МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

магістр

на тему Оптимізація укладань графів з використанням проекцій з багатовимірних просторів

Виконала: студентка 6 курсу, групи МФ-61
спеціальність 122 «Комп’ютерні науки»
освітньо-наукова програма «Інформатика»
Мисик О.Ф.
Керівник: к.ф.-м.н. Полякова Л.Ю.

Харків – 2026

АНОТАЦІЯ

У даній магістерській роботі досліджується задача візуалізації графів із використанням багатовимірного укладання та подальшої проєкції у двовимірний простір. Основну увагу приділено підходу $Nd \to 2d$, за якого граф спочатку оптимізується у просторі підвищеної розмірності, а потім отримане розташування вершин проєктується на площину. Така схема розглядається як спосіб покращення структурної організації графового укладання, зокрема для регулярних, симетричних і багатовимірно організованих графів. У роботі розглянуто основні класи методів укладання графів: силові алгоритми, методи мінімізації стресу, геометричні та детерміновані укладання, багатовимірні, багаторівневі, спектральні та нейромережеві підходи. Окремо сформульовано математичну модель багатовимірного укладання, у якій якість розташування вершин описується через узгодження геометричних відстаней між точками з топологічними відстанями у графі. Як основний інструмент оптимізації використано алгоритм Kamada–Kawai у просторі довільної розмірності, а для переходу до площинного зображення застосовано метод головних компонент PCA. Запропонований алгоритм реалізовано мовою Python у середовищі Google Colab з використанням бібліотек NetworkX, NumPy, pandas, matplotlib та openpyxl. Програмна реалізація забезпечує побудову тестових графів, виконання багатовимірного укладання, PCA-проєкцію у двовимірний простір, вимірювання часу роботи окремих етапів, формування Excel-таблиць і побудову рисунків. Код і результати експериментів розміщено у репозиторії дослідження, що забезпечує відтворюваність отриманих результатів. Експериментальне дослідження проведено на кількох класах регулярних і симетричних графів. Основним тестовим класом стали гіперкуби Q4, Q6, Q8, Q10 та Q12, для яких природна розмірність визначається безпосередньо їхньою комбінаторною структурою. Додатково розглянуто повні графи K4, K6, K8, K10, K12, граф Геммінга H(4,2), граф октаедра та тороїдальну сітку 4×4. Для аналізу використовувалися часові характеристики Kamada–Kawai, PCA та сумарного часу, а також якісне порівняння візуальної структури отриманих укладань. Отримані результати показують, що етап PCA практично не впливає на сумарний час побудови укладання, а основне обчислювальне навантаження припадає на багатовимірну оптимізацію Kamada–Kawai. Для гіперкубів попередня оптимізація у просторі більшої розмірності у низці випадків дозволяє краще виявити регулярну структуру графа: шари, симетрії, паралельні групи ребер та характерні кубічні підструктури. Водночас природна розмірність не завжди є найкращим вибором для кінцевої двовимірної проєкції, що показує необхідність експериментального підбору проміжної розмірності. Таким чином, підхід Nd → 2d є перспективним інструментом для візуалізації регулярних, симетричних і багатовимірно організованих графів. Він може бути використаний для аналізу гіперкубів, графів Геммінга, графів Келі, кристалічних і молекулярних структур, а також інших дискретних об'єктів із природною багатовимірною організацією. Подальші дослідження доцільно спрямувати на кількісне оцінювання якості укладань за додатковими метриками, порівняння PCA з іншими методами проєкції та застосування запропонованого підходу до ширших класів графів.

Загальна характеристика роботи: робота містить вступ, 4 розділи, висновки до розділів та список використаних джерел. Кількість сторінок – 30, кількість ілюстрацій – 12, кількість таблиць – 5, кількість використаних джерел – 34. Ключові слова: граф, візуалізація графів, укладання графів, багатовимірне укладання, Kamada–Kawai, PCA, проєкція, stress, гіперкуб, регулярні графи, симетричні графи.

ABSTRACT

This master's thesis investigates the problem of graph visualization using multidimensional graph layout followed by projection into a two-dimensional space. The main focus is placed on the Nd → 2d approach, in which a graph is first optimized in a higher-dimensional space and then the resulting vertex configuration is projected onto a plane. This scheme is considered as a way to improve the structural organization of graph layouts, particularly for regular, symmetric, and multidimensionally organized graphs. The thesis reviews the main classes of graph layout methods: force-directed algorithms, stress minimization methods, geometric and deterministic layouts, multidimensional, multilevel, spectral, and neural-network-based approaches. A mathematical model of multidimensional graph layout is formulated, in which the quality of vertex placement is described through the correspondence between geometric distances among points and topological distances in the graph. The Kamada–Kawai algorithm in an arbitrary-dimensional space is used as the main optimization tool, while Principal Component Analysis (PCA) is applied to obtain the final two-dimensional representation. The proposed algorithm is implemented in Python in the Google Colab environment using the NetworkX, NumPy, pandas, matplotlib, and openpyxl libraries. The software implementation supports the generation of test graphs, multidimensional layout construction, PCA projection into two-dimensional space, measurement of execution time for individual stages, creation of Excel tables, and generation of visualizations. The code and experimental results are stored in the research repository, ensuring the reproducibility of the obtained results. The experimental study was conducted on several classes of regular and symmetric graphs. The main test class consists of hypercubes Q4, Q6, Q8, Q10, and Q12, for which the natural dimensionality is directly determined by their combinatorial structure. In addition, complete graphs K4, K6, K8, K10, and K12, the Hamming graph H(4,2), the octahedral graph, and the toroidal grid 4×4 were considered. The analysis used execution-time characteristics of the Kamada–Kawai stage, the PCA stage, and the total runtime, as well as a qualitative comparison of the visual structure of the obtained layouts. The obtained results show that the PCA stage has almost no effect on the total layout construction time, while the main computational cost is associated with the multidimensional Kamada–Kawai optimization. For hypercubes, preliminary optimization in a higher-dimensional space in a number of cases makes it possible to reveal the regular structure of the graph more clearly, including layers, symmetries, parallel classes of edges, and characteristic cubic substructures. At the same time, the natural dimensionality is not always the best choice for the final two-dimensional projection, which demonstrates the need for experimental selection of the intermediate dimensionality. Thus, the Nd → 2d approach is a promising tool for visualizing regular, symmetric, and multidimensionally organized graphs. It can be applied to the analysis of hypercubes, Hamming graphs, Cayley graphs, crystalline and molecular structures, as well as other discrete objects with a natural multidimensional organization. Further research should focus on the quantitative evaluation of layout quality using additional metrics, comparison of PCA with other projection methods, and application of the proposed approach to broader classes of graphs. General characteristics of the thesis: the thesis consists of an introduction, 4 chapters, chapter conclusions, and a list of references. The thesis contains 30 pages, 12 figures, 5 tables, and 34 references. Keywords: graph, graph visualization, graph layout, multidimensional layout, Kamada–Kawai, PCA, projection, stress, hypercube, regular graphs, symmetric graphs.

ВСТУП

Візуалізація графів — це ключовий інструмент для аналізу складних структурованих систем. Граф дозволяє наочно показати елементи системи та їхні зв'язки. За останні роки сфера візуалізації графів стала необхідною для аналізу соціальних мереж, біологічних даних, телекомунікацій, машинного навчання і пошуку закономірностей у великих обсягах даних. Оскільки кількість даних і складність зв'язків постійно зростають, дослідники стикаються з викликом: як розробити гнучкі та ефективні методи побудови графових лейаутів, які точно відображають топологічну структуру мереж [1; 2; 3].

Основна мета побудови графового лейауту — створити зрозуміле й інформативне зображення, яке легко сприймати. Важливо, щоб розташування вузлів і зв'язків між ними відображало внутрішню логіку системи. Якісний лейаут допомагає побачити групи елементів, симетрії, мости між частинами мережі та навіть приховані закономірності [1; 2]. Наприклад, у соціальних мережах це дозволяє визначити найвпливовіших учасників і тематичні спільноти, у біологічних мережах — простежити метаболічні ланцюги, а в мережах передачі даних — зрозуміти характер трафіку та знайти вразливі місця інфраструктури.

Незважаючи на велике різноманіття запропонованих рішень, задача якісної побудови укладання залишається відкритою через обчислювальну важкість і наявність численних локальних екстремумів в енергетичних функціях. Традиційні алгоритмічні схеми – методи на основі фізичних аналогій (Fruchterman–Reingold, Kamada–Kawai), стрес-мінімізаційні моделі (*Stress Majorization*), спектральні та регулярні геометричні укладання – здатні генерувати естетично коректні зображення для мереж середнього масштабу [4; 5; 6; 7; 8]. Але зі збільшенням кількості вузлів і зв'язків ці алгоритми схильні зупинятися у локальних мінімумах, що призводить до погіршення точності та відтворюваності результату. Додатково заважає квадратична або кубічна залежність обчислювальної складності від кількості вершин [4; 5], що робить їхнє застосування до великомасштабних графів майже неможливим.

Шляхом подолання цих обмежень виступає концепція багатовимірного укладання: на першому кроці вершини розміщуються в просторі підвищеної розмірності $\mathbb{R}^d$, а вже після досягнення оптимуму отриманий лейаут проєктується у дво- чи тривимірний простір, зручний для безпосереднього сприйняття. Принципова перевага такого підходу полягає у тому, що у просторі вищої розмірності вершини мають суттєво більше ступенів свободи, що знижує ймовірність виникнення конфліктів між силовими обмеженнями та сприяє кращому збереженню попарних відстаней відповідно до топологічних зв'язків [9; 10; 11; 12].

Окремого розгляду заслуговує напрям, що поєднує традиційні оптимізаційні схеми з апаратом машинного навчання – передусім із графовими нейронними мережами (GNN). Завдяки здатності GNN враховувати топологічні патерни і нелінійні залежності, вони можуть ефективно формувати векторні представлення вершин (ембединги), які слугують основою для подальшої побудови якіснішого укладання [13; 14; 15; 16; 33].

Предметом роботи є розроблення та аналіз підходу до оптимізації укладання графів через проєкцію з простору вищої розмірності в 2/3D простір, що забезпечує кращу візуальну якість, обчислювальну ефективність та відтворюваність результатів. Метод базується на розмежуванні оптимізації у просторі підвищеної розмірності та відображення в нижчу розмірність за допомогою сучасних методів проєкції – наприклад, класичного MDS [9; 10; 12].

Актуальність теми

На сьогодні обробка великих графів є однією з найпоширеніших практичних задач. Дослідження мережевої поведінки у соціальних платформах, дослідження білкових взаємодій у біології, побудова інформаційних графів і моделювання транспортних потоків – усі ці задачі формалізуються у вигляді об'ємних графових структур [1; 2; 3]. Ефективне укладання графів потребує алгоритмів, що водночас зберігають їх топологічні властивості, скорочують кількість перетинів ребер і наочну геометричну

інтерпретацію отриманої схеми.

Особливо важливим стає при цьому питання масштабованості. Алгоритм Fruchterman–Reingold має асимптотичну складність $\mathcal{O}(n^2)$, Kamada–Kawai – ще вищу [4; 5]; щоб наблизити ці обидва методи до глобального оптимуму, потрібно велике число ітерацій. Підхід, що спирається на багатовимірні вкладення, здатний суттєво скоротити число необхідних ітерацій через розв'язання обмежень у багаторозмірному просторі, де вершини мають більші можливості для переміщення [9; 10].

Мета та завдання дослідження

Метою цієї роботи є розробка і практичне обґрунтування методу оптимізації укладання графів з використанням багатовимірного вкладення та проєкції у 2/3-вимірний простір.

Для цього необхідно:

1. провести огляд існуючих методів укладання графів та виявити властиві їм обмеження;
2. сформулювати математичну модель оптимізації розміщення вершин у просторі $\mathbb{R}^d$;
3. дослідити способи проєкції багатовимірного укладання у 2- або 3-вимірний простір;
4. розробити та реалізувати алгоритм оптимізації на базі бібліотек мови Python (NetworkX, igraph) [17; 18];
5. провести обчислювальні експерименти на тестових графах;
6. порівняти отримані результати з результатами відомих підходів та оцінити якість укладань за показниками *stress*, кількості перетинів ребер, стабільності та швидкості виконання алгоритму.

Об'єкт і предмет дослідження

Об'єктом дослідження є процес укладання графів – задача розміщення вершин і ребер у просторі з метою адекватного відображення структури зв'язків між елементами досліджуваної системи.

Предметом дослідження є методи багатовимірної оптимізації та проєкції, що застосовуються під час формування укладань графів.

Наукова новизна

Наукова новизна роботи обумовлена використанням багатовимірного вкладення як самостійного проміжного етапу в процесі оптимізації укладання, що дозволяє знизити кількість конфліктів між попарними відстанями у ході мінімізації енергії. На відміну від класичних алгоритмів, запропонований підхід робить задачу більш структурованою. Її можна розглядати як послідовність двох пов'язаних підзадач: глобальної оптимізації в просторі підвищеної розмірності та відображення у двовимірний простір з локальною оптимізацією конфігурації.

Практичне значення одержаних результатів

Розроблений метод може бути застосовано у задачах візуального аналізу великих мережевих структур, де традиційні методи не забезпечують прийнятної якості зображення або вимагають надмірних обчислювальних ресурсів. Зокрема, отримані результати можуть бути використані для вивчення соціальних і комунікаційних структур, аналізу мереж молекулярних взаємодій, задач представлення знань та конструювання графових баз даних.

Загальна структура роботи

Ця робота складається з чотирьох основних розділів, вступу та висновків. Перший розділ - це огляд методів укладання графів, характеристика ключових критеріїв якості та особливості сучасних алгоритмічних рішень. Другий - математична формалізації задачі багатовимірного вкладення, проєкції. Третій - алгоритм реалізації. Четвертий розділ показує результати обчислювальних експериментів та інтерпретує отримані показники.

Візуалізація графів є одним із пріоритетних напрямів сучасної інформатики. Розроблення методів, що спираються на простори вищої розмірності та сучасні оптимізаційні алгоритми, є важливим критерієм подальшого розвитку систем аналізу графів та автоматизованого візуального інтерпретування складних даних [1; 2; 9; 10].

РОЗДІЛ 1. ОГЛЯД ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ УКЛАДАННЯ ГРАФІВ

Задача візуалізації графів та критерії якості

Візуалізація графів є невід'ємною складовою вивчення структурованих даних. Порівняно з табличними та текстовими формами подання, графічне зображення мережевих зв'язків робить можливою інтуїтивну оцінку будови системи й виявлення закономірностей, що залишаються прихованими при використанні інших аналітичних засобів. У такому контексті граф представляють як сукупність об'єктів (вершин) і відношень між ними (ребер), розміщених у 2- чи 3-вимірному просторі так, щоби відповідна функція якості укладання набувала екстремального значення [1; 2; 3].

Поняття «якості укладання» є багатозначним за своїм змістом. До основних показників, за якими оцінюють графічне зображення мережі, відносять наступні:

- **мінімізація перетинів ребер** – зменшення числа взаємних накладань ліній полегшує зчитування структури мережі;
- **рівномірність розподілу вершин** – вузли не повинні надмірно концентруватися в обмежених ділянках, бо це ускладнює сприйняття;
- **відтворення симетрії та топологічних властивостей** – наявна структурна симетрія графа має бути присутня у його зображенні;
- **подібність довжин ребер** – зв'язки порівнянної смислової значущості мають бути приблизно однаковими за довжиною;
- **відповідність геометричних і топологічних відстаней** – вершини, що тісно пов'язані за структурою, повинні розташовуватися поблизу одна одної у площині зображення, а які ні - навпаки бути віддаленими [1; 2]

Силові (force-directed) методи

Силові алгоритми зображують граф як фізичну систему, де вершини відіграють роль частинок, між якими діють сили притягання (вздовж ребер) і відштовхування (між усіма парами вершин). Рівноважна конфігурація системи, що відповідає мінімуму потенційної енергії, використовується як підсумкове укладання [2; 4; 5].

Алгоритм Fruchterman–Reingold

Запропонований у 1991 році алгоритм Fruchterman–Reingold є одним із найпоширеніших у галузі укладань графів. Сила відштовхування між будь-якими двома вершинами спадає при збільшенні відстані між ними, а сила притягання вздовж ребра збільшується з ростом відстані між суміжними вершинами. Алгоритм відрізняється простотою реалізації і зазвичай видає естетично коректні укладання для мереж невеликого і середнього масштабу. Але квадратична залежність складності $\mathcal{O}(n^2)$ від кількості вершин значно обмежує можливість його застосування у випадку великих графів [4].

Алгоритм Kamada–Kawai

Підхід Kamada–Kawai представляє задачу у термінах мінімізації функції «пружної енергії»: для кожної пари вершин визначається ідеальна відстань (зазвичай пропорційна до довжини найкоротшого шляху між ними в графі), і розміщення вершин налаштовується так, щоби реальні відстані у площині максимально відповідали ідеальним. Метод забезпечує більш високу точність відтворення взаємних відстаней порівняно з алгоритмом Fruchterman–Reingold, але потребує кубічних витрат часу $\mathcal{O}(n^3)$, що сильно обмежує його масштабованість [5].

Практичні особливості силових методів

Силові методи широко використовуються в системах візуалізації – Gephi, Cytoscape, D3.js [19; 20; 21] – завдяки інтуїтивності та якості результатів для мереж помірного розміру. Також ці алгоритми схильні до застрягання у локальних мінімумах, а результат може варіюватися залежно від початкового розміщення вершин, що обмежує можливість відтворюваності укладання [1; 2].

Методи мінімізації стресу (Stress Majorization)

Методи на основі мінімізації стресу ставлять на меті безпосередню мінімізацію функції [6; 12]:

$$E(X) = \sum_{i<j} w_{ij} \big(\|X_i - X_j\| - d_{ij}\big)^2, \tag{1}$$

де d_{ij} – цільові відстані між парами вершин, w_{ij} – вагові коефіцієнти. Техніка *Stress Majorization* вирішує цю задачу через побудову послідовності задач, кожна з яких має замкнений розв'язок. Перевагою є гарантована монотонна збіжність до (локального) мінімуму [6].

Практичне застосування

Методи *stress majorization* реалізовані в спеціалізованих інструментах наукової візуалізації – Graphviz [22; 6]. Їхні розширення дозволяють використання попередніх конфігурацій як стартових точок або застосування багаторівневої апроксимації для великих мереж.

Геометричні та детерміновані укладання

Окрему категорію складають детерміновані геометричні методи, що не передбачають оптимізації за ітераціями. Такі схеми спираються на математичні правила або аналітичні вирази, що безпосередньо задають координати вершин [8; 2]. Типові приклади:

- регулярні схеми у вигляді граток для графів зі структурою сітки;
- кругові укладання;
- ієрархічні схеми для дерев або діаграм потоків даних.

Ієрархічні алгоритми на зразок методу Sugiyama [23] розбивають вершини на горизонтальні рівні відповідно до напрямку ребер. Такі схеми підходять для дерев чи DAG-структур, де пріоритетом є відображення ієрархії, а не близькості елементів за якимось критерієм. Геометричні методи простіші і швидші, але не гарантують високої візуальної якості для складних графів, де багато зв'язків і потрібна одночасна оптимізація за кількома критеріями.

Багатовимірні укладання

Багатовимірне укладання (*high-dimensional embedding*) – підхід, за якого вершини графа спочатку розміщуються у просторі високої розмірності $\mathbb{R}^d$ ($d \gg 2$), потім отримана конфігурація проєктується у 2D або 3D простір для візуалізації [9; 10]. Ідея тут полягає у тому, що в просторі високої розмірності система має менше жорстких обмежень і тому більше можливостей для знаходження лейауту з низьким значенням функції енергії.

Математичне формулювання

Задача складається головним чином з мінімізації функції [6; 9; 10]:

$$E(X) = \sum_{i<j} w_{ij} \big(\|X_i - X_j\| - d_{ij}\big)^2, \quad X_i \in \mathbb{R}^d. \tag{2}$$

Вибір розмірності d визначається необхідною точністю та часом виконання алгоритму. На практиці зазвичай розглядають простори розмірності від 4 до 50. Після досягнення рівноваги результат проєктується у 2D за допомогою методу PCA, класичного багатовимірного шкалювання (MDS) або нелінійних методів (t-SNE, UMAP) [11; 12; 24; 25].

Переваги та обмеження

+ Знижується ймовірність потрапляння у локальний мінімум [9; 10].
+ Підвищується стабільність оптимізації.
+ Краще відтворюються кластери та далекосяжні зв'язки.
– Збільшення числа параметрів може ускладнювати оптимізацію та підвищувати використання пам'яті. Але із розвитком паралельних обчислень і GPU-прискорення цей підхід стає більш практичним.

Багаторівневі методи

Багаторівневі (*multilevel*) алгоритми представляють з себе сучасну стратегію масштабування укладань для великих графів. Суть цих алгоритмів полягає у поетапному спрощенні (*coarsening*) графа до меншого за обсягом представлення, виконанні лейауту на отриманому варіанті з наступним послідовним «розгортанням» до початкового розміру із поступовим уточненням конфігурації [26; 27].

Ця ідея реалізована у методах FM^3 (*Fast Multipole Multilevel Method*) та Yifan Hu [26; 27; 28], де фізичні аналогії збережені, але взаємодії між віддаленими вершинами апроксимуються через кластери. Це знижує складність приблизно до $\mathcal{O}(n \log n)$, що дозволяє ефективно обробляти графи на сотні тисяч вузлів. Багаторівневі схеми добре поєднуються зі спектральними та stress-алгоритмами, де кожен рівень виконує роль попереднього наближення для глобальної оптимізації.

Методи на основі спектрального вкладення та GNN

Сучасний вектор розвитку графової візуалізації пов'язаний із спектральними методами і графовими нейронними мережами.

Спектральні методи

В основі спектрального укладання лежить розклад лапласіана графа:

$$L = D - A,$$

де A – матриця суміжності, D – діагональна матриця ступенів. Власні вектори матриці L, що відповідають найменшим власним значенням, визначають координати вершин у просторі. Вибір двох або трьох перших власних векторів дає 2D або 3D-укладання відповідно [7].

Спектральне укладання має міцну теоретичну базу, добре зберігає глобальну структуру графу, але може зневажати локальні деталі. Висока обчислювальна ефективність (основна операція – діагоналізація) робить такі методи популярним інструментом ініціалізації для подальшої оптимізації іншими алгоритмами [7; 2].

Вкладання на основі машинного навчання

З активним розвитком архітектур графових нейронних мереж (GNN) з'являються підходи, що дозволяють генерувати укладання на основі навчених моделей, які кодують структурну інформацію графа у вигляді векторів. Алгоритми GraphSAGE, Node2Vec, DeepWalk і GCN [31; 30; 29; 32] здатні перетворювати вершини у вектори фіксованої розмірності таким чином, що топологічно схожі вершини

опиняються поруч у векторному просторі. Отримані вектори можуть бути безпосередньо використані як початкові координати для наступної проєкції, відкриваючи перспективи для органічної інтеграції машинного навчання у традиційні задачі графової візуалізації [13; 14; 15; 16; 33].

Порівняльний аналіз методів

У таблиці 1 наведено зведений порівняльний аналіз розглянутих класів методів укладання графів [1; 2; 3].

Табл. 1: Порівняльна характеристика методів укладання графів

Клас методів	Якість укладання	Масштабованість	Переваги	Недоліки
Силові	Висока (малі графи)	Низька $\mathcal{O}(n^2)$	Інтуїтивність, наочність	Локальні мінімуми, повільність
Stress Majorization	Висока точність	Обмежена	Гарантована збіжність	Обчислювально затратний
Геометричні	Середня	Висока	Простота реалізації	Відсутність адаптивності
Багатовимірні	Висока	Залежить від d	Менше конфліктів	Підвищені ресурси
Багаторівневі	Добра	Висока	Придатні для великих графів	Складна структура алгоритму
Спектральні та GNN	Висока (при коректній проєкції)	Добра	Інтеграція з ML	Можлива втрата локальної топології

Висновки до розділу 1

Укладання графів є оптимізаційною задачею, що охоплює геометричні, аналітичні та обчислювальні аспекти. Класичні силові моделі забезпечують естетичну якість результату, але не масштабуються до великих структур. Методи *stress majorization* та спектральні спираються на строгий математичний ґрунт, проте є надмірно затратними за складністю. Натомість багатовимірні та багаторівневі схеми демонструють переваги в аналізі великих мереж: вони дозволяють уникати локальних екстремумів і підвищувати відтворюваність укладань [1; 9; 10].

Відтак актуальним залишається пошук методів, що поєднують математичну коректність, високу швидкодію та здатність до масштабування. А підхід на основі багатовимірного вкладення з подальшою проєкцією у низьковимірний простір становить перспективний напрям для подальших досліджень.

РОЗДІЛ 2. МАТЕМАТИЧНА МОДЕЛЬ БАГАТОВИМІРНОГО УКЛАДАННЯ

Формалізація задачі

Математичний опис будь-якої задачі починається з визначень. Дамо визначення графу: Граф – це невпорядкована пара $G = (V, E)$, де V – множина вершин, а E – множина ребер, які поєднують ці вершини. Також введемо позначення n – кількість вершин та m – кількість ребер (потужності множин). Ми шукаємо відображення $f\colon V \to \mathbb{R}^2$ (або $\mathbb{R}^3$), яке ставить у відповідність кожній вершині точку на площині. Координати вершини v_i позначимо як $\mathbf{x}_i = (x_{i1}, x_{i2})$. У результаті роботи алгоритму візуалізації ми отримуємо координати вершин, тобто множину $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ – множину координат вершин. Після того, як ми визначили місцезнаходження точок на площині, нам потрібно зрозуміти

наскільки добре ми визначили їх координати. Для цього потрібна числова метрика, яка в теорії графів має назву функція стресу. І нам треба знайти таку множину координат X, щоб мінімізувати цю функцію. Тепер дамо визначення функції стресу. Позначимо через d_{ij} відстань між вершинами v_i та v_j у графі (це не просто число, а топологічна міра, найчастіше, довжина найкоротшого шляху між ними), а через $\|\mathbf{x}_i - \mathbf{x}_j\|$ – евклідову відстань між їх образами у просторі розкладки (реальна відстань між точками на майбутньому лейауті). Функція stress набуває вигляду [6; 12]:

$$\text{Stress}(X) = \sum_{i<j} w_{ij}\big(\|\mathbf{x}_i - \mathbf{x}_j\| - d_{ij}\big)^2, \tag{3}$$

де w_{ij} – вагові коефіцієнти, що регулюють відносну значущість окремих пар вершин. Зазвичай їх роблять зворотньо пропорційними d_{ij}, щоб занадто великі відстані не переважували локальну структуру. Що ж означає "знайти оптимальну укладку" мовою формул? Це задача пошуку наступного компромісу: розміщення точок так, щоб геометричні відстані між ними ($\|\mathbf{x}_i - \mathbf{x}_j\|$) були якомога ближче до заздалегідь відомих d_{ij} [6].

Високовимірне вкладення

Як отримати координати вершин, які добре відображають структуру графу, у просторі великої розмірності ($\mathbb{R}^d$)? Тобто як знайти початкове наближення? Головними проблемами у задачі пошуку мінімуму стресу у просторі 2D є локальні мінімуми та "тіснота". Рахувати попарні відстані d_{ij} для великого графу і мінімізувати стрес напряму у 2D – це непроста задача і це часто має погані результати. Ці проблеми можна обійти, якщо спочатку піти у простір більшої розмірності. Конкретним рішенням є метод HDE [9]. Його суть полягає в тому, що ми обираємо m спеціальних опорних вершин (pivot), де m відповідає високій розмірності. Зазвичай обирають m у діапазоні від 50 до 200. У даному випадку координата вершини є не числом, а вектором з m чисел, кожне число в якому – це відстань від цієї вершини до конкретної pivot-вершини. Тобто, використовуючи ці опорні вершини можна отримати координати потрібної вершини у багатовимірному просторі. Для цього використовується наступна формула [9; 10]:

$$\mathbf{x}_v = \big(\|v - p_1\|,\ \|v - p_2\|,\ \ldots,\ \|v - p_m\|\big), \tag{4}$$

де v – це вершина, а $p_1, p_2, \ldots, p_m$ – це набір опорних вершин [9; 10]. Дві близькі у графі вершини будуть також близькі і в новому багатовимірному просторі, бо дві вершини, які знаходяться поруч будуть мати майже однакові відстані до всіх pivot-вершин. А значить, їх вектори ($\mathbf{x}_v$) будуть майже ідентичні. Таким чином, структуру зв'язків у графі відображено положеннями точок. Цей метод є варіацією методу MDS (Multidimensional Scaling), але, на відміну від нього, не потребує побудування додаткової матриці відстаней [9; 12]. У порівнянні з методом брутфорс HDE має перевагу у складності ($O(m \cdot (n + m))$ проти $O(n^3)$) [9; 10].

Оптимізація в просторі $\mathbb{R}^d$

Отримати початкові координати в $\mathbb{R}^d$ можна методом HDE або іншим. Наступним кроком має бути оптимізація положень вершин у відповідності до деякого критерію якості, наприклад, мінімізація сумарної сили чи енергії у пов'язаній з графом фізичній системі. Фактично ми не залишаємо точки на місці, а дозволяємо їм рухатися, щоб більш точно відповідати цільовим відстаням d_{ij}. Задача полягає в наступному. Є координати точок в $\mathbb{R}^d$. Ми хочемо змінити їх так, щоб значення функції $\text{Stress}(X)$ (див. (3)) мінімізувалося. Математичним методом, що підходить для оптимізації може бути, наприклад, градієнтний спуск [34]. Функція стресу диференційовна, можна порахувати її градієнт. Він покаже напрямок найшвидшого зросту функції. Для мінімізації функції треба рухатись у протилежному напрямку. Дамо визначення градієнту: Градієнт – це вектор часткових похідних. В кожній точці простору він визначає швидкість і напрямок найшвидшої зміни функції. Формула для часткової

похідної функції стресу по координатах однієї вершини $\mathbf{x}_i$ має такий вигляд:

$$\frac{\partial \text{Stress}}{\partial \mathbf{x}_i} = 2 \sum_{j \neq i} w_{ij} \left(1 - \frac{d_{ij}}{\|\mathbf{x}_i - \mathbf{x}_j\|}\right) (\mathbf{x}_i - \mathbf{x}_j). \tag{5}$$

Цю формулу можна фізично інтерпретувати як закон Гука для системи пружин, що з'єднують точки (сума сил). Також можна вивести ітераційну формулу:

$$\mathbf{X}_{\text{new}} = \mathbf{X}_{\text{old}} - \eta \, \nabla \text{Stress}(\mathbf{X}_{\text{old}}), \tag{6}$$

де η – це швидкість навчання, або довжина кроку. Часткова похідна для однієї вершини – це сума сил, які діють на неї від решти усіх вершин. Те, наскільки поточний стан відрізняється від цільового, впливає на ці сили. Для прискорення розрахунків використовують стохастичний градієнтний спуск (SGD) або метод імпульсу. Процес зупиниться, коли зміна функції стресу стане менше порогової, або коли буде досягнута максимальна кількість ітерацій. Цей метод можна коротко описати наступним чином: ми взяли початкове наближення розміщення точок, яке було отримане за допомогою опорних точок та покращили його математично точним методом, аби мінімізувати цільову функцію.

Проєкція у 2/3D

Наступний крок полягає в тому, щоб з багатовимірного простору $\mathbb{R}^d$ отримати укладання графа в ($\mathbb{R}^2$ або $\mathbb{R}^3$), яке, зокрема можна використати для візуалізації. Задача полягає в наступному. У нас є множина точок в $\mathbb{R}^d$. Ми шукаємо відображення $g\colon \mathbb{R}^d \to \mathbb{R}^k$ ($k = 2$ або 3). І під час проєкції ми хочемо зберегти якомога більше інформації про взаємне розміщення точок. ТОбто необхідно розв'язати задачу зниження розмірності, здійснивши відображення з високовимірного простору в низьковимірний. Головна вимога до проєкції – збереження структури взаємних відстаней. Методи проєкцій поділяють на два великих класи: лінійні та нелінійні. До лінійних методів передусім відноситься метод PCA (Principal Component Analysis) [11]. Його мета – знайти такі вісі (головні компоненти в $\mathbb{R}^d$), вздовж яких дисперсія даних максимальна. Перші дві вісі і дадуть проєкцію. Пошук осей здійснюється за допомогою коваріаційної матриці. Власні вектори, що відповідають двом найбільшими власними числами, є шуканими двома головними осями. Цей метод є швидким і це зумовлює вибір його як основного метода зниження розмірності в даній роботі. До нелінійних методів відносяться, наприклад, такі методи, як t-SNE (t-distributed Stochastic Neighbor Embedding) [24] та UMAP. Алгоритм t-SNE краще зберігає локальну структуру графа, тобто сусіди в високовимірному просторі так і залишаться сусідами в низьковимірному. Цей метод перетворює рахує відстані за допомогою ймовірносної міри та оптимізує (шукає мінімум) різницю між ймоірнісним розподілом у багато- та маловимірному просторі. Алгоритм UMAP (Uniform Manifold Approximation and Projection) [25] – сучасний нелінійний метод, який частіше працює швидше за t-SNE і краще зберігає локальну структуру. Ідея цього методу полягає в тому, що він будує граф у багатовимірному просторі ґрунтуючись на теорії перетворень Рімана, після чого знаходить його максимально схоже представлення у маловимірному.

Доцільність використання нелінійних методів варто досліджувати окремо для деяких класів графів, зважаючи в першу чергу на те, що вони потребують більше часового ресурсу для виконання. Отже, в експериментах ми будемо використовувати PCA як найпростіший і швидкий спосіб для зниження розмірності [9; 11].

Висновки до розділу 2

У розділі 2 ми поставили задачу – мінімізувати функцію стресу, обрали як отримати хороше початкове наближення у багатовимірному просторі (через відстань до опорних точок). Також було показано як покращити це наближення з точки зору математики, а саме методом градієнтного спуску. І наприкінці було описано як спроєктувати готові результати на площину, щоб можна було побачити

кінцеве укладання.

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ МЕТОДУ

Алгоритм багатовимірної оптимізації Nd → 2d

Основна ідея запропонованого підходу полягає в тому, щоб не будувати двовимірне укладання графа безпосередньо, а спочатку виконати його оптимізацію в просторі більшої розмірності. Після цього отримане багатовимірне розташування вершин проєктується на площину. Таку схему далі позначатимемо як Nd → 2d [9; 10; 13], тобто як перехід від укладання в просторі $\mathbb{R}^d$ до кінцевого укладання в $\mathbb{R}^2$.

Безпосередня побудова укладання у двовимірному просторі часто обмежує рух вершин і змушує алгоритм рано переходити до локально оптимальної, але візуально не завжди вдалої конфігурації. Якщо граф має регулярну або симетричну структуру, то в площині одночасно виникає багато конфліктів між вимогою зберегти сусідство, обмежити кількість перетинів і не зруйнувати глобальну симетрію. У просторі більшої розмірності вершини мають більше ступенів свободи, тому частину таких конфліктів можна розв'язати ще до переходу до площини.

Особливо природним цей підхід є для графів, які самі мають багатовимірну внутрішню структуру. Наприклад, вершини n-вимірного куба описуються двійковими векторами довжини n, а ребра відповідають зміні рівно однієї координати. Тому гіпотеза про доцільність оптимізації таких графів у просторі, близькому до їх природної розмірності, є змістовною і надалі перевіряється експериментально.

Вхідними даними алгоритму є граф $G = (V, E)$, розмірність проміжного простору d, цільова функція багатовимірного укладання та спосіб задання початкових координат вершин. Граф визначає топологічну структуру задачі, а параметр d задає, у просторі якої розмірності виконуватиметься оптимізація. У цій роботі в експериментах розмірність d змінюється в діапазоні від 2 до 20.

Цільова функція визначає критерій якості укладання. У загальному випадку це може бути функція енергії системи пружин у моделі Kamada–Kawai або силова функція у моделі Fruchterman–Reingold. У програмній реалізації цього дослідження використано саме алгоритм Kamada–Kawai у довільній розмірності [5; 10], що дозволяє безпосередньо порівнювати двовимірний випадок із багатовимірною оптимізацією.

Спосіб задання початкового положення також є важливим, оскільки алгоритми укладання графів зазвичай не гарантують досягнення глобального мінімуму. Початкові координати можуть бути випадковими, заздалегідь заданими або побудованими на основі HDE/pivot-based підходу [9]. Останній варіант особливо змістовний для регулярних графів, оскільки дозволяє врахувати графові відстані до набору опорних вершин уже на старті оптимізації.

Вихідними даними алгоритму є двовимірне укладання графа, тобто множина координат вершин на площині. Формально результат можна записати у вигляді множини

$$Y = \{y_1, \ldots, y_n\}, \qquad y_i \in \mathbb{R}^2.$$

Отримані координати можуть бути використані для побудови рисунка, а також для подальшого кількісного або якісного аналізу.

Оптимізація в просторі $\mathbb{R}^d$ відбувається шляхом мінімізації цільової функції. У випадку Kamada–Kawai вона має зміст узгодження геометричних відстаней між вершинами з їх графовими відстанями. Узагальнено відповідний функціонал можна записати як [5; 6]:

$$E(X) = \sum_{i<j} w_{ij} \big(\|x_i - x_j\| - d_{ij}\big)^2,$$

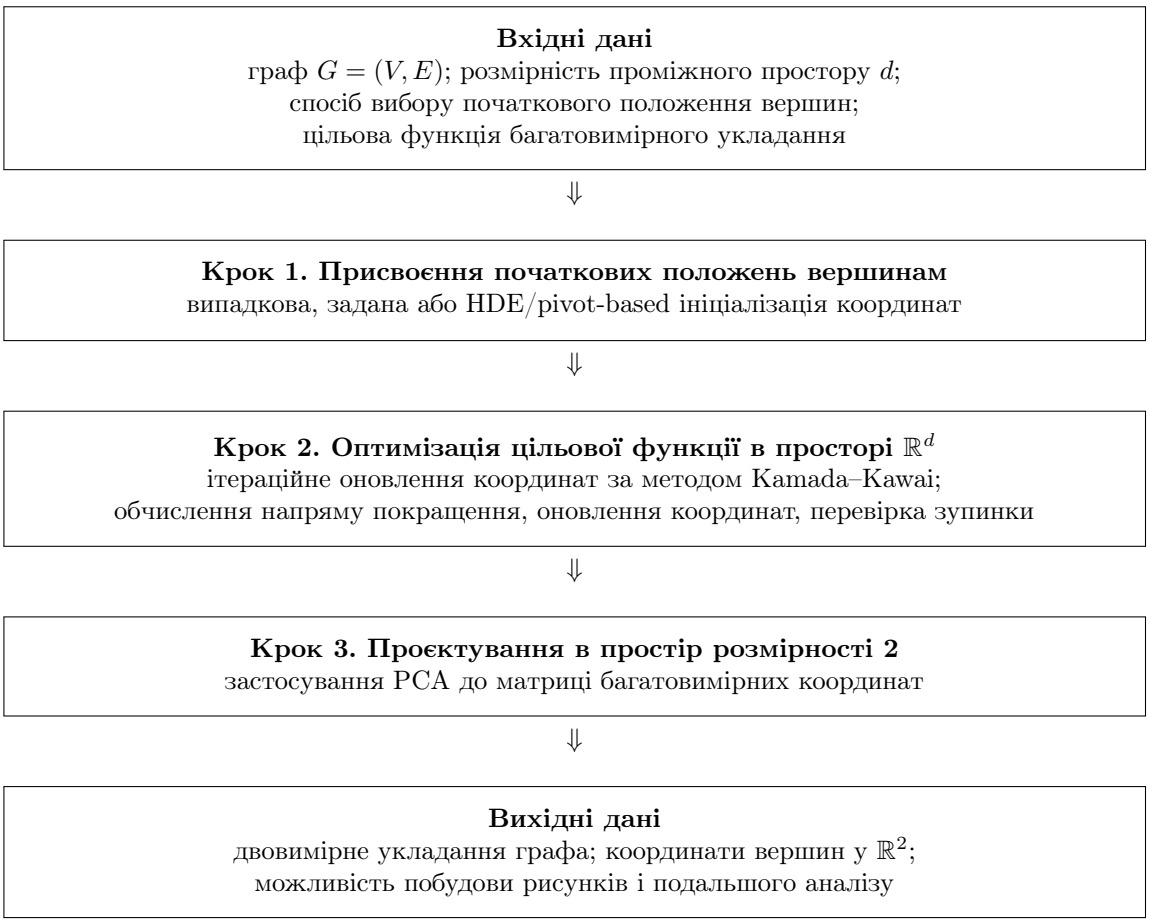

Рис. 1: Підсумкова схема алгоритму Nd $\rightarrow$ 2d

де $x_i, x_j \in \mathbb{R}^d$, d_{ij} – бажана графова відстань, а w_{ij} – ваговий коефіцієнт.

Ітераційний процес оптимізації складається з кількох логічних кроків. На кожній ітерації оцінюється напрям зменшення цільової функції, оновлюються координати вершин, за потреби коригується величина кроку та перевіряється критерій зупинки. Зі змістової точки зору це означає, що вершини послідовно переміщуються так, щоб геометрична структура укладання дедалі краще узгоджувалася з топологічною структурою графа.

Після завершення багатовимірної оптимізації отримується матриця координат $X_d \in \mathbb{R}^{n\times d}$. Якщо $d > 2$, безпосередньо використовувати її для побудови рисунка неможливо, тому виконується проєкція в $\mathbb{R}^2$. У цій роботі для цього застосовано метод головних компонент (PCA) [11], оскільки він є швидким, стабільним і не потребує складного налаштування додаткових параметрів.

PCA зберігає напрями найбільшої дисперсії багатовимірного розташування, що робить його зручним інструментом для серійних експериментів. Водночас слід підкреслити, що PCA є лінійним методом, тому він не гарантує ідеального збереження локальної структури або всіх симетрій. Саме тому в експериментальному розділі окремо аналізується вплив проміжної розмірності на вигляд фінальної площинної проєкції.

Підсумкову схему алгоритму подано на рис. 1. Вона узагальнює послідовність дій: від задання графа та параметрів побудови до отримання кінцевого двовимірного укладання.

Алгоритм також можна подати у вигляді послідовності кроків:

1. задати граф $G = (V, E)$, проміжну розмірність d, спосіб початкової ініціалізації та цільову функцію;
2. побудувати початкові координати вершин;
3. виконати оптимізацію Kamada–Kawai у просторі $\mathbb{R}^d$;

4. отримати матрицю багатовимірних координат X_d;

5. якщо $d > 2$, застосувати PCA і побудувати двовимірну проєкцію Y;

6. зберегти рисунок укладання та часові характеристики виконання.

Програмна реалізація

Програмну реалізацію запропонованого підходу виконано мовою Python у середовищі Google Colab. Такий вибір є доцільним для експериментальної роботи, оскільки Python має розвинену екосистему бібліотек для роботи з графами, числовими обчисленнями, табличними даними та візуалізацією. Середовище Google Colab дозволяє швидко запускати код, зберігати результати та зручно працювати з рисунками й таблицями.

У програмі використано бібліотеки `NetworkX`, `NumPy`, `pandas`, `matplotlib`, `openpyxl`, а також стандартні модулі `time` і `os`. Бібліотека `NetworkX` використовується для створення графів і побудови укладань Kamada–Kawai у довільній розмірності [17]. `NumPy` забезпечує роботу з матрицями координат і операціями лінійної алгебри, зокрема під час PCA-проєкції. `pandas` застосовується для збереження та групування експериментальних результатів у табличній формі.

Бібліотека `matplotlib` використовується для побудови рисунків укладань і графіків часу. `openpyxl` забезпечує запис результатів у Excel-файли. Модуль `time` використовується для вимірювання часу Kamada–Kawai, PCA та сумарного часу, а `os` – для організації файлової структури результатів. Увесь програмний код, використаний для реалізації алгоритму, побудови тестових графів, виконання експериментів, формування таблиць і побудови рисунків, розміщено у репозиторії [35]. Це забезпечує відтворюваність експериментальної частини роботи та дає змогу перевірити послідовність обчислень, описану в цьому розділі.

Код організовано у вигляді окремих модулів. Файл `imports.py` містить імпорт основних бібліотек. Файл `graphs.py` відповідає за побудову тестових графів, зокрема гіперкубів і повних графів. Файл `instruments.py` містить допоміжні математичні функції, включаючи обчислення відстаней, вибір опорних вершин, PCA-проєкцію та виклик Kamada–Kawai у просторі довільної розмірності.

Файл `experiments.py` реалізує основний сценарій одного експерименту: побудову багатовимірного укладання, проєкцію у 2D та вимірювання часових показників. Файл `excel.py` відповідає за формування Excel-таблиць і супровідних графіків. Файл `monteCarlo.py` використовується для повторних запусків експериментів на повних графах із подальшим обчисленням середніх значень і стандартних відхилень.

Таким чином, програмна реалізація безпосередньо відтворює схему

$$\text{граф} \to \text{Kamada–Kawai у } \mathbb{R}^d \to \text{PCA} \to \text{координати в } \mathbb{R}^2 \to \text{рисунки і таблиці.}$$

Це робить запропонований підхід відтворюваним і придатним для серійних експериментів з різними класами графів та різними значеннями проміжної розмірності.

Висновки до розділу 3

У третьому розділі було розроблено алгоритм Nd $\to$ 2d для побудови двовимірних укладань графів через попередню багатовимірну оптимізацію. Описано його вхідні та вихідні дані, роль початкового положення вершин, цільової функції та проєкції методом головних компонент.

Також створено програмну реалізацію мовою Python у середовищі Google Colab. Реалізований програмний pipeline забезпечує повний цикл обробки: побудову тестових графів, багатовимірне укладання, проєкцію у 2D, збереження рисунків і формування табличних результатів. Це створює основу для подальшого експериментального дослідження, наведеного в четвертому розділі.

Табл. 2: Характеристики тестових гіперкубів

Граф	Кількість вершин	Кількість ребер	Ступінь	Природна розмірність
$Q4$	16	32	4	4
$Q6$	64	192	6	6
$Q8$	256	1024	8	8
$Q10$	1024	5120	10	10
$Q12$	4096	24576	12	12

Табл. 3: Характеристики тестових повних графів

Граф	Кількість вершин	Кількість ребер	Ступінь
$K4$	4	6	3
$K6$	6	15	5
$K8$	8	28	7
$K10$	10	45	9
$K12$	12	66	11

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ

Гіпотеза

Експериментальна частина роботи спрямована на перевірку припущення, що для багатьох регулярних і симетричних графів існує природна розмірність, у якій їхня структура відображається більш адекватно, ніж у площині. Для n-вимірного куба такою природною розмірністю є n, а для інших графів вона може бути пов'язана з комбінаторною природою графа, зокрема для регулярних графів дорівнювати степеню вершин.

Гіпотезу можно сформулювати так: попередня оптимізація в просторі більшої розмірності може покращувати структурну організацію кінцевого двовимірного укладання, але не гарантує автоматичної переваги для всіх графів і всіх значень проміжної розмірності. Таким чином, природна розмірність розглядається як початковий орієнтир, доцільність вибору якого потребує експериментальної перевірки.

У межах дослідження перевіряється, чи дає схема Nd $\rightarrow$ 2d [9; 10; 13] виграш за часом, чи покращує візуальну якість укладання та чи сприяє збереженню структурних властивостей графа, таких як симетрії, шари, паралельні класи ребер і характерні підграфи.

Тестові графи

Для перевірки сформульованої гіпотези використано переважно регулярні та симетричні графи. Основним класом тестових графів стали гіперкуби $Q4$, $Q6$, $Q8$, $Q10$ та $Q12$. Вони є природним об'єктом дослідження, оскільки для кожного такого графа наперед відома природна розмірність. Характеристики тестових гіперкубів навежено в Таблиці 2.

Іншим класом тестових графів стали повні графи $K4$, $K6$, $K8$, $K10$ і $K12$. На відміну від гіперкубів, вони мають меншу кількість вершин, але значно більшу щільність зв'язків. Тому вони є зручними для дослідження меж застосовності методу у випадку дуже щільних регулярних графів. Їх характеристики наведено в Таблиці 3.

Як додаткові приклади використовуються граф Геммінга $H(4,2)$, граф октаедра та тороїдальна сітка 4×4. Графи Джонсона $J(m,n)$ і графи Келі скінченних груп у межах цієї роботи не досліджувалися експериментально, однак розглядаються як перспективні класи для подальших розширень. Характеристики графів, придатних для експериментів подано в Таблиці 4.

Табл. 4: Зведена характеристика тестових графів

Клас графів	Приклади	Причина вибору	Роль у роботі
Гіперкуби	$Q4, Q6, Q8, Q10, Q12$	Відома природна розмірність	Основний експериментальний клас
Повні графи	$K4, K6, K8, K10, K12$	Щільні регулярні графи	Перевірка поведінки на щільних структурах
Правильні багатогранники	Octahedron	Симетрична геометрична структура	Додатковий приклад
Графи Геммінга	$H(4, 2)$	Координатна регулярність	Додатковий приклад
Тороїдальні сітки	Torus 4×4	Природна двовимірна періодична структура	Контрастний приклад
Графи Джонсона	$J(m, n)$	Комбінаторна регулярність	Перспектива
Графи Келі	скінченні групи	Алгебраїчна регулярність	Перспектива

Табл. 5: Метрики оцінки укладань

Метрика	Що вимірює	Як інтерпретується	Статус у роботі
Stress	Узгодження геометричних і графових відстаней	Менше значення є кращим	Теоретично описано
Кількість перетинів	Перетини ребер у площинному зображенні	Менше значення є кращим	Якісний аналіз
Довжини ребер	Рівномірність масштабу укладання	Менший розкид є кращим	Перспектива
Кути між ребрами	Локальна читабельність	Уникаються кути, близькі до нуля	Перспектива
Граф-специфічні ознаки	Грані, шари, симетрії, паралельні ребра	Більше збереження структури є кращим	Якісний аналіз
Час Kamada–Kawai	Час багатовимірної оптимізації	Менше значення є кращим	Обчислено
Час PCA	Час проєкції у $\mathbb{R}^2$	Менше значення є кращим	Обчислено
Сумарний час	Загальний час побудови	Менше значення є кращим	Обчислено

Метрики оцінки

Для порівняння отриманих укладань доцільно використовувати як кількісні, так і якісні критерії [1; 2]. До класичних метрик належать stress (тобто функціонал якості алгоритму Камада-Каваї), кількість перетинів ребер, рівномірність довжин ребер, розподіл кутів між ребрами та граф-специфічні ознаки, пов'язані із збереженням симетрії, шарів або характерних підструктур.

У межах цієї роботи кількісно фіксувався насамперед час виконання. Інші метрики використовуються як змістовна основа для інтерпретації результатів, але не були обчислені в межах проведеного експерименту. Тому далі коректно зазначати, що stress, кількість перетинів, довжини ребер і кути використовуються переважно для якісного аналізу рисунків. Опис метрик оцінки укладань наведено в Таблиці 5.

Хід експерименту

В експерименті порівнюються два підходи: класичний Kamada–Kawai у двовимірному просторі [5] та схема Nd $\rightarrow$ 2d, у якій Kamada–Kawai спочатку застосовується у просторі $\mathbb{R}^d$, а потім виконується PCA-проєкція в $\mathbb{R}^2$ [11]. Базовим випадком є $d = 2$, а для багатовимірного підходу розглядаються значення d від 2 до 20.

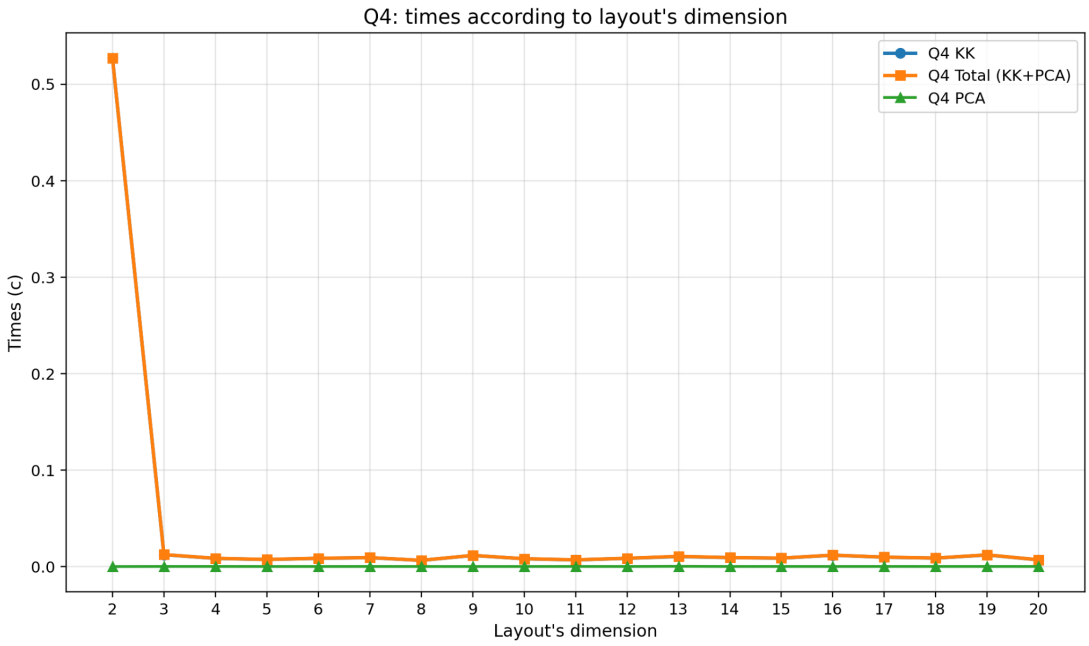

Рис. 2: Час виконання для $Q4$ залежно від проміжної розмірності

Для гіперкубів $Q4$, $Q6$, $Q8$, $Q10$, $Q12$ було побудовано часові графіки, які окремо показують час Kamada–Kawai, час PCA та сумарний час. Для повних графів $K4$, $K6$, $K8$, $K10$, $K12$ додатково проведено Monte Carlo-дослідження часу PCA, що дозволяє оцінити стабільність результатів і розкид спостережень. Усі файли з результатами експериментів, зокрема Excel-таблиці часу виконання, збережені координати укладань, побудовані рисунки та скрипти для їх генерації, розміщено у репозиторії дослідження [35]. Тому наведені нижче графіки та зображення є результатом виконання єдиного програмного pipeline, описаного в розділі 3, а не окремо підготовленими ілюстраціями.

Для $Q4$ (див. Рис. 2) видно, що після виходу за межі випадку $d = 2$ сумарний час швидко стабілізується, а внесок PCA залишається майже непомітним. Це свідчить про те, що основний обчислювальний внесок робить саме етап Kamada–Kawai.

Для $Q6$ (див. Рис. 3) часові показники є більш рівномірними, ніж для $Q4$, однак співвідношення між компонентами часу залишається тим самим: PCA має дуже малу частку в сумарному часі.

Для $Q8$ (див. Рис. 4) так само, як і для $Q6$, залежність часу від розмірності є немонотонною. Це важливий результат, оскільки він показує, що проміжна розмірність є самостійним параметром алгоритму, а не формальною технічною деталлю.

Для $Q10$ (див. Рис. 5) спостерігається суттєва різниця між окремими значеннями d. Це означає, що підбір розмірності здатний впливати не лише на структуру укладання, а й на обчислювальну ефективність.

Для $Q12$ (див. Рис. 6) час виконання різко зростає, що пояснюється великою кількістю вершин і ребер. Водночас навіть у цьому випадку PCA не є головним джерелом витрат часу, а визначальним фактором залишається етап Kamada–Kawai.

Ключовим спостереженням є те, що час роботи алгоритму $Nd \to 2d$, у випадку коли d є природною розмірністю графа не перевищує час роботи алгоритму укладання на площині.

Числові результати порівння часу роботи алгоритмів наведено в таблицях, ознайомитися з якими можна в репозиторії дослідження [35].

Monte Carlo-графік для повних графів (див. Рис. 7) підтверджує, що час PCA залишається дуже малим навіть при зміні розмірності від 2 до 20. Отже, при аналізі часу роботи основну увагу слід приділяти поведінці Kamada–Kawai, а не етапу проєкції.

Для візуального порівняння укладань, розрахованих алгоритмом $Ns- > 2d$ (у природній та

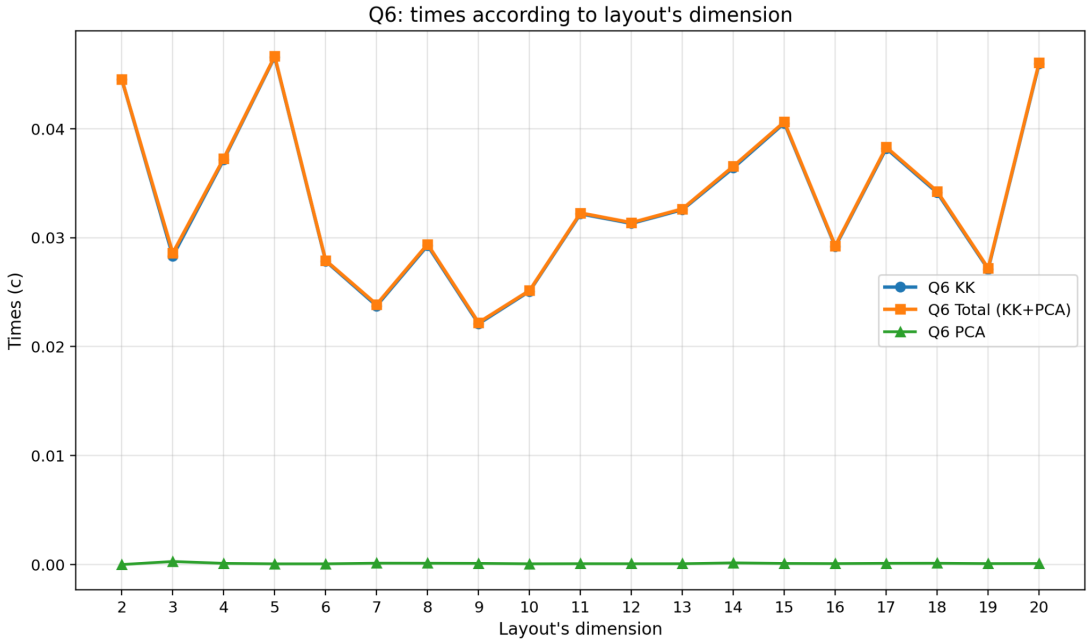

Рис. 3: Час виконання для $Q6$ залежно від проміжної розмірності

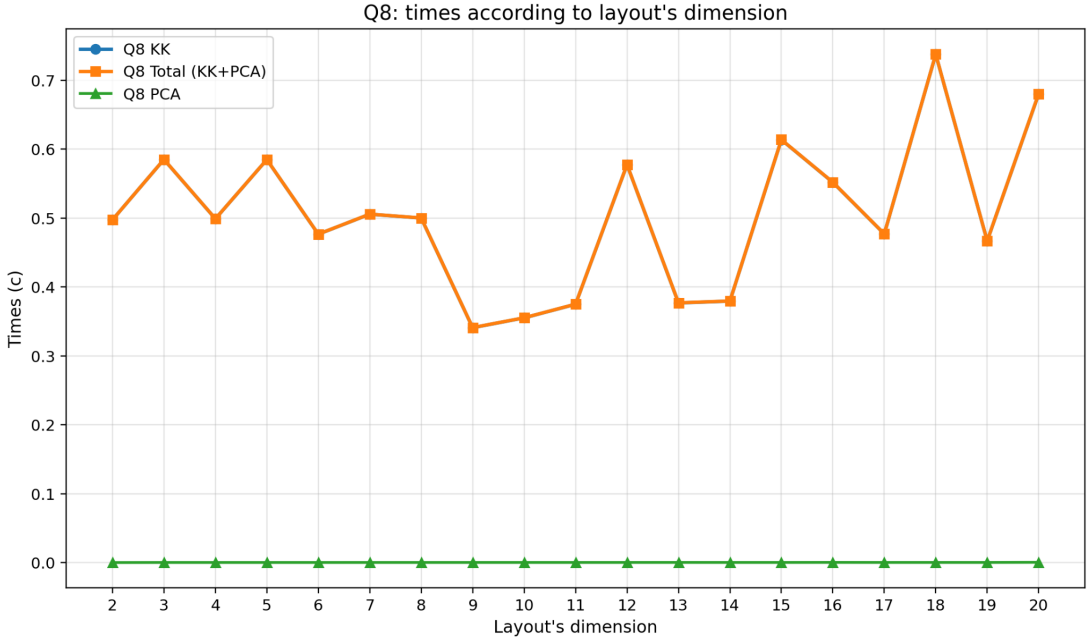

Рис. 4: Час виконання для $Q8$ залежно від проміжної розмірності

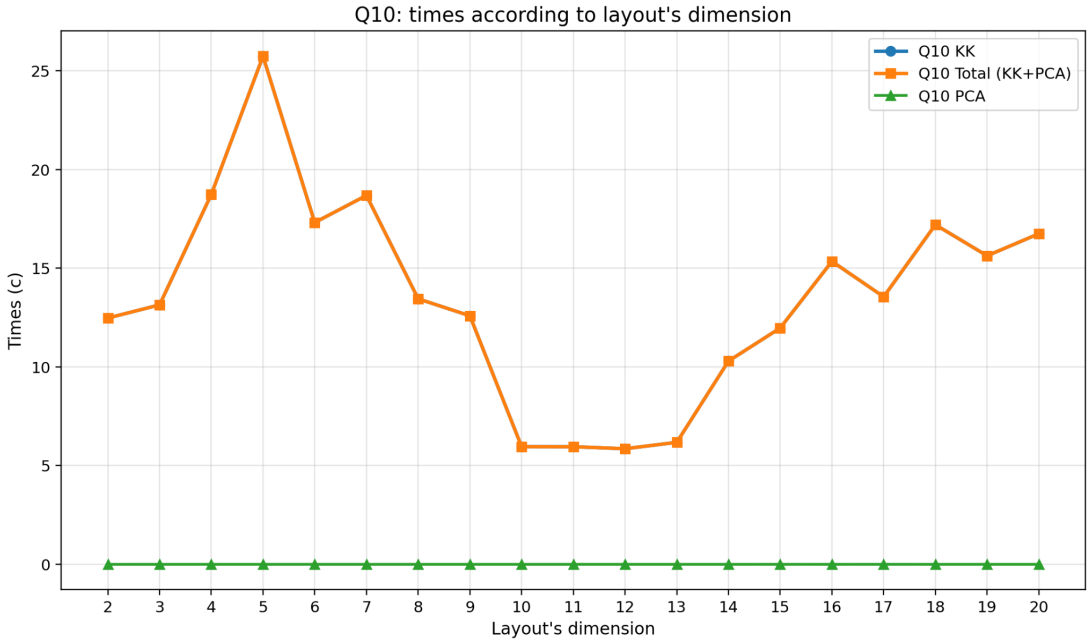

Рис. 5: Час виконання для $Q10$ залежно від проміжної розмірності

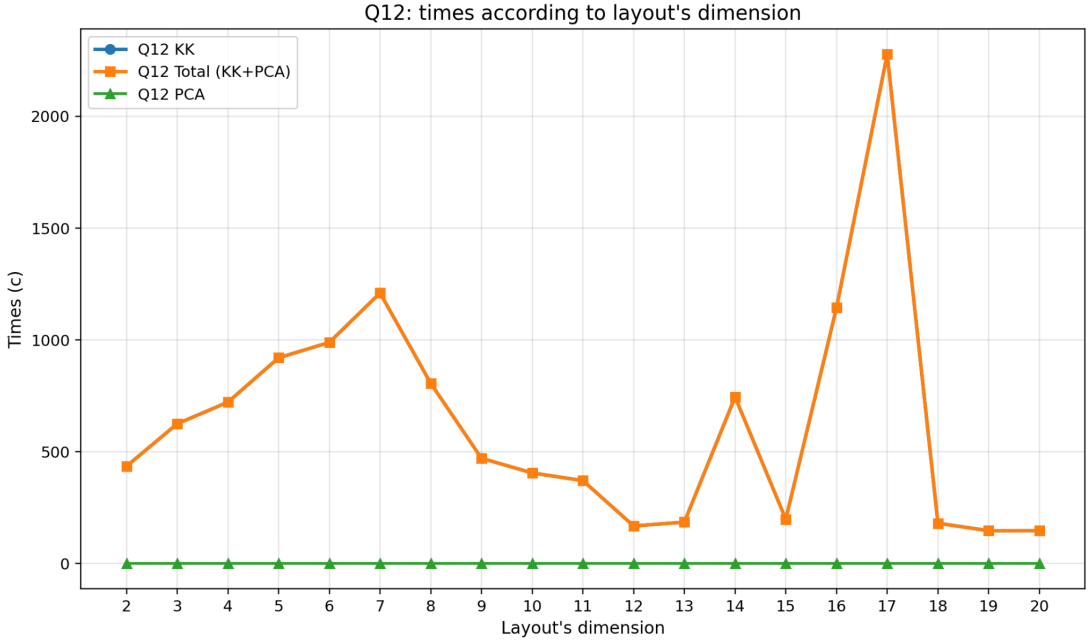

Рис. 6: Час виконання для $Q12$ залежно від проміжної розмірності

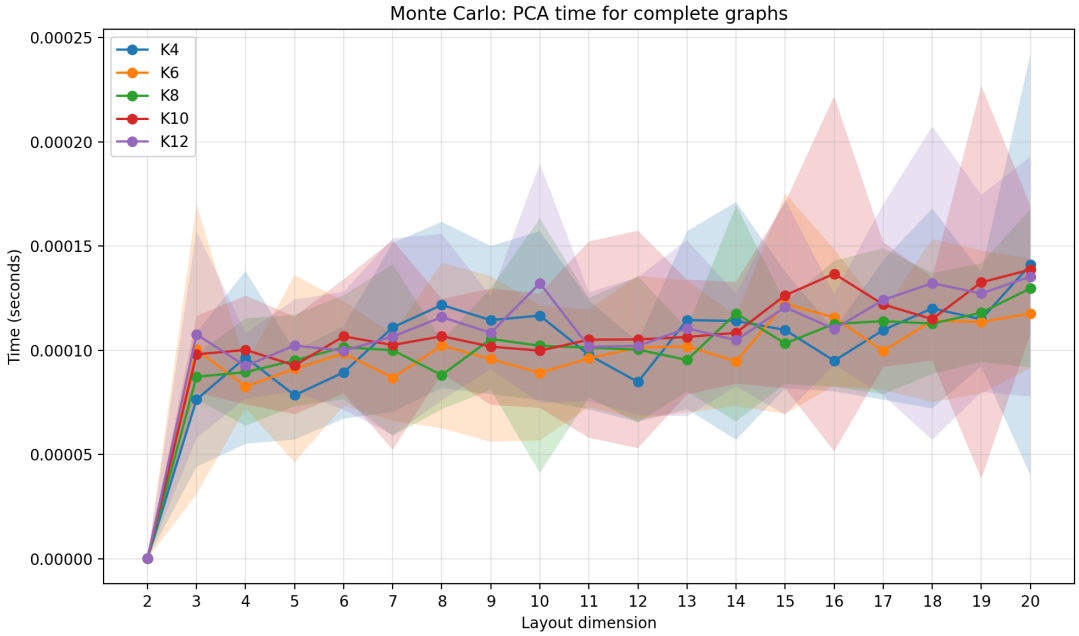

Рис. 7: Monte Carlo-дослідження часу PCA для повних графів

деяких інших розмріностях) та двовимірним алгоритмом Камада-Каваї, використано візуалізації укладань для $Q4$, $Q6$, $Q8$ та $Q10$, а також додаткові приклади для $H(4,2)$, $K12$, октаедра та тора 4×4, наверені на Рис.8-12. Кожна група рисунків дозволяє оцінити, як саме змінюється структура фінального укладання при переході до вищої проміжної розмірності.

Аналіз результатів

Отримані часові результати свідчать, що алгоритм $Nd \to 2d$ виграє за часом порівняно з прямим двовимірним укладанням для кубічних графів, отже, доцільним є тестування алгоритма на інших регулярних графах з урахування припущення про їх природну розмірність. У той же час для повних графів виграшу в часі не спостерігається, отже, актуальним залишшається питання про таке значення розмірності d, для якого $Nd \to 2d$ демонструє найкращий час, адже вплив проміжної розмірності виявився суттєвим і немонотонним.

Найстійкіший висновок полягає в тому, що PCA практично не впливає на сумарний час виконання. У всіх серіях експериментів лінії часу Kamada–Kawai і сумарного часу майже збігаються, а Monte Carlo-дослідження для повних графів підтверджує, що абсолютний час PCA залишається дуже малим. Тому головне обчислювальне навантаження припадає саме на багатовимірну оптимізацію.

Візуальний аналіз наведених укладань (див. Рис.8-11) дозволяє зробити наступні спостереження:

Для $Q4$ порівняння рисунків показує, що при переході до $d = 4$ структура гіперкуба виявляється виразнішою, ніж при прямому двовимірному укладанні. Краще помітні паралельні групи ребер і загальна шарова організація. Це узгоджується з очікуванням, що природна розмірність для цього графа є змістовною.

Для $Q6$ збереження структури також покращується, однак із ростом розміру графа повністю уникнути перетинів у площині вже неможливо. Отже, багатовимірна оптимізація покращує глобальну організацію укладання, але не усуває всіх труднощів площинної візуалізації.

Найбільш показовим є приклад $Q8$. Саме для цього графа видно, що варіант $d = 7$ дає більш виразну шарову та симетричну структуру, ніж природний варіант $d = 8$. Таким чином, параметр d та його вплив на подання графа слід розглядати як окремий об'єкт дослідження.

Для $Q10$ і $Q12$ стає очевидним обмеження підходу. Навіть після багатовимірної оптимізації

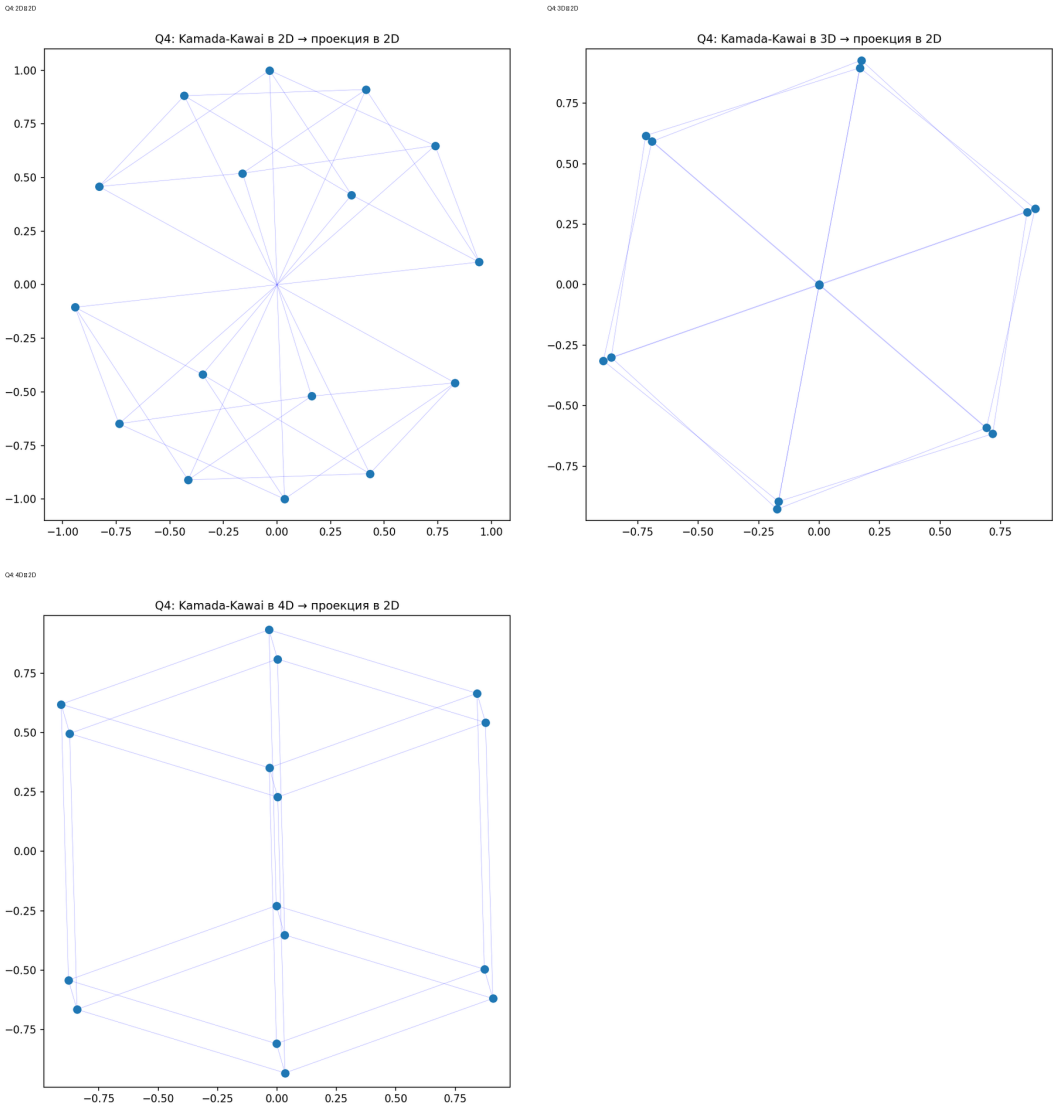

Рис. 8: Порівняння укладань для $Q4$ при $d = 2$, $d = 3$ та $d = 4$

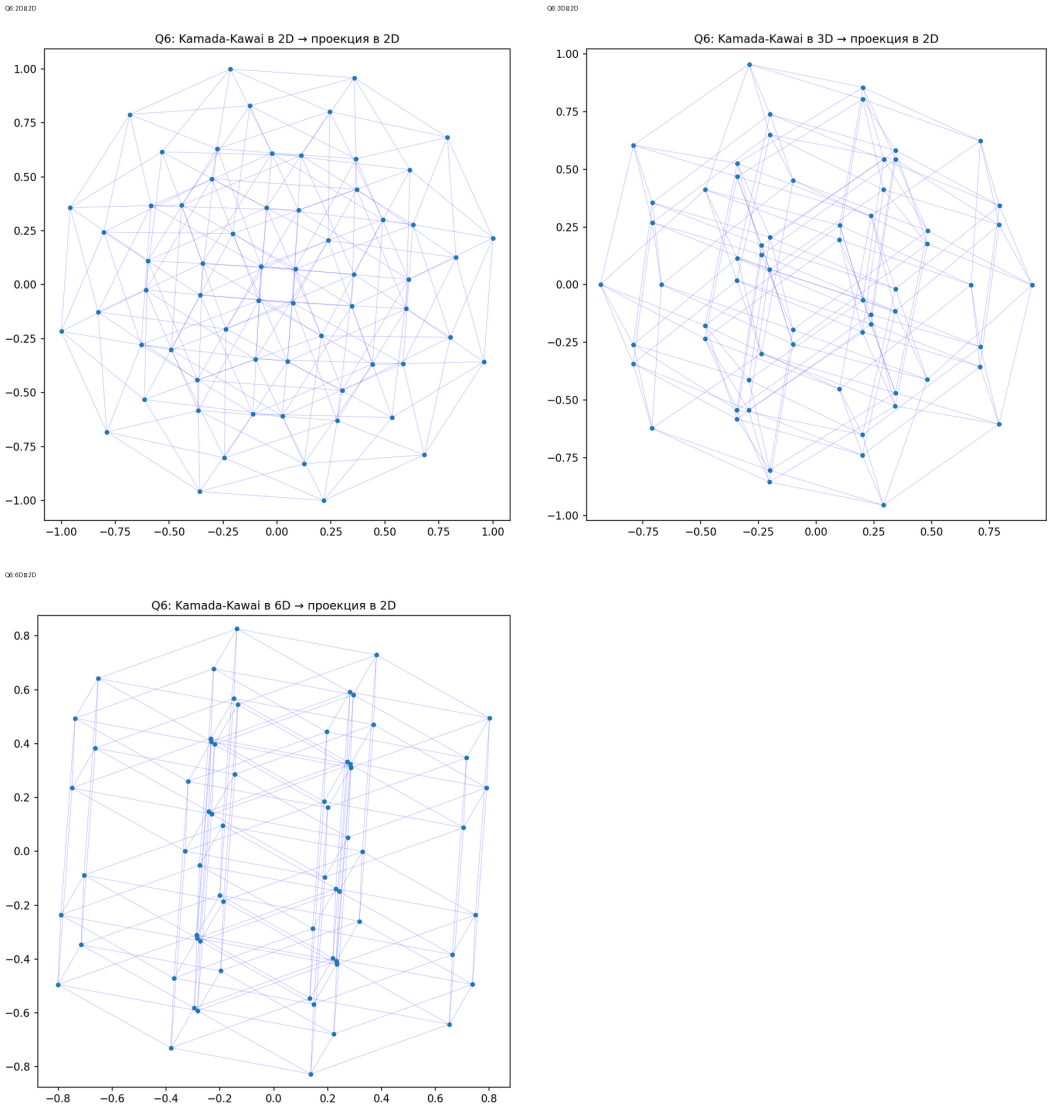

Рис. 9: Порівняння укладань для $Q6$ при $d = 2$, $d = 3$ та $d = 6$

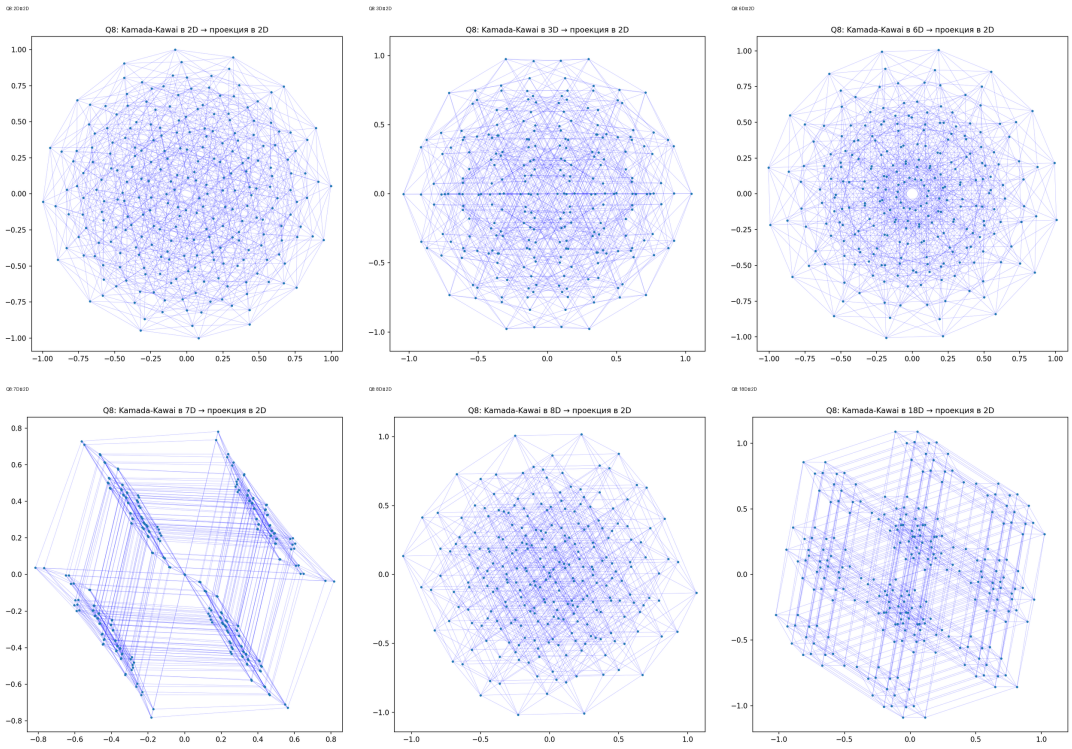

Рис. 10: Порівняння укладань для $Q8$ при $d = 2$, $d = 3$, $d = 6$, $d = 7$, $d = 8$ та $d = 18$

проєкція в $\mathbb{R}^2$ залишається дуже щільною, а рисунки важко сприймати як повністю читабельні. Це означає, що для великих гіперкубів доцільно додатково застосовувати інтерактивну візуалізацію, тривимірні представлення або методи фільтрації.

Аналіз повних графів (див. Рис. 12) показує іншу причину обмежень: не надто велика кількість вершин, а дуже висока щільність ребер. Навіть для $K12$ багатовимірна оптимізація не робить площинне зображення по-справжньому читабельним. Це пов'язано передусім із природою самого повного графа, а не тільки з недоліками методу.

Додаткові приклади (див. Рис. 12) розширюють інтерпретацію результатів. Граф $H(4, 2)$ демонструє поведінку регулярної координатної структури, близької до гіперкуба. Октаедр підтверджує доцільність підходу для малих симетричних графів, а Torus 4×4 показує, що для графів із природною двовимірною структурою перехід до вищих розмірностей не є настільки принциповим.

Слід також зазначити обмеження проведеного дослідження. По-перше, експерименти зосереджено переважно на регулярних і симетричних графах, тому узагальнення результатів на довільні графи потребує додаткової перевірки. По-друге, не всі можливі метрики були обчислені кількісно; у межах цієї роботи кількісно фіксувався насамперед час виконання, тоді як інші метрики використовувалися переважно для якісної інтерпретації рисунків.

Висновки до розділу 4

У четвертому розділі проведено експериментальне дослідження методу Nd $\rightarrow$ 2d на гіперкубах, повних графах та кількох додаткових регулярних структурах. Отримані результати частково підтверджують сформульовану гіпотезу: попередня оптимізація в просторі більшої розмірності може покращувати структурну організацію фінального двовимірного укладання.

Водночас природна розмірність графа не завжди виявляється найкращим вибором. Приклад $Q8$ показує, що близька, але не тотожна природній розмірність може давати більш виразний візуальний результат. Це дозволяє зробити висновок, що параметр d слід підбирати експериментально.

Часові графіки показують, що етап PCA практично не впливає на сумарний час побудови.

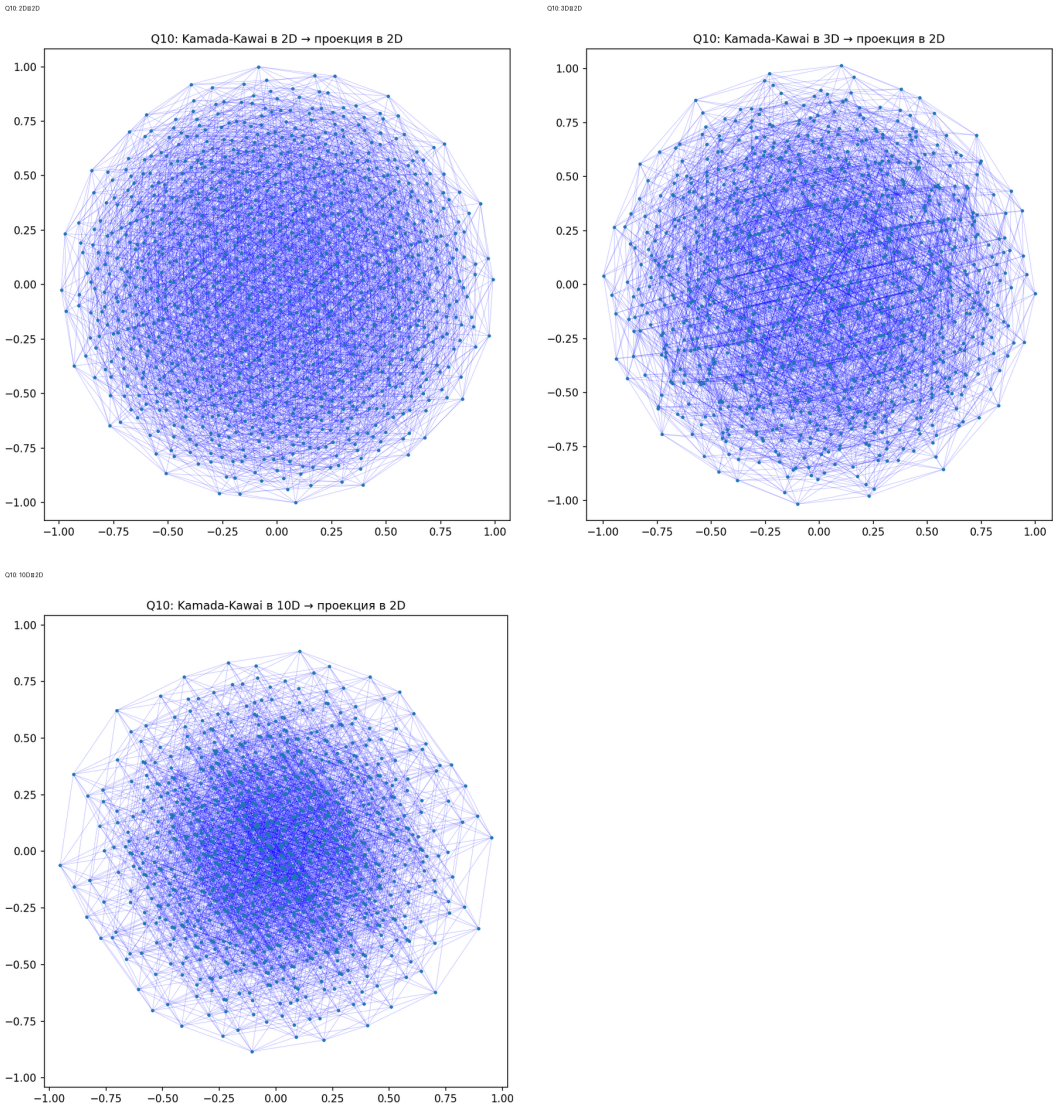

Рис. 11: Порівняння укладань для $Q10$ при $d = 2$, $d = 3$ та $d = 10$

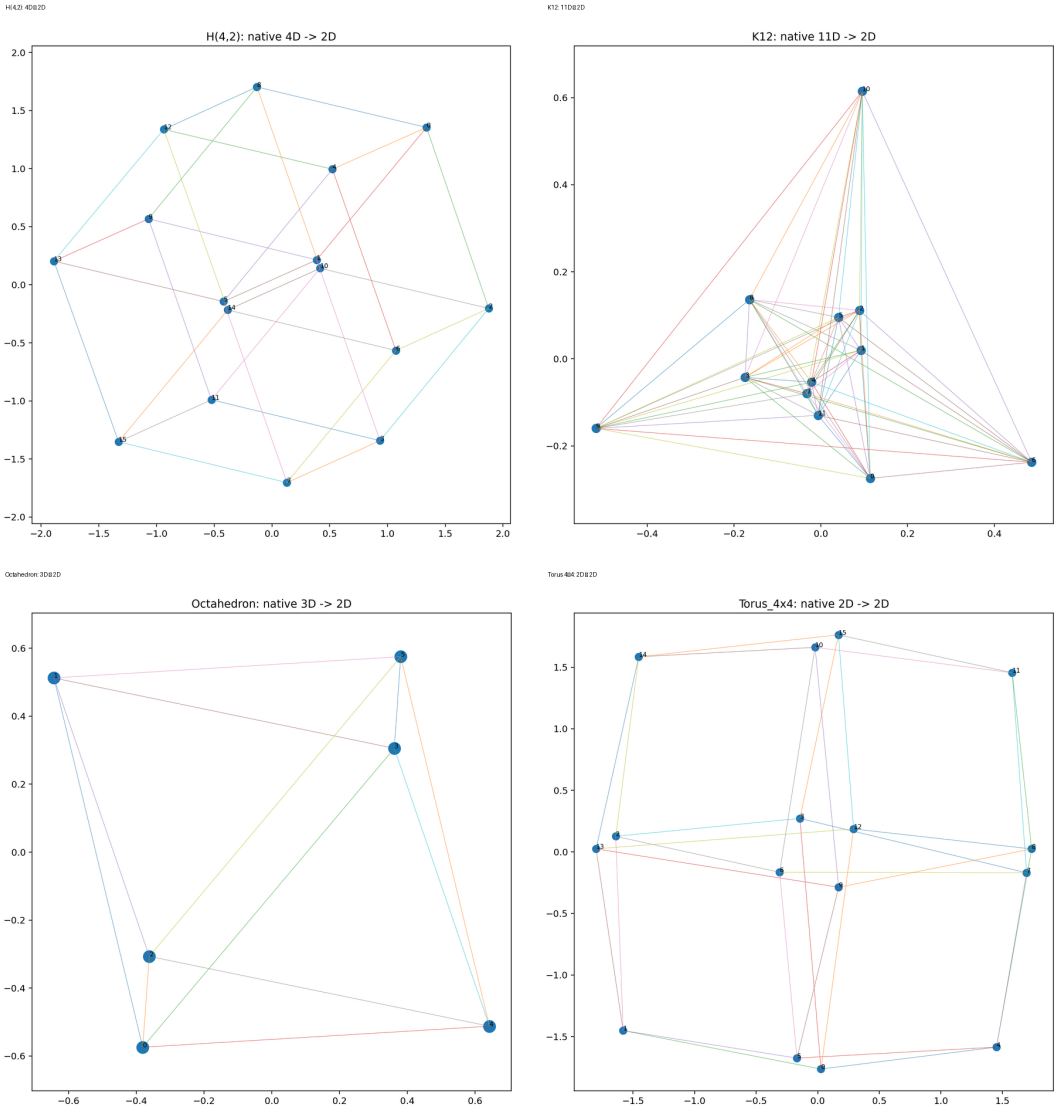

Рис. 12: Додаткові приклади: $H(4,2)$, $K12$, Octahedron, Torus 4×4

Основна обчислювальна вартість пов'язана з алгоритмом Kamada–Kawai. Для великих і щільних графів сам перехід до багатовимірної оптимізації не усуває всіх труднощів візуалізації, однак відкриває можливість краще виявляти регулярні структурні особливості.

Отже, метод Nd $\rightarrow$ 2d є перспективним насамперед для регулярних, симетричних і багатовимірно організованих графів, зокрема для гіперкубів, графів Геммінга та близьких до них класів. Подальші дослідження доцільно спрямувати на кількісний розрахунок додаткових метрик, порівняння різних методів проєкції та застосування підходу до ширшого класу графів.

ВИСНОВКИ

У магістерській роботі розглянуто проблему побудови якісних укладань графів на основі попереднього укладання у багатовимірному просторі та подальшого проєктування результату на площину. Запропонований алгоритм $Nd2d$ розглядає побудову укладання як двоетапний процес: спочатку структура графа оптимізується у просторі більшої розмірності, а потім отримане розташування вершин відображається на площину. Такий підхід дозволяє розширити можливості класичних методів візуалізації графів і дослідити, як проміжна розмірність впливає на якість та стабільність кінцевого зображення. У межах роботи було проаналізовано основні методи укладання графів, сформульовано математичну модель багатовимірного укладання, розроблено алгоритм $Nd2d$ та створено його програмну реалізацію. Реалізований програмний pipeline забезпечує повний цикл експерименту: побудову тестових графів, багатовимірну оптимізацію, проєкцію у двовимірний простір, вимірювання часу виконання та візуалізацію результатів. Це дало змогу провести відтворюване експериментальне дослідження запропонованого підходу.

Проведені експерименти показали, що попереднє укладання у просторі більшої розмірності може забезпечити виграш у часі та у якості візуалізації для окремих класів графів, зокрема, для регулярних і симетричних.

Попереднє укладання у просторі високої розмірності проводилося у припущенні, що граф має деяку природну розмірність. Гіпотеза про природну розмірність формулювалася з урахуванням структури та симетрій графа, але формальне визначення природної розмірності вимагає подальшого дослідження, так само як і залежність часу роботи алгоритму та якісних характеристик укладання від проміжної размірності як від параметра.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Di Bartolomeo S., Crnovrsanin T., Saffo D., Puerta E., Wilson C., Dunne C. Evaluating Graph Layout Algorithms: A Systematic Review of Methods and Best Practices. *Computer Graphics Forum*. 2024. Vol. 43, no. 6. e15073. DOI: https://doi.org/10.1111/cgf.15073.

[2] Gibson H., Faith J., Vickers P. A survey of two-dimensional graph layout techniques for information visualization. *Information Visualization*. 2013. Vol. 12. P. 324–357. DOI: https://doi.org/10.1177/1473871612455749.

[3] Pinki P., Shekhawat K. An annotated review on graph drawing and its applications. *AKCE International Journal of Graphs and Combinatorics*. 2023. Vol. 20, no. 3. P. 258–281. DOI: https://doi.org/10.1080/09728600.2023.2218459.

[4] Fruchterman T. J., Reingold E. M. Graph drawing by force-directed placement. *Software: Practice and Experience*. 1991. Vol. 21, no. 11. P. 1129–1164. DOI: https://doi.org/10.1002/spe.4380211102.

[5] Kamada T., Kawai S. An algorithm for drawing general undirected graphs. *Information Processing Letters*. 1989. Vol. 31, no. 1. P. 7–15. DOI: https://doi.org/10.1016/0020-0190(89)90102-6.

[6] Gansner E. R., Koren Y., North S. Graph Drawing by Stress Majorization. In: Pach J. (ed.). *Graph Drawing*. Berlin, Heidelberg : Springer, 2005. P. 239–250. DOI: https://doi.org/10.1007/978-3-540-31843-9_25.

[7] Koren Y. On Spectral Graph Drawing. In: *Computing and Combinatorics. COCOON 2003*. Lecture Notes in Computer Science. Vol. 2697. Berlin : Springer, 2003. P. 496–508.

[8] Tutte W. T. How to draw a graph. *Proceedings of the London Mathematical Society*. 1963. Vol. 3, no. 1. P. 743–768. DOI: https://doi.org/10.1112/plms/s3-13.1.743.

[9] Harel D., Koren Y. Graph Drawing by High-Dimensional Embedding. *Journal of Graph Algorithms and Applications*. 2004. Vol. 8, no. 2. P. 195–214. DOI: https://doi.org/10.7155/jgaa.00089.

[10] Gajer P., Goodrich M. T., Kobourov S. G. A multi-dimensional approach to force-directed layouts of large graphs. *Computational Geometry: Theory and Applications*. 2004. Vol. 29, no. 1. P. 3–18. DOI: https://doi.org/10.1016/j.comgeo.2004.03.014.

[11] Jolliffe I. T. *Principal Component Analysis*. 2nd ed. New York : Springer, 2002. 487 p.

[12] Kruskal J. B. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*. 1964. Vol. 29. P. 115–129.

[13] Han B., Wei Y. et al. Graph Layout Based on Network Embedding and Improved Dimensionality Reduction. In: *2020 6th IEEE International Conference on Big Data and Information Analytics (BigDIA)*. 2020. P. 125–132. DOI: https://doi.org/10.1109/BigDIA51454.2020.00028.

[14] Wang X., Yen K., Hu Y., Shen H.-W. DeepGD: A Deep Learning Framework for Graph Drawing Using GNN. 2021. arXiv:2106.15347.

[15] Tiezzi M., Ciravegna G., Gori M. Graph Neural Networks for Graph Drawing. *IEEE Transactions on Neural Networks and Learning Systems*. 2024. Vol. 35, no. 4. P. 4668–4681. DOI: https://doi.org/10.1109/TNNLS.2022.3184967.

[16] Both C., Dehmamy N., Yu R., Barabási A.-L. Accelerating network layouts using graph neural networks. *Nature Communications*. 2023. Vol. 14. Article 1560. DOI: https://doi.org/10.1038/s41467-023-37189-2.

[17] Hagberg A., Swart P., Chult D. Exploring network structure, dynamics, and function using NetworkX. Los Alamos National Laboratory, 2008. URL: https://networkx.org/.

[18] Csardi G., Nepusz T. The igraph software package for complex network research. *InterJournal. Complex Systems*. 2006. Vol. 1695, no. 5. P. 1–9. URL: https://igraph.org/.

[19] Bastian M., Heymann S., Jacomy M. Gephi: An Open Source Software for Exploring and Manipulating Networks. *Proceedings of the International AAAI Conference on Weblogs and Social Media*. 2009. Vol. 3, no. 1. P. 361–362. DOI: https://doi.org/10.1609/icwsm.v3i1.13937.

[20] Shannon P. et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*. 2003. Vol. 13, no. 11. P. 2498–2504.

[21] Bostock M., Ogievetsky V., Heer J. D^3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*. 2011. Vol. 17, no. 12. P. 2301–2309. DOI: https://doi.org/10.1109/TVCG.2011.185.

[22] Ellson J., Gansner E., Koutsofios L., North S., Woodhull G. Graphviz – Open Source Graph Drawing Tools. In: *Graph Drawing*. Berlin, Heidelberg : Springer, 2002. P. 483–484.

[23] Sugiyama K., Tagawa S., Toda M. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*. 1981. Vol. 11, no. 2. P. 109–125. DOI: https://doi.org/10.1109/TSMC.1981.4308636.

[24] van der Maaten L., Hinton G. Visualizing Data using t-SNE. *Journal of Machine Learning Research*. 2008. Vol. 9. P. 2579–2605.

[25] McInnes L., Healy J., Melville J. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. 2018. arXiv:1802.03426.

[26] Hachul S., Jünger M. Drawing Large Graphs with a Potential-Field-Based Multilevel Algorithm. In: *Graph Drawing. GD 2004*. Lecture Notes in Computer Science. Vol. 3383. Berlin : Springer, 2005. P. 285–295.

[27] Hu Y. Efficient and High Quality Force-Directed Graph Drawing. *The Mathematica Journal*. 2005. Vol. 10. P. 37–71.

[28] Barnes J., Hut P. A hierarchical O(N log N) force-calculation algorithm. *Nature*. 1986. Vol. 324, no. 6096. P. 446–449. DOI: https://doi.org/10.1038/324446A0.

[29] Perozzi B., Al-Rfou R., Skiena S. DeepWalk: Online Learning of Social Representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014. P. 701–710. DOI: https://doi.org/10.1145/2623330.2623732.

[30] Grover A., Leskovec J. node2vec: Scalable Feature Learning for Networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016. P. 855–864. DOI: https://doi.org/10.1145/2939672.2939754.

[31] Hamilton W. L., Ying R., Leskovec J. Inductive Representation Learning on Large Graphs. In: *Advances in Neural Information Processing Systems*. 2017. Vol. 30.

[32] Kipf T. N., Welling M. Semi-Supervised Classification with Graph Convolutional Networks. In: *International Conference on Learning Representations*. 2017.

[33] Shen L., Tai Z., Shen E., Wang J. Graph Exploration with Embedding-Guided Layouts. 2023. arXiv:2208.13699.

[34] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P. *Numerical Recipes in C: The Art of Scientific Computing*. 2nd ed. Cambridge : Cambridge University Press, 1992. 994 p.

[35] Mysyk, O. Project repository, 2026. Available at: https://github.com/AlexandraMysyk/Graph-Master-Diploma (accessed 01 May 2026).