УДК 519.8
Р.М. Небесний, І.В. Свистун, Р.З. Золотий, канд техн. наук, доц.
Тернопільський державний технічний університет імені Івана Пулюя

ЗАСТОСУВАННЯ ОРІЄНТОВАНОГО АЦИКЛІЧНОГО ГРАФА

R. Nebesnyy, I. Svistun, R. Zolotyi, Ph. D., Assoc. Prof.

APPLICATION OF ORIENTED ACYCLIC GRAPH

Кожен орієнтований ациклічний граф має топологічне впорядкування вершин таке, що початкова вершина кожного ребра проходить раніше в сортуванні, ніж кінцева. Загалом, це сортування не є унікальним; DAG має унікальне топологічне упорядкування тоді і тільки тоді, якщо він має орієнтований шлях, що містить всі вершини, в цьому випадку сортування таке ж, як у порядку, в якому вершини з'являються на шляху.

Комбінаторне завдання перелічення графа для підрахунку спрямованих ациклічних графів вивчав Робінсон в 1973 р.. Кількість DAG на позначеному вузлі n, де n = 0, 1, 2, 3, ..., (ці цифри можуть з'являтися в будь-якому порядку в топологічної впорядкованості DAG) становить 1, 1, 3, 25, 543, 29281, 3781503, ... Це також можна обчислити за допомогою рекурентного співвідношення

$$a_n = \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k} 2^{k(n-k)} a_{n-k} \qquad (1.1)$$

Ерік У. Вайнштайн припустив, а Маккей в 2004 р. довели [60], що ті ж числа будуть отримані в результаті обчислення бінарної матриці, для якої всі власні числа є додатними і дійсними.

Орієнтоване дерево це спрямований граф, утворений шляхом орієнтації ребра вільного дерева. Кожне орієнтоване дерево є DAG. Зокрема, це відноситься і до деревоподібних структур, сформованих орієнтацією всіх ребер в напрямку від кореня дерева. Multi-tree-structured граф являє собою орієнтований граф, в якому існує не більше одного орієнтованого шляху (в обидва напрямки) між будь-якими двома вузлами; еквівалентом є DAG, в якому для кожного вузла v, множина вузлів, досяжних з v утворює дерево.

Є ряд задач, які можна вирішити за допомогою DAG, оскільки він грає ключову роль в алгоритмі розв’язку. В роботі розглянуто саме такі алгоритми і тому ці випадки потребують детальнішого аналізу.

Алгоритмічна задача знаходження топологічного сортування може бути вирішена за лінійний час. Алгоритм Кана для топологічного сортування знаходить порядок вершин безпосередньо, шляхом збереження списку вершин, які не мають ребер, що з'єднують їх з вершинами, які ще не були перераховані, і повторно додає одну з таких вершини до кінця списку, який будується. Крім того, топологічний порядок може бути побудований за допомогою алгоритму пошуку в глибину. Також можна перевірити, який із даних орієнтованих графів є DAG за лінійний час декількома способами: або спробувати знайти топологічне сортування, а потім перевірити для кожного ребра чи є результат дійсним, або, як альтернатива, для деяких топологічних алгоритмів сортування, перевіривши, що алгоритм успішно сортує всі вершини не видаючи помилку умови.

Також складною задачею є побудова DAG. Будь-який неорієнтований граф може бути перетворений в DAG, якщо вибрати загальний порядок його вершин і орієнтацію кожного ребра з початкової кінцевої точки до більш пізньої кінцевої точки. Тим не менше, різні загальні порядки сортувань можуть привести до однієї ациклічної орієнтації. Кількість ациклічних орієнтацій рівна $|X(-1)|$, де X є хроматичним многочленом даного графа.

Будь-який орієнтований граф може бути перетворений в DAG видаленням набору зворотних вершин або набору зворотних дуг. Тим не менш, знаходження найменшого такого набору це NP-складна задача. Довільний орієнтований граф може бути перетворений в DAG, це називається стисненням, шляхом перетворення кожного з його сильно зв'язних компонентів в одну супер вершину. Коли граф вже є ациклічним, його найменша множина зворотних вершин і множина зворотних дуг порожня, і його стиснення є самим графом.

Транзитивне замикання даного DAG, з n вершинами і m ребрами, може бути побудоване за час O(mn) за допомогою або пошуку в ширину, або пошуку в глибину, щоб перевірити досяжність з кожної вершини. Крім того, воно може бути вирішене за час $O(n^{\omega})$, де $\omega < 2{,}373$ є показником для алгоритмів швидкого множення матриць; це теоретичне поліпшення в порівнянні з граничним O(mn) для щільних графів.

У всіх цих алгоритмах транзитивного замикання можна виділити пари вершин, яких можна досягнути шляхом довжиною два або більше з пар, які можуть бути пов'язані шляхом одиничної довжини. Транзитивне скорочення складається з ребер, що утворюють шлях одиничної довжини, які є єдиними шляхами, що з'єднують їх кінцеві точки. Таким чином, транзитивне скорочення може бути побудоване в тих же межах асимптотичного часу, що і транзитивне замикання.

Задача замикання приймає в якості вхідних даних орієнтований ациклічний граф з вагами на його вершинах і шукає мінімальну (максимальну) вагу закриття набір вершин з яких не виходять ребра. ЇЇ можна вирішити за поліноміальний час, використовуючи скорочення задачі про максимальний потік.

Деякі алгоритми спрощуються при використанні DAG замість загальних графів, який базується на принципі топологічного сортування. Наприклад, можна знайти найкоротший шлях і найдовший шлях від деякої вершини DAG за лінійний час шляхом обробки вершин в топологічному порядку, і обчислення довжини шляху для кожної вершини, який буде мінімальною або максимальною довжиною, отримується за допомогою будь-якого з його вхідних ребер. На відміну від цього, для довільних графів знаходження найкоротшого шляху може вимагати повільніших алгоритмів, таких як алгоритм Дейкстри або алгоритм Беллмана-Форда, а знаходження найдовшого шляху в довільному графі має NP-складність.

Представлення DAG часткового порядку часто використовується в задачах планування для систем завдань з обмеженим впорядкуванням. Наприклад, DAG можуть бути використані для опису залежностей між клітинами таблиці: якщо одна комірка обчислюється за формулою з участю значення другої комірки, потрібно намалювати ребро DAG з другої комірки до першої. Якщо вхідні значення для таблиці змінюються, всі інші значення таблиці можуть бути перераховані за допомогою однієї оцінки на клітку, шляхом топологічного сортування клітинки і переоцінки кожної клітинки згідно обрахованого порядку. Аналогічні проблеми впорядкування завдань виникають в make-файлах для компіляції програми, планування інструкцій для низькорівневої оптимізації програми і планування технічного оцінювання та аналізу програми для управління великими проектами. Залежні графи без циклічних залежностей формують орієнтовані ациклічні графи.

Література

1. Tisseur F. Parallizing The devide and conquer algorithm for the symmetric tridiagonal eigenvalue problem on distributed memory architechtures / F. Tisseur, J. Dongarra // SIAM J. SCI. COMPUT, 1998. – Vol. 20. – C. 2223–2236.

2. Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide / [Z. Bai, J. Demmel, J. Dongarra та ін.]. – Philadelphia: SIAM, 2000. – 440 с.