УДК 519.1

О.О. КУБАЙЧУК
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

РОЗВ'ЯЗАННЯ ПРИКЛАДНИХ ЗАДАЧ З ВИКОРИСТАННЯМ БАЗОВИХ АЛГОРИТМІЧНИХ ПРИМІТИВІВ ТЕОРІЇ ГРАФІВ

Розглянуто можливість постановки прикладних задач у термінах теорії графів. Зокрема, розглянуто задачі про оптимальну структуру підприємства, контроль за фінансовими операціями, визначення оптимальної послідовності робіт. Реалізовано підхід до розв'язання таких задач з використанням алгоритмічних примітивів на графах. Застосування графів як моделей багатьох прикладних задач зумовлене як наочністю поняття графа, так і наявністю точних алгоритмів на графах з поліноміальною складністю.

Процес розв'язання задачі побудови оптимальної структури послідовно включає обчислення сильно зв'язних компонент на графі задачі, побудову графа компонентів і топологічне сортування графа компонентів з метою переходу до лінійної структури. Під оптимальною структурою підприємства розуміється лінійне впорядкування його підрозділів. Метод розв'язання задачі контролю за фінансовими операціями базується на властивостях пошуку у глибину для ациклічного графа. Класифікація ребер лісу пошуку у глибину дозволяє робити припущення про приховані зв'язки за наявності зворотних ребер. Варто зазначити, що запропонована методика не дає відповіді про кількісний характер зв'язку, тільки фіксує його як факт. Частинним випадком загальної задачі лінійного програмування є задача визначення послідовності виконання робіт, задана системою різницевих обмежень. Розв'язання системи різницевих обмежень зводиться до обчислення ваги найкоротшого шляху у відповідному графі обмежень. Для обчислення найкоротшого шляху з однієї вершини застосовано алгоритм Беллмана – Форда, застосування якого дозволяє отримати результат навіть за наявності ребер від'ємної ваги.

Коректність теоретичних міркувань підтверджується практичними обчисленнями за допомогою спеціально розробленого програмного забезпечення. Обчислення проводились у середовищі "Mathcad v. 13". Розроблено відповідні процедури STRONG_COMP, DFS_CTRL, BELL_FORD_CONST.

Ключові слова: *теорія графів, алгоритми на графах, пошук у глибину, топологічне сортування, найкоротший шлях, алгоритм Беллмана – Форда.*

O.O. KUBAYCHUK
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"

SOLVING APPLIED PROBLEMS USING BASIC ALGORITHMIC PRIMITIVES OF GRAPH THEORY

The possibility of formulating applied problems in terms of graph theory is considered. In particular, the problems of the optimal structure of the enterprise, control over financial operations, and determination of the optimal sequence of works are considered. An approach to solving such problems using algorithmic primitives on graphs is implemented. The use of graphs as models of many applied problems is due to both the clarity of the concept of a graph and the availability of exact algorithms on graphs with polynomial complexity.

The process of solving the problem of constructing an optimal structure sequentially includes the calculation of strongly connected components on the graph of the problem, the construction of a graph of components, and topological sorting of the graph of components in order to transition to a linear structure. The optimal structure of the enterprise is understood as a linear ordering of its subdivisions. The method of solving the problem of control over financial operations is based on the properties of depth-first search for an acyclic graph. The classification of edges of a depth-first search forest allows us to make certain assumptions about hidden connections in the presence of reverse edges. It is worth noting that the proposed method does not give an answer about the quantitative nature of the connection, fixing only its fact. A partial case of the general linear programming problem is the problem of determining the sequence of work execution, given by a system of difference constraints. Solving a system of difference constraints is reduced to calculating the weight of the shortest path in the corresponding constraint graph. To calculate the shortest path from one vertex, the Bellman–Ford Algorithm was used, the application of which allows obtaining a result even in the presence of edges of negative weight.

The correctness of theoretical considerations is confirmed by practical calculations using specially developed software. The calculations were performed in the Mathcad v. 13 environment. The corresponding procedures STRONG_COMP, DFS_CTRL, BELL_FORD_CONST were developed.

Key words: *graph theory, algorithms on graphs, depth-first search, topological sorting, shortest path, Bellman – Ford algorithm.*

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

Постановка проблеми

Під математичною моделлю прикладної задачі зазвичай розуміють її описання та розв'язання в межах визначеної математичної теорії. Теорія графів вивчає властивості однієї з базових комбінаторних структур дискретної математики, власне графи. Застосування графів як моделей багатьох прикладних задач зумовлене як наочністю поняття графа, так і наявністю численних алгоритмічних примітивів [1; 2] теорії графів, наприклад, фундаментальних методів пошуку на графі, задач зв'язності, побудови мінімального кістякового дерева, задачі про максимальний потік, методу топологічного сортування та інших.

Комунікаційні, транспортні, інформаційні, соціальні мережі – це актуальні об'єкти, структуру яких природно зображати графом. У процесі функціонування таких мереж необхідно постає низка задач, що потребує свого вирішення. Потрібно розрізняти розв'язання таких задач у теоретико-графовій і комбінаторній постановках. В останньому випадку мова йде про задачі комбінаторної оптимізації (далі – ЗКО) [3]. ЗКО часто інтерпретують як задачу обчислення екстремумів відповідної цільової функції на заданій комбінаторній структурі, зокрема на графі прикладної задачі. Для розв'язання ЗКО розроблено цілий спектр відповідних алгоритмів комбінаторної оптимізації (далі – АКО), починаючи від поважних угорського та симплекс-алгоритму, до сучасних евристичних алгоритмів, до яких належать, зокрема, еволюційні та роєві алгоритми.

Розв'язання прикладної задачі в теоретико-графовій постановці спирається на можливості сучасної теорії графів, зокрема, *алгоритми на графах*. У даній роботі розглядаються: задача про оптимальну структуру підприємства, задача контролю за фінансовими операціями, задача визначення послідовності робіт. Розв'язання перелічених задач проводилось з використанням алгоритмічних примітивів теорії графів.

Аналіз останніх досліджень і публікацій

Можливість застосування алгоритмів на графах до розв'язання прикладних задач, моделями яких є графи, використовується давно. Зосередимось на публікаціях, тематика яких є дотичною до проблем, що вирішуються в даній роботі, та на публікаціях, які означують перспективні напрями в розвитку блокчейн-технологій, пов'язаних із застосуванням блок-графів.

У роботі [4] розглянуто частинний випадок задачі оптимізації управління методом виділення сильно зв'язних компонент на графі. У [5] висвітлено питання щодо умов стабілізації фінансової системи, моделлю якої є деяка транспортна мережа з можливістю застосування відповідних алгоритмів на графах. Задача планування будівельних робіт розв'язується в [6] на базі алгоритму топологічного сортування. Ще в далекому 1958 р. Річард Беллман презентував алгоритм пошуку найкоротших шляхів з однієї вершини навіть у разі, коли вага ребер графа є від'ємною [7]. Також алгоритм визначає наявність на графі циклу від'ємної ваги, досяжного із джерела. Алгоритм Беллмана – Форда застосовується в роботі [8], де розв'язано задачу планування послідовності виконання будівельно-монтажних робіт, визначених за допомогою системи різницевих обмежень.

Сатоші Накамото – псевдонім дослідника (чи групи дослідників), якому вперше вдалося побудувати цілком децентралізовану мережу – мережу, яка функціонувала без сервера, без централізованого керування. У своїй єдиній роботі [9] він повідомив, що побудував електронну однорангову систему грошової одиниці – біткоїну (Bitcoin). Проблема розгалуження (форку) згідно із протоколом Накамото вирішується за правилом найдовшого ланцюжка блоків, тобто валідним є тільки один ланцюжок, що має несуперечливу історію транзакцій. Перехід від лінійної структури блоків до структури зі складнішою топологією, що допускає одночасне існування декількох валідних ланцюжків, очевидно, приведе до пришвидшення опрацювання транзакцій [10]. Узагалі, тема масштабування блокчейну без втрати його захищеності є актуальною, одним із напрямів є спроба відходу від класичного блокчейну (лінійної структури) до

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

напрямлених ациклічних графів (DAG), з розробленням відповідних протоколів консенсусу [11]. Дослідження технології графчейн (блок-граф) та розв'язання задачі побудови лінійного порядку для напрямленого ациклічного графа є актуальним завданням, яке можна вирішувати, зокрема, із застосуванням алгоритмів на графах.

Мета дослідження

Метою дослідження є постановка та розв'язання прикладних задач у термінах і з використанням алгоритмів на графах.

Основні поняття, означення і алгоритми

Серед алгоритмічних примітивів теорії графів особливе місце належить групі алгоритмів, що відповідає за обхід (пошук) на графі. Такі алгоритми є невід'ємним складником багатьох інших алгоритмічних примітивів теорії графів. У задачах, що розглядатимуться в цій роботі, пошук на графах організовано як пошук у глибину (DFS, Depth-First Search), у модифікації зі складністю $O(V + E)$, що прийнята в [1].

Мітки. Відповідно до [1], у процесі обходу вершини графа розфарбовуються і для кожної вершини $v \in V$ графа $G = (V, E)$ підтримуються атрибути:

– $\pi[v]$ – попередник вершини v (вершина, з якої вершину v було відкрито, якщо попередника нема, то $\pi[v] = NIL$;

– $d[v]$ – фіксація миті відкриття вершини v (зафарбовується в сірий колір, GRAY);

– $f[v]$ – фіксація миті завершення сканування списку суміжності вершини v (вершина зафарбовується в чорний колір, BLACK).

Очевидно, *часові мітки* вершини v, $d[v]$, $f[v] \in [1, 2|V|]$, і, оскільки, для кожної вершини v існує тільки одна подія її відкриття і одна подія завершення сканування її списку суміжності, то:

$$\forall v \in V : d[v] < f[v].$$

Розфарбовування вершин відбувається таким чином, що до моменту $d[v]$ – колір вершини WHITE; між $d[v]$ і $f[v]$ – колір GRAY; після $f[v]$ – колір BLACK.

Часові мітки алгоритму пошуку у глибину є важливою інформацією для алгоритмічних примітивів теорії графів уже вищого рівня.

Ліс пошуку у глибину. Разом з атрибутами вершин підтримується підграф $G_\pi = (V, E_\pi)$ графа G, де:

$$E_\pi = \{(\pi[v], v) : v \in V, \pi[v] \neq NIL\}.$$

G_π називають *predecessor subgraph* (підграф передування, PS). На відміну від пошуку в ширину, де відповідний PS утворює дерево, G_π пошуку у глибину може складатися з декількох дерев, тобто є лісом пошуку у глибину.

Класифікація ребер. Пошуком у глибину можна скористатися для класифікації ребер вихідного графа G. Щодо лісу G_π визначають:

– ребра дерев (позначають "T") – це ребра графа G_π. Якщо в результаті дослідження ребра (u, v), відкрито білу вершину, то ребро (u, v) є ребром дерева;

– зворотні ребра (позначають "B") – це ребра (u, v), що з'єднують вершину u з її предком v у дереві пошуку у глибину. Зворотними ребрами, наприклад, є ребра-цикли;

– прямі ребра (позначають "F") – це ті ребра (u, v), які не є ребрами дерева, але з'єднують вершину u з її потомком v на дереві пошуку у глибину;

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

– перехресні ребра (позначають "C") – усі інші ребра. Вони можуть, наприклад, з'єднувати вершини в різних деревах.

У даній роботі застосовується модифікований алгоритм DFS, описаний у [4], який не розрізняє прямі та перехресні ребра, об'єднує їх в один клас, що позначається "FC". Критерієм класифікації є колір вершини v при першому зверненні до неї: білий колір – ребро дерева "T"; сірий колір – визначає зворотне ребро "B"; чорний колір – пряме чи перехресне ребро "FC".

Топологічне сортування. Топологічним сортуванням орієнтованого ациклічного графа $G = (V, E)$ є таке його зображення, за якого вершини графа G упорядковуються вздовж горизонтальної лінії (для визначеності, зліва направо) і $\forall (u, v) \in G$, вершина u за такого впорядкування розташована до вершини v. Зауважимо, що таке впорядкування можливе тільки якщо граф ациклічний.

Одним із застосувань пошуку у глибину є його використання в алгоритмі топологічного сортування TOPOLOGICAL_SORT, наведеному в [1]. Топологічно відсортовані вершини v розташовуються в лінію в порядку спадання відповідних значень $f[v]$.

Доведення коректності алгоритму TOPOLOGICAL_SORT спирається на теорему, що характеризує орієнтований ациклічний граф.

Теорема 1. Орієнтований граф G є ациклічним тоді й тільки тоді, коли пошук у глибину на G не знаходить зворотних ребер.

Сильно зв'язні компоненти. Класичним застосуванням пошуку у глибину є розкладання орієнтованого графа на сильно зв'язні компоненти (далі – СЗК). Сильно зв'язною компонентою орієнтованого графа $G = (V, E)$ є максимальна множина вершин $C \subseteq V$, така, що для будь-якої пари вершин $u, v \in C$, u і v досяжні одна з одною.

Для обчислення сильно зв'язних компонентів орієнтованого графа скористаємось алгоритмом STRONGLY_CONNECTED_COMPONENTS з [1], у якому використовується транспонування вихідного графа G (транспонований граф $G^T = (V, E^T)$, складається з ребер G, узятих у зворотному напрямку). Зазначимо, що графи G і G^T мають ті самі сильно зв'язні компоненти. Теорема 22.16 з [1] гарантує, що алгоритм ефективно обчислює СЗК.

Граф компонентів – граф, утворений з вихідного графа G стисненням у точку всіх ребер між суміжними вершинами в кожній сильно зв'язній компоненті.

Теорема 2. Граф компонент є орієнтованим ациклічним графом.

Доведення теореми випливає з леми 22.13 в [1].

Найкоротші шляхи з однієї вершини. Нехай $G = (V, E)$ – зважений орієнтований граф, $w : E \to \mathbb{R}$ – вагова функція. Вагою шляху $p = \langle v_0, v_1, ..., v_k \rangle$ називають суму ваг ребер, що його утворюють:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i) .$$

Вага найкоротшого шляху з вершини u до вершини v:

$$\delta(u, v) = \begin{cases} \min\limits_{p} w(p), \text{ якщо існує шлях з } u \text{ в } v \\ \infty, \text{ в іншому випадку} \end{cases},$$

відповідно, найкоротшим шляхом із вершини u до вершини v називають будь-який шлях із вагою $\delta(u, v)$.

Розглядається задача пошуку найкоротших шляхів з однієї вершини (single-source shortest-paths problem). Нехай $s \in V$ – така вершина (джерело). Необхідно знайти найкоротший шлях із джерела до кожної вершини $v \in V$. Алгоритмом пошуку найкоротших шляхів з однієї вершини

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

можна скористатись для вирішення багатьох інших теоретичних і прикладних проблем. Наприклад, розв'язати:

– задачу про найкоротші шляхи в заданий пункт призначення (single-destination shortest-paths problem), а саме, зміною напрямку кожного ребра;

– задачу про найкоротший шлях між заданою парою вершин (single-pair shortest-paths problem);

– задачу про найкоротші шляхи між усіма парами вершин (all-pairs shortest-paths problem), а саме, застосовують алгоритм пошуку найкоротших шляхів з однієї вершини послідовно до кожної вершини. Проте існують алгоритми, які вирішують дану задачу значно швидше.

Для пошуку найкоротших шляхів з однієї вершини скористаємось варіантом алгоритму Беллмана – Форда з [1], який дозволяє розв'язати задачу про найкоротші шляхи з однієї вершини в загальному випадку (вага ребра може бути від'ємною). Алгоритм повертає логічне значення FALSE, якщо граф містить цикл з від'ємною вагою, досяжний із джерела (якщо такий цикл є, то розв'язку не існує, інакше – повертає найкоротші шляхи та їхні ваги). Зауважимо, що найкоротший шлях не містить циклів (за наявності циклу неможливо коректно обчислити довжину (вагу) шляху). Складність алгоритму Беллмана – Форда $O(VE)$.

Прикладні задачі та результати моделювання

У даному розділі сформульовані та розв'язані із застосуванням алгоритмічних примітивів такі задачі: задача про оптимальні структури; задача контролю за фінансовими операціями та задача визначення послідовності виконання робіт, задана системою різницевих обмежень. Результати моделювання перевірено на прикладах. Обчислення проводились у середовищі *Mathcad v. 13*. Розроблено відповідні процедури STRONG_COMP, DFS_CTRL, BELL_FORD_CONST.

Задача про оптимальні структури. Під оптимальною структурою будемо розуміти лінійне впорядкування структурних підрозділів деякого підприємства.

Розглянемо компанію, у структурі якої є окремі підрозділи (об'єкти), зайняті виробництвом, вхідним продуктом для одних є вихідний продукт інших.

Нехай $G = (V, E)$ – орієнтований граф. Вершини – об'єкти. Ребра відображають зв'язки між об'єктами. Оскільки граф орієнтований, то порядок запису вершин ребра (u, v) є суттєвим. У нашому випадку це означає, що об'єкт v використовує продукцію об'єкта u.

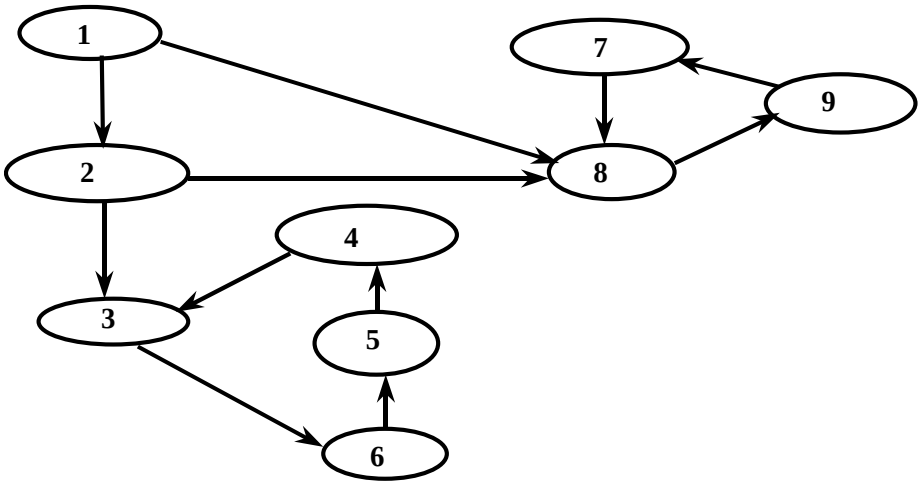

Рис. 1. Структура підприємства

Ефективний менеджмент компанії полягає, серед іншого, у впорядкуванні роботи його об'єктів. Ідеальною є ситуація, коли вдається так організувати цей процес, що підрозділи

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

у виробничому циклі розташовуються в лінійному порядку. Якщо граф ациклічний, дана задача зводиться до задачі топологічного сортування. Але, загалом, такий граф не є ациклічним.

$$\text{STRONG\_COMP(GRAF)} = \begin{bmatrix} 1 & 0 & 0 & -777 \\ 2 & (1) & (\text{"FC"}) & -777 \\ 8 & \begin{pmatrix}1\\2\\7\end{pmatrix} & \begin{pmatrix}\text{"FC"}\\\text{"FC"}\\\text{"T"}\end{pmatrix} & -777 \\ 9 & (8) & (\text{"B"}) & 5 \\ 7 & (9) & (\text{"T"}) & 3 \\ 3 & \begin{pmatrix}2\\4\end{pmatrix} & \begin{pmatrix}\text{"FC"}\\\text{"T"}\end{pmatrix} & -777 \\ 6 & (3) & (\text{"B"}) & 8 \\ 5 & (6) & (\text{"T"}) & 9 \\ 4 & (5) & (\text{"T"}) & 6 \end{bmatrix}$$

Рис. 2. Результат обчислень у Mathcad

Розв’язання поставленої задачі пропонується виконати в декілька етапів:

1. Розкладання на СЗК. В окремі компоненти виділити об’єкти, між якими склалися тісні цикли виробництва – використання продукції.
2. Побудова графа компонентів.
3. Топологічне сортування графа компонентів.

Розглянемо роботу процедури STRONG_COMP на конкретному прикладі.

Нехай об’єкти, роботу яких потрібно оптимізувати, занумеровано. Виробничі стосунки, що склалися між ними (у рамках задачі, яка поставлена), відображає орієнтований граф на рис. 1.

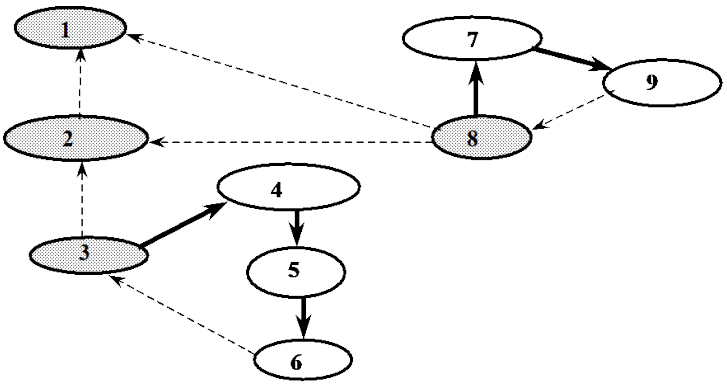

Рис. 3. Сильно зв’язні компоненти

Відповідно до результатів обчислень на рис. 2, виділимо сильно зв’язні компоненти рис. 3, де жирними стрілками зображено дерева пошуку у глибину на графі, транспонованому до вихідного графа. Заштриховані вершини – це корені дерев.

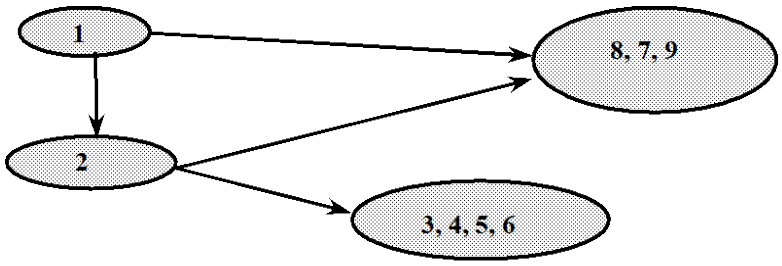

Рис. 4. Граф компонентів

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

За результатами обчислень виділяємо чотири сильно зв'язані компоненти: **1; 2; 3–4–5–6; 8–7–9.** Отже, для ефективного управління можна рекомендувати схему, зображену на рис. 4. Граф компонентів на рис. 4 є орієнтованим ациклічним графом (Теорема 2), вершини якого, за необхідності, можна топологічно відсортувати, тим самим отримати бажане на практиці лінійне впорядкування.

Задача контролю за фінансовими операціями. Результатом Теореми 1 є твердження, якщо пошук у глибину знаходить зворотні ребра, то граф містить ребра-цикли. Розглянемо застосування пошуку у глибину до розв'язання задачі контролю у фінансовій сфері.

Розглядається задача виявлення зв'язків в окремій схемі (задача контролю за фінансовими операціями). Операцією у схемі є факт перерахування коштів із рахунку i на рахунок j, де $1 \le i, j \le N$, $i \ne j$, і N – число рахунків. Операцію зручно зобразити направленим відрізком (стрілка). Тоді схема – орієнтований граф, у якого вершини – рахунки, а ребра – операції.

Значна кількість транзитних рахунків утруднює контроль. У даній роботі запропоновано процедуру DFS_CTRL, яка використовує класифікацію ребер і дозволяє відстежити приховані зв'язки між рахунками. Варто зазначити, що запропонована методика не дає відповіді про кількісний характер зв'язку, а фіксує тільки його факт.

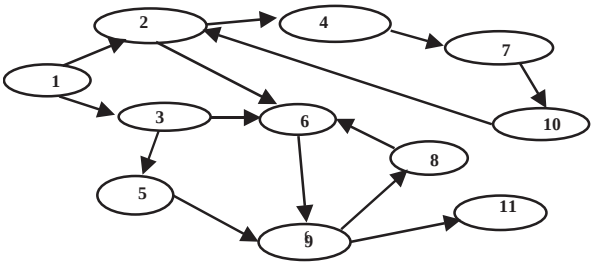

Рис. 5. Граф системи рахунків

Для схеми на рис. 5 результат DFS_CTRL зображено на рис. 6, а виявлені цикли на – рис. 7.

DFS_CTRL(GRAF) =

	1	2	3	4	5	6	7	8	9	10	11
1	0	"T"	"T"	0	0	0	0	0	0	0	0
2	0	0	0	"T"	0	"T"	0	0	0	0	0
3	0	0	0	0	"T"	"FC"	0	0	0	0	0
4	0	0	0	0	0	0	"T"	0	0	0	0
5	0	0	0	0	0	0	0	0	"FC"	0	0
6	0	0	0	0	0	0	0	0	"T"	0	0
7	0	0	0	0	0	0	0	0	0	"T"	0
8	0	0	0	0	0	"B"	0	0	0	0	0
9	0	0	0	0	0	0	0	"T"	0	0	"T"
10	0	"B"	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0

Рис. 6. Результат роботи DFS_CTRL

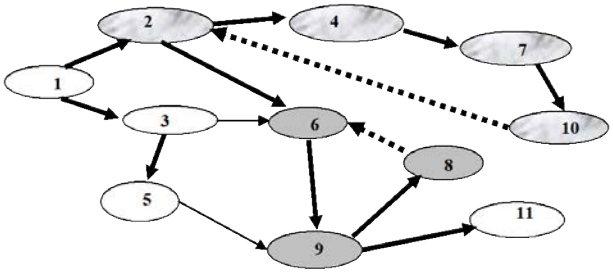

Рис. 7. Ребра-цикли (пунктир)

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

Задача визначення послідовності виконання робіт. Дана задача зводиться до розв'язання системи різницевих обмежень (далі – СРО) [6], є частинним випадком загальної задачі лінійного програмуванyя, для якої:

– система обмежень $Ax \le b$ (де A – матриця $m \times n$, b – m-компонентний вектор, x – n-компонентний вектор невідомих) має специфічний вигляд. А саме, усі обмеження матимуть вигляд $x_j - x_i \le b_k$, де $1 \le i, j \le n$ і $1 \le k \le m$. У такій СРО можлива, наприклад, така інтерпретація: невідомі величини x_i – моменти часу, коли відбуваються події; кожне обмеження – вимога, відповідно до якої між двома подіями мусить пройти деякий час (не менший або не більший, ніж за технологією);

– вигляд цільової функції неважливий, оскільки досить знайти довільний допустимий розв'язок або з'ясувати, що розв'язків немає (задача існування).

Очевидно, поставлену задачу можна розв'язати за допомогою, наприклад, симплекс-алгоритму. Але розглянемо СРО з погляду теорії графів. Беллман у [7] вказує на зв'язок задачі пошуку найкоротших шляхів з однієї вершини із системою різницевих обмежень.

Заданій системі різницевих обмежень відповідає граф обмежень – зважений орієнтований граф $G = (V, E)$, у якого:

$$V = \{v_0, v_1, ..., v_n\},$$

$$E = \{(v_i, v_j) : x_j - x_i \le b_k\} \cup \{(v_0, v_1), (v_0, v_2), ..., (v_0, v_n)\}.$$

Додаткова вершина v_0 уводиться для того, щоб із неї гарантовано була досяжною будь-яка інша вершина.

З теореми 24.9 в [1] випливає, що розв'язання СРО зводиться до обчислення ваги найкоротшого шляху у відповідному графі обмежень. Беллман і Форд запропонували алгоритм для знаходження найкоротшого шляху з однієї вершини, за допомогою застосування якого СРО можна розв'язати (отримати допустимий розв'язок, або дізнатися про його відсутність) за час $O(n^2 + nm)$. Якщо $x = (x_1, ..., x_n)$ – розв'язок, то $(x_1 + d, ..., x_n + d)$ – також розв'язок для довільної константи d.

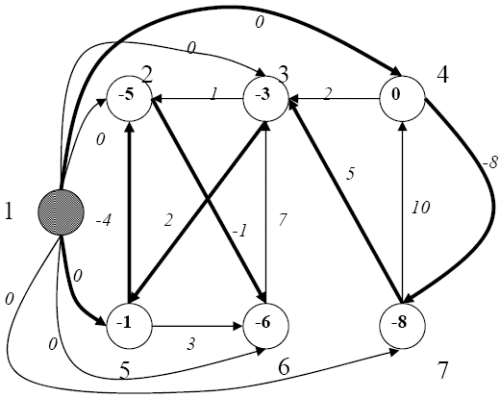

Рис. 8. Граф обмежень

Нехай деякий технологічний процес описується системою різницевих обмежень . Їй відповідає граф обмежень (рис. 8).

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

$$\begin{cases} x_1 - x_2 \le 1; & x_1 - x_4 \le -4; \\ x_2 - x_3 \le 2; & x_2 - x_5 \le 7; \\ x_2 - x_6 \le 5; & x_3 - x_6 \le 10; \\ x_4 - x_2 \le 2; & x_5 - x_1 \le -1; \\ x_5 - x_4 \le 3; & x_6 - x_3 \le -8. \end{cases} \quad \backslash^* \text{ MERGEFORMAT} \qquad (1)$$

Процедура BELL_FORD_CONST повертає розв'язок задачі (рис. 9).

$$\text{BELL\_FORD\_CONST}(GRAF, 1) = \begin{pmatrix} -5 \\ -3 \\ 0 \\ -1 \\ -6 \\ -8 \end{pmatrix}$$

Рис. 9.

Маємо: $x_1 = -5$, $x_2 = -3$, $x_3 = 0$, $x_4 = -1$, $x_5 = -6$, $x_6 = -8$. Звідси випливає, що роботи треба виконати в такій послідовності: $x_6 \Rightarrow x_5 \Rightarrow x_1 \Rightarrow x_2 \Rightarrow x_4 \Rightarrow x_3$.

Висновки

У роботі показана можливість постановки і розв'язання прикладних задач у термінах і з використанням алгоритмів на графах. Зокрема, розглянуто задачу побудови оптимальної структури на підприємстві, задачу контролю за фінансовими операціями та задачу визначення послідовності виконання робіт, що визначена системою різницевих обмежень. Коректність теоретичних міркувань підтверджується на тестових прикладах. Обчислення проведено з використанням спеціально розроблених процедур у середовищі *Mathcad v. 13*.

Список використаної літератури

1. Вступ до алгоритмів / Т.Г. Кормен та ін. Київ : К.І.С., 2023. 1288 с.
2. Hopcroft J.E., Tarjan R.E. Efficient Algorithms for Graph Manipulation. *Communications of the ACM*. 1973. № 16 (6). P. 372–378.
3. Гуляницький Л.Ф., Мулеса О.Ю. Прикладні методи комбінаторної оптимізації. Київ : ВПЦ «Київський університет», 2016. 142 с.
4. Кубайчук О.О., Теренчук С.А. Оптимізація управління методом виділення сильно зв'язаних компонентів на графах. *Техніка будівництва*. 2008. № 21. С. 87–92.
5. Кубайчук О.О., Боличев Б. Аналіз та стабілізація фінансової системи методами теорії графів. *Тенденції та перспективи розвитку науки і освіти в умовах глобалізації :* матеріали XXXIV Всеукраїнської науково-практичної інтернет-конференції, м. Переяслав-Хмельницький. 2018. Вип. 34. С. 381–383. URL: https://drive.google.com/file/d/1GeUO2LQiJEEwEQyN7slmpp1amAYRRe5_/view
6. Кубайчук О.О., Теренчук С.А., Єременко Б.М. Застосування топологічного сортування у плануванні будівельних робіт. *Містобудування та територіальне планування*. 2007. № 28. С. 102–108.
7. Bellman R. On a Routing Problem. *Quarterly of Applied Mathematics.* 1958. № 16 (1). P. 87–90.
8. Кубайчук О.О., Теренчук С.А. Застосування систем різницевих обмежень у плануванні будівельно-монтажних робіт. *Містобудування та територіальне планування.* 2007. № 27. С. 126–129.
9. United States Sentencing Commission: official site. URL: https://www.ussc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2018/Emerging_Tech_Bitcoin_Crypto.pdf (дата звернення: 24.04.2025).

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9

10. Sompolinsky Y., Zohar A. Accelerating bitcoin as transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*. 2013, 881. URL: https://eprint.iacr.org/2013/881.pdf
11. Kovalchuk L., Kaidalov D., Nastenko A., Rodinko M., Shevtsov O., Oliynykov R. Decreasing security threshold against double spend attack in networks with slow synchronization. *Computer Communications.* 2020. 154, 75–81. https://doi.org/10.1016/j.comcom.2020.01.079

References

1. Kormen, T.H., Leizerson, Ch.E., Rivest, R.L., & Stain, K. (2023). *Vstup do alhorytmiv* [Introduction to Algorithms]. Kyiv: K.I.S. [in Ukrainian].
2. Hopcroft, J.E., & Tarjan, R.E. (1973). Efficient Algorithms for Graph Manipulation. *Communications of the ACM*, 16 (6), 372–378. [in English].
3. Hulianytskyi, L.F., & Mulesa, O.Iu. (2016). *Prykladni metody kombinatornoi optymizatsii* [Applied combinatorial optimization methods]. Kyiv: VPTs "Kyivskyi universytet" [in Ukrainian].
4. Kubaichuk, O.O., & Terenchuk, S.A. (2008). Optymizatsiia upravlinnia metodom vydilennia sylno zviazanykh komponentiv na hrafakh [Optimization of control by the method of isolating strongly connected components on graphs]. *Tekhnika budivnytstva*, 21, 87–92 [in Ukrainian].
5. Kubaichuk, O., & Bolychev, B. (2018). Analiz ta stabilizatsiia finansovoi systemy metodamy teorii hrafiv [Analysis and stabilization of the financial system using graph theory methods]. *Tendentsii ta perspektyvy rozvytku nauky i osvity v umovakh hlobalizatsii*, materially XXXIV Vseukrainskoi naukovo-praktychnoi internet-konferentsii [Trends and prospects for the development of science and education in the context of globalization, Proceedings of XXXIV all-Ukrainian Scientific and Practical Conference]. Retrieved from: https://drive.google.com/file/d/1GeUO2LQiJEEwEQyN7slmpp1amAYRRe5_/view [in Ukrainian].
6. Kubaichuk, O.O., Terenchuk S.A., & Yeremenko, B.M. (2007). Zastosuvannia topolohichnoho sortuvania u planuvanni budivelnykh robit [Application of topological sorting in construction planning]. *Mistobuduvannia ta terytorialne planuvannia*, 28, 102–108 [in Ukrainian].
7. Bellman, R. (1958). On a Routing Problem. *Quarterly of Applied Mathematics,* 16 (1), 87–90 [in English].
8. Kubaichuk, O.O., & Terenchuk, S.A. (2007). Zastosuvannia system riznytsevykh obmezhen u planuvanni budivelno-montazhnykh robit [Application of differential constraint systems in planning construction and installation works]. *Mistobuduvannia ta terytorialne planuvannia*, **27**, 126–129. [in Ukrainian].
9. United States Sentencing Commission. (2018). Official site. Retrieved from: https://www.ussc.gov/sites/default/files/pdf/training/annual-national-training-seminar/2018/Emerging_Tech_Bitcoin_Crypto.pdf [in English].
10. Sompolinsky, Y., & Zohar, A. (2013). Accelerating bitcoin as transaction processing. fast money grows on trees, not chains. In: *IACR Cryptology* ePrint Archive, 2013/881. Retrieved from: https://eprint.iacr.org/2013/881.pdf. [in English].
11. Kovalchuk, L., Kaidalov, D., Nastenko, A., Rodinko, M., Shevtsov, O., & Oliynykov, R. (2020). Decreasing security threshold against double spend attack in networks with slow synchronization. *C omputer Communications*, 154, 75–81. https://doi.org/10.1016/j.comcom.2020.01.079 [in English].

Кубайчук Оксана Олексіївна – к.ф.-м.н., доцент, доцент кафедри математичного аналізу та теорії ймовірностей Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського». E-mail: o.kubaychuk@gmail.com, ORCID: 0000-0002-5135-3688.

Kubaychuk Oksana Oleksiivna – Candidate of Physical and Mathematical Sciences, Associate Professor, Associate Professor at the Department of Mathematical Analysis and Probability Theory of the National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute". E-mail: o.kubaychuk@gmail.com, ORCID: 0000-0002-5135-3688.

https://doi.org/10.32782/mathematical-modelling/2025-8-1-9