https://doi.org/10.31891/2219-9365-2025-84-40
УДК 621.391:519.85:519.17

ТІХОНОВ Віктор
Державний університет інтелектуальних технологій і зв'язку
https://orcid.org/0000-0003-4415-5234
e-mail: victor.tykhonov@suitt.edu.ua
ЯВОРСЬКА ОЛЬГА
Державний університет інтелектуальних технологій і зв'язку
https://orcid.org/0000-0002-1790-7472
e-mail: yavorskayao7@gmail.com

ЗАДАЧА ОПТИМІЗАЦІЇ РОЗПОДІЛУ ПОТОКІВ НА ВІЛЬНО-ОРІЄНТОВАНОМУ ГРАФІ

Об'єктом дослідження є процеси передавання цифрових потоків у програмно-конфігурованих телекомунікаційних транспортних мережах (SDN), що забезпечують гнучке управління трафіком та оптимізацію мережевих ресурсів. Предметом дослідження є методи та алгоритмічні підходи до оптимального розподілу цифрових потоків на вільно-орієнтованих графах із використанням принципів максимального потоку і мінімального транзиту. У роботі запропоновано модифікований ітеративний алгоритм (MIA) для розв'язання задачі оптимізації цифрових потоків у програмно-конфігурованих мережах (SDN). Алгоритм базується на принципах мінімального транзиту та максимального потоку на вільно-орієнтованих графах. Подано опис програмної реалізації мовою Python, яка дозволяє автоматизувати процес обчислення пропускної здатності та розподілу трафіку. Алгоритм забезпечує одночасне обчислення максимально можливого потоку між полюсами S та T та оптимальний розподіл цього потоку між альтернативними ST-шляхами з мінімальною кількістю проміжних вершин, реалізуючи принцип мінімального транзиту у поєднанні з принципом максимального потоку. У вихідних даних використовується нормалізований ST-граф, у якому вершини V_n нумеруються натуральними числами від 1 до N, а ребра P_k – від 1 до K; вершина V_1 ітерпретується як джерело S, а вершина з найбільшим номером – як стік T. Структура графа задається текстовим файлом graph.txt, де в першому рядку перелічені всі вершини, а в другому – непорожні ребра з їхніми вагами у форматі трійок (i, j, w). На основі цих даних алгоритм будує внутрішню модель мережі та послідовно виконує три основні блоки: 1) зчитування та валідація вхідного графа, 2) ітеративний пошук ST-шляхів фіксованої довжини з рекурсивним оновленням залишкових пропускних здатностей, 3) обчислення глобального максимуму потоку f_{max} і формування матриці залишків, яка наочно відображає використання ресурсів мережі після роботи MIA.

Ключові слова: алгоритм, цифрові потоки, оптимізація, граф, пропускна здатність, прграмно-конфігуровані мережі, телекомунікаційні мережі.

TIKHONOV Victor, YAVORSKA Olha
State University of Intelligent Technologies and Telecommunications

OPTIMIZATION PROBLEM OF FLOW DISTRIBUTION ON FREE-ORIENTED GRAPHS

The object of research is the processes of digital stream transmission in software-defined telecommunications transport networks (SDN), which provide flexible traffic management and network resource optimization. The subject of the study is methods and algorithmic approaches to the optimal distribution of digital flows on free-oriented graphs using the principles of maximum flow and minimum transit. The paper proposes a modified iterative algorithm (MIA) for solving the problem of optimizing digital flows in software-defined networks (SDN). The algorithm is based on the principles of minimum transit and maximum flow on free-oriented graphs. A description of the software implementation in Python is provided, which allows for automating the process of calculating bandwidth and traffic distribution. The algorithm provides simultaneous calculation of the maximum possible flow between poles S and T and optimal distribution of this flow between alternative ST paths with a minimum number of intermediate vertices, implementing the principle of minimum transit in combination with the principle of maximum flow. The source data uses a normalized ST graph in which the vertices V_n are numbered with natural numbers from 1 to N, and the edges P_k are numbered from 1 to K; vertex V_1 is interpreted as source S, and the vertex with the largest number is interpreted as sink T. The graph structure is specified by the text file graph.txt, where the first line lists all vertices, and the second line lists non-empty edges with their weights in triples (i, j, w). Based on this data, the algorithm builds an internal network model and sequentially performs three main blocks: 1) reading and validating the input graph, 2) iterative search for ST paths of fixed length with recursive updating of residual capacities, 3) calculation of the global maximum flow f_{max} and formation of a residual matrix that clearly reflects the use of network resources after the MIA has run.

Keywords: algorithm, digital flows, optimization, graph, bandwidth capacity, software-defined networks, telecommunications networks.

Стаття надійшла до редакції / Received 11.11.2025
Прийнята до друку / Accepted 06.12.2025

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

Сучасні програмно-конфігуровані мережі характеризуються динамічною реконфігурацією оптичних та бездротових каналів, що робить традиційні алгоритми максимального потоку неефективними. Їхня висока

обчислювальна складність не дозволяє досягти вимог реального часу в динамічних мережах [1].

Традиційна модель біполярних орієнтованих графів не відповідає вимогам для вільно-орієнтованих мереж SDN, де канали можуть реконфігуруватися в обидва напрямки. Відсутність одночасної оптимізації максимального потоку, розподілу по альтернативних шляхах та вирішення проблеми перетину ребер у класичних алгоритмах призводить до неефективного використання пропускної здатності мережі [2].

Ключовою проблемою є відсутність інтеграції алгоритмів з SDN-контролерами та програмованими протоколами, що унеможливлює розгортання на комерційних комутаторах. Складні моделі множинних потоків непридатні для динамічних 5G/6G мереж з вимогами до затримки менше 1 мс.

Задача оптимізації полягає в розробці SDN-сумісного алгоритму, який одночасно обчислює максимальний потік між полюсами S-T та оптимізує його розподіл по шляхах з мінімальним транзитом, забезпечуючи масштабованість та сумісність з перспективними мережами.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Класичні алгоритми максимального потоку, такі як Форда-Фалкерсона [3] з пошуком доповнюючого ST-шляху в залишковому графі та його покращення — Дініца з поліноміальною складністю $O(V^2E)$ [4] та Едмондса-Карпа з $O(VE^2)$ [5], заклали основу для оптимізації потоків на не орієнтованих графах, де кожне ребро замінюється двома протилежними дугами. Ці методи ефективні для статичних мереж телекомунікацій, але в програмно-конфігурованих мережах SDN їх псевдополіноміальна складність обмежує швидке реагування на динамічну реконфігурацію каналів.

Еволюція алгоритмів призвела до Push-Relabel методу Голдберга-Тар'яна [6], який використовує preflow-структуру, операції push/relabel та висотні мітки для досягнення складності $O(V^3)$ на щільних графах дата-центрів. GPU-реалізації Greco EIM та EDMGPU прискорюють обробку графів з мільйонами вершин, однак залишаються на рівні академічних прототипів для високопродуктивних обчислень без інтеграції з SDN-контролерами типу Ryu чи ONOS та комерційними комутаторами. Ці алгоритми потужні для HPC-задач (High-Performance Computing), але потребують значної адаптації для промислових SDN-розгортань [7].

Multi-Commodity Flow (MCF) узагальнює задачу максимального потоку для множинних типів трафіку (відео, IoT, голос) з різними джерелами/стоками та QoS-вимогами, маючи NP-складність з апроксимаціями $O(\log^k V/\epsilon)$. Застосовується в телекомунікаціях для Virtual Network Embedding, логістиці багатотоварних перевезень та дата-центрах [8], але інформації мало через теоретичний характер та високу обчислювальну складність. Теоретично придатний для динамічного сегментування ресурсів (SDN network slicing) , однак непрактичний для промислового впровадження (production) через повільність (100-1000мс затримка), відсутність інтеграції з Ryu/ONOS/P4Runtime та проблеми масштабу в динамічних мережах. Kamax (Capacity Max) від Motorola — окреме спеціалізоване рішення для оптимізації радіоресурсів MOTOTRBO (3000+ користувачів), не пов'язане з MCF та без SDN-програмування [9].

Аналіз існуючих алгоритмів показує, що питання вирішення задачі максимального потоку залишається актуальним для сучасних мереж, побудованих на базі SDN-архітектур та перспективних 6G-технологій, де потрібна комбінація максимальної пропускної здатності з мінімальним транзитом та повна інтеграція з контролерами Ryu/ONOS, P4Runtime і комерційними комутаторами.

ФОРМУЛЮВАННЯ ЦІЛЕЙ СТАТТІ

Метою роботи є розробка модифікованого ітеративного алгоритму (MIA) для оптимізації розподілу потоків на вільно-орієнтованих графах у SDN-мережах з принципом мінімального транзиту.

Завдання досліджень:

1. Сформувати методику ідентифікації елементів мережі на основі нормалізованого графа для введення даних.
2. Реалізувати алгоритм ітераційного пошуку шляхів за критерієм мінімальної кількості транзитних вузлів (хопів).
3. Розробити функцію динамічного перерахунку залишкового ресурсу ребер у процесі покрокового розподілу потоку.
4. Створити програмний модуль візуалізації результатів у формі матриці завантаження та залишків пропускної здатності.

ВИКЛАД ОСНОВНОГО МАТЕРІАЛУ

Пропонований модифікованоий ітеративний алгоритм, реалізований мовою Python, призначен для розв'язання задачі оптимізації потоків у програмно-конфігурованих телекомунікаційних мережах. Модель мережі представлена у вигляді нормалізованого вільно-орієнтованого графа, де вершина V_1 завжди асоційована з витоком «S», а вершина з максимальним індексом V_N — зі стоком «T». Алгоритм базується на принципі мінімального транзиту, що забезпечує вибір ST-шляхів із мінімально можливою кількістю проміжних вузлів, та принципі максимального потоку, згідно з яким на кожному шляху призначається потік, що дорівнює його «вузькій ланці».

Алгоритм складається з трьох функціональних блоків (рис. 1).

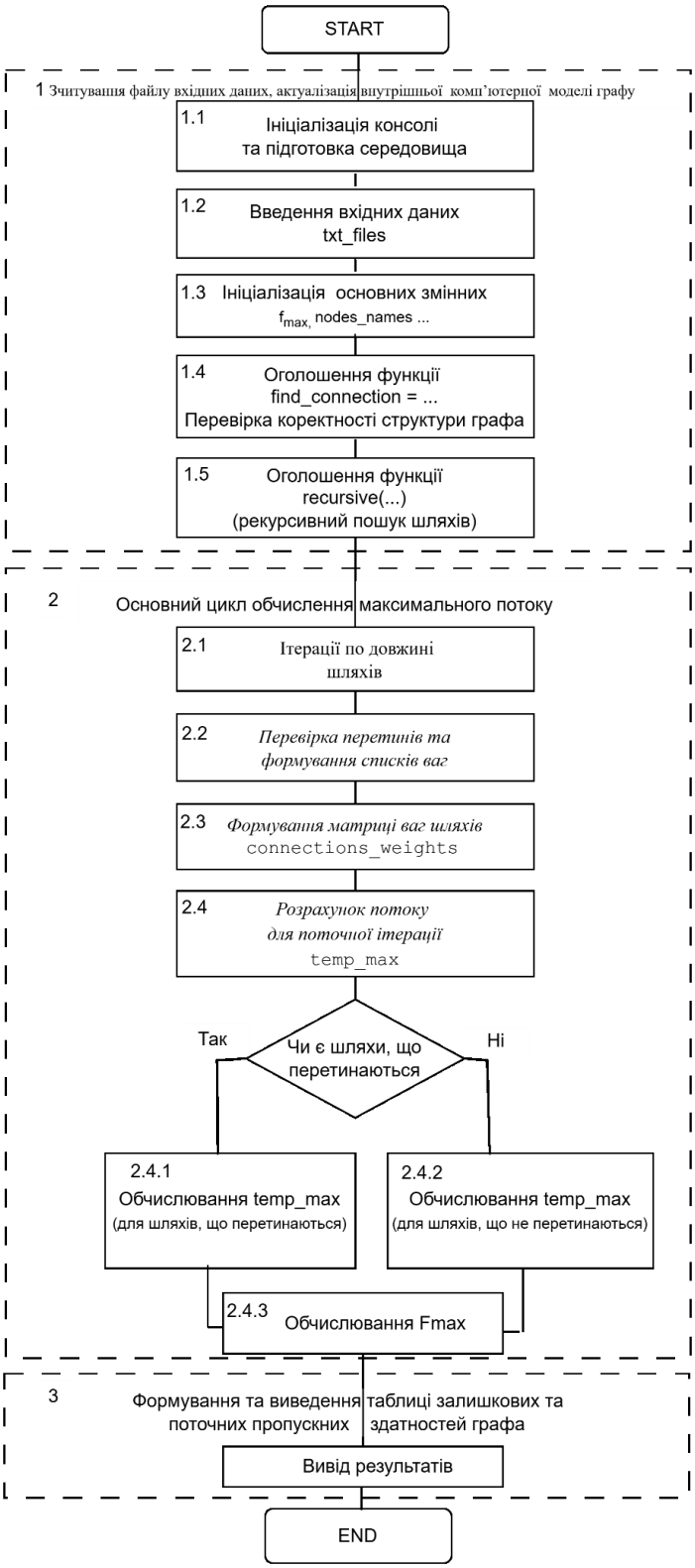

Рис. 1. Modified Incremental Algorithm

Процес виконання програмного алгоритму починається з технічної підготовки середовища та ініціалізації консолі для коректного відображення результатів. Для цього використовуються системні виклики через бібліотеку *ctypes*, що активують підтримку кольорового ANSI-виводу в Windows. На етапі введення параметрів мережі програма сканує каталог на наявність вхідних даних у форматі *.txt*. Варто зауважити, що

використання статичного текстового файлу (наприклад, *graph.txt*) обрано виключно з метою апробації алгоритму та зручності проведення тестових розрахунків.

Після вибору файлу відбувається ініціалізація основних змінних: програма зчитує та розділяє дані на імена вершин, перелік ребер та їхні початкові ваги. Створюються базові структури, такі як $f_{max} = 0$ для накопичення загального потоку, список вузлів V (*nodes_names*) для зберігання ідентифікаторів вузлів та робоча копія матриці залишкових пропускних здатностей C_{res} (*main_connections*). Далі визначається критично важлива допоміжна функція пошуку ребра *find_connection*, реалізована через лямбда-вираз: *next((idx for idx, i in enumerate(main_connections) if set(i.split()[:2]) == set([a, b])), False).* Це дозволяє миттєво знаходити індекс зв'язку між будь-якими двома вузлами *a* та *b* у вільно-орієнтованому графі.

Ключовим функціональним елементом є оголошення рекурсивної функції recursive, яка методом пошуку в глибину знаходить усі можливі шляхи p_k заданої довжини *h* (кількість хопів). Вона працює у зв'язці з новою функцією фільтрації сусідів *get_neighbors(node)*, яка аналізує поточний стан залишкової мережі і повертає лише ті вузли, зв'язок з якими має залишкову пропускну здатність більше нуля. Для кожного знайденого шляху $p_k = (e_1, e_2, .. e_h)$ обчислюється його пропускна здатність за правилом «вузької ланки» (1):

$$\delta(p_k) = \min\{c(e)\} \tag{1}$$

де *c(e)* — поточна вага ребра в залишковій мережі *main_connections*.

Основний цикл обчислення максимального потоку реалізує ітераційний перебір за довжиною шляхів h = 1, 2, … , N-1. На кожному етапі програма будує множину шляхів Ph (всі можливі маршрути між джерелом і стоком), після чого виконує перевірку на наявність перетинів (колізій). Для цього формується список усіх використовуваних ребер, і за допомогою логіки множин set() визначається, чи мають шляхи спільні канали зв'язку. Якщо знайдені на поточній ітерації шляхи є реберно-диз'юнктивними (тобто такими, що не мають жодного спільного ребра і не перетинаються), то вони вважаються незалежними. У такому випадку сумарний додатковий потік (2) на цій ітерації обчислюється як проста арифметична сума потоків кожного шляху:

$$\Delta F_h = \sum_{p_k \in P_h} \delta(p_k) \tag{2}$$

У разі виявлення перетинів ($p_i \cap p_j \neq 0$), алгоритм застосовує процедуру агрегації, де з множини потенційних потоків спочатку знаходяться мінімальні ваги для кожного шляху, а потім програма послідовно обирає найбільші з них». Загальний максимальний потік (3) у мережі накопичується як:

$$F_{max} = \sum_{h=1}^{N-1} \Delta F_h \tag{3}$$

Після кожної ітерації ваги ребер (4) у робочому масиві оновлюються за правилом

$$c_{res}(e) = c_{old}(e) - \delta(p_k) \tag{4}$$

Заключний етап полягає у формуванні матриці залишків. Програма обчислює фактично використаний потік (5) для кожного ребра f(e) як різницю між початковою ємністю C_{init} та залишком C_{res}:

$$f(e) = c_{init}(e) - c_{res}(e) \tag{5}$$

Результат виводиться у вигляді кольорової матриці, де синім кольором маркуються вузли, а світло-синім — числові значення потоків. Верхня трикутна частина матриці відображає вільний ресурс $c_{res}(e)$, а нижня — фактичне завантаження f(e), що дозволяє наочно оцінити розподіл трафіку в системі.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У роботі було розроблено та програмно реалізовано modified *iterative* algorithm для оптимізації потоків у мережах, представлених у вигляді вільно-орієнтованих графів. Алгоритм, реалізований мовою Python, забезпечує автоматичне обчислення максимальної пропускної здатності мережі та визначення найкоротших шляхів для розподілу трафіку.

У межах виконаної роботи було реалізовано такі ключові етапи:

1. *Оптимізований ітераційний пошук:* впроваджено рекурсивний перебір шляхів з функцією get_neighbors(), яка обмежує пошук лише активними сусідами з позитивною пропускною здатністю. Підхід зростання hop (1-hop → N-1-hop) гарантує пріоритет найкоротшим ST-шляхам.
2. *Динамічне оновлення залишкової мережі*: після глобального вибору шляхів оновлюються робоча копія матриці залишкових пропускних здатностей тільки для вибраних шляхів з "вузькою ланкою". Запобігає перевантаженню каналів у реальному часі.

3. *Вирішення конфліктів перетину ребер*: програмне виявлення спільних ребер між шляхами. Для кожного шляху пропускна здатність визначається найменшою здатністю його ребер. Серед шляхів, що перетинаються обирається той, у якого це обмеження є максимальним; для шляхів, що не пиританаються потоки підсумовуються.

4. *Візуалізація та верифікація*: створено систему виводу результатів у формі матриці залишкових ємностей. Використання дворівневої структури відображення дозволило розділити дані про фактичне завантаження каналів та їх вільний ресурс.

Література

1. Globa L., Skulysh M., Parhomenko D., Yakubovska K. The Approach to Flow Management in Virtual Computational Environment for Up-to-Day Telecom Networks. In: Klymash M., Beshley M., Luntovskyy A. (eds). Future Intent-Based Networking. *Lecture Notes in Electrical Engineering*, 2022, vol. 831, pp. 182–196. Springer, Cham. https://doi.org/10.1007/978-3-030-92435-5_10.

2. Lemeshko, O., Yeremenko, O., Yevdokymenko, M., Sleiman, B. (2023). Research and Development of Bilinear QoS Routing Model over Disjoint Paths with Bandwidth Guarantees in SDN. In: Hu, Z., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. *Lecture Notes on Data Engineering and Communications Technologies*, vol 181. Springer, Cham. https://doi.org/10.1007/978-3-031-36118-0_20.

3. Ford L. R., Fulkerson D. R. Maximal flow through a network // Canadian Journal of Mathematics. 1956. Vol. 8. P. 399–404. URL: https://www.semanticscholar.org/paper/Maximal-Flow-Through-a-Network-Ford-Fulkerson/794b01b2513f3610cb151ecfd07e9dc0eae7e1f3 (дата звернення: 10.11.2025).

4. Дініц Е. Алгоритм розв'язання задачі максимального потоку в мережі з оцінкою потужності // Радянські математичні доповіді. 1970. URL: https://www.researchgate.net/publication/228057696_Algorithm_for_Solution_of_a_Problem_of_Maximum_Flow_in_Networks_with_Power_Estimation (дата звернення: 10.11.2025).

5. Едмондс Дж., Карп Р. Теоретичні вдосконалення алгоритмічної ефективності для задач мережевого потоку // Журнал Асоціації обчислювальної техніки. 1972. № 19, № 2. С. 248–264. URL: https://web.eecs.umich.edu/~pettie/matching/Edmonds-Karp-network-flow.pdf (дата звернення: 10.11.2025).

6. Голдберг А., Тар'ян Р. Новий підхід до задачі максимального потоку // Журнал асоціації обчислювальної техніки. 1988. Т. 35, № 4. С. 921–940.

7. Кормен, Т., Лейзерсон, Ч., Рівест, Р., Стайн, К. Вступ до алгоритмів. — Київ: К.І.С., 2019. — 1320 с.

8. Even, S., Itai, A., Shamir, A. On the complexity of timetable scheduling and multi-commodity flow problems // SIAM Journal on Computing. — 1976. — Vol. 5, № 4. — P. 691-703. DOI: 10.1137/0205048. URL: https://epubs.siam.org/doi/10.1137/0205048 [дата звернення: 10.11.2025].ppl-ai-file-upload.s3.amazonaws.

9. Motorola Solutions. MOTOTRBO Capacity Max System Planner: Technical Documentation. — 2020. URL: https://www.motorolasolutions.com/content/dam/msi/docs/ru-ru/MOTOTRBO_Capacity_Max_DataSheet_RU.pdf (дата звернення: 10.11.2025).

References

1. Globa L., Skulysh M., Parhomenko D., Yakubovska K. The Approach to Flow Management in Virtual Computational Environment for Up-to-Day Telecom Networks. In: Klymash M., Beshley M., Luntovskyy A. (eds). Future Intent-Based Networking. Lecture Notes in Electrical Engineering, 2022, vol. 831, pp. 182–196. Springer, Cham. https://doi.org/10.1007/978-3-030-92435-5_10.

2. Lemeshko, O., Yeremenko, O., Yevdokymenko, M., Sleiman, B. Research and Development of Bilinear QoS Routing Model over Disjoint Paths with Bandwidth Guarantees in SDN. In: Hu, Z., Dychka, I., He, M. (eds) Advances in Computer Science for Engineering and Education VI. ICCSEEA 2023. Lecture Notes on Data Engineering and Communications Technologies, vol 181. Springer, Cham. https://doi.org/10.1007/978-3-031-36118-0_20ppl-ai-file-upload.s3.amazonaws.

3. Ford L. R., Fulkerson D. R. Maximal flow through a network // Canadian Journal of Mathematics. — 1956. — Vol. 8. — P. 399–404. URL: https://www.semanticscholar.org/paper/Maximal-Flow-Through-a-Network-Ford-Fulkerson/794b01b2513f3610cb151ecfd07e9dc0eae7e1f3 (дата звернення: 10.11.2025).

4. Dinic E. A. Algorithm for solution of a problem of maximum flow in networks with power estimation // Soviet Mathematics Doklady. — 1970. URL: https://www.researchgate.net/publication/228057696_Algorithm_for_Solution_of_a_Problem_of_Maximum_Flow_in_Networks_with_Power_Estimation (дата звернення: 10.11.2025).

5. Edmonds J., Karp R. Theoretical improvements in algorithmic efficiency for network flow problems // Journal of the ACM. — 1972. — Vol. 19, № 2. — P. 248–264. URL: https://web.eecs.umich.edu/~pettie/matching/Edmonds-Karp-network-flow.pdf (дата звернення: 10.11.2025).

6. Goldberg A., Tarjan R. A new approach to the maximum-flow problem // Journal of the ACM. — 1988. — Vol. 35, № 4. — P. 921–940.

7. Kormen T., Leizerson Ch., Rivest R., Stein K. Vstup do alhorytmiv. — Kyiv: K.I.S., 2019. — 1320 s.

8. Even, S., Itai, A., Shamir, A. On the complexity of timetable scheduling and multi-commodity flow problems // SIAM Journal on Computing. — 1976. — Vol. 5, № 4. — P. 691-703. DOI: 10.1137/0205048. URL: https://epubs.siam.org/doi/10.1137/0205048 (дата звернення: 10.11.2025).

9. Motorola Solutions. MOTOTRBO Capacity Max System Planner: Technical Documentation. — 2020. URL: https://www.motorolasolutions.com/content/dam/msi/docs/ru-ru/MOTOTRBO_Capacity_Max_DataSheet_RU.pdf (дата звернення: 10.11.2025).