М. А. Куприяшин, Г. И. Борзунов

АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ О РЮКЗАКЕ, ОСНОВАННЫЙ НА ОБХОДЕ ДЕРЕВА ВАРИАНТОВ УКЛАДКИ

Пусть существует набор вещей, каждая из которых имеет определенный положительный вес. Требуется найти способ укладки части этих вещей в рюкзак таким образом, чтобы вес рюкзака равнялся наперед заданному значению ω. Упорядоченное множество весов вещей называется рюкзачным вектором $\vec{a} = (a_1; a_2; \ldots a_n)$. Бинарный вектор $\vec{v}$ длины n задается таким образом, что при $v_i = 1$ предмет с номером i помещается в рюкзак, а при $v_i = 0$ этот предмет в состав рюкзака не входит. Используя эти обозначения, можно сформулировать задачу о рюкзаке: найти все возможные значения вектора $\vec{v}$, при которых выполняется: $\sum_{i=1}^{n} a_i v_i = \omega$. Задачи о рюкзаке небольшой размерности легко решаются перебором всех возможных вариантов. Однако с ростом длины рюкзачного вектора резко увеличивается сложность задачи. Задача о рюкзаке (в общем случае) считается NP-полной, решающий ее алгоритм имеет экспоненциальную временную сложность.

Одна из первых предложенных асимметричных криптосистем — криптосистема Меркля—Хеллмана [1], а также многие другие криптосистемы, основанные на решении задачи о рюкзаке, оказались нестойкими [2]. Тем не менее на сегодняшний день существуют криптосистемы на основе этой задачи, уязвимые места которых не найдены. Более того, в последние годы был предложен ряд новых систем [3], что привело к возрастанию интереса к анализу задачи о рюкзаке и ее разновидностей. В работе [4] предлагается ряд новых систем защиты информации, основанных на предложенных автором разновидностях задачи о рюкзаке, а также активно прорабатывается теория асимметричных рюкзачных систем защиты информации. Работа [5] посвящена, в частности, разработке таких рюкзачных преобразований, для которых соответствующие обратные преобразования требуют решения нескольких NP-полных задач и, таким образом, формирует надежную теоретическую основу для создания нового поколения асимметричных криптосистем, основанных на задаче о рюкзаке. Приведенные выше сведения показывают, что работы, направленные на исследования и совершенствование алгоритмов решения задачи о рюкзаке, следует считать актуальными.

В ходе изучения задачи были рассмотрены различные базовые алгоритмы ее решения. Самый простой алгоритм основывается на конструктивном перечислении и проверке всех возможных вариантов. Существует 2^n возможных значений. Проверка каждого потребует n операций умножения, $n-1$ операций сложения и 1 операцию сравнения полученного веса с заданным. Временная сложность, таким образом, составляет $T = 2^n \cdot 2n$ операций. Сокращение временной сложности решения задачи до $O(2^{n/2})$ с помощью предварительной обработки входных данных было предложено в 1974 г. Хоровицем и Сани [6]. Для этого рюкзачный вектор разбивается на два вектора, и получаются две задачи о рюкзаке, размерность которых вдвое меньше исходной. Для каждой из этих двух задач рассчитывается таблица соответствия между всеми наборами предметов и соответствующими весами рюкзака (так же, как и при полном переборе, только все значения сохраняются). Полученные списки значений сортируются и подвергаются слиянию — сложению всех возможных пар весов из первой и второй подзадач соответственно. Если какой-либо из суммарных весов совпадает с требуемым значением, значит, слияние соответствующих векторов из первой и второй подзадач является решением. Для ускорения процесса слияния один из списков просматривается в порядке возрастания, другой — в порядке убывания. Если величина суммы превышает требуемую величину, доводить итерацию слияния до конца бессмысленно [7].

Пример для $\omega = 12;\ \vec{a} = (9;7;5;3)$ приведен в таблице 1.

Таблица 1. Пример исполнения алгоритма Хоровица—Сани

Подзадача № 1	Подзадача № 2
$\vec{a} = (9;7)$ $\vec{v} = (0;0) \to \sum a_i v_i = 9*0+7*0 = 0$ $\vec{v} = (0;1) \to \sum a_i v_i = 9*0+7*1 = 7$ $\vec{v} = (1;0) \to \sum a_i v_i = 9*1+7*0 = 9$ $\vec{v} = (1;1) \to \sum a_i v_i = 9*1+7*1 = 16$	$\vec{a} = (5;3)$ $\vec{v} = (0;0) \to \sum a_i v_i = 5*0+3*0 = 0$ $\vec{v} = (0;1) \to \sum a_i v_i = 5*0+3*1 = 3$ $\vec{v} = (1;0) \to \sum a_i v_i = 5*1+3*0 = 5$ $\vec{v} = (1;1) \to \sum a_i v_i = 5*1+3*1 = 8$
(16;9;7;0)	*(8;5;3;0)*
Слияние списков: $16+0 = 16 > 12 \Rightarrow$ *итерация отменяется (16+3; 16+5; 16+8 — не рассчитывается)* $9+0 = 9 < 12$; $9+3 = 12 = 12 \Rightarrow$ *слияние векторов, соответствующих суммам 9 и 12, есть решение. 9 соответствует* $\vec{v} = (1;0)$, *3 соответствует* $\vec{v} = (0;1)$. *Таким образом, решением является* $\vec{v} = (1;0;0;1)$. *Продолжать данную итерацию бессмысленно.* $7+0 = 7 < 12;$ $7+3 = 10 < 12;$ $7+5 = 12 = 12 \Rightarrow$ *слияние векторов, соответствующих суммам 7 и 5, есть решение. 7 соответствует* $\vec{v} = (0;1)$, *5 соответствует* $\vec{v} = (1;0)$. *Таким образом, решением является* $\vec{v} = (0;1;1;0)$. *Продолжать данную итерацию бессмысленно.* $0+0 = 0 < 12;$ $0+3 = 3 < 12;$ $0+5 = 5 < 12;$ $0+8 = 8 < 12;$	

Для сокращения временной сложности базового алгоритма решения задачи о рюкзаке нами предложена вычислительная схема, основанная на использовании графа укладок. Ниже впервые описывается алгоритм, реализующий эту вычислительную схему.

Анализ получаемых решений позволяет сократить сложность решения задачи: например, если вектор $\vec{v}_k$ является решением задачи о рюкзаке, можно не рассматривать вектора большего веса $\vec{v}_z$, для которых выполнено $\vec{v}_{k_i} = 1 \Rightarrow \vec{v}_{z_i} = 1; i = \overline{1,n}$. Представим множество всех возможных решений задачи в виде графа (дерева) укладок $G(\vec{v})$. Определим граф $G(\vec{v})$, в качестве вершин которого выступают все возможные значения вектора $\vec{v}$. Ребра предлагается выбирать следующим образом: вершина $00\ldots0$ полагается корнем дерева, а вершина $\vec{v}_z$ включается в окрестность

$$\vec{v}_k \Leftrightarrow \begin{cases} \vec{v}_{z_i} = \vec{v}_{k_i}; i = \overline{1,l} \\ v_{z_j} = 0; j = \overline{(l+1),(m-1)}; \overline{(m+1);n} \\ v_{z_m} = 1 \end{cases}, \text{ где } l: \begin{cases} \vec{v}_{k_l} = 1 \\ \vec{v}_{k_p} = 0; p = \overline{l+1,n} \end{cases},$$

а m — любое число, такое, что $l < m \le n$. Другими словами, вершины следующего яруса дерева получаются путем дописывания «1» в вектор $\vec{v}$ справа. Например, между вершинами 1010 и 1011 имеется ребро, а между 1010 и 1110 — нет. Вершина 1000 связана с 1100, 1010 и 1001, а вершина 0001 является листом. Для рюкзачного вектора размерности 4 описанная структура может быть представлена в следующем виде (см. рис. 1).

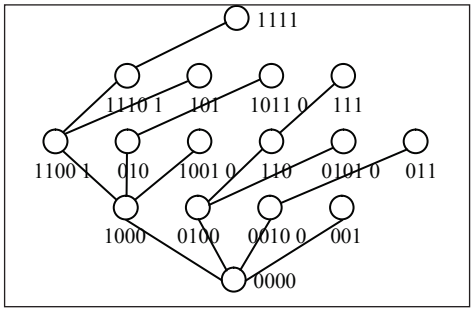

Рис. 1. Пример структуры дерева укладок

По мере удаления от корня дерева вес рюкзака может только увеличиваться. Это означает, что, если для определенной вершины вес рюкзака превысит ω, дальнейшее движение «в глубь» дерева заведомо не принесет решений. Это позволяет существенно сократить количество рассматриваемых решений при не слишком больших значениях ω. Для поиска решений можно воспользоваться алгоритмом обхода графа в глубину. Пример: для $\omega = 12;\ \vec{a} = (9;7;5;3)$ приведен в таблице 2, соответствующая структура дерева поиска решений отражена на рис. 2.

Таблица 2. Пример исполнения алгоритма обхода дерева графа укладок

1. Обход дерева начинается в корневой вершине 0000. Так как $\omega > 0$ (в противном случае решение тривиально), корневая вершина не является решением. В стек заносится ее окрестность: (1000; 0100; 0010; 0001).
2. Из стека извлекается первая вершина 1000. В ней вес рюкзака составляет $9+0+0+0=9<\omega$. Заносится в стек окрестность этой вершины: (1100; 1010; 1001; 0100; 0010; 0001).
3. Из стека извлекается вершина 1100. В ней вес рюкзака составляет $9+7+0+0=16>\omega$. Это означает, что данная вершина, равно как и последующие, не является решением задачи. Соответственно, нет нужды описывать ее окрестность. Содержимое стека не обновляется: (1010; 1001; 0100; 0010; 0001).
4. Из стека извлекается вершина 1010. В ней вес рюкзака составляет $9+0+5+0=14>\omega$. Это означает, что данная вершина, равно как и последующие, не является решением задачи. Соответственно, нет нужды описывать ее окрестность. Содержимое стека не обновляется: (1001; 0100; 0010; 0001).
5. Из стека извлекается вершина 1001. В ней вес рюкзака составляет $9+0+0+3=12=\omega$. Это означает, что данная вершина является решением. В последующих вершинах значение ω будет больше. Соответственно, нет нужды описывать окрестность. Содержимое стека не обновляется: (0100; 0010; 0001).
6. Из стека извлекается первая вершина 0100. В ней вес рюкзака составляет $0+7+0+0=7<\omega$. Заносится в стек окрестность этой вершины: (0110; 0101; 0010; 0001).
7. Из стека извлекается вершина 0110. В ней вес рюкзака составляет $0+7+5+0=12=\omega$. Это означает, что данная вершина является решением. В последующих вершинах значение ω будет больше. Соответственно, нет нужды описывать окрестность. Содержимое стека не обновляется: (0101; 0010; 0001).

8. Из стека извлекается первая вершина 0101. В ней вес рюкзака составляет $0+7+0+3=10<\omega$. *Окрестность этой вершины пуста.*
9. Из стека извлекается первая вершина (0;0;1;0). В ней вес рюкзака составляет $0+0+5+0=5<\omega$. *Заносится в стек окрестность этой вершины:(0011;0001)*
10. Из стека извлекается первая вершина 0011. В ней вес рюкзака составляет $0+0+5+3=8<\omega$. *Окрестность этой вершины пуста.*
11. Из стека извлекается первая вершина 0011. В ней вес рюкзака составляет $0+0+0+3=3<\omega$. *Окрестность этой вершины пуста.*
12.Стек пуст, обход закончен, найдены решения: 1001; 0110.

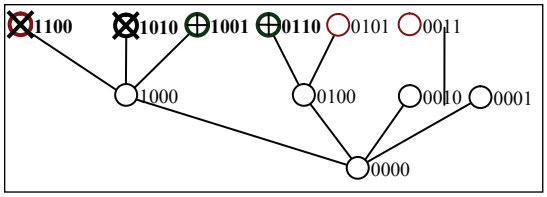

Рис. 2. Структуры дерева укладок для вычислительного примера

Знаком «X» обозначены вершины, в которых вес рюкзака превысил ω. Знаком «+» обозначены вершины, оказавшиеся решениями. Всего потребовалось обойти 11 вершин из 16. Таким образом, временная сложность решения задачи о рюкзаке понижается на 31 %, что обеспечивает коэффициент ускорения 1,45 по сравнению с методом полного перебора.

Предложенный подход к решению задачи о рюкзаке имеет и другие преимущества. В частности, поиск решения может осуществляться с использованием параллельных вычислений: каждая ветка дерева — независимая подзадача.

СПИСОК ЛИТЕРАТУРЫ:

1. *Merkle R., Hellman M.* Hiding Information and Signatures in Trapdoor Knapsacks // IEEE Transactions on Information Theory. Vol. IT-24. M., 1978. С. 525–530.
2. *Шнайер Б.* Прикладная криптография. М.: Триумф, 2012. – 815 с.
3. *Куприяшин М. А., Борзунов Г. И.* Анализ состояния работ, направленных на повышение стойкости шифрсистем на основе задачи о рюкзаке // Безопасность информационных технологий. 2013. № 1. С. 111–112.
4. *Осипян В.О.* Разработка математических моделей систем передачи и защиты информации. М., 2006. – 371 с.
5. *Подколзин В. В.* Моделирование систем на основе односторонних рюкзачных отображений. АКД. М., 2011. – 24 с.
6. *Horowitz E., Sahni S.* Computing Partition with Applications to the Knapsack Problem // Journal of the ACM. 1974. № 21. С. 277–292.
7. *Woeginger G. J.* Exact Algorithms for NP-Hard Problems: A Survey // Combinatorial Optimization / M. Jünger et al. Springer-Verlag Berlin, 2003. P. 185–207.

REFERENCES:

1. *Merkle R., Hellman M.* Hiding Information and Signatures in Trapdoor Knapsacks // IEEE Transactions on Information Theory. Vol. IT-24. M., 1978. P. 525–530.
2. *Schneier B.* Prikladnaya cryptography. M.: Triumf, 2012. – 815 p.
3. *Kupriyashin M. A., Borzunov G. I.* Analiz sostoyaniya rabot, napravlennyh na povyshenie stojkosti shifrsistem na osnove zadachi o ryukzake // bezopasnost informatsionnyh tehnologiy. 2013. № 1. P. 111–112.
4. *Osipyan V. O.* Razrabotka matematicheskih modelej sistem peredachi i zashity informacii. M., 2006. – 371 p.
5. *Podkolzin V. V.* Modelirovanie sistem na osnove odnostoronnih ryukzachniykh otobrazenii. AKD. M., 2011. – 24 p.
6. *Horowitz E., Sahni S.* Computing Partition with Applications to the Knapsack Problem // Journal of the ACM. 1974. № 21. P. 277–292.
7. *Woeginger G. J.* Exact Algorithms for NP-Hard Problems: A Survey // Combinatorial Optimization / M. Jünger et al. Springer-Verlag Berlin, 2003. P. 185–207.