МЕТОДИ І АЛГОРИТМИ СУЧАСНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

УДК 621.382

Р. Базилевич, М. Влах, Н. Пелих
Національний університет “Львівська політехніка”,
кафедра програмного забезпечення

ОСОБЛИВОСТІ ОПРАЦЮВАННЯ ДАНИХ ДЛЯ ІЄРАРХІЧНОЇ КЛАСТЕРИЗАЦІЇ СКЛАДНИХ СХЕМ

Пропонується гнучкий і універсальний підхід для опису електричних схем та дерева згортання, за яким можна оптимізувати виконання ключових етапів ієрархічної кластеризації.

Ключові слова – ієрархія, кластеризація, структури, дані

A flexible and universal approach for presentation of electric circuits and reduction tree, which can optimizes the performance of key stages of hierarchical clustering is proposed.

Keywords - hierarchy, clustering, structures, data

Вступ

Існує багато прикладних задач великої розмірності, що належать до важкорозв’язуваних комбінаторних задач (клас NP). Ефективним підходом для отримання якісних розв’язків є ієрархічна кластеризація. Суть цього підходу полягає у розбитті задачі на частини (кластери) за певними критеріями і оперування частинами при пошуку розв’язку. Це дає змогу зменшити затрати часу на пошук наближеного до оптимального розв’язку та отримати якісні розв’язки для задач великих розмірностей, де інші методи є менш придатними.

Ієрархічна кластеризація схеми надає інформацію про внутрішню її структуру, про існуючі взаємозалежності між її компонентами. Аналіз дерева кластерів та його розрізи формують початкове розбиття схеми. Таку інформацію можна надалі використовувати для організації процесів обміну між кластерами та їх фрагментами, що дасть змогу покращити якість розв’язків.

Описано організаційну структуру даних для ієрархічної кластеризації. Запропоновано гнучкий і ефективний метод для процесу згортки, який надає інформацію про структуру ієрархічно вкладених кластерів.

Опис вхідних даних

Опис схем з багатоарними зв’язками (гіперграф) подається у вигляді файлів [1], які містять таку інформацію:

- кількість вузлів схеми;
- кількість міжвузлових зв’язків;
- опис зв’язків множиною складових елементів;

Опис зв’язків надано переліком, в якому кожен з них представляється множиною складових елементів:

$$e_i = \{p_{i1}, p_{i2}, \dots p_{in}\},$$

де e_i – ідентифікатор зв’язку (його порядковий номер); $p_{i1}, p_{i2}, \dots p_{in}$ – елементи схеми, які об’єднує зв’язок.

При об’єднанні кількох елементів в кластер зв’язки поділяють на зовнішні, внутрішні та комбіновані [2]. Зовнішні зв’язки містять один елемент в кластері, а інші поза його межами. Внутрішні зв’язки складаються лише з елементів єдиного кластера. Комбіновані зв’язки містять частину елементів в кластері (більше одного), а решту за його межами. На рис. 1 наведено приклад умовного представлення електричної схеми у вигляді гіперграфу. Тут присутні *n*-арні ($e_1 = \{p_1, p_2, p_3\}$, $e_2 = \{p_1, p_2, p_6\}$) і бінарні ($e_3 = \{p_2, p_3\}$, $e_4 = \{p_4, p_5\}$, $e_5 = \{p_4, p_6\}$) зв’язки.

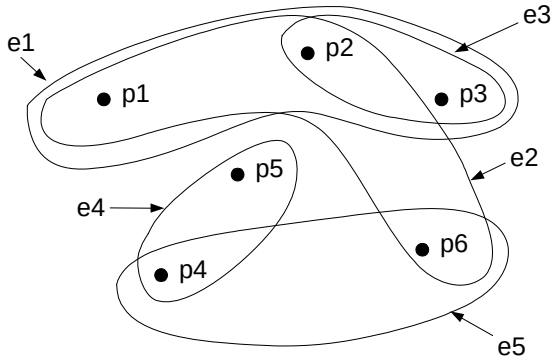

Рис. 1. Представлення електричної схеми

Дерево згортання

Результатом роботи алгоритму ієрархічної кластеризації є деревиста структура – дерево згортання [2, 3, 4], яка відображає взаємне входження елементів та кластерів. У процесі програмної реалізації життєвий цикл кластера розпочинається тоді, коли два вузли схеми об’єднуються в спільну структуру, і завершується в момент його включення в інший кластер. Для відображення такої інформації в пам’яті комп’ютера використовують рекурсивні структури. Оскільки кластер може містити інші кластери, а також елементи схеми, то варто помістити в його опис посилання на структурні компоненти, які його утворили. Це дасть змогу маніпулювати отриманою структурою як прямим аналогом дерева згортання без затрат на додаткові перетворення.

Кластери складаються із елементів схеми та інших кластерів (рис. 2), що викликає певні незручності, пов’язані із програмною реалізацією. Суть проблеми полягає в тому, що дві різні структури (два різні класи в об’єктно-орієнтованих мовах програмування) мають опрацьовуватись однотипно в межах єдиного контейнера. Усунути такий недолік можна по-різному, використовуючи "алгоритмічні умовності", механізми наслідування тощо.

Запропоновано використовувати певну абстракцію, суть якої полягає в тому, що елемент схеми можна представити у формі кластера 0-го рівня дерева згортання, який містить лише його. Така концепція є гнучкою, універсальною і функціонально придатною для багатьох прикладних задач. Це скорочує обсяг програмного коду і зменшує затрати часу та ресурсів на додаткові операції, які необхідно вводити при використанні інших підходів.

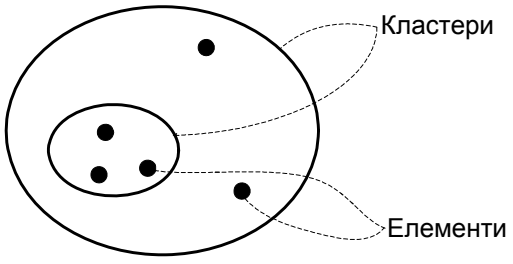

Рис. 2. Структура кластера

Опис структур даних

Визначимо дві основні структури даних для опису схеми та процесу її кластеризації: контейнер зв’язків і кластер, які в поєднанні із деякими допоміжними будуть використовуватись для реалізації процесу ієрархічної кластеризації. Зв’язки схеми необхідно зберігати в пам’яті так, щоб зменшити необхідність їх постійної модифікації при кластеризації. Представлятимемо їх списком, який містить інформацію про номер зв’язку та список посилань на інцидентні до нього елементи. В процесі згортки елементи схеми об’єднуються в кластери, і на наступних ітераціях необхідно замість них розглядати вже спільний кластер. Модифікувати для цієї мети зв’язки недоцільно, оскільки це вимагає додаткових обчислювальних затрат. Для підтримки записів про зв'язки в актуальному стані запропоновано додати до кожного запису лічильник кількості наявних елементів зв’язку на цій стадії процесу згортки (*UseCounter*). На початку його значення дорівнює числу елементів, які об’єднує зв’язок. В процесі згортки цей лічильник модифікується. Його значення постійно відображатиме число елементів, які об’єднує зв’язок для поточного рівня дерева згортання. На рис. 3 наведено схематичне зображення структури, яка описує зв’язки схеми.

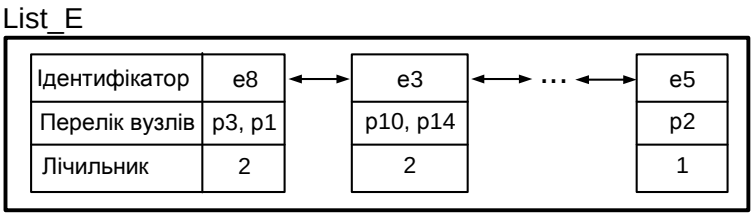

Рис. 3. Структура даних для опису зв’язків схеми

Впорядкованість зв’язків за зростанням їх ідентифікатора (порядкового номера) не є обов’язковою умовою, так само, як і впорядкованість посилань на вузли схеми. Необхідність впорядкування записів визначає алгоритм кластеризації, який використовує ці структури. Цей спосіб представлення зв’язків зручний тим, що дає змогу заповнити інформаційні поля за одну операцію зчитування вхідного файла.

Основним об’єктом маніпуляції в процесах згортки є елемент і кластер. У цьому алгоритмі ці поняття ідентичні і представляються єдиною сутністю – кластером (*Cl*). Елемент схеми – це кластер 0-го рівня дерева згортання. Для опису кластерів та структури їх взаємного входження використовується така інформація:

1. Список кластерів, які його утворили (*CompList*).
2. Списки зв’язків кластера:

– список внутрішніх зв’язків (IntList);

– список інших зв’язків (ExtCombList), який містить інформацію про комбіновані і зовнішні зв’язки кластера, які позначено маркерами: Comb – комбінований, Ext – зовнішній.

Перелік зв’язків відсортовується за зростанням їх порядкового номеру.

3. Перелік посилань на кластери (*NeighbList*). Перелік необхідний для оцінки доцільності згортання пар кластерів.

При згортанні схеми можна використати декілька критеріїв згортання *Cr*, більшість з яких маніпулює трьома числовими величинами:

- число зовнішніх зв’язків потенційного кластера (*NExt*);
- число внутрішніх зв’язків (*NInt*);
- число комбінованих зв’язків (*NComb*);

Список *NeighbList* містить також значення цих параметрів.

4. Список кластерів *ClList*, в якому вони впорядковуються за зростанням ідентифікаторів. Це унеможливлює дублювання інформації про пари в кластерах, наприклад, пари (p_1, p_2) і (p_2, p_1) в кластерах Cl_1 і Cl_2 відповідно. Поточний кластер міститиме в списку *NeighbList* інформацію лише про пов'язані з ним кластери, які мають порядковий номер, більший за його власний. Відповідно кластер "знає" параметри згортки лише для своїх сусідів, що знаходяться з правого боку від нього в списку *ClList*. Тобто скорочується об'єм використовуваної пам'яті і за рахунок зменшення інформації знижуються затрати часу її опрацювання та число вимушених змін в структурах даних.

5. Список *LeftNeighbList*, який містить інформацію про кластери, які знаходяться зліва від кожного кластера списку *ClList* . Цей список надає інформацію про оточення кластера.

6. Впорядкована черга кластерів Rat^{Cr} побудована за критерієм *Cr*. Кожний кластер оцінює усі пари, які він може утворити із сусідніми кластерами за певним критерієм, і пропонує найкраще з-поміж доступних значень критеріїв. Відповідно до цього значення він займає своє місце в черзі Rat^{Cr}. Це оптимізує вибір потрібного кластера для згортки.

Рейтинг можна задати як спискову структуру, елементи якої містять два поля: *значення критерію (ключ)* і *посилання на кластер*. Оскільки критеріїв може бути декілька і вони можуть застосовуватись одночасно, то і рейтингів може бути декілька. Є можливість використання суміщених за кількома критеріями рейтингів, їх лексикографічне впорядкування. Допускається використання підпорядкованих рейтингів і рейтингів, організованих в черги.

Інформація про значення критеріїв для пари кластерів (p_i, p_j) міститься в кластері p_i, відповідно до принципу впорядкованості кластерів за ідентифікаційним номером (тут $i < j$). Кожен кластер містить набір внутрішніх рейтингів сусідніх кластерів *IntRat*, впорядкованих за значенням критерію. Перші елементи цих рейтингів визначають положення кластера в чергах *Rat*.

7. Множина вказівників на його позицію в кожній із рейтингових черг *Pointers to Rat*. Це дасть змогу прискорити процес видалення кластера із черги.

Структуру даних для кластера наведено на рис. 4. Усі кластери нульового рівня створюються після зчитування з файла значення N_p (число вузлів) та заносяться в список *ClList* і під час зчитування зв'язків одночасно заповнюються. Підготувати усю інформацію про схему для проведення ієрархічної кластеризації можна за один цикл читання файла з даними.

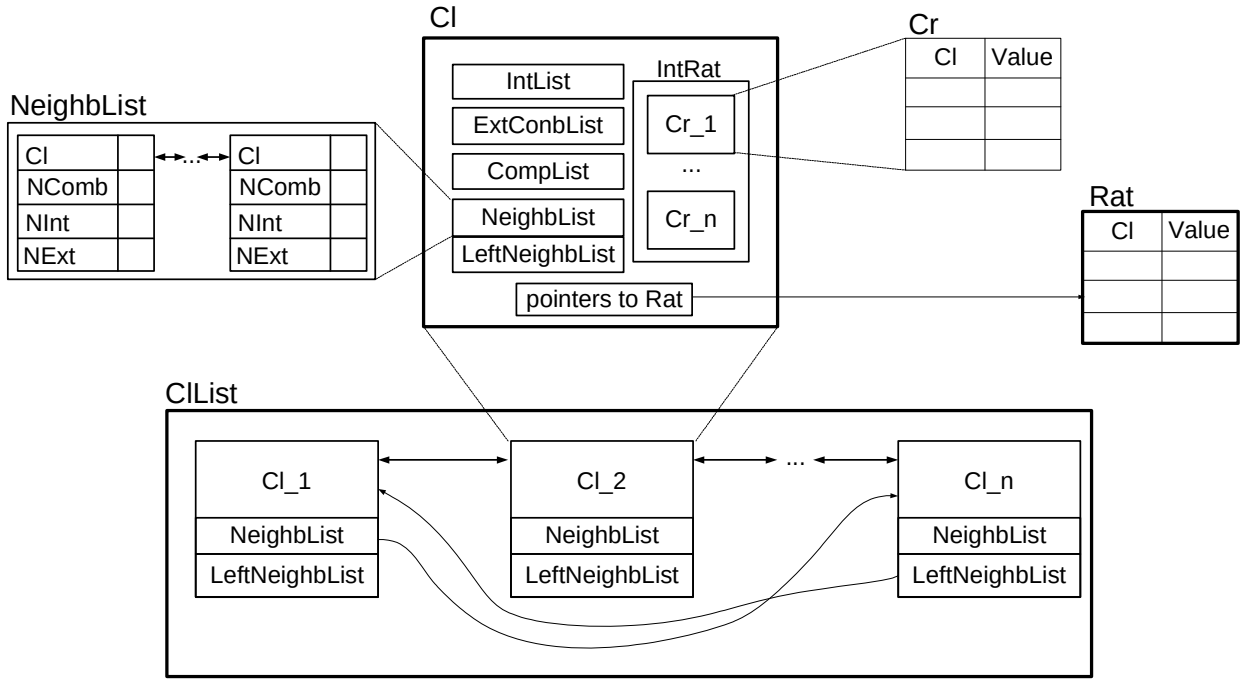

Рис. 4. Структура даних для кластера

Відображення дерева згортання

Запропоновані структури даних для ієрархічної кластеризації надають інформацію про процес згортки і можливості обходу дерева згортання. На останній ітерації отримується один або декілька кластерів, які представляють всю схему. Декілька кластерів на останньому кроці згортання виникають у тому випадку, якщо схему утворено незв'язаними частинами. Замість одного утворюються декілька дерев згортання («ліс»). Описи кластера міститимуть посилання на кластери попереднього рівня, які їх сформували, тобто на своїх "предків" тощо. Рекурсивний ітераційний процес відновлення компонентів кластерів створює деревисту ієрархічну структуру, яка формує дерево (ліс) згортання схеми, що може бути використано для розв'язання таких задач: розбиття схеми на частини, пакування в підсхеми з заданими обмеженнями, розміщення елементів, модуляризації та тестування програмного забезпечення та інших.

Висновки

Ієрархічна кластеризація – складний і ресурсозатратний процес, який вимагає значного обсягу часу на виконання. Проте вона здатна надати інструментарій для вирішення широкого кола задач комбінаторної оптимізації високих та надвисоких розмірностей. Запропонований принцип організації структур даних відповідає поставленим вимогам. Його гнучкість полягає у наявності можливості використання широкого кола критеріїв згортки, можливості одночасного їх застосування, динамічної зміни. Підхід передбачає можливість одночасного злиття більш ніж двох кластерів. Ефективність розробленого підходу обумовлена скороченням числа операцій, потрібних для злиття кластерів, обчислення критеріїв згортки, пошуку кращого рішення. Скорочено число операцій, покликаних оновлювати інформацію в кластерах і зв'язках після кожного злиття кластерів. Ієрархічність отриманої структури надає необхідні засоби для аналізу дерева згортання. Можливості протоколювання процесів згортки за допомогою додаткових компонентів надає належну інформацію про структуру і властивості задачі в цілому, а також її локальні характеристики.

1. Charles J. Alpert. The ISPD98 circuit benchmark suite, IBM Austin Research Laboratory. 2. Bazylevych R.P., Melnyk R.A., Rybak O.G.Circuit Partitioning for FPGAs by the Optimal Circuit Reduction Method, Vlsi Design, OPA (Overseas Publishers Association) N.V., 2000. 3. Базилевич Р.П. Декомпозиционные и топологические методы автоматизированного конструирования электронных устройств. – Львів: Вища школа, 1981. – 168 с. 4. Bazylevych R. The Optimal Circuit Reduction Method as an Effective Tool to Solve Large and Very Large Size Intractable Combinatorial VLSI Physical Design Problems. – 10th NASA Symposium on VLSI Design, 2002.