УДК 519.718

К.т.н., доцент Романкевич В.О., аспірант Фесенюк А.П., магістрант Шкира Д.С.

Національний технічний університет України «Київський політехнічний інститут»

АНАЛІЗ СКЛАДНОСТІ АЛГОРИТМУ ГЕНЕРУВАННЯ РІВНОВАЖНИХ ВЕКТОРІВ НА ОСНОВІ ПЕРЕТВОРЕННЯ ПСЕВДОВИПАДКОВИХ ЦІЛИХ ЧИСЕЛ

Abstract

Vitaliy O. Romankevych, assoc. prof., PhD; Andriy P. Fesenyuk, postgraduate, Dmytro S. Shkyra, student

Complexity analysis of generation of equilibrium vectors algorithm, based on conversion of integer numbers

The algorithm which is used for computation of reliability with using of method of statistic experiments is present in this paper. Complexity of the algorithm is calculated and practical suitability of the algorithm is proved.

Вступ

Надійність комп'ютерних систем є важливою їх характеристикою. Здатність системи безвідмовно працювати протягом заданого часу в заданих умовах визначає можливість використання системи у реальному житті. Особливо високими є вимоги до надійності систем реального часу, наприклад, систем керування літаками або космічними апаратами, систем керування банківськими операціями.

Розрахунок показників надійності відмовостійких систем дозволяє встановити відповідність їх надійності нормативно-технічній документації, прогнозувати їх роботу протягом тривалого проміжку часу, знайти умови, за яких такі системи можуть успішно виконувати поставлені задачі.

В роботах [1,2] запропонований метод розрахунку надійності відмовостійких багатопроцесорних систем шляхом проведення статистичних експериментів. Для реалізації цього методу використовуються графологічні (GL) моделі поведінки системи в потоці відмов, генератор станів системи та засоби для збору статистики, виконання розрахунків і оцінки похибок. При проведенні експериментів стан системи характеризується двійковим вектором, котрий визначає роботоздатність компонентів системи.

Постановка задачі

При розрахунку надійності великих систем кількість випробувань, необхідних для проведення вичерпного дослідження, є дуже великою. Це може перешкодити точному розрахунку надійності за прийнятний час. Тому проводиться обмежене число статистичних експериментів і розраховується похибка отриманого результату. Як показано в роботах [1,2], похибка розрахунку може бути зменшена при проведені експериментів для кожної кратності відмов окремо. Крім того, необхідно забезпечити випадковість (або псевдовипадковість) вибору векторів стану системи і відсутність повторень векторів.

Для вирішення даної проблеми, у загальному випадку, необхідні керовані генератори псевдовипадкових послідовностей векторів. Керовані генератори, це такі генератори, для яких можуть бути задані характеристики їх вихідних послідовностей (наприклад, тип розподілу, вага векторів по Хемінгу, відсутність повторень).

У статті [3] був запропонований один з таких алгоритмів, а саме алгоритм генерування псевдовипадкових послідовностей векторів із заданою вагою по Хемінгу, без повторення генерованих векторів, котрий використовує існуючі алгоритми для досягнення псевдовипадковості.

Для того, щоб визначити межі практичного застосування цього алгоритму, потрібно провести аналіз його складності. Основними характеристиками, які мають бути оцінені, є час роботи алгоритму та пам’ять, необхідна для нього [4,5,6,7].

Генерування рівноважних векторів шляхом відображення цілих чисел у рівноважні вектори

Основна ідея, на якій базується алгоритм, полягає у тому, що генерується послідовність псевдовипадкових цілих чисел із заданого проміжку з необхідними статистичними характеристиками. Числа у цій послідовності не повторюються. Згенероване число відображається у вектор, що має задану довжину та вагу [3].

Для можливості відображення цілих чисел у послідовність векторів вони мають бути пронумеровані.

Першим вектором вибирається вектор, у якого m молодших розрядів є одиничними, а інші $n-m$ – нульовими. Останнім вектором є вектор у якого m старших розрядів є одиничними. Наступний вектор для будь-якого вектора, відмінного від останнього, визначається за таким правилом: знаходиться позиція першої справа послідовності "01". Ця пара

замінюється на "10", а усі одиниці, що були справа від знайденої пари, переміщаються та займають молодші розряди вектора (рис. 1).

0000111	0
0001011	1
0001101	2
0001110	3
0010011	4
0010101	5
0010110	6
0011001	7
…	
1011000	29
1100001	30
1100010	31
1100100	32
1101000	33
1110000	34

Рис 1. Нумерація векторів

Необхідною умовою роботи генератора є відсутність повторень у вхідній послідовності. Для цього підходять генератори виду $x_i = f(x_{i-1})$, для $i > 0$, $0 \le x < M \Rightarrow 0 \le f(x) < M$. Такі генератори мають період, що не перевищує M.

Для перетворення номеру вектора власне у вектор використовується матриця ваг розрядів (P) розміром $m \times (n - m + 1)$.

Елементи цієї матриці визначаються за формулою:

$$P_{i,j} = \begin{cases} 0, & j = 0, \\ C_{i+j}^{j-1}, & j > 0. \end{cases} \qquad (1)$$

m						
	0	1	2	3	4	5
	0	1	3	6	10	15
	0	1	4	10	20	35
	0	1	5	15	35	70

$n - m + 1$

Рис 2. Матриця ваг розрядів

Заповнення матриці відбувається послідовно починаючи з лівого верхнього кута з використанням тотожності $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$ (рис. 2).

Перетворення числа x відбувається за таким алгоритмом:

1. Ініціалізація лічильників $i := m-1$ – індекс одиничного розряду, що вибирається, $j := n-m$ – зміщення поточного розряду;
2. Доки елемент $P_{i,j} > x$, $j := j-1$;
3. Поставити 1 в $i+j$ позиції результату, $x := x - P_{i,j}$;
4. Якщо $i > 0$, то $i := i-1$ та перейти до пункту 2;
5. Кінець.

При реалізації алгоритму можна врахувати симетричність матриці ваг розрядів та зберігати неповну матрицю.

Аналіз складності алгоритму

Знайдемо асимптотичну оцінку кількості пам'яті M та часу, необхідного для роботи алгоритму T в залежності від характеристик векторів, що генеруються – довжини n та ваги m.

Для генерування таких векторів використовується матриця ваг розрядів розміром $m \times (n-m+1)$. Для зберігання значень у цій матриці можуть використовувати комірки однакової довжини. Значення у комірці матриці визначається за формулою (1). Максимальне значення буде мати крайня права комірка матриці, виходячи з формули за якою визначається це значення. В цій комірці буде записано значення $C_{n+1}^{n-m} = \frac{(n+1)!}{(n-m)!(n+1)!}$. Величина цього значення дозволяє оцінити кількість бітів, необхідних для збереження даних в одній комірці матриці. За грубу оцінку цього значення можна прийняти $C_{n+1}^{n-m} < 2^n$, що випливає з таких комбінаторних співвідношень: $\sum_{i=0}^{n} C_n^i = 2^n$ та $C_n^m = C_{n-1}^m + C_{n-1}^{m-1}$. Очевидно, що для представлення такого числа у пам'яті потрібно не більше $\log_2(2^n) = n$ біт.

Отже, кількість пам'яті ($M(n,m)$), необхідної для роботи алгоритму,

$$M(n,m) \le nm(n-m+1) \le nm^2$$

Асимптотична оцінка складності алгоритму по пам'яті:

$$M(n,m) = O(nm^2)$$

Запропонований алгоритм може бути модифікований шляхом використання зменшеної матриці ваг розрядів. При цьому асимптотична оцінка не зміниться, оскільки модифікований алгоритм буде відрізняється від немодифікованого на константний множник.

Проведемо оцінку часу, необхідного для роботи алгоритму.

Алгоритм складається з операцій віднімання та порівняння десяткових чисел великої розрядності, операцій доступу до пам'яті (читання матриці ваг розрядів, формування результуючого вектору). Віднімання та порівняння є найскладнішими операціями у цьому алгоритмі. Тому саме кількість таких операцій буде визначати асимптотичну складність алгоритму.

Довжина чисел у двійковому представленні, як було показано, не перевищує n двійкових розрядів. Позначимо складність однієї операції віднімання у залежності від n як $S(n)$, складність операції порівняння позначимо як $C(n)$ (складність цих операції не залежить від m).

Згідно алгоритму порівняння відбуваються послідовно від правого нижнього до лівого верхнього кута матриці (рис. 2); у випадку переходів вверх (зменшення i) відбувається віднімання, а у випадку переходу вліво – порівняння. Таким чином число операцій віднімання дорівнює m, а порівняння $n+1$.

Загальна складність: $T(n,m) = m \cdot S(n) + (n+1) \cdot C(n)$. Враховуючи, що складність операцій віднімання та порівняння пропорційна до довжини операндів, асимптотична оцінка складності алгоритму по часу:

$$T(n,m) = O(n^2)$$

Висновки

Проведена оцінка складності алгоритму дозволяє оцінити кількість пам'яті, необхідної для роботи алгоритму, та час його роботи в залежності від параметрів генерованих векторів. Розглядається гірший випадок. Отримана оцінка є оцінкою зверху.

Алгоритм має поліноміальну складність, що дозволяє його використовувати для розв'язання практичних задач.

Розглянуто декілька модифікацій алгоритму, які дозволяють дещо зменшити ресурси (час процесора та пам'ять), необхідні для його роботи. Але у загальному вони не впливають на асимптотичну складність алгоритму.

Література

1. *Романкевич А.М., Гроль В.В., Карачун Л.Ф., Орлова М.Н., Романкевич В.А.* Об одном подходе к расчету надежности отказоустойчивых многопроцессорных систем.
2. *Гроль В.В., Орлова М.Н., Романкевич В.А*. Об одной особенности тестирования моделей отказоустойчивых многопроцессорных систем при расчете надежности.

3. Науковий вісник Чернівецького національного університету. Комп'ютерні системи та компоненти. - 2010. - Т.1, вип.2. С. 21-24
4. *Кнут Д.Э.* Искусство программирования. Том 2. Получисленные алгоритмы. М.: Вильямс, 2007. – 788с.
5. *Кнут Д.Э.* Искусство программирования. Том 1. Основные алгоритмы. М.: Вильямс, 2008. – 720с.
6. *Donald E. Knuth.* The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams. 2009.
7. *Ахо А, Хопкрофт Дж., Ульман Дж*. Построение и анализ вычислительных алгоритмов. – М.: Мир, 1976. – 536с.