# 1.Класичні постановки задач паралельного впорядкування

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, що 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.

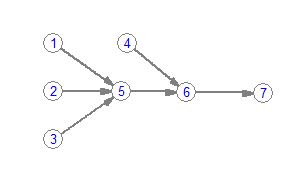


Рис.1.1. Вигляд графу

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

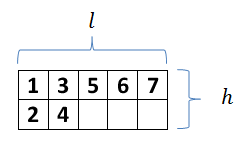


Рис. 1.2. Паралельне впорядкування вершин графу *G*

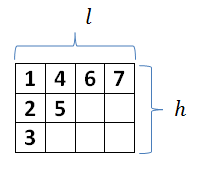


Рис. 1.3. Паралельне впорядкування вершин графу *G*

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].

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.

Впорядкування та знадобляться нам при розв’язанні і узагальнених задач паралельного впорядкування.

# 2. Постановка узагальненої задачі 1 паралельного впорядкування

Введемо деякі обмеження на вигляд впорядкування . Розглянемо вектор . Тоді будемо будувати впорядкування так, щоб потужність множини не перевищувала , .

Для прикладу розглянемо деякий граф , що зображений на рис. 1.4.

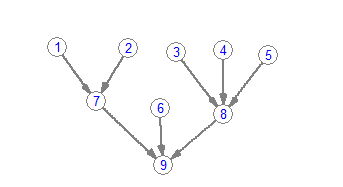


Рис. 1.4. Вигляд графу

Задамо цілочисельний вектор .

Побудуємо деякі впорядкування так, щоб потужність множини не перевищувала , . Наприклад так, як записано в таблицях 1.1 і 1.2:

*Таблиця 1.1.*

Допустиме впорядкування *S1*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *i=1* | 1 | 2 |  |  |  |
| *i=2* | 3 | 4 | 5 | 6 | 7 |
| *i=3* | 8 |  |  |  |  |
| *i=4* | 9 |  |  |  |  |

*Таблиця 1.2.*

Допустиме впорядкування *S2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 3 | 4 |  |  |
| *i=2* | 1 | 2 | 5 | 6 |
| *i=3* | 7 |  |  |  |
| *i=4* | 8 |  |  |  |
| *i=5* | 9 |  |  |  |

Розглянемо також деякий граф , що зображений на рис. 1.5.

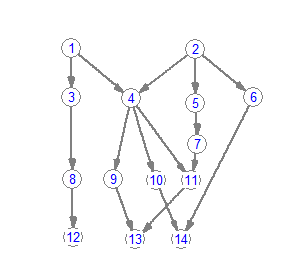


Рис.1.5. Вигляд графу

Задамо вектор .

Побудуємо деякі впорядкування так, щоб потужність множини не перевищувала , . Наприклад так, як записано в таблицях 1.3 і 1.4:

*Таблиця 1.3.*

Допустиме впорядкування *S1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 1 |  |  |  |
| *i=2* | 2 | 3 |  |  |
| *i=3* | 4 | 5 | 6 |  |
| *i=4* | 8 | 9 | 10 | 7 |
| *i=5* | 11 | 12 |  |  |
| *i=6* | 13 | 14 |  |  |

*Таблиця 1.4.*

Допустиме впорядкування *S2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 2 |  |  |  |
| *i=2* | 6 | 5 |  |  |
| *i=3* | 1 | 7 |  |  |
| *i=4* | 3 | 4 |  |  |
| *i=5* | 8 | 9 |  |  |
| *i=6* | 10 | 11 |  |  |
| *i=7* | 12 |  |  |  |
| *i=8* | 13 |  |  |  |
| *i=9* | 14 |  |  |  |

Бачимо, що відповідні впорядкування можуть мати різні довжини.

Будемо називати впорядкування оптимальним, якщо воно має мінімальну довжину.

Позначимо відповідну задачу .

## 2.1. Алгоритми, засновані на рівневому принципі

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. Цей точний алгоритм поліноміальної складності був розроблений для дерева (лісу) для випадку, коли .

Розглянемо деякий довільний граф. Наприклад, зображений на рис. 2.1 граф :

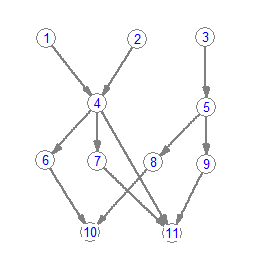


Рис. 2.1. Вигляд графу

Задамо деяку послідовність цілих чисел - вектор .

Побудуємо впорядкування згідно з рівневим принципом. Для цього побудуємо для впорядкування , що зображене в таблиці 2.1. Номер строки *k* характеризує номер місця вершини в упорядкуванні, а відповідно – і її рівень.

*Таблиця 2.1.*

Впорядкування вершин графу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *k=1* | 1 | 2 | 3 |  |
| *k=2* | 4 | 5 |  |  |
| *k=3* | 6 | 7 | 8 | 9 |
| *k=4* | 10 | 11 |  |  |

Тепер побудуємо деякі впорядкування згідно з рівневим принципом. Їх зображено в таблиці 2.2 та 2.3.

*Таблиця 2.2.*

Впорядкування вершин графу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 3 |  |  |  |
| *i=2* | 1 | 2 | 5 |  |
| *i=3* | 4 | 8 | 9 |  |
| *i=4* | 6 | 7 |  |  |
| *i=5* | 10 |  |  |  |
| *i=6* | 11 |  |  |  |

*Таблиця 2.3.*

Впорядкування вершин графу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 1 |  |  |  |
| *i=2* | 2 | 3 |  |  |
| *i=3* | 4 | 5 |  |  |
| *i=4* | 8 | 9 |  |  |
| *i=5* | 6 |  |  |  |
| *i=6* | 7 |  |  |  |
| *i=7* | 10 |  |  |  |
| *i=8* | 11 |  |  |  |

В обох впорядкуваннях вершини заносилися в упорядкування згідно з рівневий принципом. Але в першому випадку довжина впорядкування , а в другому - .

З’ясуємо, чи можливо, керуючись рівневим принципом, побудувати неоптимальне впорядкування для випадку, коли заданий граф являється деревом.

Розглянемо зображений на рис. 2.2 граф . Задамо деякий вектор .

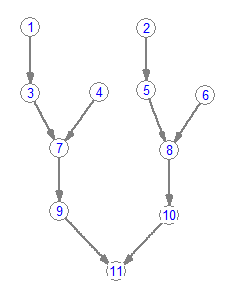


Рис. 2.2. Вигляд графу

Побудуємо впорядкування згідно з рівневим принципом. Для цього побудуємо для впорядкування , що зображене в таблиці 2.4.

*Таблиця 2.4.*

Впорядкування вершин графу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *k=1* | 1 | 2 |  |  |
| *k=2* | 3 | 4 | 5 | 6 |
| *k=3* | 7 | 8 |  |  |
| *k=4* | 9 | 10 |  |  |
| *k=5* | 11 |  |  |  |

Побудуємо деякі впорядкування згідно з рівневим принципом. Їх зображено в таблиці 2.5 та 2.6.

*Таблиця 2.5.*

Впорядкування вершин графу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 1 | 2 |  |  |
| *i=2* | 4 | 6 |  |  |
| *i=3* | 3 | 5 |  |  |
| *i=4* | 7 |  |  |  |
| *i=5* | 8 | 9 |  |  |
| *i=6* | 10 |  |  |  |
| *i=7* | 11 |  |  |  |

*Таблиця 2.6.*

Впорядкування вершин графу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 1 | 2 |  |  |
| *i=2* | 3 | 4 |  |  |
| *i=3* | 5 | 6 | 7 |  |
| *i=4* | 8 |  |  |  |
| *i=5* | 9 | 10 |  |  |
| *i=6* | 11 |  |  |  |

В першому випадку довжина впорядкування , а в другому - . Як бачимо, різниця в довжині викликана різним вибором вершин одного рівня.

Розглянемо ще одне дерево , зображене на рис. 2.3. Задамо вектор .

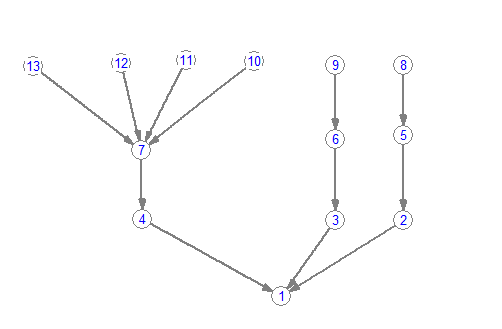


Рис. 2.3. Вигляд графу

Побудуємо для впорядкування , що зображене в таблиці 2.7.

*Таблиця 2.7.*

Впорядкування вершин графу

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *k=1* | 8 | 9 | 10 | 11 | 12 | 13 |
| *k=2* | 5 | 6 | 7 |  |  |  |
| *k=3* | 2 | 3 | 4 |  |  |  |
| *k=4* | 1 |  |  |  |  |  |

Побудуємо деякі впорядкування: перше згідно з рівневим принципом, друге – довільно, з порушенням принципу. Отримані впорядкування зображено відповідно в таблицях 2.8 та 2.9.

*Таблиця 2.8.*

Впорядкування вершин графу , побудоване згідно з рівневим принципом

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 8 | 9 | 10 | 11 |
| *i=2* | 12 | 13 |  |  |
| *i=3* | 5 | 6 | 7 |  |
| *i=4* | 3 |  |  |  |
| *i=5* | 2 |  |  |  |
| *i=6* | 4 |  |  |  |
| *i=7* | 1 |  |  |  |

*Таблиця 2.9.*

Впорядкування вершин графу , побудоване довільно

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *i=1* | 8 | 9 | 10 | 11 |
| *i=2* | 5 | 6 |  |  |
| *i=3* | 13 | 12 | 2 | 3 |
| *i=4* | 7 |  |  |  |
| *i=5* | 4 |  |  |  |
| *i=6* | 1 |  |  |  |

Вершини 5 і 6 були внесені в упорядкування раніше за вершини 12 і 13, хоча вони були вільними. В першому випадку довжина впорядкування , а в другому - .

Як бачимо, можливі порушення рівневого принципу, тобто впорядкування, побудовані згідно з рівневим принципом можуть бути не оптимальними. Як бачимо з прикладів, таке порушення можливе в двох випадках:

1. Різний вибір вершин одного рівня приводить до різної довжини впорядкування.
2. Якщо на певному кроці ставити в упорядкування вершину, що має не максимальний з можливих рівень, довжина впорядкування стає меншою, ніж впорядкування, побудоване згідно з рівневим принципом.

Тобто, існують такі графи і обмеження , що рівневий принцип не завжди будує оптимальні впорядкування.

# 3. Постановка узагальненої задачі 2 паралельного впорядкування

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.

Якщо дозволені переривання у виконанні робіт, то таку задачу можна легко звести до задачі з завданнями з однаковим часом виконання. Для цього 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, збільшиться кількість вершин в графі , а отже, збільшиться і складність розв’язання такої задачі, однак ми зможемо і надалі користуватися тими же методами, які будують впорядкування для завдань з однаковим часом виконання.

Цікавим є питання, чи доцільно дозволяти переривання у системі, тобто, чи належать оптимальні впорядкування належати до класу впорядкувань з перериваннями.

# Умови доцільності дозволу переривань

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 .

Узагальнемо 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 .

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 .

**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)

, – довільне ціле число, ; (1.2)

*;*  (1.3)

1. ; (2)
2. безпосередньо слідує за вершиною, яка має максимальний допустимий час завершення серед всіх вершин , тобто ; (3)

r не слідує після i

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.

# Список літератури

1. Турчина В.А. Алгоритми перерахування всіх паралельних упорядкувань фіксованої довжини: зб. наук. пр. / В.А. Турчина, Ю.С. Зозуля, А.К. Підаш // Питання прикладної математики і математичного моделювання. – Д., 2013. – С. 256 – 261.
2. Коффман Э.Г. Введение в теорию расписаний / Э.Г. Коффман. – М., «НАУКА», 1984. – 333 с.
3. Лазарев А.А. Теория расписаний. Задачи и алгоритмы / А.А. Лазарев, Е.Р. Гафаров. – М., Московский государственный университет им. М.В. Ломоносова (МГУ), 2011. – 222 с.
4. Бурдюк В.Я. Алгоритмы параллельного упорядочения / В.Я. Бурдюк, В.А. Турчина. – Днепропетровск,- 1985. – 84 с.