УДК 519.8

В.А. Турчина, Ю.С. Зозуля, А.К. Підаш
Дніпропетровський національний університет імені Олеся Гончара

АЛГОРИТМИ ПЕРЕРАХУВАННЯ ВСІХ ПАРАЛЕЛЬНИХ УПОРЯДКУВАНЬ ФІКСОВАНОЇ ДОВЖИНИ

Запропоновані нові алгоритми направленого перебору для побудови множини всіх допустимих паралельних впорядкувань заданої довжини.

Предложены новые алгоритмы направленного перебора для построения множества всех допустимых параллельных упорядочений заданной длины.

New algorithms of the directed search for construction of the set of all possible parallel orderings of desired length are proposed.

Ключові слова: паралельні впорядкування, теорія розкладів, оптимальні впорядкування.

Вступ. Серед задач дискретної оптимізації особливе місце належить задачам теорії впорядкувань. Оптимальні розв'язки спеціальних дискретних задач на графах дозволяють економити ресурси у системах, моделі яких являють собою завдання, на послідовність виконання яких накладено умови часткового порядку. Прикладами таких систем є послідовності робіт, послідовності операцій, які виконуються ЕОМ. У зв'язку з активним розвитком багатопроцесорних ЕОМ та багатоядерних процесорів задачі теорії впорядкувань стають дедалі більш затребуваними. Розв'язання оптимізаційних задач теорії впорядкувань дозволяє суттєво економити ресурси (кількість робітників, процесорів, ядер процесора) або ж суттєво скорочувати час, необхідний для виконання всіх робіт. Якщо говорити про ЕОМ, то, розв'язавши задачу теорії впорядкувань, ми можемо збільшити швидкодію, а це сприятиме більш ефективній роботі людини за комп'ютером. То ж ми можемо досягти більшої продуктивності праці. Також стає можливим оптимальне використання наявних ресурсів. Так, у випадку ЕОМ, для того щоб завдання виконалося швидше, не обов'язково збільшувати кількість процесорів або ядер процесора, часто достатньо скласти розклад виконань завдань так, аби відповідне впорядкування було оптимальним.

Задачі теорії впорядкувань мають для дослідників як теоретичний інтерес, оскільки вдається отримати цікаві математичні результати, що стосуються дослідження моделей, множин допустимих розв'язків, оригінальні підходи до методів розв'язання цих задач, так і практичний, оскільки багато реальних задач можна звести до задач теорії впорядкувань.

За допомогою алгоритмів зможемо побудувати всі допустимі впорядкування для заданого графа, і після цього, за необхідності, серед них обрати саме ті, що задовольняють нас за деякими критеріями.

Постановка задачі. Введемо до розгляду граф $G = \{V, U\}$, де $V = \{1,2,...,n\}$ – множина вершин, а U – множина дуг.

Впорядкування S множини V вершин графа G назвемо паралельним впорядкуванням вершин орграфа G, якщо з того, що $(i, j) \in U$ випливає, що вершина i знаходиться в S лівіше за вершину j.

Для існування паралельного впорядкування необхідно й достатньо, щоб граф G був ациклічним.

Будемо називати впорядкування оптимальним, якщо воно або при заданій довжині має мінімальну ширину, або при заданій ширині – мінімальну довжину [1].

Можна показати, що цим оптимізаційним задачам на графах можна однозначно поставити у відповідність оптимізаційні задачі теорії розкладів. Надалі будемо будувати оптимальні розклади за допомогою вирішення відповідних оптимізаційних задач на графах.

Класичні методи пошуку розв'язків. Загальна задача паралельного впорядкування належить до класу NP-повних задач, тому виникає необхідність побудови та дослідження не лише точних, а й наближених алгоритмів.

Для даної задачі найбільш поширені є такі точні методи: метод гілок та меж, зведення задачі до задачі про умовний максимальний потік, а також відомий підхід, що зводить дану задачу до задачі цілочисельного лінійного програмування [3].

Точні алгоритми, що мають поліноміальну складність, відомі лише для дуже обмежених класів задач (граф G – це кореневе дерево, граф G – без транзитивних дуг та h=2).

Тому для розв'язання практичних задач, що мають велику розмірність, доцільно використовувати наближені алгоритми, а особливо ті, точність яких можна встановити апріорно. Використання наближених методів дозволяє скоротити час або ресурси (наприклад, пам'ять), необхідні для побудови впорядкувань.

© *В.А. Турчина, Ю.С. Зозуля, А.К. Підаш, 2013*

Крім того, заслуговують на увагу відомості про множину всіх допустимих упорядкувань заданої довжини, а також множини всіх оптимальних упорядкувань заданої довжини. Алгоритмам побудови цих множин і присвячена дана робота.

При побудові цих множин допоміжними виступають упорядкування $\underline{S}, \overline{S}$, довжини яких рівні й дорівнюють критичній довжині. В упорядкуванні $\underline{S}$ крайнє ліве допустиме місце позначимо $\underline{\mu_i}$, а крайнє праве – $\overline{\mu_i}$. Тоді допустимі місця μ_i для вершини *i* в допустимому впорядкуванні належать проміжку $\left\lfloor \underline{\mu_i}, \overline{\mu_i} \right\rfloor$ [2].

Алгоритми перерахування всіх паралельних упорядкувань фіксованої довжини. Алгоритми засновані на комбінаторному аналізі елементів кінцевої множини інтервалів. Загальною ідеєю алгоритму є рекурсивний обхід вершин графа, розміщених у певному порядку, «вперед» і «назад». Йдучи «вперед», ми аналізуємо, наскільки були максимально «зміщені» всі пов'язані з поточною вершиною вершини, які знаходяться у списку вершин раніше, та з множини допустимих місць вершини в упорядкуваннях μ_i видаляємо цю кількість вершин. Записуємо в результат перше «вільне» значення з цілочисельного проміжку μ_i для даної вершини і відразу видаляємо це значення з допустимого проміжку для даної вершини. Якщо це не остання вершина, – йдемо далі.

Якщо поточна вершина – це остання вершина в нашому списку вершин, то заносимо поточне впорядкування в «головний результат» і йдемо «назад» до тих пір, поки не зустрінемо вершину, в якої її μ не пусте. Тепер вважаємо, що йдемо «вперед», видаляємо всі входження наступних за списком вершин із впорядкування, а також для цих вершин присвоюємо їх μ_i такі значення, які були на початку виконання алгоритму.

Алгоритм

1. Відомий граф. Обираємо форму представлення даних про граф у пам'яті (матриця суміжності, списки ребер, матриця інцидентності). Задаємо довжину впорядкування *l* (якщо вона не задана, то довжина буде дорівнювати критичній, тобто l_0).

1.1 . Будуємо допоміжні впорядкування $\underline{S}, \overline{S}$.

1.2 . Для кожної вершини орграфа визначаємо множину її допустимих місць в упорядкуваннях. Для *i*-ї вершини це значення визначається формулою $\mu_i \in \left\lfloor \underline{\mu_i}, \overline{\mu_i} + l - l_0 \right\rfloor$,

де $\underline{\mu_i}$ – місце вершини *i* в упорядкуванні $\underline{S}$; $\overline{\mu_i}$ – місце вершини *i* в упорядкуванні $\overline{S}$.

2. Формуємо список вершин у потрібному нам порядку, а саме:

2.1 . k:=1.

2.2 . До списку вершин заносимо вершини з k-го місця в упорядкуванні $\underline{S}$.

2.3 . Якщо k менше за кількість місць у впорядкуванні $\underline{S}$, то k:=k+1, інакше – список вершин побудовано.

3. За поточну вершину вважаємо першу вершину у списку вершин. Вважаємо, що йдемо «вперед».

4. Якщо допустимий проміжок для поточної вершини пустий, то:

- якщо поточна вершина – це перша вершина у списку вершин, то йдемо на пункт 6;
- інакше поточною вважаємо попередню до поточної вершину з списку вершин; йдемо на пункт 5.

Інакше, якщо це шлях назад, то:

- вважаємо, що йдемо вперед;
- із результату видаляємо всі значення, що відповідають наступним після поточної вершинам у списку вершин;
- допустимі проміжки, що відповідають наступним після поточної вершинам у списку вершин, робимо такими, як були сформовані в пункті 1.

5. Для всіх вершин, які зв'язані з поточною та які знаходяться в списку вершин раніше поточної, знаходимо різницю між її місцем знаходження та першим значенням із допустимого проміжку, яке було знайдене на кроці 1. Обираємо максимальну різницю. Якщо для поточної вершини перше значення з її допустимого проміжку менше, ніж відповідне значення, знайдене на першому кроці, то за допустимий проміжок для даної вершини беремо зсунутий на максимальну різницю початковий проміжок. До результату додаємо поточну вершину на місце, що відповідає першому значенню з поновленого допустимого проміжку.

Обчислюємо величину Δ як різницю між першим значенням із допустимого проміжку поточної вершини і відповідним першим значенням, знайденим на кроці 1.

Із допустимого проміжку для поточної вершини видаляємо перше значення.

Знаходимо всі вершини, що пов'язані з поточною, та видаляємо з їх допустимих проміжків перші Δ значень.

Якщо поточна вершина – остання, то записуємо в «головний результат» отримане упорядкування та за поточну обираємо попередню вершину в списку вершин; вважаємо, що йдемо шляхом «назад». Йдемо на пункт 4. Інакше – як поточну обираємо наступну вершину у списку вершин.

6. Результатом роботи алгоритму вважаємо «головний результат». Якщо він пустий, то алгоритм не знайшов допустимих упорядкувань. Кінець.

Алгоритм побудови всіх упорядкувань фіксованої довжини і мінімальної ширини є частинним випадком задачі побудови множини всіх допустимих паралельних впорядкувань. Головна ідея алгоритму будується на рекурсивній функції перебору всіх можливих позицій для вершини i на кожному кроці. За допомогою рекурсивної функції обробляємо всі вершини, допоки не перевищимо h, отримане з попереднього кроку , шляхом рекурсивного перебору всіх можливих позицій для кожної вершини з наступним кроком. Функція спочатку робить прямий хід, обробляючи від 1 до n-ї вершини. Далі алгоритм робить зворотний хід. Обчислює всі можливі позиції n-ї вершини згідно з умовами, що задовольняють позиціям інших (n-1) вершин та, ураховуючи результат упорядкування мінімальної ширини, обчислений на попередньому кроці; якщо ширина менша, то змінюємо значення h та припиняємо хід або йдемо далі. Потім рекурсивна функція переходить на вершину (n-1) і таким самим чином обробляє всі можливі позиції для вершин (n-2) і для n-ї вершини. Продовжуємо алгоритм, поки не перелічимо всі позиції для першої вершини.

Алгоритм

1. За матрицею суміжності перевіряємо граф на ациклічність; якщо є цикли, – тоді кінець.
2. Обробляємо матрицю суміжності і будуємо $\underline{S}, \overline{S}$.
3. З отриманих $\underline{S}, \overline{S}$ будуємо допустимі місця μ_i; i -номер вершини.

μ_i - множина можливих позицій в упорядкуванні для i-вершини, тобто $\mu_i \in \lfloor \underline{\mu_i}, \overline{\mu_i} + l - l_0 \rfloor$.

4. За допомогою певного алгоритму обираємо ті i вершин, які не лежать на критичному шляху, а також відповідні їм μ_i.

5. За допомогою рекурсивної функції обробляємо всі i вершин, отримані з попереднього кроку з відповідними μ_i, шляхом рекурсивного перебору всіх можливих позицій для кожної i -ї вершини з наступним кроком.

Функція спочатку робить прямий хід, обробляючи від 1 до n-ї вершини.

Далі алгоритм робить зворотний хід, обчислює всі можливі позиції n-ї вершини згідно з умовами, що задовольняють позиціям інших (n-1) - вершин. Потім рекурсивна функція переходить на вершину (n-1) і таким самим чином обробляє всі можливі позиції для вершин (n-2) і для n-ї вершини. Продовжуємо алгоритм, поки не перелічимо всі позиції для першої вершини.

6. Виведення результатів. Кінець.

Висновки. У даній роботі було запропоновано нові алгоритми направ-леного перебору для побудови множини всіх допустимих паралельних упорядкувань заданої довжини. Вони засновані на аналізі структури графів, які моделюють послідовності операцій в обчислювальних і технологічних процесах.

Має інтерес також задача побудови множини всіх допустимих упорядкувань для задач, у яких вершини графа мають різні помітки, що відповідають різному часу виконання операцій.

Бібліографічні посилання

1. **Коффман Э.Г.** Введение в теорию расписаний / Э.Г. Коффман. – М.: Наука, 1984. – 333 с.
2. **Бурдюк В.Я.** Алгоритмы паралельного упорядочения / В.Я. Бурдюк, В.А. Турчина. – Днепропетровск, 1985. – 84 с.
3. **Беллман Р.** Динамическое программирование / Р. Беллман – М.: Иностр. лит-ра, 1960. – 400 с.

Надійшла до редколегії 25.06.2013 р.