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?

Оглавление

[Оглавление 2](#_Toc440539805)

[Розділ 1. Місце задач паралельного упорядкування в класі задач теорії розкладів 3](#_Toc440539806)

[1.1. Постановки задач теорії розкладів 3](#_Toc440539807)

[1.2. Розклади без переривань і з перериваннями 5](#_Toc440539808)

[1.3. Задачі паралельного упорядкування в класі задач теорії розкладів 6](#_Toc440539809)

[Розділ 2. Узагальнені задачі паралельного упорядкування 9](#_Toc440539810)

[2.1. Постановки узагальнених задач 9](#_Toc440539812)

[2.2. Алгоритми розв’язання узагальнених задач для дерева 11](#_Toc440539813)

[2.3. Підкласи оптимальних упорядкувань, що допускають переривання (?) 19](#_Toc440539814)

[2.4. Методи декомпозиції при аналізі узагальнених задач паралельного упорядкування 20](#_Toc440539815)

[2.5. Підкласи графів, оптимальні упорядкування яких не вимагають переривань 27](#_Toc440539816)

# Розділ 1. Місце задач паралельного упорядкування в класі задач теорії розкладів

## Постановки задач теорії розкладів

В загальному формулюванні задача теорії розкладів 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 ряді 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.

## 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 необхідності в додатковому часі виконання, пов’язаному зі збереженням та подальшим зчитуванням інформації про стан задачі.

## Задачі паралельного упорядкування в класі задач теорії розкладів

Задачі теорії розкладу природнім чином пов’язуються з задачами впорядкувань. Покажемо це. Для початку введемо деякі визначення.

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 слідує, що розташовано в лівіше за .

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

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

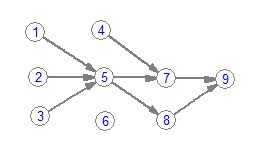


Рис.1.1. Вигляд графу

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*



Рис. 1.4. 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].

Наприклад, впорядкування, зображене на рис. 1.2. є оптимальним для задачі , а на рис. 1.4. – для задачі .

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 мінімальній кількості виконавців, необхідних для виконання робіт ().

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

# Розділ 2. Узагальнені задачі паралельного упорядкування



## 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 число - час виконання роботи. 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 .

Розглянемо граф, зображений на рис. 2.1. Позначки біля вершин відповідають часу виконання роботи .

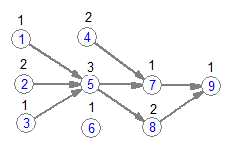


Рис. 2.1. Вигляд графу

Оптимальне впорядкування, побудоване для задачі , можемо бачити в таблиці 2.1.

*Таблиця 2.1.*

Оптимальне впорядкування вершин графу G для узагальненої задачі

## **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 на прикладі 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 к-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 перш за все ту вільну вершини, яка має максимальний рівень. Якщо таких вершин декілька – беремо будь-яку.

Розглянемо граф , зображений на рис. 2.2.

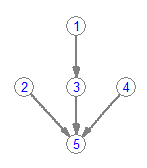


Рис. 2.2. Вигляд графу

Побудуємо впорядкування згідно з рівневим принципом для задачі . Для цього побудуємо для впорядкування , що зображене в таблиці 2.2. Номер строки *k* характеризує номер місця вершини в упорядкуванні, а відповідно – і її рівень.

*Таблиця 2.2.*

Впорядкування вершин графу

|  |  |  |  |
| --- | --- | --- | --- |
| *k = 1* | 1 |  |  |
| *k = 2* | 2 | 3 | 4 |
| *k = 3* | 5 |  |  |

Тепер побудуємо впорядкування згідно з рівневим принципом (таблиця 2.3)

*Таблиця 2.3.*

Впорядкування вершин графу , побудоване згідно з рівневим принципом

|  |  |
| --- | --- |
| **1** | **2** |
| **3** | **4** |
| **5** |  |

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

Розглянемо граф , зображений на рис. 2.3.

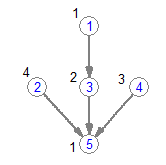


Рис. 2.3. Вигляд графу

Побудуємо оптимальне впорядкування без переривань для задачі (таблиця 2.4.)

*Таблиця 2.4.*

Оптимальне впорядкування без переривань

|  |  |
| --- | --- |
| **1** | **2** |
| **4** | **2** |
| **4** | **2** |
| **4** | **2** |
| **3** |  |
| **3** |  |
| **5** |  |

Довжина отриманого впорядкування – 7.

Виявляється, що дозвіл на переривання у виконанні робіт може зменшити довжину.

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 ланцюг вершин , де кількість елементів цього ланцюга буде дорівнювати кратності часу виконання даного завдання до часу . Таким чином, отримали допоміжну задачу паралельного впорядкування у класичній постановці (час виконання всіх завдань однаковий).

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 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:

**Теорема:**

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

Тоді , де – константа, що залежить тільки від .

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

Проілюструємо описаний підхід прикладами.

Допоміжну задачу можна зобразити наступним чином (рис 2.4.).

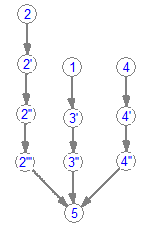


Рис. 2.4. Вигляд графу для допоміжної задачі

Тепер побудуємо впорядкування згідно з рівневим принципом для допоміжної задачі. Результат видно у таблиці 2.5.

*Таблиця 2.5.*

Оптимальне впорядкування з перериваннями



Довжина впорядкування – 6. 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. Окрім того, впорядкування з перериваннями можна будувати базуючись на рівневому принципі, який є алгоритмом поліноміальної складності, тоді як при побудові впорядкувань без переривань рівневий принцип не дає оптимальних результатів.

Також постає питання, чи можливо зменшити кількість переривань в упорядкуванні, при цьому не збільшуючи довжину.

При побудові впорядкувань згідно з рівневий принципом, на кожному кроці беремо перш за все ту вільну вершину, яка має максимальний рівень. Якщо таких вершин декілька – беремо будь-яку.

Дещо модифікуємо рівневий принцип. А саме, при побудові впорядкування будемо брати перш за все ту вільну вершину, яка має максимальний рівень. Якщо таких вершин декілька – беремо перш за все ту, яка вже стоїть на попередньому місці в упорядкуванні.

Даний алгоритм не суперечить рівневому принципу, адже по суті ми лише визначаємо спосіб обрання вершини у випадку, коли декілька вершин мають максимальний рівень, тоді як в класичному рівневому принципі цей вибір довільний. Таким чином, для модифікованого алгоритму зберігаються всі властивості, справедливі для рівневого принципу, але при цьому кількість переривань є меншою.

Продемонструємо роботу даного алгоритму на прикладі.

Як було показано раніше, впорядкування для графу з рис. 2.3, побудоване класичним рівневим принципом з дозволом переривань, зображене в таблиці 2.5. Його довжина – 6, а кількість переривань – 2.

В таблиці 2.6. різними кольорами показані роботи, що перериваються.

*Таблиця 2.6.*

Оптимальне впорядкування з перериваннями



Тепер побудуємо впорядкування модифікованим способом. Результат наведено в таблиці 2.7.

*Таблиця 2.7.*

Оптимальне впорядкування з перериваннями, побудоване згідно з модифікованим рівневим принципом

|  |  |
| --- | --- |
| **2** | **1** |
| **2** | **4** |
| **2** | **4** |
| **2** | **3** |
| **4** | **3** |
| **5** |  |

Як бачимо, воно містить лише одне переривання. При цьому довжина, звичайно, залишилася незмінною.

Отже, для побудови впорядкувань з перериваннями для випадку, коли граф обмежень – дерево, можна рекомендувати використовувати модифікований рівневий принцип.

Розібратися, чому лише наближення. Переривання в довільних місцях? Придумати приклад, коли переривати треба не по тактам, як зазвичай.

## Підкласи оптимальних упорядкувань, що допускають переривання (?)

## Методи декомпозиції при аналізі узагальнених задач паралельного упорядкування

Як відомо, для задач паралельного впорядкування у класичній постановці існують точні алгоритми поліноміальної складності для випадку, коли заданий граф обмежень – дерево. Так як точні алгоритми для задачі в загальному випадку мають експоненційну складність, інтерес представляють наближені методи. Вони дозволяють будувати близькі до оптимальних упорядкування за прийнятний час.

Запропоновано наближений метод, який базується на рівневому принципі. Загальною ідеєю є виділення остовного дерева для заданого графу обмежень, побудова оптимального впорядкування для нього, і модифікація отриманого впорядкування таким чином, аби воно було допустимим для початкового графу.

Необхідною умовою застосування методу є зв’язність графу.

//а корнева вершина?

Нехай задано граф , що зображений на рис.

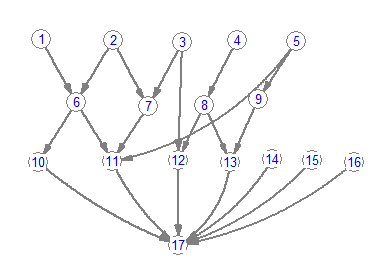


Рис. 2.\_\_. Вигляд графу для допоміжної задачі

// **Остовное дерево** — ациклический [связный](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84) [подграф](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%B3%D1%80%D0%B0%D1%84) данного связного [неориентированного графа](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D0%B3%D1%80%D0%B0%D1%84),

Для нього знаходимо довільне остовне дерево, наприклад таке, як на рис.

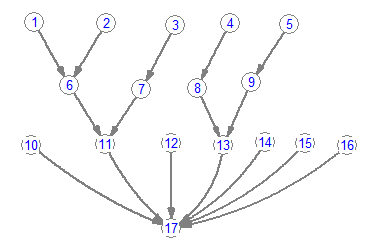


Рис. 2.\_\_. Вигляд графу для допоміжної задачі

Знаходимо для нього оптимальне впорядкування за допомогою алгоритму, що базується на рівневому принципі. Результат наведено в таблиці\_

*Таблиця 2.\_.*

Оптимальне впорядкування, побудоване згідно з рівневим принципом для остовного дерева

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** | **4** | **5** |
| **2** | **7** | **8** | **9** |
| **6** | **10** | **12** | **13** |
| **11** | **14** | **15** | **16** |
| **17** |  |  |  |

Як бачимо, воно не є допустимим (наприклад, вершини 2 та 7, а також 6 та 10 не можуть стояти на одному місці в упорядкуванні). Довжина – 5.

Тож необхідно деяким чином модифікувати впорядкування.

Довільним чином це можна зробити, наприклад, так, як показано в таблиці 2.\_

*Таблиця 2.\_.*

Модифіковане впорядкування

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** | **4** | **5** |
| **2** | **8** | **9** |  |
| **7** | **6** | **12** | **13** |
| **10** | **11** | **14** | **15** |
| **16** |  |  |  |
| **17** |  |  |  |

Отримане впорядкування має більшу довжину (6), але є допустимим. Хотілося б отримувати як можна більш близькі до оптимального впорядкування.

Запропоновано наступний алгоритм, який реалізує зазначену ідею.

Нехай задано зв’язний граф . Побудуємо для нього остовне дерево , керуючись наступним правилом:

Для кожної вершини знаходимо список всіх вершин, що за нею слідують.

Залишаємо лише одну (довільну дугу) зі знайдених. Решту видаляємо.

Побудований вказаним способом граф буде обов’язково деревом, так як кожна вершина матиме лише одну вихідну дугу, а також міститиме всі вершини початкового графу, так як початковий граф є зв’язним. Таким чином отриманий граф є остовним деревом для початкового графу.

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

Як було зазначено вище, отримане впорядкування може не бути допустимим для початкового графу . Конфліктною парою будемо називати таку пару вершин із множини , 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 нашого прикладу конфліктною парою будуть пари вершин (2,7) та (6,10).

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 необхідності збільшення довжини, мінімізувати її відхилення від оптимальної, 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 входить до однієї з конфліктних пар, то вибираємо відповідну дугу і йдемо на пункт 3. Якщо конфліктних пар не знайдено, - поточне впорядкування допустиме, йдемо на пункт 8.
3. Якщо у початкової вершини зі знайденого конфлікту немає попередників на попередньому місці в упорядкуванні, то:
   1. Якщо на попередньому місці в упорядкуванні є вільне місце, то ставимо туди початкову вершину. Таким чином, ми позбавилися від даного конфлікту, додатково не порушивши умов часткового порядку, та не збільшивши довжину впорядкування. Йдемо на пункт 2.
   2. Якщо на попередньому місці в упорядкуванні є вершина , яка не має наступників на поточному місці, то 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 2.
4. Якщо у кінцевої вершини зі знайденого конфлікту немає наступників на наступному місці в упорядкуванні, то:
   1. Якщо на наступному місці в упорядкування є вільне місце, то ставимо туди кінцеву вершину. Таким чином, ми позбавилися від даного конфлікту, додатково не порушивши умов часткового порядку, та не збільшивши довжину впорядкування. Йдемо на пункт 2.
   2. Якщо на наступному місці в упорядкуванні є вершина , яка не має попередників на поточному місці, то міняємо місцями кінцеву 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 2.
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 або на щойно додане (тобто поки на нього вже не поставили вершин), якщо в графі є вільні вершини, то сортуємо їх, віддаючи пріоритет початковим вершинам з конфліктних пар, і переставляємо їх по одній на відповідне місце.
7. Йдемо на пункт 2.
8. Проходимо по рівням впорядкування. Якщо є вільні місця на поточному рівні, перевіряємо, чи є вільні вершини (тобто такі, попередники яких містяться в упорядкування на попередніх рівнях). Якщо є – ставимо їх на вільні місця.
9. Кінець алгоритму.

Проілюструємо даний алгоритм, беручи за основу вищенаведений приклад графу, остовного дерева та впорядкування для нього.

Нагадаємо вигляд упорядкування, отриманого для остовного дерева (таблиця \_)

*Таблиця 2.\_.*

Оптимальне впорядкування, побудоване згідно з рівневим принципом для остовного дерева

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **3** | **4** | **5** |
| **2** | **7** | **8** | **9** |
| **6** | **10** | **12** | **13** |
| **11** | **14** | **15** | **16** |
| **17** |  |  |  |

1. Конфліктні пари - (2,7) та (6,10).
2. Знаходимо вершину 2 та відповідну їй вершину 7.
3. У вершини немає 2 попередників на попередньому місці. Тож переходимо на пункт 3.1.
   1. На попередньому місці в упорядкуванні вільних місць немає.
   2. На попередньому місці в упорядкуванні є вершина 1, яка не має наступників на поточному місці. Тож міняємо місцями початкову вершину зі знайденого конфлікту 2, і вершину 1.

Впорядкування набуває вигляду, зображеного в табл.

*Таблиця 2.\_.*

Поточне впорядкування

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **3** | **4** | **5** |
| **1** | **7** | **8** | **9** |
| **6** | **10** | **12** | **13** |
| **11** | **14** | **15** | **16** |
| **17** |  |  |  |

Йдемо на пункт 2.

1. Знаходимо вершину 6 та відповідну їй вершину 10.
2. У вершини 6 є попередники на попередньому місці. Йдемо на пункт 4.
3. У вершини 10 немає наступників на наступному місці в упорядкуванні, тож йдемо на пункт 4.1.
   1. На наступному місці в упорядкуванні вільних місць немає.
   2. На наступному місці в упорядкуванні є вершина 14, яка не має попередників на поточному місці, тож міняємо місцями вершину 10 і вершину 14.

Впорядкування набуває вигляду, зображеного в табл.

*Таблиця 2.\_.*

Поточне впорядкування

|  |  |  |  |
| --- | --- | --- | --- |
| **2** | **3** | **4** | **5** |
| **1** | **7** | **8** | **9** |
| **6** | **14** | **12** | **13** |
| **11** | **10** | **15** | **16** |
| **17** |  |  |  |

Йдемо на пункт 2.

1. Конфліктів немає, йдемо на пункт 8.

8. Вільних місць на рівнях, окрім останнього, немає.

9. Кінець алгоритму.

Таким чином вдалося отримати оптимальне впорядкування. Його довжина – 5.

Даний алгоритм є наближеним, тож в деяких випадках будемо отримувати лише наближені розв’язки. Він має лише незначні ускладнення порівняно з рівневим принципом, тож можемо говорити про невелику складність алгоритму.

# Підкласи графів, оптимальні упорядкування яких не вимагають переривань

Цікавим є питання, чи доцільно дозволяти переривання у системі, тобто, чи належать оптимальні впорядкування належати до класу впорядкувань з перериваннями. Відповідь на це питання важлива перед усім через те, що в залежності від того, дозволяються переривання чи ні, може обиратися різна система, на якій будуть виконуватися завдання. Також алгоритми розв’язання задач належать зовсім до різних класів.

Тож метою було створення деякого критерію, який дозволяє з’ясувати, до якого класу належить впорядкування, що має меншу довжину.

Перш за все введемо деякі позначення.

Для задачі паралельного впорядкування у класичній постановці важливим поняттям є множина її допустимих місць в упорядкування. Будуємо допоміжні упорядкування , , а потім для кожної вершини орграфу визначаємо величини – місце вершини в упорядкуванні , та – місце вершини в упорядкуванні .

Узагальнимо ці величини для задачі з різним часом виконання.

Позначимо час виконання роботи за .

Дещо змінимо алгоритм побудови впорядкувань та .

Алгоритм побудови впорядкування :

1. Вважаємо в всі місця пустими.
2. В орграфі шукаємо вершини, які не мають вхідних дуг і записуємо їх на -те місце в . . Якщо - 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 .

Аналогічно будуємо .

Для прикладу розглянемо граф, зображений на рис. \_

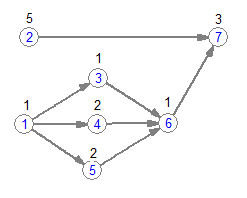


Рис. 2.\_\_. Вигляд графу

Впорядкування наведене в таблиці \_

Позначимо - крайнє ліве положення вершини в упорядкуванні ,

- крайнє праве положення вершини в упорядкуванні ,

- крайнє ліве положення вершини в упорядкуванні ,

- крайнє праве положення вершини в упорядкуванні .

[Приклад ]

**Твердження:**

Нехай маємо граф . Маємо розв’язати узагальнену задачу (час виконання робіт в загальному випадку різний). Якщо в є підграф , такий, що:

* 1. при цьому ця робота може виконуватися одночасно з деякою іншою роботою з ; (1.1)
  2. , – довільне ціле число, ; (1.2)
  3. ; (1.3)

1. ; (2)
2. безпосередньо слідує за вершинами, які мають максимальний допустимий час завершення серед всіх вершин , тобто , при цьому не слідує після ; (3)

а також в немає вільних вершин,

то оптимальне впорядкування належить до класу впорядкувань з перериваннями.

*Доведення:*

Нехай переривання у виконанні операцій заборонені.

З (2) слідує, що виконання жодної з робіт не може початися раніше чи одночасно з роботою . Нехай почалося обслуговування роботи виконавцем . Поки не завершиться її виконання, решту робіт може виконувати лише інший виконавець b.

З умов (1.2) та (1.3) слідує, що час виконання роботи та cумарний час виконання всіх робіт з однаковий і дорівнює . Так як обслуговування робіт з почалось пізніше за на один такт, то і завершиться виконання на один такт пізніше. В цей час виконавець буде простоювати, так як згідно з умовами в немає вільних вершин.

Після обслуговування всіх робіт з може початися виконання роботи яка задовольняє умові (3.1). При цьому виконання інших робіт в цей час не відбувається. То ж один з виконавців простоює.

Загальний час простою дорівнює два такти.

Нехай переривання у виконання операцій дозволені.

Так як згідно з умовою (1.1) в існує вершина з часом виконання 1, яка може виконуватися одночасно з деякою іншою роботою з , то ми можемо перервати обслуговування роботи на користь вказаної роботи. Кількість тактів для виконання робіт з буде дорівнювати , після чого одразу може початися виконання роботи , яка задовольняє умові (3.1). Час від початку до завершення виконання роботи буде дорівнювати . Тобто таким чином виконання робіт та завершиться одночасно, простоїв немає, час завершення обслуговування всіх згаданих робіт зменшиться на 1.

Отже, оптимальне впорядкування належить до класу впорядкувань з перериваннями.

[Приклади ]