Порівняння стратегій редагування бінарних діаграм рішень для роботи з графовими даними

Володимир Міхав
Кафедра кібербезпеки та програмного забезпечення, Центральноукраїнський національний технічний університет,
м. Кропивницький, Україна
mihaw.wolodymyr@gmail.com

Єлизавета Мелешко
Кафедра кібербезпеки та програмного забезпечення, Центральноукраїнський національний технічний університет,
м. Кропивницький, Україна
elismeleshko@gmail.com

Abstract. In this work, we research binary decision diagrams' editing. We describe the main algorithm, show the evolution way of the BDD editing strategies, and compared their productivity.

Ключові слова: бінарні діаграми рішень, BDD, графи, зберігання даних, обробка даних.

Вступ

Бінарні діаграми рішень (БДР) – це економна форма представлення булевих функцій у вигляді орієнтованого ациклічного графу. Вершини графу представляють аргументи функції, листки – її двійкові значення [1]. Можна вважати, що БДР отримується із бінарного дерева рішень шляхом вилучення усіх надлишкових підструктур (ізоморфних підграфів). За допомогою БДР можна зберігати лише булеві та псевдобулеві функції, а також дані, що можуть бути приведені до таких даних, наприклад граф соціальної мережі. БДР дають можливість зберігати дані у стисненому вигляді та швидко отримувати значення функції за її параметрами, але редагування БДР вимагає складних обчислень.

Основна частина

Для занесення нового значення до БДР або зміни існуючого, її необхідно відредагувати. Ми використовуємо наступний алгоритм редагування БДР:

1. Створити термінал з новим значенням, якщо він ще не існує.
2. Здійснити спуск від кореню БДР до терміналу відповідно до значень параметрів.
3. Визначити перший рівень l, на якому вузол належить кільком різним значенням. Такий вузол має кілька посилань з верхніх рівнів або відсутній у рівні (відповідно до правил побудови БДР це означає, що цей вузол мав однакових нащадків). Якщо такого рівня немає – направити останній вузол на новий термінал і припинити роботу.
4. Побудувати нову гілку, яка за вказаними параметрами вказуватиме на новий термінал. Для підтримки редукованості БДР ми маємо використовувати вузли повторно. Для рівнів від l і вище гарантується, що вузол не використовується у іншій гілці, тож за необхідності його можна змінити або видалити.

Рівні БДР утворені за допомогою однозв'язних списків. Кожен вузол має посилання на свого сусіда у рівні. Один рівень об'єднує усі вузли, які відносяться до однієї булевої змінної. Під час тестів на великих об'ємах даних було виявлено, що цей підхід вимагає дуже багато обчислювальних ресурсів, оскільки для пошуку вузлів для повторного використання або видалення вузлів із однаковими нащадками доводиться здійснювати повний обхід рівня.

Перша застосована нами оптимізація полягала у винесенні підтримки редукованості із процедури редагування. Це спричиняє більші витрати пам'яті через створення надлишкових вузлів, проте дозволяє рідше виконувати складну процедуру підтримки редукованості.

Перший алгоритм, який було застосовано

для підтримки редукованості – алгоритм, запропонований Д. Кнутом у роботі [2]. Він дає можливість видалити зайві вузли за один прохід без застосування додаткової пам'яті. Але для роботи цього алгоритму потрібні поля, які ми використовуємо для зв'язку вузлів у рівні, тому після редукування БДР потрібно здійснити ще один обхід для відновлення списків. Подальше тестування показало, що редукування на місці сповільнює роботу редагування. Тому після редукування було додано ще один крок, який полягає у наступному:

- Виділяється блок пам'яті, еквівалентний блоку, виділеному для вузлів БДР.
- БДР відтворюється у новому блоці рівень за рівнем.
- Старий блок пам'яті видаляється і замінюється новим.

Таким чином вузли групуються у пам'яті за рівнем, завдяки чому збільшується швидкість оперування даними.

Наступна оптимізація полягає у покращенні кроку компресії вузлів. Оскільки компресія вимагає повного обходу усіх вузлів за рівнями, цей обхід можна використати і для пошуку надлишкових вузлів. При копіюванні вузла у нову область пам'яті у старий вузол заноситься його позиція у новому блоці пам'яті. Якщо нові позиції дочірніх вузлів співпадають – вузол не копіюється, а його нова позиція позначається як нова позиція дочірнього вузла. Для пошуку дубльованих вузлів використовується хеш-таблиця – значенням виступає позиція вузла, а ключем – хеш від позицій дочірніх вузлів. Таким чином, якщо вузол із такими дочірніми вузлами уже занесений до хеш-таблиці, то його не потрібно заносити до нової області пам'яті знову. При цьому підході уже немає необхідності у окремому кроці редукування, що дає можливість відмовитися від двох повних обходів БДР, але вимагає додаткової пам'яті для зберігання хеш-таблиці для одного рівня БДР.

Такий підхід до пошуку надлишкових вузлів дозволив здійснити наступну оптимізацію. Оскільки хеш-таблиця дає можливість знайти вузол за позиціями дочірніх вузлів за сталий час, немає необхідності у підтримці зв'язку між вузлами у одному рівні, тому і видалення вузла можна здійснити за сталий час. У нашій реалізації вузол хеш-таблиці займає 8 байтів, що вдвічі менше ніж вузол БДР. При цьому, для підтримки нормальної швидкодії хеш-таблиці, вона має бути заповнена не більш ніж на 50%. Таким чином, хеш-таблиця займатиме стільки ж місця, як і БДР. Проте цей підхід позбавляє необхідності здійснювати компресію вузлів, тому хеш-таблиця буде зберігатися на місці, яке займала б копія БДР.

Таблиця 1. Порівняння часу, витраченого для заповнення БДР при різних стратегіях редагування

Кількість записів	Стратегія редукування і компресії, мс	Стратегія редуку-вання під час компресії, мс	Стратегія хешу-вання, мс
2621440	215	1024	670
16777216	17836	10872	6446
50331648	66188	34761	20686
100663296	147702	74708	40344

У таблиці вище наведено порівняння швидкодії різних стратегій редагування БДР. Як бачимо, стратегія хешування дала найкращі результати. Стратегія редукування і компресії працює за найдовших час, оскільки вимагає найбільшої кількості обходів вузлів БДР.

Висновки

Було реалізовано та порівняно декілька стратегій редагування бінарних діаграм рішень. Показано, що стратегія хешування вузлів БДР має найвищу швидкість роботи, проте вимагає більшого об'єму пам'яті на постійній основі, а стратегія редукування і компресії БДР має найменшу швидкодію.

Література

[1] Карпов, Ю.Г. (2010), "Model checking. Верификация параллельных и распределенных программных систем", СПб.: БХВ-Петербург, С. 295-366

[2] Кнут Д.Э. (2013) "Искусство программирования, Том 4А. Комбинаторные алгоритмы, часть 1", М.: Вильямс, 960 с.