УДК 004.6

РОЗРОБКА АСОЦІАТИВНОГО МАСИВУ ДЛЯ ЗБЕРІГАННЯ РОЗРІДЖЕНИХ ОРІЄНТОВАНИХ ГРАФІВ НА ОСНОВІ БІНАРНИХ ДІАГРАМ РІШЕНЬ З ПОДАВЛЕННЯМ НУЛІВ

Міхав Володимир

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

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

Ефективне представлення розріджених орієнтованих графів у пам'яті комп'ютера дає можливість будувати моделі складних мереж із великою кількістю об'єктів, у тому числі – соціальних мереж. У статті описується спосіб представлення розріджених орієнтованих графів за допомогою асоціативного масиву на основі бінарних діаграм рішень, який дає можливість ефективно зберігати графи великої розмірності, але вимагає значної кількості обчислень для внесення змін. Для оптимізації використання обчислювальних ресурсів використовується додаткове сховище, у якому будуть накопичуватися зміни. Також у статті запропоновано алгоритм управління пам'яттю при роботі з бінарними діаграмами рішень.

Ключові слова: орієнтований граф, асоціативний масив, бінарна діаграма рішень.

Development of an associative array for storage of sparse oriented graphs based on zero-suppressed binary decision diagrams

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 sparse oriented graphs in computer memory makes it possible to build models of complex networks with a large number of objects, including social networks. In the article, we describe the way to represent sparse oriented graphs using associative arrays based on binary decision diagrams. This way allow you to effectively store large dimensional graphs, but requires a large amount of computations to make changes. In order to optimize using of computing resources, we propose the using of an additional repository that will accumulate changes. In addition, we offer a memory management algorithm for working with binary decision diagrams.

Key words: oriented graphs, associative array, binary decision diagram, BDD, ZDD.

Постановка проблеми. Моделі складних мереж, у тому числі – соціальних мереж, часто створюються з використанням розріджених орієнтованих графів. Матриця суміжності – один із найзагальніших способів представлень графів, проте вона вимагає великого об'єму пам'яті, що значно обмежує максимальний розмір моделі. Але, оскільки матриця суміжності – це, фактично, булева функція, її можна ефективно представити за допомогою бінарної діаграми рішень (БДР). Проте БДР визначає лише спосіб збереження даних, тому необхідно обрати абстрактний тип даних, який визначає спосіб взаємодії з даними. У ролі інтерфейсу ми пропонуємо використовувати асоціативний масив.

Аналіз останніх досліджень і публікацій. Проблемі компактного представлення булевих функцій приділяється багато уваги в роботах зарубіжних і вітчизняних дослідників. Дональд Кнут у своїй роботі [3] наводить результати ґрунтовних досліджень на тему бінарних діаграм рішень та пропонує псевдокод процедур для виконання базових операцій над БДР. У роботі [2] 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 способу представлення розріджених орієнтованих графів за допомогою асоціативного масиву на основі бінарних діаграм рішень.

Асоціативний масив – абстрактний тип даних, який дає можливість зберігати дані у вигляді пар (ключ, значення). Він має підтримувати наступні операції:

- INSERT(ключ, значення) – додавання пари;
- FIND(ключ) – пошук значення за ключем;
- REMOVE(ключ) – видалення значення за ключем.

Найчастіше асоціативний масив реалізують за допомогою бінарного дерева пошуку або хеш-таблиці. Наприклад, у стандартній бібліотеці мови C++ контейнер map реалізований на основі червоно-чорного дерева (різновид бінарного дерева пошку). Використання бінарного дерева пошуку дає

можливість використовувати такі операції, як пошук пари із мінімальним чи максимальним значенням ключа та перегляд усіх пар за зростанням чи спаданням ключа. Проте основні операції у цій структурі даних у середньому виконуються за час $O(\log n)$ [4, 308].

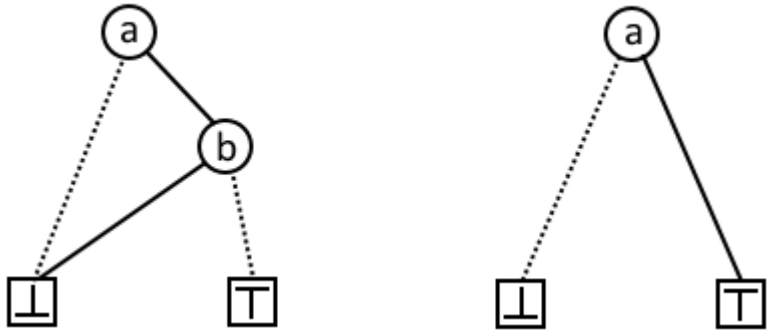
Використання **хеш-таблиці** дає можливість виконувати операції пошуку та видалення в середньому за час $O(1)$. Середній час додавання нової пари також оцінюється як $O(1)$, проте в гіршому випадку час додавання оцінюється як $O(n)$. Час додавання пари зростає зі зростанням коефіцієнту заповненості хеш-таблиці, оскільки збільшується кількість колізій. Тому при досягненні певного значення коефіцієнту заповненості необхідно створити хеш-таблицю більшого розміру та заново додати до неї існуючі пари. Порогове значення коефіцієнту залежить від хеш-функції та способу розв'язання колізій (метод ланцюжків дає можливість заповнювати хеш-таблицю на 300-500%, тоді як при використанні методу відкритої адресації бажано перебудовувати хеш-таблицю при заповнені на 70-80%) [1, 116-128].

Для представлення **орієнтованих графів** можна використовувати різні структури даних. Вибір структури залежить від операторів, які будуть застосовуватися до верших і дуг оргафу. Одним із найбільш загальних представлень є матриця суміжності. Матриця суміжності для орграфу G – це матриця A розміру $n \times n$ зі значеннями булевого типу, де $A[i, j] = true$ тоді і лише тоді, коли існує дуга із вершини i до вершини j.[1, 184] Проте таке представлення вимагає великого об'єму пам'яті, що особливо помітно, якщо граф розріджений. Але, оскільки матриця суміжності – це, фактично, булева функція, її можна ефективно представити за допомогою бінарної діаграми рішень.

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 [2]. 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 розріджених булевих функцій пропонується модифікація бінарних діаграм рішень - **бінарні діаграми рішень з подавленням нулів** (ZDD, НДР) [3, 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

Перехід від матриці суміжності до БДР. Щоб перетворити матрицю суміжності у БДР, необхідно перетворити номери вузлів у групи булевих змінних, номери стовпчиків та рядків нумеруються окремо. Позначимо номери стовпчиків та рядків булевими векторами a та b відповідно. Наприклад, якщо у графі 4 вузли, то третій стовпчик можна позначити як a_1=1, a_0=0, а другий рядок – як b_1=1, b_0=0. Тоді, якщо існує дуга від вузла 3 до вузла 2, булева функція з цими параметрами буде істинною.

Ми орієнтуємось на роботу із розрідженими графами, тому для представлення матриці суміжності ми будемо використовувати НДР.

Реалізація основних операцій асоціативного масиву. Висота НДР залежить від кількості змінних, тому середній час визначення, чи є дуга між вузлами, можна оцінити як $O(1)$. Операції видалення та додавання значень можна реалізувати лише за допомогою виконання булевої операції над двома НДР, тому надалі вони будуть розглядатися як операція зміни значення.

Для виконання зміни значення необхідно згенерувати коригуючу НДР та злити її із існуючою. Алгоритм злиття двох БДР описаний у роботі [3, 259-266]. Кількість вузлів у НДР f позначимо як $B(f)$. Час роботи операції злиття основної НДР f та коригуючої НДР g має порядок $B(f)$ та вимагає розміщення порядку $B(f)$ додаткових вузлів (ми нехтуємо розміром коригуючої НДР $B(g)$, оскільки у загальному випадку він буде значно меншим за $B(f)$), тому внесення змін безпосередньо у НДР нераціонально.

Для **оптимізації внесення змін** ми пропонуємо використати ідею холодного та гарячого сховищ. У ролі гарячого сховища у нас виступатиме хеш таблиця, у якій зберігатимуть останні зміни графу. Холодне сховище (НДР) використовується для збереження всього графу. У такому випадку робота з графом відбуватиметься наступним чином:

- для отримання значення потрібно перевірити, чи є це значення у хеш-таблиці; якщо ні – його необхідно прочитати з НДР;
- змінені значення фіксуються у хеш-таблиці;
- після закінчення логічного блоку операцій або внесення певної кількості змін необхідно створити нову НДР на основі хеш-таблиці та злити її із існуючою.

Для того, щоб замінити старі дані в НДР на нові, не пошкодивши при цьому інші дані, ми скористалися алгоритмом обміну значень двох змінних без використання допоміжної змінної (рис. 2).

```
a  ⊕  0 1 0 1 1 0 1 0
b     1 0 0 0 1 1 0 1
-------------------------
a     1 1 0 1 0 1 1 1

b  ⊕  1 0 0 0 1 1 0 1
a     1 1 0 1 0 1 1 1
-------------------------
b     0 1 0 1 1 0 1 0

a  ⊕  1 1 0 1 0 1 1 1
b     0 1 0 1 1 0 1 0
-------------------------
a     1 0 0 0 1 1 0 1
```

Рис. 2. Обмін значень змінних *a* і *b* без використання допоміжної змінної

Цей алгоритм працює завдяки використанню операції *XOR* (виключне або). Таблиця істинності булевої функції XOR зображена на рис. 3.

x	y	xor
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 3. Таблиця істинності *XOR*

Якщо проінтерпретувати *x* як збережені дані, а *y* – як коригуючі, то стає очевидним, що у коригуючій НДР істинними мають бути встановлені лише ті значення, до яких необхідно внести зміни в існуючій НДР. При цьому коригуюча НДР у більшості випадків буде дуже малою. Для того, щоб мати можливість побудувати коригуючу НДР, нам необхідно у хеш-таблиці зберігати як нове, так і старе значення. Тоді, за допомогою *XOR*, ми зможемо визначити, які саме зміни були внесені.

Збереження ваги дуги. Особливо корисним використання XOR стає, коли ми хочемо зберігати інформацію не лише про факт наявності дуги між вузлами, а й зберігати вагу цієї дуги. Для цього ми пропонуємо виділити додаткову групу змінних, яка відповідатиме за представлення ваг у НДР. Позначимо такі змінні як булевий вектор *c*. Значення 0 резервується для позначення відсутності дуги. Такий підхід не передбачає збереження дробових чисел, проте можна розробити нумерацію дробових чисел і зберігати не саме число, а його номер.

Для того, щоб прочитати значення ваги, під час проходу по НДР необхідно:

- зафіксувати, які змінні з групи *c* трапляються на шляху;
- перевірити, чи створює така змінна альтернативний маршрут: якщо так, то необхідно перевірити обидва маршрути та відкинути хибний.

На основі зафіксованих змінних формується значення ваги. Ми вважаємо, що кожній парі номерів вузлів може відповідати лише одне значення ваги. У загальному випадку ми можемо отримати декілька дійсних маршрутів у НДР і, відповідно, декілька значень. Така ситуація може виникнути, якщо замість

номерів двох вузлів у запиті буде вказано номер вузла та значення ваги: у цьому разі ми зможемо отримати номери усіх вузлів, до яких є дуги від вказаного вузла із вказаною вагою. Можливість створення таких запитів вигідно виділяє НДР серед інших способів представлення графів.

Пошук з порівнянням. Зберігання значень ваг у бінарному вигляді не дає можливості вибирати значення, які більші чи менші за вказане. Проте, якщо планується використовувати невеликі цілі значення ваг, можна перейти від бінарної до унарної системи числення. Це збільшить кількість змінних у НДР, проте мало вплине на її розмір, оскільки більшість нових змінних будуть присутніми в діаграмі лише раз. Завдяки унарній системі числення можна здійснювати вибірки за умовами ≥ та <. Наприклад, щоб вибрати усі дуги з вагою менше 5, достатньо злити основну НДР із функцією $\neg c_5$. Окрему змінну можна виділити для зберігання знаку.

Використання рядкових ключів. У роботі [5] пропонується модифікація НДР - послідовні бінарні діаграми рішень (SeqBDD). Вона дуже зручна для зберігання рядків. Замість змінної у вузлі зберігається предикат $x_k=C$, де x_k – k-ий символ рядка, C – певний символ. Такий тип діаграм зручний для встановлення відповідності між рядками, але має вузли більшого розміру і вигідний лише коли ключі мають спільні підрядки. В інших випадках вигідніше зберігати рядки в іншому сховищі (наприклад у хеш-таблиці), а у НДР зберігати номер рядка.

Алгоритми пошуку кращого порядку змінних. Вагомим недоліком БДР є те, що їм потрібні евристичні алгоритми для впорядкування змінних перед обробкою. Для багатьох булевих функцій відомі евристичні алгоритми попереднього впорядкування змінних не дають можливості оперувати з БДР в обмеженому обсягові пам’яті. Для подолання цієї проблеми використовують динамічне впорядкування змінних.

Річард Рудель у своїй роботі [6] описує два алгоритми пошуку кращого порядку змінних: віконні перестановки та розроблений ним алгоритм просіювання. Алгоритм віконних перестановок вибирає k змінних та здійснює

k! попарних обмінів, вичерпуючи всі можливі перестановки вибраних змінних, а також не більше *k**(*k*-1)/2 обмінів для відновлення кращої знайденої послідовності. Алгоритм просіювання базується на пошуку кращої позиції для змінної з припущенням, що інші змінні лишаються нерухомими. Якщо у БДР є *n* змінних, то є *n* потенційних позицій для розташування змінної (в тому числі і її нинішня позиція). Змінна просувається до одного краю, потім до іншого, а потім до найкращої з переглянутих позицій. В алгоритму просіювання є перевага, яка полягає у тому, що змінна може переміститись на більшу відстань відносно своєї нинішньої позиції. В процесі його роботи розмір діаграми може спочатку зрости, а потім зменшитися нижче початкового рівня.

Під час дослідження було виявлено, що порядок, в якому будуть просіюватися змінні, може впливати на ефективність самого алгоритму. Тому було вирішено здійснювати вибір змінної, яку будемо просіювати наступною, випадково. Це робить результат роботи алгоритму дещо нестабільним (для однієї й тієї ж ВБДР розмір отриманої діаграми може відрізнятись), проте гарантує, що розмір діаграми буде зменшений, якщо це можливо.

Керування пам’яттю при роботі з НДР. Під час проведення операції злиття діаграм 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). Таким чином, ми можемо адресувати 2^{31} вузлів, які займуть 32 ГБ. Але зміна розміру такого масиву – дуже дорога операція, тому ми використовуємо сторінковий масив – діаграма має масив посилань на сторінки, а кожна сторінка складається із 1024 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 асоціативного масиву для зберігання розріджених орієнтованих графів на основі бінарних діаграм рішень з подавленням нулів, розглянуто переваги та недоліки такого підходу. Акцент зроблено на особливостях реалізації НДР для оптимізації використання пам'яті.

Список літератури

1. Ахо А. Структуры данных и алгоритмы / А. Ахо, Д. Хопкрофт, Д. Ульман; пер. с англ. А. Минько. – М.: ООО "И. Д. Вильямс", 2000. – 384 с.

2. Карпов Ю. Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем / Юрий Глебович Карпов. – СПб.: БХВ-Петербург, 2010. – С. 295–366

3. 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.

4. Introduction to Algorithms / T.Cormen, C. Leiserson, R. Rivest, C. Stein; 3-rd edition – The MIT Press, 2009. – 1292 p.

5. Loekito E. A Binary decision diagram based approach for mining frequent subsequences / E. Loekito, J. Bailey, J Pei – Knowledge and Information Systems, 2009. – Volume 24. – Issue 2. – P. 235-268

6. Rudell R. Dynamic Variable Ordering for Ordered Binary Decision Diagrams / Richard Rudell. // Proceedings of the 1993 IEEE/ACM International Conference on Computer-aided Design. – 1993. – P. 42–47.