**Аналіз узагальнених задач паралельного упорядкування, що допускають переривання**

Наказ №2084-с від 20. 10

Земляна

1. Місце задач паралельного упорядкування в класі задач теорії розкладів
   1. Постановки задач теорії розкладів
   2. Розклади без переривань і з перериваннями
   3. Задачі паралельного упорядкування в класі задач теорії розкладів
2. Узагальнені задачі паралельного упорядкування
   1. Постановки узагальнених задач
   2. Алгоритми розв’язання узагальнених задач для дерева
   3. Підкласи оптимальних упорядкувань, що допускають переривання
   4. Методи декомпозиції при аналізі узагальнених задач паралельного упорядкування
   5. Підкласи графів, оптимальні упорядкування яких не вимагають переривань
3. Програмна реалізація алгоритмів
   1. Програмна генерація графів
   2. Виділення підкласів графів
   3. Побудова оптимальних упорядкувань з перериваннями і побудова оптимальних упорядкувань без переривань

Про p, np?

1. **Місце задач паралельного упорядкування в класі задач теорії розкладів**
   1. **Постановки задач теорії розкладів**

В загальному формулюванні задача теорії розкладу полягає в наступному:

за допомогою деякої множини ресурсів (обслуговуючих приладів, робочих і т.д.) має бути виконана деяка фіксована система завдань. Метою є побудова розкладу, тобто деякої послідовності виконання завдань з інформацією, коли та на якому ресурсі має виконуватися завдання, - такого, що послідовність виконання завдань задовольняє технологічних вимогам, і при цьому є оптимальним.

Загальна модель задач теорії розкладів складається з сукупності моделей, що описують ресурси, систему завдань, обмеження при складанні розкладів, а також міри їх оцінки.

У більшості моделей ресурси – це набір процесорів. Позначимо їх . В залежності від задачі вони можуть мати як однакові характеристики, так і бути різними за функціональністю та/або швидкодією. В деяких випадках описують також додаткові ресурси. Це може бути оперативна пам'ять, зовнішня пам'ять, з’єднання з мережею Інтернет, бібліотеки, тощо.

Система завдань в найбільш загальному випадку може бути представлена як система

що побудована наступним чином:

1. – набір завдань, які потрібно виконати.
2. позначає задане на відношення часткового порядку (не рефлексивне), котре визначає обмеження на послідовність виконання завдань. Запис означає, що має бути виконане раніше, ніж почнеться виконання .
3. – матриця розміром , елементи якої представляють собою час виконання завдання ( ) на процесорі ( ). , якщо не може бути виконане на процесорі . Для кожного існує , для якого . Якщо всі процесори ідентичні, то матриця вироджується в вектор .
4. – набір, –та координата якого являє собою кількість ресурсу типу , необхідного для виконання завдання .
5. – вага, що інтерпретується як вартість перебування завдання в системі.

При дослідженні задач розглядають два підходи, згідно з якими завдання виконуються з перериваннями або без переривань.

Зазвичай у якості міри ефективності розкладу розглядають максимальний час завершення задач (тобто довжину розкладу) чи середньозважений час завершення завдань.

Наведена модель є загальною. Зазвичай всі реальні задачі – це її частковий випадок. В подальшому будемо вважати, що тривалість виконання роботи не залежить від виконавця.

Величина називається довжиною розкладу. В цей момент виконання всіх робіт завершено.

Також надалі будемо називати розклад оптимальним, якщо він або має задану довжину () і вимагає мінімальної кількості виконавців (), або при заданій кількості виконавців має мінімальну довжину.

* 1. **Розклади без переривань і з перериваннями**

Якщо в системі всі завдання мають однакові часи виконання, то можна вважати, що всі ці часи рівні одиниці. У цьому випадку ми маємо так звану UET-систему (unit-execution-time). Це класична постановка задачі. В такому випадку задача переривань не виникає.

Якщо ж час виконання робіт у загальному випадку різний, то виділяють два підходи – розв’язання задач з дозволом чи забороною переривань.

Розклад без переривань передбачає, що завдання після початку виконання вже не може бути зупинене до повного завершення. Розклад з перериваннями дозволяє знімати задачу з ресурсу (наприклад, з процесора) до її завершення, з подальшим відновленням виконання завдання. При цьому вважаємо, що сумарний час виконання завдання залишається незмінним, тобто завдання відновлюється саме з того моменту, на якому було зупинене. При перериванні з’являється необхідність знімати задачу з виконання до її завершення і зберігати інформацію про неї протягом часу очікування. Але такі затрати зазвичай є незначними. Також вважаємо, що не виникає необхідність у додатковому часі виконання, пов’язаному зі збереженням та подальшим зчитуванням інформації про стан задачі.

* 1. **Задачі паралельного упорядкування в класі задач теорії розкладів**

Задачі теорії розкладу природнім чином пов’язуються з задачами впорядкувань. Покажемо це.

Впорядкуванням скінченої множини , що містить елементів, будемо називати розміщення цих елементів по місцям, розташованих в лінію так, що кожен елемент з розміщується лише на одному місці.

Кількість непустих місць в впорядкуванні називається довжиною упорядкування та позначається

Нехай – множина тих елементів з , які розташовані в на -му місці.

Шириною впорядкування називається кількість елементів найбільшої по потужності множини , і позначається

Введемо до розгляду граф , де – множина вершин, а – множина дуг.

Впорядкування множини вершин графу назвемо паралельним впорядкуванням вершин орграфу , якщо з того, то розташовано в лівіше за .

Для існування паралельного впорядкування необхідно і достатньо, або граф був ациклічним.

Для існування паралельного впорядкування необхідно і достатньо, або граф був ациклічним [1].

Для прикладу, розглянемо граф, зображений на рис. 1.1.

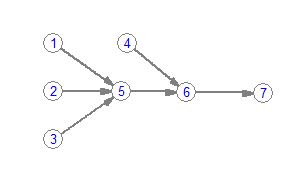


Рис.1.1. Вигляд графу

Побудуємо деякі паралельні впорядкування вершин цього графу. Їх зображено на рис. 1.2 та рис. 1.3.

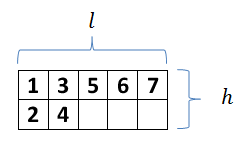


Рис. 1.2. Паралельне впорядкування вершин графу *G*

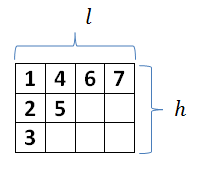


Рис. 1.3. Паралельне впорядкування вершин графу *G*

Як бачимо, при довільному допустимому розміщенні вершин графу *G* ми можемо отримувати допустимі впорядкування різної довжини та ширини.

Будемо називати впорядкування оптимальним, якщо воно або при заданій довжині має мінімальну ширину, або при заданій ширині має мінімальну довжину. Позначимо описані задачі відповідно та [2, 3].

Оскільки кожному завданню можна поставити у відповідність вершину графу, а порядку слідування робіт – дуги графу, то можна задачі теорії розкладів інтерпретувати як оптимізаційні задачі на графах.

Ациклічний граф може слугувати математичною моделлю обмежень на порядок виконання робіт. Множина вершин графу буде відповідати множині завдань , множина ребер – технологічним обмеженням, заданим відношенням при цьому з того, що слідує, що робота має бути завершена не пізніше, ніж почнеться виконання роботи .

Таким чином, паралельні впорядкування вершин графу будуть однозначним чином задавати допустимі розклади. При цьому довжина упорядкування , а ширина впорядкування відповідатиме мінімальної кількості виконавців,необхідних для виконання робіт .

Таким чином, оптимізаційним задачам на графах можна однозначно поставити у відповідність оптимізаційні задачі в теорії впорядкувань. Надалі будемо будувати оптимальні розклади за допомогою вирішення відповідних оптимізаційних задач на графах.

1. Узагальнені задачі паралельного упорядкування
   1. Постановки узагальнених задач
   2. Алгоритми розв’язання узагальнених задач для дерева
   3. Підкласи оптимальних упорядкувань, що допускають переривання
   4. Методи декомпозиції при аналізі узагальнених задач паралельного упорядкування
   5. Підкласи графів, оптимальні упорядкування яких не вимагають переривань
   6. . Постановки узагальнених задач

У більшості реальних задач виникає необхідність розглядати системи завдань, у яких час виконання задач різний. Такому випадку відповідає узагальнена задача паралельного впорядкування, яка формулюється наступним чином:

Задано граф , де – множина вершин, а – множина дуг. Цей граф задає відношення часткового порядку виконання робіт. Кожній вершині поставлено у відповідність час виконання роботи - число . У загальному випадку часи виконання різні.

Необхідно побудувати таке допустиме впорядкування множини вершин графу , яке або при заданій довжині має мінімальну ширину, або при заданій ширині має мінімальну довжину. Позначимо описані задачі відповідно та .

* 1. .Алгоритми розв’язання узагальнених задач для дерева

Для розв’язання задачі паралельного упорядкування у класичній постановці існують точні алгоритми поліноміальної складності, які є точними для випадку, коли – тобто, граф обмежень є лісом або деревом, а також коли ширина бажаного впорядкування

Для того, аби перейти до розгляду алгоритмів побудови оптимальних впорядкувань для узагальненої задачі, розглянемо підхід, заснований на рівневому принципі, у контексті задач у класичній постановці.

Розглянемо питання про існування точного розв’язку для загальної задачі паралельного впорядкування, а саме задачу побудови паралельного впорядкування для орграфу , де .

Розглянемо алгоритм побудови деякого впорядкування

Алгоритм

1. Вважаємо в всі місця пустими.
2. В орграфі шукаємо вершини, які не мають вхідних дуг і записуємо їх на -те місце в . Викреслюємо ці вершини та дуги, що з них виходять. Граф, що залишився, позначимо .
3. Якщо множина вершин графу пуста, то кінець.
4. , переходимо на крок 3, обираючи в якості орграфу орграф .

Стверджуємо, що впорядкування, побудоване для даної задачі, є оптимальним.

Довжину отриманого впорядкування позначимо , а саме впорядкування - . Впорядкування буде також оптимальним, якщо шукане впорядкування будувати справа наліво, тобто на кожному кроці шукати в орграфі вершини, що не мають вхідних дуг, і розміщувати їх починаючи з місця . Таке впорядкування позначатимемо [4].

Таким чином, ми отримали деякі точні рішення для задачі паралельного впорядкування, але лише для випадку, коли по суті не обмежене.

Але нас цікавлять саме впорядкування та , які знадобляться нам надалі.

Нехай задано граф . Побудуємо для нього впорядкування . Характеризувати рівень вершини буде величина, що обчислюється за формулою:

де довжина впорядкування ;

- номер місця вершини в упорядкуванні.

Вільною будемо називати таку вершину, всі попередники якої вже є в упорядкуванні. При побудові впорядкування будемо брати ту вільну вершину, яка має максимальний рівень. Якщо таких вершин декілька – беремо будь-яку.

У випадку узагальненої задачі розрізняють підходи з дозволом та без дозволу переривань.

[Нехай задано граф, побудуємо без переривання (приклад)]

Побудуємо впорядкування з перериваннями. Для цього розглянемо допоміжну задачу.

Множину вершин будемо називати ланцюгом, якщо .

Знайдемо деякий час , який є спільним дільником для часу виконання кожного з завдань, і в графі для кожного завдання замінимо вершину на ланцюжок вершин , де кількість елементів цього ланцюжка буде дорівнювати кратності часу виконання даного завдання до часу . Таким чином отримали допоміжну задачу паралельного впорядкування у класичній постановці (час виконання всіх завдань однаковий).

Тепер побудуємо впорядкування згідно з рівневим принципом для допоміжної задачі.

[приклад тут]

Як бачимо, отримане впорядкування має меншу довжину. Воно і є впорядкуванням з перериваннями для початкової задачі.

Отже, впорядкування з перериваннями можуть мати меншу довжину, ніж впорядкування, побудовані без дозволу переривань.

Варто зазначити, що …[написати про те, що завжди можемо розбити на ланцюжок, с 97]

Позначимо як граф, що отримуємо з заміною вершини ланцюгом , де . Попередники стають попередниками , а її наступники – наступниками . Після цього кожна вершина має час виконання , що дозволяє застосовувати до алгоритми побудови впорядкувань без переривань, розроблені для задач у класичній постановці.

Вершини можуть бути ще раз розділені і подані у вигляді ланцюгів вершин з однаковим часом виконання. Позначимо як граф, що отримуємо з заміною кожної вершини ланцюгом з вершин, що мають однаковий час виконання. При збільшенні можна очікувати, що оптимальне впорядкування з перериваннями для системи, що представляється графом стає все ближчим до розкладу з перериваннями для системи, що представляється графом .

Нехай – довжина оптимального розкладу без переривань для системи, заданої графом , що виконується виконавцями, а – з перериваннями.

Тоді справедлива наступна теорема, що

Теорема:

Нехай маємо граф , , для кожної вершини задано час виконання (у загальному випадку різний), – додатнє дійсне число, котрому пропорційні всі , .

Тоді , де – константа, що залежить тільки від .

З теореми слідує, що кожного разу, коли ми можемо побудувати оптимальний розклад без задачі паралельного упорядкування у класичній постановці, ми можемо побудувати розклад з перериваннями і для узагальненої задачі, котре буде наскільки завгодно близьким до оптимального.

**Алгоритм**

Нехай задано звязний граф . Побудуємо для нього довільне остовне дерево .

Для отриманого графу побудуємо впорядкування згідно з рівневим принципом. Як відомо, рівне вий принцип для дерева є точним алгоритмом, що має поліноміальну складність.

Отримане впорядкування може не бути допустимим для початкового графу . Конфліктною парою будемо називати таку пару вершин, яка не задовольняє відношення часткового порядку, що задається графом . Вершину, яка має бути поставлена в упорядкування раніше з двох, називатимемо початковою, а яка пізніше – кінцевою.

Для того, аби знайти всі конфлікти, достатньо продивитися всі дуги, що містяться в графі , але не увійшли в остовне дерево , тобто елементи множини .

Для того, аби зробити впорядкування допустимим, і при цьому збільшити довжину впорядкування якомога менше, запропоновано наступний алгоритм:

1. Знаходимо всі конфлікти. Якщо конфліктних пар не знайдено, - поточне впорядкування допустиме, кінець алгоритму. Інакше - обходимо впорядкування з початку.
2. Якщо знаходимо вершину, яка є однією з конфліктної пари, шукаємо і другу та йдемо на пункт 3. Якщо конфліктних пар не знайдено, - поточне впорядкування допустиме, кінець алгоритму.
3. Якщо на попередньому місці в упорядкуванні є вершина , яка не має наступників на поточному місці, то йдемо на пункт 4. Інакше – на пункт 5.
4. Міняємо місцями початкову вершину зі знайденого конфлікту, і вершину . Таким чином, ми позбавилися від даного конфлікту, додатково не порушивши умов часткового порядку, та не збільшивши довжину впорядкування. Йдемо на пункт 6.
5. Збільшуємо довжину впорядкування, додавши місце в упорядкування відразу за поточним. Ставимо на нього кінцеву вершину з конфліктної пари.
6. Допоки є можливість додавати вершини на поточне місце (тобто поки на нього вже не поставили вершин), перевіряємо, чи є ще такі конфліктні пари, початкову вершину яких можна поставити на вільне місце. Для вказаної перевірки для кожної конфліктної пари перевіряємо, чи є всі її попередники в упорядкуванні на вже проглянутих рівнях). Якщо такі вершини знайдені – ставимо їх в упорядкування.
7. Допоки є можливість додавати вершини на поточне місце (тобто поки на нього вже не поставили вершин), якщо в графі є вільні вершини, переставляємо їх на поточне.
8. Йдемо на пункт 2.