УДК 519.8 doi: 10.15421/322207

К.Д. Караваєв, В.А. Турчина
Дніпровський національний університет імені Олеся Гончара

УЗАГАЛЬНЕННЯ ЗАДАЧ УПОРЯДКУВАННЯ З УРАХУВАННЯМ НЕПОВНОГО ЗАВАНТАЖЕННЯ

У статі розглядається узагальнення задачі упорядкування на випадок неповного завантаження. Проводиться порівняння з іншими класами задач упорядкування, запропоновано модифікацію алгоритму, заснованого на максимальному паросполученні, яка враховує обмеження на структуру шуканого розв'язку. Показано, що відомі та модифікований алгоритм для цієї задачі є наближеними.

Ключові слова: дискретна оптимізація, теорія розкладів, оптимальні упорядкування, максимальне паросполучення, неповне завантаження.

K.D. Karavaiev, V.A. Turchyna
Oles Honchar Dnipro National University

GENERALISATION OF SEQUENCING PROBLEMS FOR THE CASE OF INCOMPLETE WORKLOAD

A natural formalization of the practical tasks associated with the optimization of the execution order of technologically interrelated jobs to minimize overall completion time is the optimal sequencing problem.

In this article we consider a generalization of the sequencing problem for the case of incomplete workload. An additional constraint is introduced, reflecting the existence of a predefined set of weekends, during which the executors cannot perform their jobs. The proposed generalization is compared with the classical problem, the problem with variable sequencing width and the problem with job assignment.

The problem is studied for the case of a sequencing width equal to two. A modification of the algorithm based on the maximum matching which takes into account the constraints on the structure of the sought solution is proposed. It is shown that the introduced changes in the reachability graph generation do not allow relying on the proof of optimality of the original algorithm. The approach to the proof of the accuracy of the algorithms for these problems through the check of its optimality for the case of problems with dense sequencing is proposed.

A method for reducing given problem to the instance of scheduling problem with job assignment based on introduction of a dummy executor that allows fixing isolated vertices corresponding to weekends at given places in the sequencing is described. It is shown that due to uncertainty of performing of some steps, the well-known algorithms based on maximal matching and lexicographic labeling and the proposed modified algorithm for this case are approximate. Analysis of examples of suboptimality of the mentioned algorithms substantiates the prospect of investigating the possibility of introducing level principle into the modified algorithm.

Keywords: discrete optimization, scheduling theory, optimal sequencing, maximum matching, incomplete workload.

К.Д. Караваев, В.А. Турчина
Днипровский национальный университет имени Олеся Гончара

ОБОБЩЕНИЕ ЗАДАЧ УПОРЯДОЧЕНИЯ ДЛЯ СЛУЧАЯ НЕПОЛНОЙ ЗАГРУЖЕННОСТИ

В статье рассматривается обобщение задачи упорядочения для случая неполной загруженности. Проводится сравнение с другими классами задач упорядочения, предложена модификация алгоритма, основанного на максимальном паросочетании, которая учитывает ограничения на структуру искомого решения. Показано, что известные и модифицированный алгоритм в этом случае являются приближенными.

Ключевые слова: дискретная оптимизация, теория расписаний, оптимальные упорядочения, максимальное паросочетание, неполная загруженность.

Початок дослідження задач оптимального упорядкування вершин графів був пов'язаний із проблемою оптимізації порядку виконання робіт на конвеєрах автомобільного виробництва у середині минулого сторіччя. У наступні десятиріччя були досліджені численні узагальнення цієї задачі, розроблені точні та наближені методи їх розв'язання. У загальному випадку задача виявилася NP-важкою [1], тому особливу увагу науковців приділено пошуку спеціальних випадків, для яких існують поліноміальні точні алгоритми знаходження розв'язку.

Розглянемо класичну постановку задачі оптимального упорядкування та деякі її узагальнення, що виникли з необхідності враховувати додаткові обмеження та умови.

Нехай маємо n завдань, для яких відомі деякі виробничі зв'язки, що обумовлюють відносний порядок їх виконання. Математичну модель цих зв'язків природно подавати у вигляді орієнтовного графу $G(V,U)$, у якому вершини відповідають завданням, а дуги – виробничим зв'язкам. В класичній постановці, з якої розпочинилось дослідження, вважалося, що витрати часу на виконання кожного завдання однакові.

Для подальшої формалізації задач, що виникають, наведемо деякі відомі означення і постановки класичної задачі та деяких її узагальнень [2].

Означення 1. Паралельним упорядкуванням вершин орієнтовного графу $G=(V,U)$ називається таке упорядкування його вершин по місцях, розташованих у лінію, при якому з того, що пара $(i,j)\in U$ випливає, що вершина i розташовується в упорядкуванні S лівіше вершини j, тобто з того, що $(i,j)\in U \wedge (i\in S[p], j\in S[q])$ випливає, що $p<q$.

Означення 2. Довжиною l упорядкування S називається число непорожніх місць в ньому: $l(S)=\sum_{i=1}^{n} sign|S[i]|$, де $S[i]$ – множини елементів, що знаходяться в упорядкуванні S на місці i.

Означення 3. Шириною h упорядкування S називається величина, що дорівнює найбільшій кількості елементів, що розташовані на одному місці: $h(S)=\max\limits_{1\le i\le n}|S[i]|$.

Задача 1 (класична). По заданим графу G і значенню ширини h побудувати паралельне упорядкування мінімальної довжини.

Задача 2 (узагальнена). По заданим графу G і вектору значень ширини $h_i, i=1,\ldots,n$ побудувати паралельне упорядкування мінімальної довжини, в якому на i-ому місці буде стояти не більше h_i вершин.

Задача 3 (з призначенням). По заданим графу G, значенню ширини h та вектору призначень $f_i \in \{1,\ldots,h\}$ $(i=1,\ldots,n)$, який визначає виконавця, що має виконати відповідну роботу, побудувати паралельне упорядкування мінімальної довжини.

Розглянемо також деякі інші модифікації задач.

Означення 4. Тривалістю виконання роботи p_i називається натуральне число, що визначає кількість місць в упорядкуванні, на яких має стояти відповідна вершина.

Означення 5. Задачі, в яких вершини з тривалістю виконання більше одиниці можуть бути розташовані лише на суміжних місцях, називаються задачами без переривань. Якщо ця умова не є обов'язковою, тоді маємо задачу із можливістю переривання.

Відзначимо, що у випадку задачі із можливістю переривання, вона може бути зведена до еквівалентної задачі, в якій тривалості виконання всіх робіт дорівнюють одиниці, шляхом заміни вершин на ланцюжки з довжинами, рівними тривалостям виконання відповідних робіт. У подальшому будемо розглядати лише такі задачі.

Перейдемо тепер до дослідження нового класу задач упорядкування.

Нехай для деякої підмножини виконавців визначені заплановані вихідні, періоди профілактики, тощо, під час яких вони не можуть виконувати роботи, тобто на відповідних місцях в упорядкуваннях обов'язково мають бути пропуски. Зрозуміло, що у випадку, коли роботи не призначені конкретному виконавцю, отримаємо узагальнену задачу. Тому доцільним є розглядати лише випадок задачі із призначенням.

Задача 4 (з вихідними). По заданим графу G, значенню ширини h, вектору призначень $f_i \in \{1,\ldots,h\}$ $(i=1,\ldots,n)$ та множинам вихідних W^j, $j=1,\ldots,h$, які визначають місця, що в упорядкуванні мають бути порожніми в упорядкуванні для i-го виконавця, побудувати паралельне упорядкування мінімальної довжини.

Розглянемо різницю між задачами 2-4 на наступному прикладі.

Приклад. Нехай маємо граф G, зображений на рис. 1, $p_1=2, p_i=1 (i=2,\ldots,7)$.

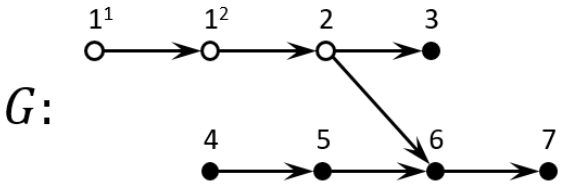

Рис. 1. Граф для прикладу демонстрації відмінностей між задачами 2-4

Для задачі 2 при $h_i = \{2,1,2,1,2\}$ отримаємо наступне оптимальне упорядкування:

$$S_1 = \left\{ \begin{matrix} 1^1 \\ 4 \end{matrix}, 1^2, \begin{matrix} 2 \\ 5 \end{matrix}, 6, \begin{matrix} 3 \\ 7 \end{matrix} \right\},\ l(S_1) = 5.$$

Для задачі 3 при $h = 2$ та $f_i = \{1,1,2,2,2,2,2\}$ (на рис. 1 роботи призначені для першого виконавця зображені колами, для другого – кругами) оптимальним буде наступне упорядкування (крапками позначені вимушено порожні позиції):

$$S_2 = \left\{ \begin{matrix} 1^1 \\ 4 \end{matrix}, \begin{matrix} 1^2 \\ \circ \end{matrix}, \begin{matrix} 2 \\ 5 \end{matrix}, 6, 3, 7 \right\},\ l(S_2) = 6.$$

Розглянемо тепер задачу 4 при $h = 2, W^1 = \{2\}, W^2 = \{4\}$ (відповідають $h_i = 1$ з прикладу до задачі 2) та вектору призначень f_i, аналогічним попередній задачі, отримаємо наступне оптимальне упорядкування (хрестиками додатково позначені вихідні):

$$S_3 = \left\{ \begin{matrix} 1^1 \\ 4 \end{matrix}, \begin{matrix} \times \\ 5 \end{matrix}, \begin{matrix} 1^2 \\ \circ \end{matrix}, \times, \begin{matrix} 2 \\ 6 \end{matrix}, 3, 7 \right\},\ l(S_3) = 7 \qquad (1)$$

Бачимо, що в усіх випадках були отримані упорядкування різної довжини, а отже додаткові умови є суттєвими. Можна вважати, що задача з вихідними є деякою комбінацією задач 2 та 3. Варто відмітити також, що у випадку $W^1 = \{3\}$ на третьому місці обидві позиції виявилися б порожніми, проте, з огляду на те, що вихідні є вимушеними перервами у процесі виконання робіт, довжину упорядкування, як час необхідний для виконання всіх завдань, доцільно вважати рівною 7, що суперечить вихідному означенню довжини упорядкування. В той же час додавання вихідного після 7 місця не має впливати на довжину, оскільки всі роботи на той час вже будуть виконані. Тому необхідно ввести скореговане означення для довжини упорядкування для задачі 4.

Означення 6. Довжиною l упорядкування S для задачі з вихідними називається число непорожніх місць та порожніх місць з вихідними, праворуч від яких є непорожні місця.

У подальшому довжину упорядкування будемо розуміти у значенні, відповідному до задачі, що розглядається, без додаткового уточнення. Також для подальших досліджень можливості розв'язання наведених задач знадобиться наступне означення.

Означення 7. Граничними термінами d_i^s, d_i^e для i-ої роботи називаються числа, що визначають проміжок місць, на якому може стояти відповідна вершина в упорядкуванні, де d_i^s – найменший номер місця, на якому вона може стояти, а d_i^e – найбільший.

Відомо, що при $h = 2$ розв'язок класичної задачі може бути отриманий за поліноміальний час. Першим з алгоритмів, які дозволяють знайти оптимальне упорядкування в цьому випадку, був алгоритм, заснований на максимальному паросполученні [3]. Розглянемо, як цей алгоритм може бути модифіко-

ваний для розв'язання задачі із вихідними на прикладі графу з рис. 1 та додаткових умов з прикладу до задачі 4.

Схема алгоритму, заснованого на максимальному паросполученні

1. Для графу $G(V,U)$ будуємо неорієнтований граф $\overline{G}(V,E)$, де $(i,j)\in E$, якщо у графі G немає шляху ані прямого, ані зворотного шляху між вершинами i та j. Такий граф у подальшому будемо називати графом досяжності.

2. В отриманому графі $\overline{G}$ знаходимо максимальне паросполучення, яке позначимо $M\subseteq E$, тобто підмножину ребер E, що не мають спільних вершин, з максимальною потужністю.

3. В шуканому упорядкуванні S^* вважаємо всі місця вільними і покладаємо $k=1$.

4. Якщо G порожній, то кінець алгоритму.

5. Можливий один з наступних випадків:

а) Серед вільних вершин графу G існує така, яка не належить жодному з ребер у M, тоді обираємо для розташування її.

б) У множині M існує таке ребро (i,j), що вершини i та j є відкритими, тоді обираємо їх для розташування і видаляємо (i,j) з M.

в) У множині M знайдеться така пара ребер (i,p) та (j,q), що вершини i та j є відкритими, а між вершинами p та q є ребро у графі $\overline{G}$, тоді для розташування обираємо вершини i та j, ребра (i,p) та (j,q) видаляємо з M, а ребро (p,q) додаємо у M.

6. Обрані вершини розташовуємо на місце k в упорядкуванні S^* та видаляємо їх з графу G разом з вихідними дугами, якщо вони є. Приймаємо $G:=G, k:=k+1$ та переходимо на 4.

Важливо зазначити, що для знаходження M для довільних неорієнтованих графів відомі алгоритми поліноміальної складності, наприклад алгоритм стиснення квіток.

Схема модифікованого алгоритму

1. Для графу $G(V,U)$ будуємо неорієнтований граф $\overline{G}(V,E)$, де $(i,j)\in E$, якщо у графі G немає ані прямого, ані зворотного шляху між вершинами i та j (граф досяжності).

2. Видаляємо з графу $\overline{G}$ ребра, що з'єднують вершини, що відповідають роботам, які призначені різним робітникам.

3. Додаємо до графу $\overline{G}$ вершини, що відповідають вихідним, та з'єднуємо їх також з усіма вершинами, що відповідають роботам, які призначені іншим виконавцям, та з вихідним іншого виконавця, призначеним на те ж місце в упорядкуванні, якщо він існує. Для кожної вершини вводимо граничні терміни, що відповідають місцям, які вона має займати в упорядкуванні.

4. В отриманому графі $\overline{G}$ знаходимо максимальне паросполучення, яке позначимо $M\subseteq E$, тобто підмножину ребер E, що не мають спільних вершин, з максимальною потужністю.

5. В шуканому упорядкуванні S^* вважаємо всі місця вільними і покладаємо $k=1$.

6. Якщо G порожній, то кінець алгоритму.

7. Якщо є вихідні, які відповідають k-ому місцю, то віддаємо їм перевагу (якщо вони поодинокі) або ребрам, до яких вони відносяться.

8. Можливий один з наступних випадків:

а) Серед вільних вершин графу G існує одна чи декілька таких, що не належать жодному з ребер у M, тоді обираємо для розташування їх.

б) У множині M існує таке ребро (i,j), що вершини i та j є відкритими, тоді обираємо їх для розташування і видаляємо (i,j) з M.

в) У множині M знайдеться така пара ребер (i,p) та (j,q), що вершини i та j є відкритими та відповідають роботам, призначеним різним виконавцям, тоді для розташування обираємо вершини i та j, ребра (i,p) та (j,q) видаляємо з M, а ребро (p,q) додаємо у M, якщо воно є у графі $\overline{G}$.

г) Серед вільних вершин графу G існує одна вершина i, що не належать жодному з ребер у M, та деяка інша j, що належить деякому ребру (j,q) у M, та може бути виконана одночасно із i, тоді обираємо для розташування їх, а вершину q додаємо до поодиноких.

9. Об'єднуємо поодинокі вершини у пари, якщо між ними є ребра у графі досяжності, та додаємо отримані пари у M.

10. Обрані вершини розташовуємо на місце k в упорядкуванні S^* та видаляємо їх з графу G разом з вихідними дугами, якщо вони є. Приймаємо $G:=G, k:=k+1$ та переходимо на 4.

Відмітимо також, що додатковою мотивацією для модифікації та застосування саме алгоритму, заснованого на максимальному паросполученні, є те, що для задач 3 та 4 відповідний граф досяжності є дводольним. Це безпосередньо випливає з другого етапу алгоритму: залишаємо ребра лише між вершинам, що відповідають роботам різних виконавців. Дводольність графу досяжності в свою чергу спрощує пошук максимального паросполучення.

Приклад розв'язання. На першому етапі алгоритму необхідно побудувати граф досяжності (неорієнтовний граф, вершини якого з'єднані ребром, якщо у вихідному графі між ними немає орієнтовного шляху у будь-якому напрямку). Отримаємо граф $\overline{G_1}$ з рис. 2.

Перед тим, як перейти до пошуку максимального паросполучення, згідно алгоритму, врахуємо спочатку той факт, що в шуканому упорядкуванні вершини, які відповідають роботам, що призначені різним робітникам, не можуть стояти на одному місці. Це можна зробити, видаливши з $\overline{G_1}$ ребра, що з'єднують вершини, які призначені одному виконавцю. Таким чином ці ребра не зможуть потрапити до максимального паросполучення, а отже в отриманому за алгоритмом упорядкуванні на кожному місці зможуть знаходитись

лише вершини, що призначені різним виконавцям. Отримаємо граф $\overline{G_2}$, штрихові лінії позначають видалені ребра.

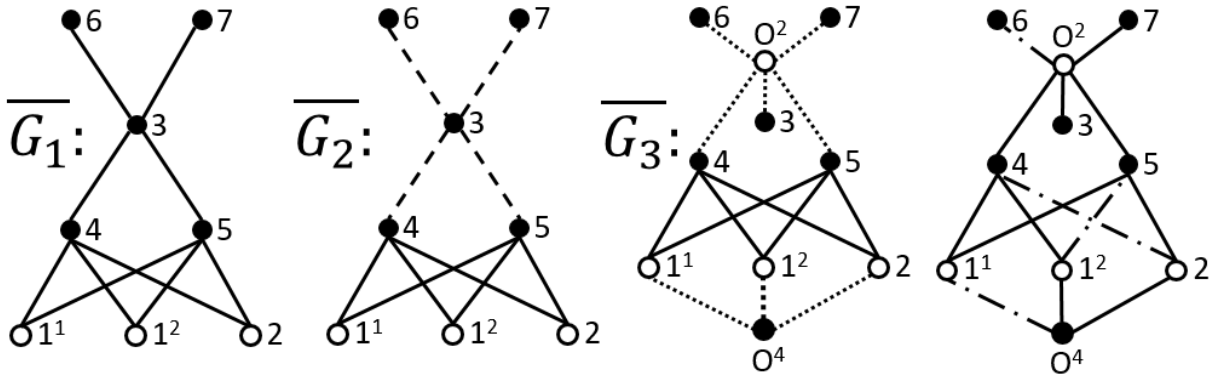

Рис. 2. Етапи перебудови графу досяжності з прикладу розв'язання задачі 4

Врахуємо тепер наявність вихідних. Додамо до вихідного графу дві ізольовані вершини o^2 та o^4, які відповідають вихідним першого та другого виконавця відповідно. Для того, щоб врахувати, що вихідні призначені на конкретний день, визначимо для них граничні терміни рівні місцям в упорядкуванні, які вони мають займати. Маємо додати ці вершини також до графу досяжності разом із ребрами, що з'єднують їх з усіма вершинами, призначеними для іншого виконавця. Отримаємо граф $\overline{G_3}$, додані ребра позначені пунктирними лініями. Наявність граничних термінів буде врахована пізніше.

Знайдемо тепер максимальне паросполучення в графі $\overline{G_3}$, ним буде $M = \{(1^1, o^4), (1^2, 5), (2,4), (o^2, 6)\}$ (на рис. 2 зображене штрихпунктирною лінією), при цьому вершини 3 та 7 не ввійшли до жодної з пар.

Перейдемо до побудови шуканого упорядкування.

Відкритими вершинами у вихідному графі є 1^1 та 4, вони належать до пар $(1^1, o^4)$ та $(2,4)$, тому, відповідно до алгоритму, розташовуємо їх в упорядкуванні на перше місце: $S[1] = \{1^1, 4\}$ та додаємо до паросполучення пару $(2, o^4) \to M$. Зазначимо, що вершини o^2 та o^4 також були відкритими, проте, відповідно, до граничних термінів вони не можуть бути розміщенні на першому місці в шуканому упорядкуванні.

Відповідно до умов, у першого виконавця має бути вихідний на 2 місці, тому на цьому етапі маємо обов'язково розмістити вершину o^2, яка належить парі $(o^2, 6)$. Іншими відкрити вершинами є 1^2 та 5, які входять до однієї пара, та o^4, розміщення якої на 4 місці порушить граничні терміни. Аналогічно першому кроку об'єднаємо ці пари, тоді $S[2] = \{o^2, 5\}$. Проте пара $(1^2, 6)$ не може бути додана до M, оскільки між цими вершинами немає ребра у графі $\overline{G_3}$, тому додамо їх до вершин без пар.

На цьому етапі відкритими є лише вершини 1^2 та o^4, яку не можна розмістити на 3 місці, тому можливим є лише розміщення $S[3]=1^2$.

На 4 місці обов'язково має бути розміщена вершина o^4, вона утворює пару із вершиною 2, що також є відкритою, тому розташовуємо їх на це місце $S[4]=\{2,o^4\}$, а отже всі пари з множини M були використані.

Залишається розташувати вершини без пар, отримаємо: $S[5]=6$, $S[6]=3$, $S[7]=7$. На цьому побудова упорядкування завершується.

Отже, упорядкування, отримане за модифікованим алгоритмом, збігається з оптимальним з прикладу для задачі 4.

Варто зробити наступні зауваження. По-перше, при об'єднанні пар важливо додатково слідкувати, аби на одне місце не потрапили вершини, що призначені одному виконавцю. По-друге, задачі із граничними термінами, на відміну від задач 1-4, взагалі кажучи, можуть не мати розв'язку, проте легко переконатися, що будь-який допустимий розв'язок задачі 4 задовольняє введеним граничним термінам, згідно означенню вихідного дня, а отже цей перехід є еквівалентним. По-третє, аналогічні модифікації алгоритму можуть бути застосовані до пошуку розв'язків для узагальненої задачі при $h_i \in \{1,2\}$ та задачі з призначенням: для задачі 2 необхідно ввести ізольовані вершини з граничними термінами, які відповідають місцям з $h_i=1$; для задачі 3 достатньо лише видалити ребра у графі досяжності, що з'єднують вершини-роботи, що призначені одному виконавцю. По-третє, запропонований алгоритм має принципові відмінності від алгоритму для класичної задачі, що не дозволяють стверджувати про оптимальність упорядкування, отриманого за модифікованим алгоритмом, у загальному випадку. Справді, бачимо, що оптимальний розв'язок (1) містить лише три заповнені місця, а отже він відповідає паросполученню з 3 парами вершин, яке не є максимальним. В зв'язку з цим при розміщенні вершин на 2 місце не змогли утворити нову пару та додали вершини до поодиноких, що також суперечить алгоритму.

З усього переліченого випливає необхідність обґрунтування оптимальності модифікованого алгоритму для задачі 4.

Розглянемо, що буде відбуватися із оптимальним упорядкуванням, якщо додамо до графу ізольовані вершини, що відповідають тим завданням, які можуть виконувати обидва виконавці. Слід відмітити, що поки в оптимальному упорядкуванні лишаються місця, на яких розташовано менше двох вершин, його довжина не буде змінюватися, оскільки додані вершини завжди можна розташувати на одну з вільних позицій. Отже, після приєднання деякої кількості таких вершин отримаємо упорядкування, в якому на всіх місцях розташовано по 2 вершини. Таке упорядкування назвемо щільним.

Зрозуміло, що кількість таких вершин, необхідна для отримання щільного упорядкування, заздалегідь не відома. Проте легко побачити, що парність кількості необхідних вершин буде збігатися з парністю суми тривалостей виконання робіт та кількості вихідних. Окрім цього, якщо до графу із щільним

упорядкуванням додати парну кількість ізольованих непризначених вершин, то оптимальне упорядкування для такого графу також буде щільним, оскільки надлишкові вершини можна розмістити разом.

Зроблені зауваження дозволяють довести твердження, аналогічне теоремі про алгоритми, що точно знаходять щільні упорядкування [4], а отже для доведення оптимальності модифікованого алгоритму достатньо показати, що він знайде щільне упорядкування, якщо воно існує, яке буде відповідати максимальному паросполученню.

Приклад. Нехай маємо граф G^2, зображений на рис. 3, $W^1 = \varnothing$, $W^2 = \{1\}$, $h = 2$.

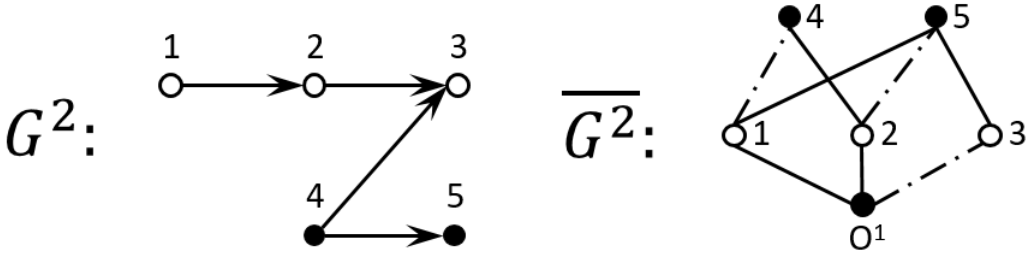

Рис. 3. Приклад застосування алгоритму для задачі з щільним упорядкуванням

Призначення робіт виконавцям, граф досяжності $\overline{G^2}$, отриманий за модифікованим алгоритмом, та максимальне паро сполучення $M^2 = \{(1,4),(2,5),(3,o^1)\}$ зображені на рис. 3, згідно з попередніми позначеннями.

Відповідно до алгоритму, на першому кроці маємо обов'язково розмістити вершину o^1. Вона належить до ребра $(3,o^1)$, в якому вершина 3 є закритою. Обидві відкриті вершини належать ребру $(1,4)$, тому розмітимо на перше місце в упорядкуванні пару $S[1] = \{1,o^1\}$, а вершини 3 та 4 вимушені додати до поодиноких, оскільки у $\overline{G^2}$ між ними немає ребра.

На другому кроці відкритими є вершини 2 та 4, перша з яких належить до ребра $(2,5)$, а друга є поодинокою. Для того, аби розташувати обидві вершини на друге місце, маємо розірвати пару: $S[2] = \{2,4\}$. Причому дві поодинокі вершини 3 та 5 у $\overline{G^2}$ є суміжними, а отже можемо додати це ребро до M^2.

Таким чином, на останньому кроці розташовуємо вершини, що належать ребру $(3,5)$ на третє місце: $S[3] = \{3,5\}$. Вихідне упорядкування є щільним, а отже оптимальним:

$$S_4 = \left\{\begin{matrix}1\\ \times\end{matrix}, \begin{matrix}2\\ 4\end{matrix}, \begin{matrix}3\\ 5\end{matrix}\right\},\ l(S_4) = 3.$$

Отримане упорядкування є оптимальним, але при цьому було порушено схему класичного алгоритму. Крім того слід відмітити, що не кожне початкове максимальне паросполучення гарантує нам отримання розв'язку без порушень класичного алгоритму навіть у випадку щільного упорядкування.

Відзначимо також, що задача 4 може бути зведена до задачі 3 аналогічно тому, як задача 2 може бути зведена до задачі 1. Введемо фіктивного виконавця, призначені роботи якого утворюють ланцюжок, довжина якого дорівнює довжині оптимального упорядкування. Зрозуміло, що оптимальне упорядкування для отриманої задачі при $h' = h + 1$ можна побудувати шляхом приєднання до кожного місця оптимального упорядкування вихідної задачі однієї вершини з доданого ланцюжка. А отже кожна вершина-робота фіктивного виконавця буде однозначно відповідати деякому місцю в оптимальному упорядкуванні. З'єднаємо тепер кожну вершину, що відповідає вихідному, вхідною дугою з вершиною ланцюжка, що відповідає попередньому місцю в упорядкуванні до вихідного, та вихідною з вершиною, що відповідає наступному місцю в упорядкуванні після вихідного. Легко впевнитись, що розширене упорядкування після введення додаткових дуг також буде допустимим та оптимальним. При цьому вершини-вихідні тепер можуть бути розташовані лише на відповідних місцях, тому можемо виключити з задачі ці обмеження. Отримана задача є прикладом задачі 3, невідому довжину оптимального упорядкування можна знайти за допомогою бінарного пошуку, оскільки шукана довжина – мінімальна довжина ланцюжка, при якій на кожному місці в отриманому упорядкуванні буде знаходитись одна вершина з доданого ланцюжка.

Тому у подальших дослідженнях увага приділялася лише задачі 3.

Насправді, можемо також взагалі не отримати оптимальне упорядкування через неоднозначність обрання пар вершин для розміщення при застосуванні алгоритму, навіть для найпростіших графів.

Приклад. Розглянемо задачу 3 для графів з рис. 4 та $h = 2$, призначення робіт визначається помітками вершин на рисунку.

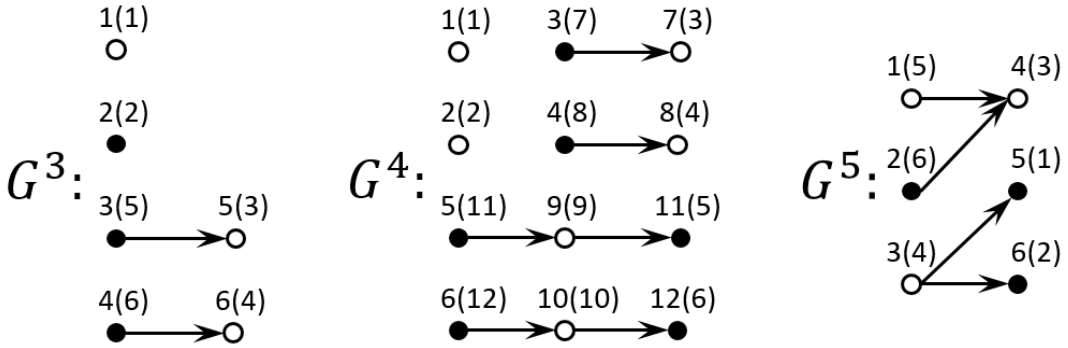

Рис. 4. Приклади неоптимальності алгоритму для задач з щільним упорядкуванням

Легко побачити, що допустимим максимальним паросполученням для графу G^3 буде $M^3 = \{(1,2),(6,3),(5,4)\}$. Відповідно до алгоритму можемо розмістити на перше місце пару вершин $(1,2)$, проте тоді нерозміщеними та відкритими залишаться лише вершини 3 та 4, які не можуть бути розміщені разом, оскільки відповідають роботам, що мають бути виконані одним і тим же виконавцем. Тому зможемо розмістити лише одну з них на другому місці, на-

приклад, вершину 3. На подальших кроках алгоритму зможемо розмістити пару вершин $(5,4)$ та вершину 6. У результаті отримаємо упорядкування:

$$S_5 = \begin{matrix} 1 & \circ & 5 & 6 \\ 2, & 3, & 4, & \circ \end{matrix}, \; l(S_5) = 4.$$

Отримане упорядкування не є оптимальним. Справді, якби на першому етапі замість розміщення пари вершин $(1,2)$ розмістили б пару $(1,3)$ та додали до M^3 нову пару $(6,2)$, то отримали б упорядкування меншої довжини:

$$S_5^* = \begin{matrix} 1 & 5 & 6 \\ 3, & 4, & 2 \end{matrix}, \; l(S_5^*) = 3.$$

Отже, бачимо, що не змогли отримати оптимальний розв'язок через невдале обрання пари вершин для розміщення. Насправді, в алгоритмі немає жодних вказівок щодо пріоритетності розміщення вершин, відповідно до випадків а-г. В свою чергу для класичного алгоритму доведено, що обрання вершин для розміщення не вплине на оптимальність отриманого розв'язку.

З аналізу попереднього прикладу можна зробити припущення, що для отримання оптимального розв'язку потрібно забороняти розміщення ізольованих вершин парами, якщо у графі є інші неізольовані вершини. Таким чином матимемо вільні вершини для утворення пар на більш пізніх етапах побудови упорядкування. Виявляється, що ця умова не є достатньою для подолання проблеми неоднозначності вибору.

Перейдемо тепер до графу G^4. Максимальним паросполученням для нього буде: $M^4 = \{(10,5),(9,6),(7,11),(8,12),(1,3),(2,4)\}$. Бачимо, що для цього графу ізольовані вершини належать до різних пар. При застосуванні алгоритму можемо розташувати спочатку пари $(1,3)$ та $(2,4)$. Далі маємо можливість лише комбінувати пари: розміщуємо пари $(7,5)$ і $(8,6)$ та додаємо до M^4 пари $(10,11)$ і $(9,12)$. Таким чином прийшли до підграфу та пар, що є аналогічними до попереднього прикладу. Результуюче упорядкування для цього прикладу:

$$S_6 = \begin{matrix} 1 & 2 & 7 & 8 & 9 & 10 & \circ \\ 3, & 4, & 5, & 6, & \circ, & 11, & 12 \end{matrix}, \; l(S_6) = 7.$$

У цьому прикладі також змогли б зменшити довжину при іншому порядку розташування пар. Розпочнемо з комбінування пар $(1,3)$ і $(10,5)$ та $(2,4)$ і $(9,6)$, тоді на перші два місця будуть розміщені пари $(1,5)$ і $(2,6)$, а пари $(10,3)$ і $(9,4)$ додані до M^4. Далі розміщуємо пари, що залишились у M^4, тоді отримаємо оптимальне упорядкування:

$$S_6^* = \begin{matrix} 1 & 2 & 9 & 10 & 7 & 8 \\ 5, & 6, & 4, & 3, & 11, & 12 \end{matrix}, \; l(S_6^*) = 6.$$

Аналізуючи цей приклад, можемо дещо уточнити попереднє припущення. Побачили, що для отримання оптимального упорядкування мали спочатку розмістити вершини 5 та 6, що є початками найдовших ланцюжків у графі. В той же час при розміщенні на першому етапі вершин 3 та 4, що також є почa-

тками ланцюжків, проте коротших, отримали неоптимальне упорядкування. Ці спостереження разом з необхідністю збереження ізольованих вершин є основними положеннями рівневого принципу побудови упорядкувань [5].

Відомим алгоритмом, який використовує рівневий принцип, є алгоритм, заснований на лексикографічному помічені вершин [6]. Він також є поліноміальним та точним для класичної задачі при $h=2$ за умови, що у графі немає транзитивних дуг.

Насправді, легко переконатися, що для попередніх двох прикладів за алгоритмом було б отримано оптимальне упорядкування (пріоритети вершин, отримані за алгоритмом, зазначені на рис. 4 у дужках). Проте виявляється, що можемо отримати неоптимальний розв'язок через довільність обрання міток для вершин без вихідних дуг.

Розглянемо граф G^5, нехай вершини без вихідних дуг мають наступні помітки: вершина 4 – помітку 3, вершина 5 – помітку 1, а вершина 6 – помітку 2. Відповідно до алгоритму, для вершини 1 послідовність поміток безпосередніх послідовників (3), для вершини 2 – (3), для вершини 3 – (2,1). Тоді, відповідно до лексикографічного упорядкування послідовностей, маємо $(3)\succ=(3)\succ=(2,1)$, а отже вершини 3 отримає помітку 4, вершина 2 – помітку 5, а вершина 1 – помітку 6 (отримані помітки зображені на рис. 4 у дужках). При побудові упорядкування на перше місце розмістимо вершини 1 та 2, на друге місце можемо розмістити лише вершину 3, оскільки вершини 3 та 4 відповідають роботам, що мають бути виконані одним і тим самим виконавцем; після чого розміщуємо разом вершини 4 та 5 і на останньому місці буде вершина 6. Отримаємо наступне упорядкування:

$$S_7=\begin{Bmatrix}1 & 3 & 4 & \circ\\ 2, & \circ, & 5, & 6\end{Bmatrix},\ l(S_7)=4.$$

Аналогічно попереднім прикладам, могли б отримати коротше упорядкування, якщо б розмістили спочатку вершини 2 та 3, потім змогли б розташувати пари вершин 1 та 5 і 4 та 6. При цьому результуючим упорядкуванням було б наступне:

$$S_7^*=\begin{Bmatrix}3 & 5 & 6\\ 2, & 1, & 4\end{Bmatrix},\ l(S_7^*)=3.$$

Отже, бачимо, що обидва алгоритми є в загальному випадку лише наближеними. До переваг модифікованого алгоритму можна віднести легкість впровадження обмежень, обумовлених структурою шуканого розв'язку, а до недоліків значну невизначеність виконання кроків при побудові упорядкування. В той же час алгоритм, заснований на лексикографічному помічені, має недоліками переваги попереднього та навпаки.

В подальшому цікавим є дослідження питання можливостей поєднання переваг розглянутих алгоритмів в одному, чисельне визначення частоти знаходження точних розв'язків алгоритмами для різних класів графів, а також пошук та обґрунтування класів графів, для яких зазначені алгоритми є точними.

Бібліографічні посилання

1. Ullman J. D. NP-complete scheduling problems. *J. of Comput. And Syst. Sci.* 1975. P. 384–393.
2. Бурдюк В. Я., Турчина В.А. Алгоритмы параллельного упорядочения: Учебное пособие. Днепропетровск: ДГУ, 1985. 83 с.
3. Fujii, M., Kasami T., Ninomija K. Optimal sequencing of two equivalent processors. *SIAM J. Appl. Math.* 1969. № 4. P. 784–789.
4. Turchyna V., Karavaiev K. Analysis of algorithms for constructing dense sequencing of digraphs vertices. *CEUR Workshop Proceedings 2608, CEUR-WS.org*. 2020. P. 690-703.
5. Hu T.C. Parallel sequencing and assembly line problems. *Oper. Res.* 1961. P. 841–849.
6. Fujii M. Optimal sequencing of two equivalent processors. *SIAM J. Appl. Math*. 1971. № 1. P. 141–162.

Надійшла до редколегії 29.09.2022.