе $(A_M(y \oplus y_0) : y \in V^{(1)} \backslash \{y_0\})$, которое равно $2^{-t}(N-1)$. Отсюда на

основании формул (11), (12) и независимости случайных элементов A, ξ и $\mathcal{M}$ следует, что

$$\Pr(\Omega_4) \leqslant \sum_{i=l}^{l} \sum_{\substack{M \subseteq \overline{1,m}: \\ |M|=t}} \Pr(U_i(\xi, \mathcal{M}) \cap V(A, \mathcal{M}_i) \cap \{\mathcal{M}_i = M\}) =$$

$$= \sum_{i=l}^{l} \sum_{\substack{M \subseteq \overline{1,m}: \\ |M|=t}} \Pr(U_i(\xi, \mathcal{M}) \cap \{\mathcal{M}_i = M\}) \Pr(V(A, M)) \leqslant$$

$$\leqslant 2^{-t} N \sum_{i=l}^{l} \sum_{\substack{M \subseteq \overline{1,m}: \\ |M|=t}} \Pr(U_i(\xi, \mathcal{M}) \cap \{\mathcal{M}_i = M\}) = 2^{-t} N \sum_{i=l}^{l} \Pr(U_i(\xi, \mathcal{M})) \leqslant 2^{-t} N.$$

Итак, справедливо неравенство (10), что и требовалось доказать.

Из оценок (6)–(10) непосредственно следует неравенство (5). ■

Исследуем асимптотическое поведение трудоёмкости алгоритма при $k \to \infty$ для каждого набора значений (p, δ, ρ).

Теорема 2. Пусть δ, δ_1, δ_2, δ_3, δ_4 — положительные числа, такие, что

$$\delta_1 + \delta_2 + \delta_3 + \delta_4 \leqslant \delta < 1/2. \tag{13}$$

Для любых $p \in (0, 1/2)$, $\rho \in \{3, 4, \ldots\}$ и $k > \rho$ положим

$$t = \max\left\{\left\lceil \log(N{\delta_1}^{-1}) \right\rceil, \lfloor (1/2 - p)^{-1} \rfloor + 1\right\}, \varepsilon = p + t^{-1}; \tag{14}$$

$$m = \max\left\{\left\lceil 1/2 \cdot t^2 \ln({\delta_2}^{-1}) \right\rceil, \lceil 1/2 \cdot (1/2 - \varepsilon)^{-2} \ln(\widetilde{N}^2 {\delta_3}^{-1}) \rceil, \lfloor t(1-\varepsilon)^{-1} \rfloor + 1\right\}; \tag{15}$$

$$l = \left\lceil \left(1 - \varepsilon - tm^{-1}\right)^{-t} \ln({\delta_4}^{-1}) \right\rceil. \tag{16}$$

Тогда предложенный алгоритм восстанавливает истинное решение СУ (1) с вероятностью не менее $1 - \delta$, а его трудоёмкость при $k \to \infty$ составляет

$$T = \mathrm{O}\left((1-2p)^{-2} k^{\lceil \rho/2 \rceil (1 + \log\left(\frac{1}{1-p}\right))} \log^2 k^{\lceil \rho/2 \rceil}\right) \tag{17}$$

элементарных операций, где O зависит только от δ_j ($j \in \overline{1,4}$).

Доказательство. Из соотношений (14), (15) следует, что параметры ε и t удовлетворяют ограничениям, указанным в условии теоремы 1. Неравенство $P_{\text{ош}} \leqslant \delta$ вытекает непосредственно из формул (5) и (13)–(16).

Убедимся в справедливости равенства (17) (на протяжении оставшейся части доказательства символ O обозначает некоторую постоянную, зависящую разве что от параметров $\delta_j, j \in \overline{1,4}$).

Обозначим $\mu = \varepsilon + tm^{-1}$, $C = 1 + 2/\ln({\delta_2}^{-1})$ и зафиксируем число $k_0 \in \mathbb{N}$, такое, что $1/2 \cdot \log k_0 > \max\{C, 1 + (1/2 - p)^{-1}\}$. Из определения параметров N и t вытекает, что для любого $k \geqslant k_0$ справедливы следующие соотношения:

$$2(1 + (1/2 - p)^{-1}) < \log N \leqslant t = \lceil \log(N{\delta_1}^{-1}) \rceil \leqslant \log(2N{\delta_1}^{-1}). \tag{18}$$

Кроме того, согласно формуле (15), $tm^{-1} \leqslant 2/(t \ln({\delta_2}^{-1}))$, откуда следует, что для любого $k \geqslant k_0$

$$\mu = p + t^{-1} + tm^{-1} \leqslant p + t^{-1}\left(1 + \frac{2}{\ln({\delta_2}^{-1})}\right) \leqslant p + \frac{C}{\log N}.$$

Оценим сверху значение $(1-\mu)^{-t}$ при $k \geqslant k_0$. Заметим, что поскольку $p < 1/2$ и $1/2 \cdot \log N > C$, то $1-\mu \geqslant 1-p-C/\log N \geqslant (1-p)\,(1-2C/\log N) > 0$. Отсюда, используя верхнюю оценку параметра t из формулы (18), получим

$$(1-\mu)^{-t} \leqslant (1-p)^{-t}\left(1-\frac{2C}{\log N}\right)^{-t} = 2^{t\log(1-p)^{-1}}2^{t\log(1-2C/\log N)^{-1}} \leqslant$$
$$\leqslant (2N{\delta_1}^{-1})^{\log(1-p)^{-1}}\left(2N{\delta_1}^{-1}\right)^{\log(1-2C/\log N)^{-1}} \leqslant$$
$$\leqslant N^{\log(1-p)^{-1}}(2{\delta_1}^{-1})(2N{\delta_1}^{-1})^{\log(1-2C/\log N)^{-1}}.$$

Наконец, применяя известное неравенство $\ln(1-x) \geqslant -x(1-x)^{-1}$, $x \in (0,1)$, получим

$$\ln\left((2N{\delta_1}^{-1})^{\log(1-2C/\log N)^{-1}}\right) = \ln\left(\left(1-\frac{2C}{\log N}\right)^{-\log(2N{\delta_1}^{-1})}\right) \leqslant$$
$$\leqslant \log(2N{\delta_1}^{-1})\frac{2C}{\log N}\left(1-\frac{2C}{\log N}\right)^{-1} = \mathrm{O}(1), \quad k \to \infty.$$

Итак,

$$l = \mathrm{O}\left((1-\mu)^{-t}\right) = \mathrm{O}\left(N^{\log(1-p)^{-1}}\right), \quad k \to \infty. \tag{19}$$

Далее, на основании соотношений (18) для любого $k \geqslant k_0$ справедливы неравенства $t(1/2-p) > 2(1+(1/2-p)^{-1})(1/2-p) > 2$, из которых следует, что

$$1/2-\varepsilon = 1/2-p-t^{-1} = (1/2-p)(1-t^{-1}(1/2-p)^{-1}) > 1/2\cdot(1/2-p).$$

Отсюда на основании формулы (15) получаем

$$m = \mathrm{O}((1/2-p)^{-2}\log^2\widetilde{N}), \quad k \to \infty. \tag{20}$$

Кроме того, согласно формуле (14),

$$t = \log N + O(1), k \to \infty. \tag{21}$$

Подставляя выражения в правых частях равенств (19)–(21) в формулу (3) и принимая во внимание соотношения $N \leqslant \widetilde{N} \leqslant k^{\lceil\rho/2\rceil}$, $\rho\widetilde{N} \leqslant 1/2\cdot k^{\lceil\rho/2\rceil}$, $\rho \in \overline{3,k-1}$, получаем

$$T = \mathrm{O}(l\widetilde{N}(t\log N + \rho m)) = \mathrm{O}\left(\rho\widetilde{N}^{1+\log(1-p)^{-1}}(1/2-p)^{-2}\log^2\widetilde{N}\right) =$$
$$= \mathrm{O}\left((1-2p)^{-2}k^{\lceil\rho/2\rceil\left(1+\log(1-p)^{-1}\right)}\log^2 k^{\lceil\rho/2\rceil}\right), \quad k \to \infty.$$

Итак, равенство (17), а вместе с ним и теорема полностью доказаны. ■

Замечание 1. Как видно из доказательства теоремы 2, при выполнении равенств (14)–(16) и $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 0{,}25\delta$ трудоёмкость описанного алгоритма ограничена сверху полиномиальной функцией от δ^{-1} — величины, обратной к верхней границе вероятности ошибки алгоритма. Вопрос о том, обладают ли указанным свойством алгоритмы, изложенные в [2, 3], в этих публикациях не обсуждается.

Замечание 2. Трудоёмкость описанного алгоритма можно уменьшить, если при выполнении процедуры 1 хранить таблицу D_i в «распределённом» виде, записывая по адресу $A_{\mathcal{M}_i}y$ номер вектора $y \in V^{(1)}$ (в случае многократного обращения по одному и тому же адресу сохраняется последняя запись). При выполнении процедуры 2 для нахождения вектора $y \in V^{(1)}$ со свойством $A_{\mathcal{M}_i}y = b(z)_{\mathcal{M}_i}$ достаточно обратиться в память по адресу $b(z)_{\mathcal{M}_i}$, $z \in V^{(2)}$, $i \in \overline{1,l}$. Нетрудно видеть, что вероятность ошибки модифицированного таким образом алгоритма оценивается сверху по формуле (5). При этом его трудоёмкость составляет

$$\widetilde{T} = \mathrm{O}(\widetilde{N}l\rho m) \tag{22}$$

элементарных операций, а объём используемой памяти —

$$\widetilde{S} = \mathrm{O}(2^t t + km) \ \text{ бит.} \tag{23}$$

2. Вычислительные эксперименты

Описанный алгоритм (в модифицированном виде; см. замечание 2) реализован программно и применён к решению ряда систем уравнений с искажёнными правыми частями. Вычислительные эксперименты проводились на ЭВМ с процессором Intel Core i5-2450 (2,5 ГГц) и объёмом оперативной памяти 6 Гб RAM (DDR3) на базе Windows 7. При этом использовалась среда разработки Microsoft Visual C++ Studio 2008.

В табл. 1 и 2 показаны типичные результаты, полученные в двух сериях экспериментов. Слева в таблицах приведены теоретические значения параметров t, ε, l и m, $\log\widetilde{T}$, $\log\widetilde{S}$, рассчитанные при $\delta_1 = \delta_2 = \delta_3 = \delta_4 = 0{,}25\delta$ по формулам (14)–(16) и (22), (23). Справа указаны значения параметров t', ε', m', выбранные для проведения экспериментов, а также результаты, полученные в ходе их выполнения.

Таблица 1

Сравнение теоретических и экспериментальных результатов
($k = 200$, $\rho = 4$, $\delta = 0{,}1$)

p	Теория						Эксперимент					
	t	ε	l	m	$\log\widetilde{T}$	$\log\widetilde{S}$	t'	ε'	$l'_{\mathrm{ср}}$	$l'_{\max}$	m'	$T_{\mathrm{ср}}$, с
0,005	20	0,055	21	738	30,20	24,33	20	0,055	1,15	3	600	0,0056
0,008	20	0,058	22	738	30,27	24,33	20	0,058	1,20	6	600	0,0082
0,010	20	0,060	23	738	30,33	24,33	20	0,060	1,27	6	600	0,0089
0,020	20	0,070	29	738	30,67	24,33	20	0,070	1,59	12	600	0,0215
0,050	20	0,100	57	738	31,64	24,33	20	0,100	2,66	20	600	0,0604

Для каждой пары значений (p, m'), приведённых в табл. 1 и 2, 225 раз выполнялась следующая процедура: независимо друг от друга, случайно равновероятно генерировались $m' \times k$ -матрица A и вектор x_0 веса ρ, по которым формировалась СУ вида (1) с искажениями в правой части, распределёнными по закону (2). Затем к полученной СУ применялся описанный выше (модифицированный) алгоритм с параметрами l, ε' и t', значения которых приведены в таблицах. В результате выполнения алгоритма для каждой из 225 систем уравнений определялось фактическое число шагов (выбираемых множеств $\mathcal{M}$, $i \in \overline{1,l}$) до нахождения оценки истинного решения этой системы.

Таблица 2

Сравнение теоретических и экспериментальных результатов
($k = 2000$, $\rho = 4$, $\delta = 0{,}1$)

p	Теория						Эксперимент					
	t	ε	l	m	$\log \widetilde{T}$	$\log \widetilde{S}$	t'	ε'	$l'_{\text{ср}}$	$l'_{\max}$	m'	$T_{\text{ср}}$, с
0,005	27	0,042	21	1345	37,72	31,76	26	0,042	1,17	4	1000	19,516
0,008	27	0,045	23	1345	37,85	31,76	26	0,045	1,20	4	1000	21,312
0,010	27	0,047	25	1345	37,97	31,76	26	0,047	1,39	7	1000	30,572
0,020	27	0,057	33	1345	38,39	31,76	26	0,057	1,56	5	1000	43,866
0,050	27	0,087	79	1345	39,63	31,76	26	0,087	4,15	31	1000	153,721

Параметры $l'_{\text{ср}}$ и $l'_{\max}$ в таблицах равны соответственно среднему арифметическому и наибольшему из этих чисел. В последней колонке табл. 1 и 2 приведено среднее время выполнения алгоритма в секундах (при этом надёжность восстановления истинных решений сгененированных СУ составляет 100 %).

Как видно из таблиц, значения $l'_{\text{ср}}$ и $l'_{\max}$ заметно меньше теоретической верхней границы l количества шагов, выполняемых до нахождения оценки истинного решения СУ. Например, согласно табл. 2, истинное решение веса 4 системы из 1000 уравнений от 2000 неизвестных с вероятностью искажения в правой части $p = 0{,}050$ находится в среднем не более чем на пятом (а в худшем случае — на тридцать первом) шаге со 100 %-й надёжностью, в то время как теоретическое значение числа шагов $l = 79$ (при большем количестве уравнений $m = 1345$ и нижней границе надёжности алгоритма 90 %). Среднее время восстановления решения такой СУ составляет около 2,5 мин.

Экспериментально установлено, что параметры алгоритма могут быть выбраны в широком диапазоне в зависимости от ограничений относительно трудоёмкости и количества уравнений в системе. При этом на практике значения указанных параметров заметно меньше теоретических (при той же нижней границе надёжности).

В табл. 3–5 приведены результаты сравнения предложенного алгоритма с методом максимума правдоподобия. Данные в табл. 3 получены следующим образом. Сначала экспериментальным путём для каждого значения p, указанного в таблице, были получены оценки $m_1(p)$ и $m_2(p)$ наименьшего числа уравнений в (случайно сгенерированной) системе (1), при котором вероятность правильного восстановления её истинного решения с помощью метода максимума правдоподобия и соответственно предложенного алгоритма составляет не менее 90 %. Затем для каждого p 225 раз выполнялась процедура, в ходе которой к случайно сгенерированной СУ (1), состоящей из $m_2(p)$ уравнений, применялся предложенный алгоритм. Потом из этой СУ удалялись последние $m_2(p) - m_1(p)$ уравнений и полученная система решалась методом максимума правдоподобия. Данные в табл. 4 и 5 получены аналогично, с тем отличием, что для каждого исходного значения p вместо 225 запусков процедуры выполнялось только 10. Последнее обстоятельство обусловлено заметным увеличением времени решения СУ методом максимума правдоподобия.

Таблица 3

Сравнение алгоритмов решения систем с искажёнными правыми частями ($k = 128$, $\rho = 4$)

p	Метод максимума правдоподобия		Предложенный алгоритм	
	Среднее время выполнения, с	Количество уравнений, $m_1(p)$	Среднее время выполнения, с	Количество уравнений, $m_2(p)$
0,005	3,6795	40	0,0008	56
0,008	3,6923	40	0,0027	56
0,010	3,8418	40	0,0072	56
0,020	3,9712	48	0,0113	160
0,050	5,8443	72	0,0453	200

Таблица 4

Сравнение алгоритмов решения систем с искажёнными правыми частями ($k = 128$, $\rho = 5$)

p	Метод максимума правдоподобия		Предложенный алгоритм	
	Среднее время выполнения, с	Количество уравнений, $m_1(p)$	Среднее время выполнения, с	Количество уравнений, $m_2(p)$
0,005	94,48	40	0,0732	56
0,008	99,73	40	0,0899	56
0,010	105,96	40	0,0995	56
0,020	106,16	48	0,3466	72
0,050	136,60	72	1,4910	160

Таблица 5

Сравнение алгоритмов решения систем с искажёнными правыми частями ($k = 128$, $\rho = 6$)

p	Метод максимума правдоподобия		Предложенный алгоритм	
	Среднее время выполнения, с	Количество уравнений, $m_1(p)$	Среднее время выполнения, с	Количество уравнений, $m_2(p)$
0,005	2056,94	40	0,4013	56
0,008	2154,86	40	0,4551	64
0,010	2138,73	40	0,4278	64
0,020	2197,63	48	0,7418	120
0,050	3691,22	72	2,7181	200

Как видно из таблиц, предложенный алгоритм в сотни (а в ряде случаев — в тысячи) раз превосходит по эффективности метод максимума правдоподобия, причём с ростом значения ρ это различие становится всё более заметным.

В целом, предложенный алгоритм представляется более простым и эффективным с точки зрения программной реализации по сравнению с алгоритмами, описанными в [2, 3], и может быть применён на практике для восстановления истинных решений малого веса слабоискажённых систем линейных булевых уравнений от нескольких сотен или тысяч неизвестных.

Авторы благодарны рецензенту за критические замечания, учёт которых позволил существенно улучшить качество работы, избежать неточностей в формулировках и усилить первоначальную версию теоремы 2.

ЛИТЕРАТУРА

1. *Балакин Г. В.* Введение в теорию случайных систем уравнений // Труды по дискретной математике. М.: ТВП, 1997. Т. 1. С. 1–18.
2. *Grigorescu E., Reysin L., and Vempala S.* On noise-tolerant learning of sparse parities and related problems // Proc. 22nd Intern. Conf. of Algorithmic Learning Theory. Berlin, Heidelberg: Springer Verlag, 2011. P. 413–424.
3. *Valiant G.* Finding correlations in subquadratic time, with applications to learning parities and juntas // Proc. 53rd Annual IEEE Symp. on the Foundations of Computer Science, New Brunswick, New Jersey, October 2–23, 2012. P. 11–20.
4. *Andoni A. and Indyk P.* Near-optimal hashing algorithms for approximate nearest neighbor in high dimension // Proc. 47th Symp. on Foundations of Computer Science, Berkeley, California, October 21–24, 2006. P. 459–468.
5. *Алексейчук А. Н., Грязнухин А. Ю.* Метод восстановления систематических линейных кодов по наборам искаженных кодовых слов // Прикладная радиоэлектроника. 2013. Т. 12. № 2. С. 128–132.
6. *Greene D., Parnas M., and Yao F.* Multi-index hashing for information retrieval // Proc. 35th Annual IEEE Symp. on the Foundations of Computer Science, Santa Fe, New Mexico, November 20–22, 1994. P. 722–731.
7. *Dolev D., Harari Y., and Parnas M.* Finding the neighborhood of a query in a dictionary // Proc. 2nd Israel Symp. on Theory and Computing Systems, Natanya, Israel, June 7–9, 1993. P. 33–42.
8. *Indyk P. and Motwani R.* Approximate nearest neighbors: towards removing the curse of dimensionality // Proc. Symp. on Theory of Computing. New York, NY, USA: ACM, 1998. P. 604–613.
9. *Де Брёйн Н. Г.* Асимптотические методы в анализе: пер. с англ. М.: ИЛ, 1961. 247 с.
10. *Babbage S. H.* Improved “exhaustive search” attacks on stream ciphers // European Convention on Security and Detection, Brighton, May 16–18, 1995. P. 161–166.
11. *Golić J.* Cryptanalysis of alleged A5 stream cipher // Advances in Cryptology. Proc. EUROCRYPT’97. Springer Verlag, 1997. P. 239–255.
12. *Рейнгольд Э., Нивергельт Ю., Део Н.* Комбинаторные алгоритмы. Теория и практика: пер. с англ. М.: Мир, 1980. 476 с.
13. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов: пер. с англ. М.: Мир, 1979. 535 с.
14. *Ширяев А. Н.* Вероятность: учеб. пособие для вузов. М.: Наука, 1989. 640 с.