УДК 004.021 https://doi.org/10.15407/pp2020.02-03.138

ОБ ИСПОЛЬЗОВАНИИ ОСОБЫХ СТРУКТУР ДАННЫХ В АЛГОРИТМАХ ПОКРЫТИЯ

О.Н. Паулин, Н.О. Комлевая

Цель данной работы – это повышение эффективности методов и алгоритмов решения задачи нахождения покрытия. Под эффективностью понимается минимальная задержка процедуры, которая реализует данный метод. Для повышения эффективности метода «Разложение по столбцу» в процедуру построения дерева решения вводится характеристический вектор (ХВ), полученный суммированием единиц в столбцах/строках таблицы покрытия (ТП); он характеризует текущее состояние таблицы покрытия. Идея этого метода состоит в поэтапном разложении ТП на подтаблицы с использованием их сокращения по определённым правилам. Рассматриваются 3 способа сокращения исходной таблицы/текущих подтаблиц в методах: 1) «Граничный перебор по вогнутому множеству»; 2) «Использование свойств таблицы покрытия»; 3) «Минимальный столбец – максимальная строка». В последнем способе впервые применен ХВ, который позволил до полутора раз ускорить процедуру нахождения покрытия. Вычисляются оценки сложности для рассмотренных методов покрытия; имеем: S_1=O(n^3); S_2=O(2^n); S_3=O(n^2), где n – определяющий параметр задачи о покрытии (количество столбцов), и определяются границы применимости данных методов. Показывается, что применение характеристических векторов в методах 1 и 2 нецелесообразно.

Ключевые слова: покрытие, эффективность, метод, дерево решения, характеристический вектор, сокращение таблицы покрытия, оценка сложности процедуры.

Мета даної роботи – це підвищення ефективності методів і алгоритмів вирішення задачі знаходження покриття. Під ефективністю розуміється мінімальна затримка процедури, яка реалізує даний метод. Для підвищення ефективності методу «Розкладання по стовпцю» в процедуру побудови дерева рішення вводиться характеристичний вектор (ХВ), отриманий підсумовуванням одиниць в стовпцях/рядках таблиці покриття (ТП); він характеризує поточний стан таблиці покриття. Ідея цього методу полягає в поетапному розкладанні ТП на підтаблиці з використанням їх скорочення за певними правилами. Розглядаються 3 способи скорочення вихідної таблиці / поточних підтаблиць в методах: 1) «Граничний перебір по увігнутій множини»; 2) «Використання властивостей таблиці покриття»; 3) «Мінімальний стовпець – максимальний рядок». В останньому способі вперше застосований ХВ, який дозволив до півтора разів прискорити процедуру знаходження покриття. Обчислюються оцінки складності для розглянутих методів покриття; маємо: S1 = O (n ^ 3); S2 = O (2 ^ n); S3 = O (n ^ 2), де n – визначальний параметр завдання про покриття (кількість стовпців), і визначаються межі застосування даних методів. Показується, що застосування характеристичних векторів в методах 1 і 2 недоцільно.

Ключові слова: покриття, ефективність, метод, дерево рішення, характеристичний вектор, скорочення таблиці покриття, оцінка складності процедури.

The aim of this work is to increase the efficiency of methods and algorithms for solving the problem of finding coverage. Efficiency is understood as the minimum delay of the procedure that implements this method. To increase the efficiency of the “Columnization” method, a characteristic vector (CV) is introduced into the decision tree construction procedure, obtained by summing the units in columns / rows of the coverage table (CT); it characterizes the current state of the coverage table. The idea of this method is to gradually decompose CT into sub-tables using their reduction according to certain rules. We consider 3 ways to reduce the original table / current sub-tables in the methods: 1) "Border search over a concave set"; 2) "Using the properties of the coverage table"; 3) "The minimum column is the maximum row." In the latter method, CV was used for the first time, which made it possible to accelerate the coating finding procedure up to one and a half times. The complexity estimates for the considered coating methods are calculated; we have: S1 = O (n ^ 3); S2 = O (2 ^ n); S3 = O (n ^ 2), where n is the determining parameter of the coverage problem (number of columns), and the applicability limits of these methods are determined. It is shown that the use of CV in methods 1 and 2 is impractical.

Key words: coverage, efficiency, method, decision tree, characteristic vector, reduction of the coverage table, assessment of the complexity of the procedure.

Постановка проблемы и анализ последних исследований

Задача покрытия относится к классу оптимизационных комбинаторных задач; она сводится к задаче поиска в некотором конечном множестве определённых подмножеств с заданными свойствами. Эта задача возникает при необходимости оптимального выбора таких компонент из множества возможных, которые покрывают заданную их совокупность. Например, в задаче набора сотрудников в бюро переводов требуется выбрать минимальный состав переводчиков, каждый из которых владеет несколькими иностранными языками.

Предметные области применения алгоритмов покрытия весьма разнообразны. Одна из них – функциональное и структурное тестирование программного обеспечения. Как показано в [1], функциональное тестирование методом «черного ящика» предполагает полное (чаще всего кратчайшее) покрытие тест-кейсами программных функций. Структурное тестирование методом «белого ящика» заключается в полном покрытии тестами всех структурных элементов программы [2]. Другой популярной в настоящее время областью применения алгоритмов покрытия является использование гетерогенных коммуникационных технологий для умных городов и менее масштабных инфраструктур, позволяющих им общаться друг с другом посредством сетевого соединения. При этом разрабатываются специальные методики для сбора данных и обеспечения повсеместной сети связи, включающие помимо стандартных алгоритмов покрытия системные сценарии для работы в реальном времени [3]. Ряд работ посвящен исследованию эффективности и повышению скорости работы алгоритмов покрытия в задачах, требующих больших вычислительных ресурсов. Так, в работе [4]

приведен подход, основанный на распараллеливании обработки исходных данных. При этом глобальная цель разбивается на подцели, которые достигаются разными параллельными процессами, с последующим получением интегрированного результата. В работах [5, 6] предложено снижать сложность алгоритмических задач, в том числе задачи о покрытии, за счет перераспределения процессов обработки с привлечением инструментов интеллектуальной обработки данных. Таким образом, проведенный анализ литературных источников показал, что задача поиска покрытия по-прежнему актуальна. При этом в работах используются стандартные алгоритмы в качестве инструментария для решения прикладных задач; не ставится целью улучшение известных методов покрытия.

Эффективность известных методов может быть повышена с использованием структур данных, обеспечивающих минимальную сложность процедуры, которая реализует соответствующий метод [7]. В ряде работ в качестве структуры данных используется характеристический вектор, представляющий собой массив однотипных данных – чаще всего, логического типа. В статье [8] характеристические вектора использованы для описания вершин деревьев с учетом их уровней и потомков в алгоритме определения изоморфизма XML-схем. В работе [9] предложено сохранять в виде характеристического вектора преставление графических образов символов с последующим распознаванием символов на основании их геометрических и топологических свойств. Таким образом, применение характеристических векторов как эффективных структур данных является целесообразным.

В работе [10] рассмотрены задача покрытия и перечислены методы её традиционного решения; здесь же достаточно подробно рассмотрены методы и алгоритмы перебора строк таблицы покрытия: полного перебора и граничного перебора по вогнутому множеству. В работе [11] продолжено рассмотрение методов и алгоритмов покрытия. Статья посвящена рассмотрению метода «использование свойств таблицы покрытия» и алгоритма на его основе. В обеих работах из рассмотренных вычислительных процессов (ВП) выделены макрооперации (МО) и составлен общий список МО.

В работе [12] предложен интересный подход к перебору строк ТП небольшой размерности: рассмотрение совокупности (сочетания) строк по две, по три, и т. д., до m-1, где m – количество строк ТП. При этом исключены из рассмотрения варианты одиночных строк, которые не могут создавать покрытие, и вариант всех строк, дающий гарантированное, но не оптимальное покрытие. Эта идея здравая и может быть использована для любого метода нахождения покрытия. Однако авторами не выполнена оценка сложности предложенного ими метода, которая на деле равна: $S=O(2^m - m - 1)\approx O(2^m)$. Без использования данной формулы невозможно указать предельное значение для m; по нашему мнению, $m<=10$.

В работе [13] впервые введены характеристические векторы в приближённый метод покрытия «минимальный столбец – максимальная строка», что позволило ускорить алгоритм, построенный на данном методе.

Таким образом, задача повышения эффективности методов и алгоритмов решения задачи нахождения покрытия является актуальной.

Цель и задачи работы

Цель данной работы – повышение эффективности методов покрытия «Граничный перебор», «Использование свойств ТП» и «Разложение таблицы покрытия по столбцу» за счёт введения в процесс нахождения покрытия характеристического вектора (ХВ). Под эффективностью метода понимается снижение времени процедуры нахождения покрытия по данному методу.

Для достижения цели необходимо решить следующие задачи:

- проверить возможность использования ХВ для организации ВП нахождения покрытия;
- разработать процедуру по методу «Разложение таблицы покрытия по столбцу» с использованием ХВ;
- определить сложность улучшенных алгоритмов на основе перечисленных методов;
- формализовать рекомендации по выбору способа сокращения ТП в дереве решения при реализации метода «Разложение таблицы покрытия по столбцу».

Основная часть

Рассмотрим методы нахождения покрытия и используемые в соответствующих им алгоритмах структуры данных: метод сокращения таблицы покрытия с использованием свойств ТП; метод граничного перебора по вогнутому множеству; метод «минимальный столбец – максимальная строка» и приведём их словесные описания.

Метод и алгоритм сокращения таблицы покрытия с использованием свойств ТП [11]

ТП может обладать (или же не обладать) следующими свойствами: содержать строку – *ядерную* или *антиядерную*, столбец – *поглощающий*, строку – *поглощаемую*.

Метод состоит в последовательном рассмотрении перечисленных свойств и при их наличии – в сокращении ТП за счёт удаления определённых строк и/или столбцов. Эти свойства определяются соответствующими теоремами, представленными далее.

Структуры данных:

M – матрица принадлежности или таблица покрытия, M[i, j] – её элемент, i=1..m, j=1..n;

P – общий признак наличия изменений в ТП, P = $p_1 \vee p_2 \vee p_3 \vee p_4$;

P_k – частный признак наличия изменений в ТП для подпроцесса, k=1..4;

L – список ядерных строк.

Словесное описание алгоритма (СОА), реализующего данный метод

1. Общая процедура состоит в строгой последовательности применения теорем (1-я, 2-я, 3-я, 4-я и снова 1-я, 2-я, и т. д., пока возможны изменения в ТП).

2. Результат процедуры для каждой теоремы – удаление строк или\и столбцов ТП. Но возможны случаи неприменимости теорем.

3. Процедура для каждой теоремы заканчивается определением значения признака p_k (p_k = 1 в случае наличия изменений в ТП).

4. Формируется общий признак P наличия изменений в ТП (P=$p_1 \vee p_2 \vee p_3 \vee p_4$). Если в результате прохода вычислительного процесса по всем частным процедурам ТП не изменилась (P=0), то вычислительный процесс закончен. Иначе общая процедура повторяется.

Рассмотрим частные процедуры (подпроцессы вычислений).

Сокращение ТП на основе теоремы о ядерной строке

Теорема 1. Если в столбце ТП содержится единственная единица, то строка, содержащая эту единицу, входит во все покрытия и называется *ядерной*. Ядерная строка (ЯС) вычёркивается из ТП с запоминанием, вычёркиваются также все столбцы, покрываемые единицами ЯС.

Назовём единицу строки ТП *особенной*, если она в своём столбце является единственной.

Доказательство. Без особенной единицы покрытие принципиально невозможно.

Словесное описание процедуры нахождения ЯС

0. Признаку p_1 присваивается значение 0.

1. Перебираются строки ТП в цикле по i, i=1..m

2. В текущей строке перебираются её элементы. Если M[i, j]=0, то переход на п. 2.

3. Для очередной «1» строки просматривается весь столбец ТП, содержащий эту «1» (i=1..m). При этом нулевые элементы игнорируются, а единичные суммируются в S. Если S>=2, то переход на п. 2.

4. Если она особенная (S=1), то текущая строка является ядерной; её номер заносится в список ЯС.

5. Удаляются покрывающие столбцы ЯС (t=1..m), при M[i, t]=1.

6. Удаляются ЯС. Признаку p_1 присваивается значение 1.

7. Если перебраны все строки, то возврат в основную программу.

Сокращение ТП на основе теоремы об антиядерной строке

Теорема 2 (об антиядерной строке). *Антиядерной* называется строка, содержащая нулевые элементы. Эту строку можно удалить из ТП без запоминания.

Доказательство. Если строка является нулевой, то она не покрывает ни одного столбца. Следовательно, она является пустой (бесполезной) для покрытия.

Словесное описание процедуры нахождения антиядерной строки (АЯС)

0. Признак p_2 принимает значение 0.

1. Перебираются строки ТП. Если перебраны все строки, то переход на п. 4.

2. Просматриваются элементы текущей строки (j=1..n). При этом возможны два варианта: досрочный выход из цикла при M[i, j]=1 с переходом на п. 1 либо полный перебор элементов строки при всех M[i, j]=0, т. е. получается антиядерная строка.

3. АЯС удаляется из ТП; p_2:=1. Переход на п. 1.

4. Возврат.

Прежде чем перейти к рассмотрению третьей и четвёртой теорем, познакомимся с понятием «поглощение».

Определение 1. Вектор $E=(e_1,\ldots e_m)$ *поглощает* вектор $F=(f_1,\ldots,f_m)$, $E \geq F$, если для всех компонент e_i, f_i этих векторов можно одновременно записать $e_i \geq f_i$. В противном случае векторы называются *несравнимыми*.

Сокращение ТП на основе теоремы о поглощающих столбцах

Теорема 3. В ТП могут быть вычеркнуты *все* поглощающие столбцы (рассматриваемые как двоичные векторы) без ущерба для построения всех безизбыточных покрытий.

Доказательство. Если в столбце не содержится особенная единица, то имеет смысл искать столбец с минимальным числом единиц. Тогда все столбцы, поглощающие выбранный столбец, имеют большее число единиц и являются избыточными относительно выбранного столбца. Следовательно, их можно удалить.

СОА «Нахождение поглощающего столбца»

0. Признак p_3 принимает значение 0; g_1:=0, g_2:=0.

1. Формируем пары столбцов. Для этого:

1а. фиксируется *j*-й столбец в качестве первого компонента пары, *j*=1..*n*-1.

1б. перебираются остальные столбцы в качестве второго компонента пары с номером *j'*, *j'*=*j*+*k*, *k* =1..*n*-*j*.

2. Представляем столбцы пары в виде двоичных векторов (V, V') с номерами *j* и *j'* соответственно.

3. В цикле по *i*, *i*=1..*m*, сравниваем элементы пары векторов.
Если V[i]=V'[i], то переход на п. 3.
Если V[i]>V'[i], то g_1:=1, иначе g_2:=1.
Если g_1&g_2:=1, то переход на п. 1б.

4. По окончании поэлементного сравнения проверяем g_1=1? Если – да, то удаляем столбец с номером *j*, иначе удаляем столбец с номером *j'*. p_3:=1. Переходим на п. 2.

5. Если просмотрены все пары, то возврат в основную программу.

Теорема 4. (*о поглощаемых строках при поиске одного кратчайшего покрытия*). Если при решении задачи о покрытии достаточно гарантировать получение хотя бы одного кратчайшего покрытия, то можно удалить *все* поглощаемые строки.

Доказательство. Для эффективного покрытия целесообразно оставить строку с максимальным количеством единиц**,** поскольку поглощаемые ею строки имеют меньшее число единиц; их можно вычеркнуть без ущерба для покрытия.

СОА «Нахождение поглощаемой строки»

0. p_4:=0; g_1:=0; g_2:=0.

1. Формируются пары строк. Для этого:

1а. фиксируется *i*-я строка в качестве первого компонента пары, *i*=1..*m*-1.

1б. перебираются остальные строки (второй компонент пары с номером *i'*, *i'*=*i*+*k*, *k* =1..*m*-*i*).

2. Представляем строки пары в виде двоичных векторов (V, V') с номерами *i* и *i'* соответственно.

3. В цикле по *j*, *j*=1..*n*, сравниваем элементы пары векторов.
Если V[j]=V'[j], то переход на п. 3.
Если V[j]>V'[j], то g_1:=1, иначе g_2:=1.
Если g_1&g_2:=1, то переход на п. 1б.

4. По окончании поэлементного сравнения проверяем g_1=1? Если – да, то удаляем строку с номером *i'*, иначе удаляем строку с номером *i*. p_4:=1. Переходим на п. 1б.

5. Если просмотрены все пары, то возврат в основную программу.

Отметим, что приведенный порядок применения теорем обосновывается тем, что при этом наиболее эффективно упрощается таблица покрытия.

Разновидностью теоремы 4 является

Теорема 5 (*о поглощающих строках при построении минимальных покрытий*). Если при решении задачи о покрытии достаточно гарантировать построение всех (хотя бы одного) минимального покрытия, то можно вычеркивать поглощаемую строку, если цена её больше или равна цене поглощающей строки.

В этом случае п. 4 должен включать в себя проверку по цене строк.

Используя теоремы 1..4, упрощаем ТП. При этом возможны два исхода.

1. ТП после упрощения становится пустой (вычеркнуты все столбцы). В этом случае множество ядерных строк – требуемое покрытие.

2. Остаток ТП более не упрощается приёмами из теорем 1..4. Получаем *циклический остаток* (ЦО) таблицы покрытий. Покрытие ЦО таблицы можно строить методами граничного перебора либо разложения по столбцу.

Полное решение (покрытие исходной таблицы) состоит из ядерных строк и строк покрытия ЦО.

Метод и алгоритм граничного перебора по вогнутому множеству [10]**.** Алгоритм основан на методе генерации подмножеств и их целенаправленном отборе. Алгоритм генерации подмножеств должен гарантировать, что при его последовательном применении можно построить, начиная с некоторого начального подмножества, все возможные подмножества; при этом не должно быть повторений и должен существовать критерий окончания перебора.

Граничный перебор заключается в следующем: сначала строятся все подмножества D_i, содержащие A_1, затем – содержащие A_2, но не содержащие A_1; если построено подмножество D_i, то за ним строится подмножество D_{i+j} целиком содержащее D_i ($D_i \supset D_{i+j}$).

Во многих приложениях достаточно находить только безызбыточные покрытия, что естественным образом сокращает перебор. Однако не удаётся найти простой и эффективный алгоритм, не требующий построения всех избыточных покрытий. Идея улучшения алгоритма перебора (генерации) подмножеств заключается в следующем: если текущее подмножество – покрытие, то в это подмножество не нужно вводить новые элементы.

Теорема 6. Если P покрытие, то и P', $P \supseteq P'$, также является покрытием, т. е. множество всех возможных покрытий *вогнуто*.

Таким образом, безызбыточные покрытия – это граница вогнутого множества всех покрытий,

СОА перебора по вогнутому множеству

Структуры данных:

T– таблица покрытия (ТП), содержащая m строк и n столбцов;

D – текущее подмножество (список), элементами которого являются номера строк ТП;

L – список номеров строк, дающих покрытие.

Операции над списком L :

- *вставить* элемент x ($x \to L$);
- *удалить* элемент x ($x \leftarrow L$);
- *считать* значение элемента x ($x \leftrightarrow L$) с сохранением его в списке.

Отметим, что при вставке/удалении элемента x модифицируется указатель, а сам список упорядочивается (происходит перенумерация элементов).

Словесное описание алгоритма граничного перебора:

1. Текущее подмножество D:={A_1}, i:=0.

2. Выясняем, является ли D покрытием, для чего формируем признак p покрытия (ФПП). Если очередное построение в D – покрытие (p=1), то упорядочиваем список покрытий (УСП).

3. Находим наибольший номер j элемента в подмножестве D

3.1. Если $j \neq n$ и D – не покрытие (p=0), то выполняем п. 5.

3.2. Если $j \neq n$ и D – покрытие (p=1), то выполняем п. 4.

3.3. Если $j = n$, то удаляем A_n из D ($A_n \leftarrow D$), и если D = ∅, то заканчиваем построение всех безызбыточных покрытий и переходим на п. 6, иначе – снова находим наибольший номер j элемента в D.

4. Удаляем элемент A_j из D ($A_j \leftarrow D$).

5. j:=j+1. Вводим элемент A_j в D ($A_j \to D$) и выполняем п.2.

6. Выводим на печать список безызбыточных покрытий.

7. Конец.

Из полученных безызбыточных покрытий можно выбрать покрытия с минимальным количеством строк (кратчайшее покрытие) либо покрытие с минимальной ценой (минимальное покрытие).

Рассмотрим алгоритмические модули ФПП и УСП.

Описание модуля ФПП. Сформулируем признак покрытия: p = 1, если в процессе перебора строк ТП, заданных подмножеством (списком) D, выяснится, что все 0-позиции покрыты. Покрытие может наступить до исчерпания строк, входящих в подмножество D.

Структуры данных:

T – таблица покрытия, содержащая m строк и n столбцов;

D – текущее подмножество (список) из s номеров строк ТП;

L – список из r позиций, на которых находятся нули (0-позиции);

V – двоичный вектор;

t – номер 0-позиции, $t = L[j]$;

q – номер строки в ТП.

Словесное описание вычислительного процесса в модуле ФПП.

1. Если в D всего одна строка (принимаем по умолчанию, что одна строка не образует покрытия), то переход на п. 8 (возврат значения признака p = 0).

2. Определяется номер первой в D строки (q := D[1]) и считывается сама строка A_q из ТП ($A_q \leftrightarrow T$). Она представляется в виде двоичного вектора V, ($V \div A_q$).

3. Просматриваются позиции V; номера 0-позиций заносятся в список L ($V[j] = 0 \Rightarrow j \to L$).

4. Определяется номер следующей строки из D и, соответственно, считывается сама строка, которая далее представляется в виде двоичного вектора, $V \div A_q$.

5. Просматриваются в V все те позиции t, которые занесены в L; если в текущей позиции находится 1, то номер этой позиции удаляется из L ($t \leftarrow L$).

6. Если L=∅, то *p*:=1 (*D* – покрытие); переход на п. 8;

7. Если не все строки в *D* обработаны, то переход на п.4, иначе *p*:=0.

8. Возврат значения *p* в основную программу.

Описание вычислительного модуля УСП. Идея упорядочения списка покрытий состоит в удалении по очереди из ранее построенных покрытий тех покрытий, которые поглощают (включают в себя) *D*, т.е. избыточных покрытий, при этом уменьшая *i* каждый раз на 1; запоминаем последнее *D* как покрытие P_i (*i*:=*i*+1; P_i:= *D*).

Отметим, что возможны 4 варианта результата проверки на включение множеств *A* и *B*.

1. $A \subset B$.

2. $A \supset B$.

3. $A = B$.

4. A и B несравнимы. В данном случае возможны по построению подмножеств D только 1-й и 4-й варианты. Следовательно, достаточно убедиться в том, что все элементы B входят/не входят в множество A.

Примем по умолчанию, что множества A и B представлены списками и упорядочены по возрастанию.

Структуры данных:

C – список покрытий;

P^* – текущее покрытие;

i – номер покрытия, $i = 1 \ldots q$.

Исходные данные: список покрытий C, очередное покрытие P^*, не вошедшее ещё в список C, номер последнего в списке покрытия – q.

Словесное описание вычислительного процесса в модуле УСП.

1. Организуем цикл по i перебора покрытий в списке C ($i := i+1$). Если $i > q$, то переходим на п. 5.

2. Считываем очередное покрытие из C ($P_i \leftrightarrow C$).

3. Определяем, не является ли оно избыточным относительно P^* (сравнение покрытий P_i и P^* на избыточность – модуль СПИ). Если P_i не является избыточным ($g = 0$), то переходим на п. 1.

4. Удаляем избыточное покрытие P_i из списка C ($P_i \leftarrow C$), переходим на п. 1.

5. Процедура оканчивается и возвращается результирующий список покрытий C'.

Описание вычислительного модуля СПИ. Переобозначим: $A := P_i$, $B := P^*$. Отметим, что количество элементов в P_i всегда больше, чем в P^*, что определяется методом построения подмножеств D; при этом число вариантов сравнения уменьшается до двух: избыточно/неизбыточно. Обозначим g признак избыточности. Тогда $g = 1$ означает избыточность P_i относительно P^*.

Идея решения состоит в исключении из обоих множеств их пересечения $A \cap B$; в случае $B = \varnothing$ это будет означать, что P_i избыточно относительно P^*.

Словесное описание алгоритма для модуля СПИ.

1. Перебираем элементы y_j множества B.

2. Просматриваем элементы x_k множества A.

3. Если элементы x_k и y_j совпадают, то они удаляются из обоих множеств; переход к п.2. Иначе множества несравнимы ($g := 0$) – переход на п. 5.

4. Проводится модификация указателей множеств и перенумерация элементов (по умолчанию – автоматическая).

5. Если $B = \varnothing$, то $g := 1$, иначе переход на п. 1.

6. Процесс заканчивается; возвращается значение g.

Метод «минимальный столбец – максимальная строка» и алгоритм на его основе [13]

Данный метод является приближённым, то есть не даёт оптимального покрытия; однако полученное покрытие – достаточно близкое к кратчайшему. Его идея заключается в выборе из строк, определяемых единицами минимального столбца, строки с максимальным количеством единиц. Процедура, реализующая этот метод, используется многократно.

В алгоритме используются следующие структуры данных.

Отметим, что кроме известных в стандартном алгоритме структур данных, нами для ускорения процесса нахождения покрытия введены новые структуры – *целочисленные характеристические векторы* VCOL и

VROW, содержащие вначале суммы единиц для всех столбцов и строк соответственно, тем самым *заготавливая* описание *начального состояния* ТП. В процессе обработки ТП эти вектора отражают *текущее состояние* ТП, при том, что сама *ТП не изменяется*.

На рис. 1 показана схема взаимодействия введенных векторов и ТП. Здесь x = {0, 1} – значения функции принадлежности элементов строки элементам опорного множества, т. е. именам столбцов; y = Σx – значения сумм элементов столбцов, z = Σx – значения сумм элементов строк, y, z = {0, 1, 2, …}. Рядом с изображением вектора на рис. 1 проставлены номера элементов.

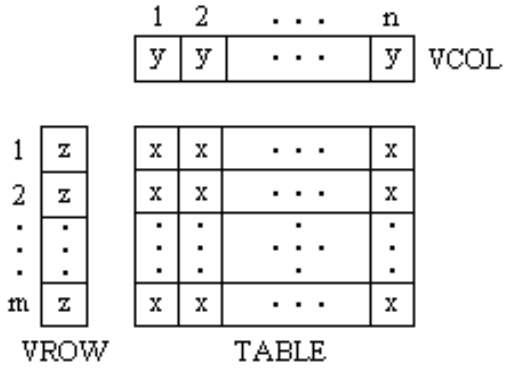

Рис. 1. Иллюстрация применения векторов VCOL и VROW

Для сокращения перебора элементов минимального столбца ТП введём маску, в качестве которой используем модифицированный вектор строк VROW', MASKA[i]=VROW'[i]; модификация состоит в том, что целочисленные ненулевые элементы вектора строк VROW[i] мы заменяем единицами (рис. 2,а).

Рассмотрим пример работы маски (рис. 2,б). Результирующий вектор определяется по формуле VREZ[i] = MASKA[i]⊗TABLE[i,k], где ⊗ – операция поэлементного перемножения векторов, *k* – номер минимального столбца.

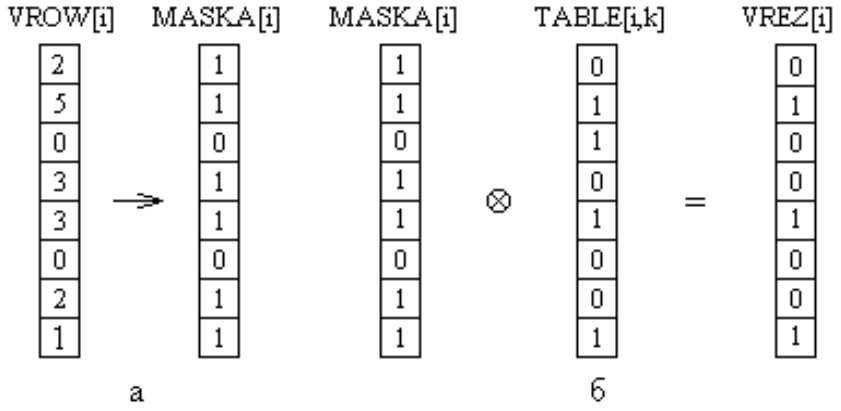

Рис. 2. Пример работы маски на *k*-ом столбце ТП

Структуры данных:

TABLE – таблица покрытий;

VCOL – целочисленный характеристический вектор столбцов, *j*-й элемент которого является суммой единиц *j*-го столбца;

VROW – целочисленный характеристический вектор строк, *i*-й элемент которого является суммой единиц *i*-й строки;

MASKA – двоичный вектор, отображающий вектор VSTR, первоначально MASKA является единичной;

VMIN – вектор, отображающий текущий минимальный столбец ТП;

VREZ – результирующий вектор, полученный из VMIN при маскировании;

LNR – текущий список номеров строк, отображающий VREZ;

LCOV – список номеров покрывающих строк (список покрытий);

counter – счётчик для подсчёта нулевых элементов вектора столбцов;

k, l – переменные, обозначающие номера минимального столбца либо максимальной строки соответственно.

i, j – параметры циклов.

СОА по методу «минимальный столбец – максимальная строка»

0. Ввод таблицы покрытий размерностью *mxn*.

1. Инициализация. Обнуляются все вектора, списки и счётчик нулевых элементов (counter); формируем начальную маску, заполняя её единицами.

2. Заполняем характеристический вектор (ХВ) VCOL информацией о столбцах. Для этого перебираем столбцы и для выбранного столбца суммируем ненулевые элементы. Если при этом сумма равна 0, то выдаём сообщение «Покрытие отсутствует» и переходим на п.14.

3. Заполняем ХВ VROW информацией о строках. Для этого перебираем строки ТП и в выбранной строке суммируем ненулевые элементы.

4. Организуем основной цикл с постусловием окончания: «вектор VCOL является нулевым», что имитирует вычёркивание в ТП всех столбцов – см. п.12.

5. Определяем минимальный элемент ХВ VCOL и фиксируем его номер *k*; в случае нескольких одинаковых минимальных элементов выбираем первый встретившийся.

6. Представляем столбец ТП с номером *k* в виде вектора: VMIN[i]:=TABLE[i,k], и маскируем его: VREZ[i]:=MASKA[i]⊗VMIN[i].

7. Обнуляем текущий список номеров строк (LNR) и записываем в него те строки, которые содержат единицы в VREZ[i].

8. Определяем максимальный элемент в LNR, используя VREZ, и фиксируем его номер *l*; в случае нескольких одинаковых максимальных элементов выбирается первый встретившийся.

9. Записываем номер *l* в текущий список покрытий LCOV.

10. Обнуляем элементы с номером *l* в ХВ VROW и в маске. Это имитирует удаление из ТП строки с номером *l*.

11. Обнуляем *j*-е элементы ХВ VCOL, для которых TABLE[l, j]=1. Это имитирует удаление из ТП покрываемых столбцов. При каждом обнулении в counter добавляем 1.

12. Проверяем окончание процесса решения по условию "Counter< *n*?". Если да – переходим на п. 4.

13. Выводим список покрытий LCOV.

14. Конец.

Исследования на ТП со случайным их заполнением показали, что введение ХВ позволило получить ускорение от 1,12 для ТП малой размерности (8*8) до 1,56 для ТП размерностью 32*32. Кроме того, показано, что сложность модифицированного алгоритма составляет S=O(m*n).

Ниже нами впервые рассматривается метод разложения по столбцу и исследуется возможность использования ХВ для эффективного сокращения ТП.

Метод разложения по столбцу [14]

Отметим прежде всего, что метод разложения по столбцу входит составной частью в процедуру построения дерева решения (ДР), в каждой вершине которого находится сокращенная на предыдущем этапе процедуры таблица/циклический остаток (ЦО) покрытия.

Построение ДР осуществляется обходом его вершин сверху вниз (прямой проход). Обработка вершин ДР одного уровня (проход в ширину) включает в себя сокращение подтаблиц/ЦО (в корне – исходной ТП). Для сборки списка покрытий осуществляется обход его вершин снизу вверх (обратный проход).

Основная идея метода разложения по столбцу заключается в нахождении совокупности строк, содержащей хотя бы одну единицу в каждом столбце таблицы. Решение задачи, реализующей идею, состоит в разбиении исходной ТП на *k* подтаблиц с помощью процедуры разложения по столбцу, которая состоит в следующем: выбирается столбец с наименьшим числом единиц; этим единицам соответствуют *k* строк – *k* вариантов разложения ТП. Каждая подтаблица получается в результате её сокращения – вычёркивания соответствующей строки и покрытых ею столбцов; номер вычёркиваемой строки приписывается данному варианту и включается в будущее покрытие.

Упрощение процесса решения. Для упрощения процесса решения данной задачи полученная подтаблица сокращается в несколько итераций, пока не исчерпаются сокращения – получится ЦО. Имена полученных покрывающих строк дописываются к имени данного варианта. Так мы поступаем для всех вариантов. Каждый вариант оканчивается ЦО и множеством покрывающих строк; к ЦО вновь применяется описанная процедура разложения по столбцу, приводящая к своим вариантам (подвариантам) разложения. Решение для подварианта продолжается, пока его ЦО не станет пустым.

Отметим, что при рассмотрении процедуры текущего разложения в очередной вариант остатка таблицы не включаются строки, выбранные в покрытие для предыдущих вариантов разложения.

Дерево процесса решения. Процесс решения фиксируется в виде дерева решения (ДР), в каждой вершине которого записаны множество введенных в решение строк и соответствующий ЦО. К корню дерева приписаны ЦО и множество покрывающих строк исходной ТП. Из корня выходит столько ветвей дерева

решения, сколько единиц в выбранном столбце; каждая ветвь соответствует своему варианту разложения. Дерево достраивается разложением ЦО, приписанного к вершине, в которой разложение ещё не проведено и ЦО ещё не пуст.

Вершина, в которой ЦО пуст, называется *конечной* (*листом*), для неё разложение не проводится.

Отметим, что листья могут появиться на разных уровнях дерева решения.

Вершины\листья дерева нумеруются в соответствии с обходом дерева в ширину.

Итак, решение задачи состоит в чередовании применения метода разложения по столбцу и сокращения подтаблицы до ЦО; при этом образуется множество покрывающих/ядерных строк.

Построение (сборка покрывающих строк) покрытий по дереву решения. По дереву решения легко получить результат – множество покрытий. Для этого достаточно по каждой ветви дерева пройти от конечной вершины к корню (снизу вверх), объединяя множество строк, введенных в покрытия на каждой пройденной вершине. Каждое такое множество – покрытие исходной ТП. Для выделения оптимальных решений нужно привести подобные, удалить избыточные покрытия и выбрать из оставшихся оптимальные покрытия по соответствующему критерию – кратчайшие либо минимальные.

СОА по [11] для метода «Использование свойств ТП»

1. Исходная ТП упрощается и полученные ЦО и ядерные строки приписываются начальной вершине (корню) дерева решения.

2. Находится не конечная очередная вершина, при которой имеется не пустой участок ТП. Если таких вершин нет, то процесс построения дерева решения заканчивается и выполняется п. 3, иначе находится столбец с меньшим числом единиц и проводится разложение по этому столбцу. В каждом варианте разложения остаток таблицы сокращается, и если он оказывается пустым, то соответствующая вершина объявляется конечной. Снова выполняется п. 2.

3. По дереву решения строятся покрытия, их оценивают и выбирают наилучшие (*в примере – минимальные*).

Модификация метода разложения по столбцу. В [13] предложено отмечать результаты всех манипуляций с ТП в характеристическом векторе (ХВ), не изменяя самой таблицы покрытия. Первоначально элементы ХВ получаются суммированием единиц в столбце/строке; манипуляции состоят в удалении столбцов и строк, что в ХВ отражается обнулением соответствующего элемента. Этот подход будет реализован и в методе разложения по столбцу.

Предварительно рассмотрим вопрос сокращения ТП. Сокращение ТП можно выполнить следующими способами: 1) используя свойства ТП (наличие ядерных и антиядерных строк; возможность поглощения столбцов/строк, представленных в виде двоичных векторов); 2) используя метод граничного перебора (устранение избыточных покрытий); 3) используя модифицированный приближённый метод «минимальный столбец – максимальная строка».

Оценим эти способы по сложности. При этом нас в оценке интересует скорость роста параметра, наиболее полно характеризующего процедуру.

В способе 1 наиболее сложным по числу операций является выявление поглощающего столбца/поглощаемой строки. При этом надо сравнить все возможные пары столбцов\строк. Например, перебор пар столбцов даёт сложность $S=n(n-1)/2$, где n – число столбцов. Поскольку поглощение может быть двоякое (A>B либо B>A), то количество операций удваивается и составляет $S=n(n-1)\approx n\text{^}2$.

Определим число операций при сравнении элементов двоичных векторов. Пусть число элементов в столбце равно m. Тогда число операций сравнения равна m, а оценка общей сложности нахождения поглощающих столбцов равна $S_o=O(m*n\text{^}2)$.

Для случая перебора строк оценка сложности примет вид $S_o= O(n*m\text{^}2)$.

Для выявления ядерной строки необходимо просмотреть все столбцы и все элементы в столбцах. Число операций составляет m*n.

Для выявления антиядерной строки также необходимо m*n операций (просмотр строк и значений элементов в строке).

В итоге получаем (для простоты примем $m=n$) $S=n\text{^}3+2n\text{^}2$, что при $n>5$ позволяет отбросить второе слагаемое: $S_o\approx n\text{^}3=O(n\text{^}3)$.

В способе 2 за счёт исключения δ избыточных покрытий сложность составляет $S_o=2\text{^}m-\delta$, причём $\delta<<2\text{^}m$. Таким образом,

$S_o\approx 2\text{^}m=O(2\text{^}m)\rightarrow O(2\text{^}n)$.

В способе 3, если использовать модификацию алгоритма введением ХВ [13], оценка сложности составит $S_o=O(m*n) \rightarrow O(n\text{^}2)$.

Наилучшей оценкой обладает третий способ. Что касается первых двух способов, то при малых значениях n предпочтительней способ 2, при $n=10$ оценки близки, при $n>10$ способ 2 проигрывает способу 1.

Рассмотрим возможность применения ХВ в методе «Использование свойств ТП». ХВ имеет смысл использовать, если для проведения процесса обработки ТП необходимо знать суммы значений элементов

столбцов\строк. Анализ вычислительного процесса для каждой теоремы показывает, что такой необходимости нет. Следовательно, нет и необходимости в использование ХВ.

Для метода граничного перебора анализ показал, что и здесь нет необходимости в суммировании «1» столбцов\строк. Следовательно, нет необходимости и в ХВ.

Словесное описание модифицированного алгоритма по методу «минимальный столбец – максимальная строка».

0. Вводим таблицу покрытий размерностью m*n.

1. Инициализация: заполняем поэлементно характеристические вектора значениями сумм строк/столбцов.

2. Упрощаем таблицу покрытия одним из рассмотренных способов.

3. Применяем разложение по столбцу к полученному циклическому остатку. Для этого:

3.1. Находим в ХВ для столбца первый встретившийся минимальный элемент.

3.2. Для каждой «1» в минимальном столбце запоминаем номер соответствующей строки и имитируем вычёркивание данной строки и всех покрываемых ею столбцов обнулением соответствующих элементов ХВ как для строки, так и для столбцов. Получаем ЦО (подтаблицу).

3.3. Приписываем полученному ЦО номер удалённой строки. Сформирована новая вершина ДР.

4. Для каждой сокращённой ТП проводим операции, описанные в пп. 2 и 3. Получаем новый уровень ДР.

5. Перебираем вершины ДР (проход ДР в ширину). Если в одной из вершин процедура закончена (вычеркнуты все столбцы), то далее считываем все номера сокращенных ТП, приписанных вершинам на пути от листа до корня ДР (обратный проход по вершинам ДР).

6. Объединяем списки. Для выделения оптимальных решений нужно привести подобные, удалить избыточные покрытия и выбрать из оставшихся оптимальные покрытия по соответствующему критерию – кратчайшие либо минимальные.

7. Конец процедуры.

Выводы

Поставленные задачи выполнены, при этом получены следующие результаты:

1. Рассмотрены 3 способа сокращения ТП по методам: «Граничного перебора», «Использования свойств ТП», «Минимальный столбец – максимальная строка». Использование ХВ для организации ВП нахождения покрытия целесообразно только в методе «Минимальный столбец – максимальная строка» [13], для которого получено ускорение до полутора раз.

2. Разработана новая процедура по методу «Разложение таблицы покрытия по столбцу», в которой применена достаточно сложная структура данных – характеристический вектор с целочисленными значениями.

3. Для трёх рассмотренных способов сокращения ТП определена сложность алгоритмов: для 1-го способа оценка составляет S_o=O(n^3), для 2-го – S_o=O(2^n) и для 3-го – S_o= O(n^2).

4. Проведен анализ и предложены рекомендации по выбору способа сокращения ТП в дереве решения при реализации метода «Разложение таблицы покрытия по столбцу»:

– при $n \le 6$ можно применять способ «Граничный перебор по вогнутому множеству»; полученные покрывающие строки дописываются ко всем покрывающим строкам ветви;

– при $6 < n \le 10$ можно применять способ «Использование свойств ТП»; при этом получаем ЦО на каждой ветви ДР, к которым снова применяем разложение по столбцу;

– при n>10 целесообразно применять способ «Минимальный столбец – максимальная строка». При этом надо помнить, что данный метод является довольно быстрым, тем более что применение ХВ даёт дополнительное ускорение, но не гарантирует оптимальности полученного покрытия.

Литература

1. Mathur A.P. Foundations of software testing: fundamental algorithms and techniques. *New Delhi: Dorling Kindersley*. 2013. 324 p.
2. Mansoor A. Automated Software Test Data Optimization Using Artificial Intelligence. *Int. J. Inf. Commun. Technol. Trends*. 2013. Vol. 9. P. 5–19.
3. Ahuja K., Khosla A. A novel framework for data acquisition and ubiquitous communication provisioning in smart cities. *Future Generation Computer Systems – The International Journal Of Science*. 2019. Vol. 101. P. 785–803.
4. Sanchez-Gomez J.M., Vega-Rodríguez M.A., Pérez C.J. Parallelizing a multi-objective optimization approach for extractive multi-document text summarization. *Journal of Parallel and Distributed Computing*. 2019. Vol. 134. P. 166–179.
5. Zhanyang X., Xihua L., Gaoxing J., Bowei T. A time-efficient data offloading method with privacy preservation for intelligent sensors in edge computing. *Eurasip Journal On Wireless Communications And Networking*. 2019. Vol. 1. P. 236– 46.
6. Hui L., Haining L., Shu Z., Zhaoman Z., Jiang C. Intelligent learning system based on personalized recommendation technology. *Neural Computing & Applications*. 2019. Vol. 31, Is. 9. P. 4455–4462.
7. Кузюрин Н.Н., Фомин С.А. Эффективные алгоритмы и сложность вычислений. К.: Наука. 2011. 363 с.

8. Сергеев А.П. Алгоритм определения изоморфизма XML-схем. *Проблеми програмування*. 2010. № 2–3. С. 530–536.
9. Прохоров В.Г. Распознавание графических образов текстовых символов, представленных в виде характеристических векторов. *Проблеми програмування*. 2007. № 3. С. 97–106.
10. Паулин О.Н. Вычислительные модели алгоритмов покрытия. *Информатика и математические методы в моделировании*. 2016. Т. 6, № 4. С. 385–396.
11. Паулин О.Н. Методы и алгоритмы покрытия (Часть 2). *Информатика и математические методы в моделировании*. 2017. Т. 7, № 4. С. 333–338.
12. Канцедал С.А., Костикова М.В. Один алгоритм решения задачи о покрытии. *Автоматизированные системы управления и приборы автоматики*. 2011. № 155. С. 49–53.
13. Paulin O.N., Komleva N.O., Sinegub N.I., Sarafaniuk D.E. About modification the coverage algorithm using the "minimum column – maximum row" method. *Вчені записки ТНУ імені В.І. Вернадського. Серія: технічні науки*. 2020. Т. 31 (70), № 1. (в печати)
14. Новосёлов В.Г., Скатков А.В. Прикладная математика для инженеров-системотехников. Дискретная математика в задачах и примерах: Учебное пособие. К.: УМК ВО, 1992. 200 с.

References

1. Mathur A.P. (2013) Foundations of software testing: fundamental algorithms and techniques. *New Delhi: Dorling Kindersley*. 324 p.
2. Mansoor A. (2013) Automated Software Test Data Optimization Using Artificial Intelligence. *Int. J. Inf. Commun. Technol. Trends*. 9. P. 5–19.
3. Ahuja K. & Khosla A. (2019) A novel framework for data acquisition and ubiquitous communication provisioning in smart cities. *Future Generation Computer Systems – The International Journal Of Science*. 101. P. 785–803.
4. Sanchez-Gomez J.M., Vega-Rodríguez M.A. & Pérez C.J. (2019) Parallelizing a multi-objective optimization approach for extractive multi-document text summarization. *Journal of Parallel and Distributed Computing*. 134. P. 166–179.
5. Zhanyang X., Xihua L., Gaoxing J. & Bowei T. (2019) A time-efficient data offloading method with privacy preservation for intelligent sensors in edge computing. *Eurasip Journal On Wireless Communications And Networking*. 1. P. 236– 46.
6. Hui L., Haining L., Shu Z., Zhaoman Z. & Jiang C. (2019) Intelligent learning system based on personalized recommendation technology. *Neural Computing & Applications*. 31 (9). P. 4455–4462.
7. Kuzyurin N.N. & Fomin S.A. (2011) Efficient algorithms and computational complexity. Kyiv. 363 p.
8. Sergeev A.P. (2010) Algorithm for determining isomorphism of XML schemas. *Program problems*. 2–3. P. 530–536.
9. Prohorov V.G. (2007) Recognition of graphic images of text characters represented in the form of characteristic vectors. *Program problems*. 3. P. 97–106.
10. Paulin O.N. (2016) Computational models of coverage algorithms. *Computer science and mathematical methods in modeling*. 6 (4). P. 385–396.
11. Paulin O.N. (2017) Methods and algorithms for coverage (Part 2). *Computer science and mathematical methods in modeling*. 7 (4). P. 333–338.
12. Kancedal S.A. & Kostikova M.V. (2011) One algorithm for solving the coverage problem. *Automated control systems and automation devices*. 155. P. 49–53.
13. Paulin O.N., Komleva N.O, Sinegub N.I. & Sarafaniuk D.E. (2020) About modification the coverage algorithm using the "minimum column – maximum row" method. *Scientific notes of TNU Vernadsky. Series: Technical Sciences*.. T. 31 (70), N 1. (in press)
14. Novosyolov V.G. & Skatkov A.V. (1992) Applied Mathematics for Systems Engineers. Discrete Mathematics in Problems and Examples: A Study Guide. Kyiv. 200 p.

Получено 17.02.2020

Об авторах:

Паулин Олег Николаевич,
доктор технических наук, доцент.
Количество научных публикаций в украинских изданиях – более 100.
Количество научных публикаций в зарубежных изданиях – 7.
https://orcid.org/0000-0002-2210-8317,

Комлевая Наталия Олеговна,
кандидат технических наук, доцент.
Количество научных публикаций в украинских изданиях – 46.
Количество научных публикаций в зарубежных изданиях – 4.
http://orcid.org/0000-0001-9627-8530.

Место работы авторов:

Одесский национальный политехнический университет,
проспект Шевченко, 1, г. Одесса, 65044.
Тел.: (38)(048) 705-85-66.
E-mail: kaf.spz@opu.ua,
nkomlevaya@gmail.com