УДК 658.52.011.56

С.А.КАНЦЕДАЛ, М.В.КОСТИКОВА

ОДИН АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ О ПОКРЫТИИ

Для решения задач о покрытии среднего размера предлагается простой алгоритм, реализующий ограниченный перебор сочетаний покрывающих объектов. Приводится способ вычислений предельного количества основных операций, выполняемых алгоритмом.

1. Постановка задачи

Рассмотрим самую общую постановку задачи о покрытии. Для конечного множества $M = \{m_1, m_2, ..., m_m\}$ и некоторого конечного семейства его подмножеств S_j, $j = 1, 2, ..., n$ требуется найти подсемейство $S_j' \subseteq S_j$ с минимальным набором подмножеств S_j такое, что каждый элемент исходного множества M принадлежит, по крайней мере, одному из этих подмножеств. Найденные подмножества S_j' обычно называют покрывающими, а семейство S_j' – наименьшим покрытием M.

В том случае, когда покрывающим элементам приписаны некоторые веса, рассматривают взвешенную задачу о покрытии, предполагающую поиск наименьшего взвешенного покрытия M. Когда же покрывающие подмножества попарно не пересекаются, задача о наименьшем покрытии трактуется как задача о разбиении множества M на наименьшее количество классов.

Цель исследования – разработка нового алгоритма задачи о покрытии.

Задача для достижения цели – обзор существующих методов решения такой задачи.

Приведем простейший пример взвешенной задачи о покрытии. Некоторой организации требуется нанять переводчиков с ряда языков на английский. Имеется пять кандидатур таких переводчиков A, B, C, D, E, делающих переводы только с части языков и требующих свою оплату труда.

Необходимо определить, каких переводчиков следует взять на работу, чтобы обеспечить возможность перевода со всех языков на английский и при этом гарантировать наименьшие расходы организации.

Исходные данные задачи представлены двумя таблицами (табл. 1, 2), в первой из которых в соответствующих ее клетках возможности переводчиков, т.е. перевода ими с приведенного перечня языков на английский, отмечены единицами, невозможности – нулями. Очевидно, что множество M, для которого требуется найти наименьшее взвешенное покрытие, образует перечень языков. Кандидатуры переводчиков представляют собой семейство подмножеств $S_j = \{A, B, C, D, E\}$, подсемейство $S_j' \subseteq S_j$ – перечень переводчи-

ков, обеспечивающих возможность перевода, т.е. покрытия M с минимальными затратами.

По существу первая таблица без верхней строки и крайнего левого столбца представляет собой нуль-единичную матрицу. Строки матрицы интерпретируются как языки, столбцы – как переводчики. Поэтому задачу можно толковать как поиск такого множества столбцов матрицы, что в целом они обеспечивают наличие единиц во всех ее строках, образно говоря, покрывают строки этой матрицы и приводят к наименьшим расходам.

Таблица 1. Возможность переводчика переводить на английский язык с указанного языка

Переводчики / Язык	A	B	C	D	E
Французский	1	0	1	1	0
Немецкий	0	1	0	0	1
Греческий	0	1	0	0	0
Итальянский	1	0	0	1	0
Испанский	0	0	1	0	0
Русский	0	1	1	0	1
Китайский	1	0	0	1	1

Таблица 2. Оплата труда переводчиков

Переводчики	A	B	C	D	E
Оплата труда	35000	40000	38000	45000	47000

Другим примером является задача о развозке грузов. Она моделируется так, что в целом граф G представляет сеть дорог. Одна из вершин графа представляет склад, а остальные его вершины – торговые точки (магазины, кафе и т.п.). Транспортное средство (автомобиль), покидающее склад, снабжает товарами некоторые из этих точек, получающие весь необходимый ассортимент. После этого автомобиль возвращается на склад.

Заранее с учетом практических ограничений проезда выбрано некоторое число маршрутов $1, 2, ..., j, ..., n$ со стоимостями $C_1, C_2, ..., C_j, ..., C_n$.

Требуется определить, сколько машин следует использовать на маршрутах, чтобы в течение дня доставить все грузы со склада потребителям и обеспечить наименьшую суммарную стоимость всех замкнутых маршрутов.

В этой задаче исходное множество M – это перечень торговых точек, семейство подмножеств S_j – перечень маршрутов, $S_j^{'} \subseteq S_j$ – подсемейство маршрутов, обеспечивающих минимальное покрытие M.

Как и выше, получаем нуль-единичную матрицу, строки которой $1, 2, ..., i, ..., m$ интерпретируются как торговые точки, а столбцы – как маршруты. Наличие единицы на пересечении некоторой строки и столбца в этой матрице означает, что данная торговая точка включена в соответствующий маршрут. Наличие нуля отображает тот факт, что она не входит в рассматриваемый маршрут.

Таким образом, задача сводится к тому, чтобы в указанной матрице найти то количество столбцов, которые покрывают строки матрицы и минимизируют расходы на перевозки.

Чисто формально задача о наименьшем покрытии легко представима в терминах целочисленного линейного программирования. Исходим из того, что задана нуль-единичная матрица $P = \|P_{ij}\|$, строки которой $i = 1, 2, ..., m$ – это индексы элементов исходного множества M, столбцы $j = 1, 2, ..., n$ – индексы элементов семейства S, а элементы P_{ij} равны единице или нулю в зависимости от того, покрывает j-й столбец i-ю строку или нет. Введем булевы переменные $x_j, j = 1, 2, ..., n$ такие, что $x_j = 1$, если j-й столбец в искомом покрытии входит в него и $x_j = 0$ в противном случае. С переменными x_j свяжем веса

c_j (в рассматриваемом примере оплаты переводчиков). В результате получаем задачу: минимизировать функцию

$$Z = \sum_{j=1}^{n} c_j x_j$$

при условиях

$$P_{ij} = \begin{cases} 1, & \text{если } j\text{ - й столбец покрывает } i\text{ - ю строку,} \\ 0 & \text{в противном случае,} \end{cases}$$

$$x_j = \begin{cases} 1, & \text{если } j\text{ - й столбец входит в покрытие,} \\ 0 & \text{в противном случае,} \end{cases}$$

$$\sum_{j=1}^{n} P_{ij} x_j \geq 1, \; i = 1, 2, ..., m.$$

При этом последнее неравенство гарантирует покрытие строк матрицы P элементами x_j.

2. Метод решения задачи

В связи с тем, что задача целочисленного линейного программирования NP-полная, задача о наименьшем покрытии также обладает этим свойством, т.е. точно может быть решена только алгоритмами экспоненциального характера.

Первыми методами решения задачи были методы отсечения Гомори, среди которых наилучшим оказался усовершенствованный его алгоритм – алгоритм Мартина [1]. Позже для решения задачи были предложены несколько модификаций метода ветвей и границ [2, 3].

Для задач относительно небольшого размера ($m \leq 30$, $n \leq 20$) мы предлагаем простейший алгоритм ее решения – последовательной генерации и проверки сочетаний. Его действия продемонстрируем на задаче набора переводчиков.

На первом шаге сформируем все варианты пар переводчиков (A, B), (A, C), (A, D), (A, E), (B, C), (B, D), (B, E), (C, D), (C, E), (D, E) и для каждой пары установим возможность покрытия строк исходной матрицы. Если такая возможность существует больше, чем для одной пары, выбираем ту из них, которая определит наименьшие затраты и завершим поиск. В противном случае, т.е. когда покрытия парами нет, сформируем последовательность троек переводчиков (A, B, C), (A, B, D), (A, B, E), (A, C, D), (A, C, E), (A, D, E), (B, C, D), (B, C, E), (B, D, E), (C, D, E), определим возможность покрытия, в зависимости от чего завершим решение задачи или продолжим формирование состава переводчиков. Описанные действия продолжим до обнаружения первого покрывающего их сочетания. Если такого сочетания нет, задача не имеет решения. Заметим, что проверка возможности покрытия выполняется достаточно просто. В каждой строке матрицы суммируются элементы столбцов, образующих сочетание, и при обнаружении первой нулевой суммы в некоторой ее строке делается вывод о том, что данное сочетание не обеспечивает покрытие. В противном случае покрытие существует.

Применительно к данной задаче убеждаемся, что покрытие обеспечивает две тройки переводчиков: (A, B, C) и (B, C, D). Для первой тройки расходы равны $35000 + 40000 + 38000 = 113000$ денежных единиц, для второй – $40000 + 38000 + 45000 = 123000$ таких единиц. Следовательно, искомым решением является прием на работу переводчиков A, B, C.

3. Оценка трудоемкости метода

Оценим сверху трудоемкость алгоритма, который легко может быть построен путем реализации описанных действий. Для этого подсчитаем количество основных операций – сложений и сравнений, – которые он будет выполнять для установления того факта, имеет ли задача решение или нет.

В связи с тем, что число вариантов двоек, троек, $(n-1)$-к столбцов матрицы P равно числу сочетаний C_n^2, C_n^3, C_n^{n-1}, всего для выбора получим $C_n^2 + C_n^3 + ... + C_n^{n-1}$ вариантов

набора ее столбцов. В свою очередь для проверки любого варианта на покрытие в худшем случае требуется выполнить m сравнений и некоторое количество сложений, зависящее от числа элементов в сочетании. Так, для вариантов, образуемых парами столбцов матрицы P, для каждой ее строки необходимо выполнить одно сложение и одно сравнение для выявления нуля, т.е. в целом для матрицы $2m$ операций. Варианты, образуемые тройками столбцов, для каждой строки матрицы P требуют выполнить два сложения и одно сравнение, т.е. всего $3m$ операций. Варианты с $(n-1)$ столбцами требуют выполнить $n-2$ сложения и одно сравнение, т.е. $(n-1)$-у операцию, в каждой строке P, а в целом для матрицы P $-(n-1)m$ операцию. Наконец, последний вариант с n столбцами в каждой строке P требует выполнить $n-1$ сложение и одно сравнение, т.е. в целом для матрицы $n \cdot m$ операцию.

Таким образом, предельное количество основных операций, выраженное через число сочетаний, следующее $\left[2C_n^2 + 3C_n^3 + ... + (n-1)C_n^{n-1} + n\right]m$. Окончательно для получения верхней оценки числа операций, выполняемых алгоритмом, необходимо элементы приведенного выражения умножить на число операций, которое требуется выполнить для генерации одного сочетания. Согласно [4], где приводится два алгоритма построения множеств сочетаний, генерация одного сочетания потребует то число операций, которое не больше, чем значение верхнего индекса символа сочетания операций. В результате окончательно для верхней оценки получим $W = \left[4C_n^2 + 9C_n^3 + ... + (n-1)^2 C_n^{n-1} + n\right]m$.

Безусловно, полученная оценка очень завышена, так как определяет то число операций, которое нужно выполнить для установления факта, существует ли решение или нет. В действительности число операций должно быть существенно меньше, поскольку предполагается, что часть задач будут иметь решения. К сожалению, найти аналитическое выражение для вычисления такого числа операций не представляется возможным, так как нельзя предсказать, сколько столбцов матрицы P определит решение. Единственная возможность установить среднее и максимальное число этих операций, а также их рост в зависимости от размера задачи – набор компьютерной статистики, ее обработка и аппроксимация полученных данных подходящей эмпирической кривой. Прикидочный расчет числа операций по приведенной формуле для задачи $m = 20, n = 10$ даст

$$W = \left(4 \cdot \frac{10!}{2!\cdot 8!} + 9 \cdot \frac{10!}{3!\cdot 7!} + 16 \cdot \frac{10!}{4!\cdot 6!} + 25 \cdot \frac{10!}{5!\cdot 5!} + 36 \cdot \frac{10!}{6!\cdot 4!} + 49 \cdot \frac{10!}{7!\cdot 3!} + 64 \cdot \frac{10!}{8!\cdot 2!} + 81 \cdot \frac{10!}{9!} + 10\right) \times$$

$$\times 20 = (180 + 1080 + 3360 + 6300 + 7560 + 5880 + 2880 + 810 + 10) \cdot 20 = 28060 \cdot 20 = 561200$$

операций, что вполне приемлемо для решения задач среднего размера на современных компьютерах.

В заключение решим предложенным алгоритмом задачу о наименьшем взвешенном покрытии, исходные данные (P, C) которой приведены ниже.

	1	2	3	4	5	6
1	1	1	1	0	0	1
2	1	0	1	0	0	1
3	0	0	0	1	0	0
4	0	1	0	0	1	0
5	0	0	0	0	1	1
6	1	1	0	0	0	0

$P =$

$C =$ 4 7 5 8 3 2

Составим сочетания для всех пар столбцов матрицы P и, если пара покрывает ее строки, отметим ее знаком плюс (+), в противном случае – знаком минус (-). Имеем $(1,2)^-$, $(1,3)^-$, $(1,4)^-$, $(1,5)^-$, $(1,6)^-$, $(2,3)^-$, $(2,4)^-$, $(2,5)^-$, $(2,6)^-$, $(3,4)^-$, $(3,5)^-$, $(3,6)^-$, $(4,5)^-$, $(4,6)^-$, $(5,6)^-$. В связи с тем, что пары индексов столбцов матрицы P не образуют покрытия, переходим к формированию сочетаний из трех индексов P. В результате получаем $(1,2,3)^-$, $(1,2,4)^-$, $(1,2,5)^-$, $(1,2,6)^-$, $(1,3,4)^-$, $(1,3,5)^-$, $(1,3,6)^-$, $(1,4,5)^+$, $(1,4,6)^-$, $(1,5,6)^-$, $(2,3,4)^-$, $(2,3,5)^-$, $(2,3,6)^-$, $(2,4,5)^-$, $(2,4,6)^+$, $(2,5,6)^-$, $(3,4,5)^-$, $(3,4,6)^-$, $(3,5,6)^-$, $(4,5,6)^-$.

Таким образом, покрытие дают сочетания столбцов 1, 4, 5 со стоимостью $C_1 + C_4 + C_5 = 4 + 8 + 3 = 15$ и 2, 4, 6 со стоимостью $C_2 + C_4 + C_6 = 7 + 8 + 2 = 17$, откуда оптимальное решение задачи – столбцы 1, 4, 5 матрицы P и стоимость 15. Как видим, это решение достигается при его выборе из $C_6^2 + C_6^3$ сочетаний, на что затрачивается $W = \left(4 \cdot C_6^2 + 9 \cdot C_6^3 + 6\right) \cdot 6$, т.е. не более, чем 1476 основных операций.

Выводы

Научная новизна работы – получен новый алгоритм решения задачи о покрытии для задач среднего размера.

Практическая значимость – алгоритм легко программируем, в связи с чем он может быть без затруднений использован в производственных целях.

Список литературы: 1. *Корбут А.А., Финкельштейн Ю.Ю.* Дискретное программирование. М.: Наука, 1969. 368 с. **2.** *Кристофидес Н.* Теория графов. Алгоритмический подход. М.: Мир, 1978. 432 с. **3.** *Lemke C.E., Salkin H.M., Spielberg K.* Set covering by single branch enumeration with linear programming Subproblems // Oper. Res. 1971. V. 19. P. 998 – 1015. **4.** *Липский В.* Комбинаторика для программистов. М.: Мир, 1988. 213 с.

Поступила в редколлегию 04.06.2011

Канцедал Сергей Андреевич, д-р техн. наук, профессор, зав. кафедрой финансы и кредит Западнодонбасского института экономики и управления. Научные интересы: математическое моделирование, теория расписаний и её применение. Адрес: Украина, Павлоград, ул. Днепровская, 400.

Костикова Марина Владимировна, канд. техн. наук, доцент кафедры информатики Харьковского национального автомобильно-дорожного университета. Научные интересы: математическое моделирование, теория расписаний и её применение. Адрес: Украина, 61002, Харьков, ул. Петровского, 25, тел. 707-37-74.