УДК 004.6

АЛГОРИТМ РЕДАГУВАННЯ БІНАРНИХ ДІАГРАМ РІШЕНЬ НА МІСЦІ

Міхав Володимир

Науковий керівник: канд. ф.-м. наук, доцент Паращук С. Д.

Центральноукраїнський державний педагогічний університет імені Володимира Винниченка, м. Кропивницький, Україна

Ефективне представлення матриць у пам'яті комп'ютера дає можливість будувати і опрацьовувати моделі складних об'єктів. Бінарні діаграми рішень компактні, але вимагають значних обчислень для редагування значень У статті ми описуємо спосіб редагування бінарних діаграм рішень на місці, який дає можливість змінити значення діаграми на одному наборі параметрів. Описаний алгоритм дозволяє балансувати кількість обчислень і витрати пам'яті, що сприяє оптимізації витрат ресурсів. Також у статті ми пропонуємо алгоритм управління пам'яттю при роботі з бінарними діаграмами рішень.

Ключові слова: матриця, бінарна діаграма рішень.

Algorithm of editing binary decision diagram in place

V. Mikhav

Scientific supervisor: Candidate of Physics and Mathematics Sciences, Docent Paraschuk S. D.

The Volodymyr Vynnychenko Central Ukrainian State Pedagogical University, Kropyvnytsky, Ukraine

Efficient representation of matrices in computer memory makes it possible to construct and process models of complex objects. In the article, we describe how to edit binary decision diagrams in place, which makes it possible to change the values of the diagram in one set of parameters. The described algorithm allows you to balance the calculation and memory consumption, which helps to optimize the consumption of resources. In addition, we offer a memory management algorithm for working with binary decision diagrams.

Key words: matrix, binary decision diagram, BDD, MTBDD.

Постановка проблеми. Бінарні діаграми рішень (БДР) дають можливість ефективно представляти у пам'яті матриці з великою кількістю повторюваних елементів. Проте зміна елемента матриці, представленої за допомогою БДР – це складна операція, що вимагає повторного синтезу діаграми. Це значно ускладнює застосування БДР у програмах, які вимагають частої зміни значень

матриці. Ми пропонуємо метод зміни БДР, який дає можливість уникнення повторного синтезу.

Аналіз останніх досліджень і публікацій. Проблемі компактного представлення булевих функцій приділяється багато уваги в роботах зарубіжних дослідників. Дональд Кнут у своїй роботі [2] наводить результати ґрунтовних досліджень на тему бінарних діаграм рішень та пропонує псевдокод процедур для виконання базових операцій над БДР. У роботі [1] Ю. Карпов описує основи роботи з БДР та наводить приклади їх застосування, у тому числі представлення матриць за допомогою БДР.

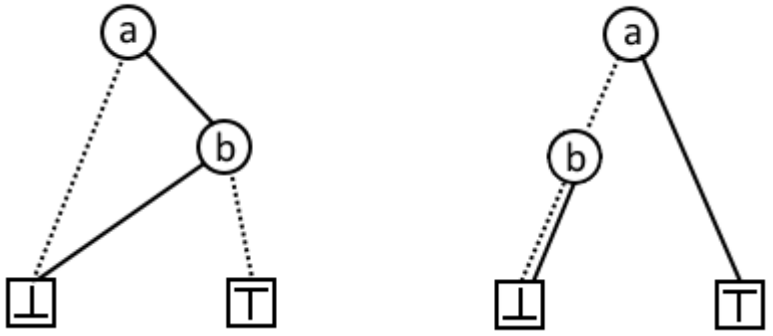
Постановка завдання. Мета статті полягає у висвітленні розробленого нами алгоритму зміни значення БДР на одному наборі параметрів на місці, без повторного синтезу БДР.

Виклад основного матеріалу

Бінарні діаграми рішень (БДР, binary decision diagram, BDD) – це економна форма представлення булевих функцій у вигляді орієнтованого ациклічного графа. Вершини графа представляють аргументи функції, листки – її двійкові значення [1]. При представленні булевих функцій у формі БДР стало можливим розв’язувати багато проблем, які при традиційних представленнях структур нерозв’язні через значну розмірність таких представлень і складність операцій над ними. БДР можуть успішно застосовуватися фактично в кожній галузі, де потрібно обробляти дискретні структури даних, у тому числі і в комбінаториці. Можна вважати, що БДР отримується з бінарного розв’язуючого дерева функції викиданням усіх надлишкових підструктур (ізоморфних підграфів). Щоб позбутися надлишковості, потрібно видалити ізоморфні підграфи (операція вилучення), лишивши лише один, та об’єднати однакові термінальні вершини (операція злиття). Окрім того, якщо з якоїсь вершини обидва виходи ведуть до однієї і тієї ж підструктури, то цю вершину можна вилучити. Повторне застосування операцій злиття і вилучення в будь-якому порядку до бінарного розв’язуючого дерева булевої функції чи будь-якого отриманого з нього графа до тих пір, поки не буде усунено всі

надлишковості, призводить до редукованого представлення цієї функції, яке і називається БДР. Ключовою вимогою до БДР є вимога впорядкованості (для редукованих впорядкованих БДР (РВБДР, ROBDD) послідовність міток змінних функції на всіх шляхах має бути в будь-якому, але спільному глобальному порядку, і на одному шляху змінні не можуть повторюватись).

Для представлення булевих функцій з великою кількістю нулів пропонується модифікація бінарних діаграм рішень - **бінарні діаграми рішень з подавленням нулів** (ZDD, НДР) [2, 293-295]. У цій структурі даних правило видалення дещо змінене: вершина видаляється, якщо нащадок, до якого переходимо при значенні змінної *істина* – це нуль-термінал. При цьому допускається, щоб виходи вершини співпадали. Ця структура даних дає можливість доволі ефективно представляти розріджені графи.



а б

Рис. 1. БДР (а) та НДР (б) для функції a ∧ ¬b

Багатотермінальна бінарна діаграма рішень (MTBDD) – це модифікація бінарних діаграм рішень, яка може мати більше двох терміналів [3]. Завдяки цьому така БДР може представляти функції, у яких область значень – це скінченна множина цілих чисел. Цей вид БДР зручний для представлення векторів і матриць.

Перехід від матриці до БДР. Для представлення вектору за допомогою БДР необхідно представити номери елементів вектору у бінарній системі числення і використати їх як параметр функції. Таким чином, для представлення вектору з 4 елементів функція повинна мати 2 параметра. Для

представлення матриці необхідно додати ще одну групу параметрів для задання рядка матриці. Багатотермінальна БДР дає можливість одразу зберігати цілочисельні значення для матриці. Для представлення значень у звичайній БДР необхідно додати ще одну групу параметрів, за допомогою яких буде кодуватися значення.

Зміна значення БДР. Для виконання зміни значення необхідно згенерувати коригуючу БДР та злити її із існуючою. Алгоритм злиття двох БДР описаний у роботі [2, 259-266]. Кількість вузлів у БДР f позначимо як $B(f)$. Час роботи операції злиття основної БДР f та коригуючої БДР g має порядок $B(f)$ та вимагає розміщення порядку $B(f)$ додаткових вузлів (ми нехтуємо розміром коригуючої БДР $B(g)$, оскільки у загальному випадку він буде значно меншим за $B(f)$). Тому злиття для зміни одного значення БДР нераціональне.

Для оптимізації внесення змін можна використати ідею холодного та гарячого сховищ. У ролі гарячого сховища виступає хеш таблиця, у якій зберігаються останні зміни графу. Холодне сховище (БДР) використовується для збереження всього графу. Такий підхід дає можливість накопичувати зміни і фіксувати їх за необхідності, завдяки чому зменшує кількість операцій злиття БДР. Проте він вимагає значного ускладнення проекту та додаткових витрат пам’яті для реалізації гарячого сховища.

Зміна значення БДР на окремому наборі параметрів. Ми пропонуємо алгоритм зміни значення БДР, який працює на місці, підтримуючи цілісність БДР без операції злиття і надлишкових витрат пам’яті.

Перед внесенням змін до БДР її потрібно підготувати, відсортувавши її вузли за рівнями. Під рівнем вузла БДР ми розуміємо номер параметра функції, якому відповідає цей вузол. Рівень 0 зарезервовано, тому починаємо рахувати рівні з 1. Сортування БДР, яка має n рівнів, здійснюється за наступним алгоритмом:

1. Виділити масив вказівників для підтримки n однозв’язних списків.
2. Для кожного невідвіданого вузла, починаючи з кореня:
   2.1. Занести вузол до списку, який відповідає рівню вузла.

2.2. Помітити вузол як відвіданий.

2.3. Збільшити лічильники посилань для нащадків вузла.

2.4. Повторити для невідвіданих нащадків вузла.

Таке сортування здійснюється за один прохід, тому має складність *O(B(f))*. Воно зв'язує вузли одного рівня між собою, а також дає можливість визначити кількість посилань на кожен вузол. Наший алгоритм редагування залишає вузли відсортованими, тому, якщо між редагуваннями не виконується жодна інша операція, повторне сортування здійснювати не потрібно.

Алгоритм редагування складається з наступних років:

1. Здійснити кишенькове сортування вузлів БДР за рівнями.
2. Прохід від кореню БДР до термінала:
   2.1. Якщо корінь БДР належить не до першого рівня – встановити рівень розділення *forkLevel* на рівень 0.
   2.2. На кожному кроці обрати наступний елемент для переходу залежно від значення параметру функції, якому відповідає поточний вузол. Зафіксувати посилання на обраний елемент у масиві *path*.
   2.3. Якщо *forkLevel* ще не заданий, а наступний елемент – це або термінал, або вузол з кількома батьками, або поточний і наступний вузли відрізняються більш ніж на один рівень, то встановити *forkLevel* на рівень поточного вузла.
3. Знайти термінал, який відповідає новому значенню БДР.
4. Для рівнів від *n* до *forkLevel*:
   4.1. Знайти вузол, на який має продовжувати посилатись нова версія гілки.
   4.2. Якщо вузли з нової і старої гілки співпадають – перейти до наступного рівня. Інакше знайти на поточному рівні вузол, що посилається на потрібні вузли. Якщо такого вузла немає - створити новий.
5. Якщо *forkLevel* посилається на рівень 0 – перевстановити корінь БДР на нову гілку.

6. Для рівнів від *forkLevel* до 1:
   6.1. Якщо вузли нової гілки і *path* почали співпадати – перервати цикл.
   6.2. Якщо вузли з нової і старої гілки співпадають – видалити вузол, зафіксований у збереженому маршруті. Якщо цей вузол був коренем – перемістити корінь на нову гілку.
   6.3. Знайти на поточному рівні вузол, що посилається на потрібні вузли. Якщо такий є – видалити вузол, зафіксований у *path*.

Цей алгоритм потребує лише $O(n)$ пам'яті для збереження маршруту від кореня до терміналу. Складність алгоритму складає $O(B(f))$ у загальному випадку і $\Omega(b(n))$ у найкращому, де $b(i)$ – кількість елементів у i-му рівні, $b(n) \leq 2^t$, t – кількість терміналів.

Поділ БДР на дві частини за рівнем *forkLevel* має важливе значення у підтримці цілісності БДР. Якщо різниця між рівнями батьківського і дочірнього вузлів більше одиниці, або дочірній елемент має кількох батьків, то до дочірнього елементу є декілька маршрутів, тому його не можна видаляти чи редагувати, але можна використати повторно. При цьому батьківський вузол можна редагувати і необхідно видалити, якщо на нього більше не буде посилань.

У випадку, якщо потрібно оновлювати багато значень за один раз, можна зменшити складність зміни одного значення до $O(n)$, якщо додавати нові вузли без перевірки на можливе дублювання. Через це після кожної зміни значення БДР збільшуватиметься на n елементів. Після закінчення внесення змін необхідно видалити усі надлишкові вузли. У роботі [2, 257-259] описано алгоритм, який приводить БДР до коректного формату шляхом видалення надлишкових вузлів.

Керування пам'яттю при роботі з БДР. При редагуванні діаграми потрібно часто видаляти та створювати вузли, тому для покращення швидкодії необхідно мати пул попередньо виділених вузлів, щоб зменшити навантаження на менеджер пам'яті. Збереження вузлів у єдиному масиві дає можливість використовувати номер вузла замість його адреси, завдяки чому розмір вузла

зменшується на 43% (з 28 до 16 байт). Але зміна розміру такого масиву – дуже дорога операція, тому ми використовуємо сторінковий масив – діаграма має масив посилань на сторінки, а кожна сторінка складається з фіксованої кількості вузлів.

Для швидкого пошуку вільних вузлів у сторінці ми створюємо зв'язний список вузлів, зберігши голову списку у параметрах сторінки. Він працює за наступним алгоритмом:

- для виділення вузла беремо номер вільного вузла з голови списку та переміщуємо голову на наступний вузол;
- при видаленні вузла ми приєднуємо його як голову списку.

Для швидкого пошуку сторінки із вільним вузлом ми також створюємо зв'язний список, голова якого зберігається у першій сторінці. Він працює за наступним алгоритмом:

- якщо у заповненій сторінці звільняється вузол, вона заноситься у голову списку;
- новий вузол виділяється зі сторінки в голові списку;
- при заповненні сторінки вона видаляється зі списку;
- якщо список порожній, шукаємо вільне місце у масиві вказівників. Якщо вільного місця немає – розширюємо масив. Після цього створюємо нову сторінку і заносимо її список;
- якщо всі вузли сторінки звільняються, а у списку є інші сторінки – вона видаляється

Запропонований алгоритм керування вузлами використовує невелику кількість додаткової пам'яті, проте значно прискорює виділення і звільнення вузлів, а також зменшує навантаження на системний менеджер пам'яті.

Висновки. У статті описано спосіб реалізації редагування бінарної діаграми рішень «на місці», розглянуто переваги та недоліки такого підходу. Акцент зроблено на особливостях реалізації операцій над БДР для оптимізації використання пам'яті.

Список літератури

1. Карпов Ю. Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем / Юрий Глебович Карпов. – СПб.: БХВ-Петербург, 2010. – С. 295–366
2. Кнут Д. Э. Искусство программирования, том 4, А. Комбинаторные алгоритмы, часть 1 / Дональд Эрвин Кнут; пер. с англ. И. В. Красикова. – М.: ООО "И. Д. Вильямс", 2013. – 960 с.
3. Fujita M. Multi-terminal binary decision diagrams: An efficient data structure for matrix representation / M. Fujita, P. McGeer, J. Yang. // Formal methods in system design. – 1997. – Volume 10. – P. 149–169.