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 графових укладань
для мереж із структурою спільнот

Виконав: студент 6 курсу, групи МФ-61
спеціальність 122 «Комп'ютерні науки»
освітньо-наукова програма
«Інформатика»

Лінник О.С.
Керівник Полякова Л.Ю.

Харків – 2026

АНОТАЦІЯ

У даній магістерській роботі досліджується задача візуалізації графів із акцентом на відображення їхньої спільнотної структури. Запропоновано підхід, заснований на поєднанні алгоритму Камада–Каваї з нейромережевим прискоренням (KamadaNN), який дозволяє отримувати вкладення графів у евклідовий простір із збереженням як глобальних, так і локальних структурних властивостей. Особливу увагу приділено оцінюванню якості отриманих укладок у контексті подальшого виявлення спільнот.

Експериментальне дослідження проведено на кількох сімействах графів, зокрема симетричних графах із чітко вираженими спільнотами, несиметричних графах зі спільнотною структурою, а також графах без чітких спільнот. Для кожного графа застосовано різні алгоритми виявлення спільнот, включаючи Louvain, Girvan–Newman, greedy modularity та Label Propagation. Якість результатів оцінювалась за сукупністю метрик, серед яких коефіцієнт силуету, індекс Девіса–Болдіна, нормалізовані показники компактності та сепарованості, метрика найближчих сусідів, а також оцінка перекриття випуклих оболонок.

Отримані результати показують, що для графів із чіткою спільнотною структурою алгоритм KKNN демонструє стабільну перевагу над методом NeuLay-2 за всіма розглянутими метриками. Для більш складних випадків, зокрема несиметричних графів та графів без виражених спільнот, обидва підходи демонструють конкурентні результати, підкреслюючи різні аспекти структури графа: KKNN краще відображає глобальну компактність і узгодженість кластерів, тоді як NeuLay-2 у ряді випадків ефективніше зберігає локальні зв’язки між вершинами.

Таким чином, запропонований підхід є ефективним інструментом для

візуалізації графів із різною структурою та може бути використаний у задачах аналізу складних мереж, де важливим є коректне відображення спільнотної організації.

Загальна характеристика роботи: робота містить вступ, 3 частини, висновки та список використаної літератури. Кількість сторінок – 56, кількість ілюстрацій – 15, кількість використаних джерел – 37.

Ключові слова: граф, алгоритм, нейронні мережі, укладка, Камада-Каваї, силова модель, спільноти,.

ABSTRACT

In this master's thesis the problem of graph visualization with a focus on accurately representing community structure is considered. A method based on combining the Kamada–Kawai algorithm with neural network acceleration (KKNN) is proposed, enabling the construction of graph embeddings in Euclidean space while preserving both global and local structural properties. Particular attention is given to evaluating the quality of the obtained embeddings in the context of subsequent community detection.

The experimental study is conducted on several families of graphs, including symmetric graphs with clearly defined communities, asymmetric graphs with community structure, and graphs without well-defined communities. For each graph, multiple community detection algorithms are applied, including Louvain, Girvan–Newman, greedy modularity, and Label Propagation. The quality of the results is evaluated using a set of metrics, such as the silhouette coefficient, Davies–Bouldin index, normalized compactness and separability, the k-nearest neighbors metric, and convex hull overlap.

The results show that for graphs with well-defined community structure, the KKNN algorithm consistently outperforms NeuLay-2 across all considered metrics. For more complex cases, including asymmetric graphs and graphs without clear community structure, both approaches demonstrate competitive performance, highlighting different aspects of graph structure: KKNN better captures global compactness and cluster coherence, while NeuLay-2, in some cases, more effectively preserves local relationships between nodes.

In conclusion, the proposed approach is an effective tool for visualizing graphs with varying structural properties and can be applied to the analysis of complex networks where accurate representation of community structure is

essential.

Overall work characterization: the thesis comprises an introduction, three sections, conclusions, and a list of references. Number of pages – 56, number of illustrations – 15, number of utilized sources – 37.

Keywords: graph, algorithm, neural networks, layout, Kamada-Kawai, force model, communities, ….

ЗМІСТ

ВСТУП

Серед класичних підходів до побудови розкладок графів значне місце займають алгоритми, засновані на фізичних моделях, зокрема Камада-Каваї [1], який базується на моделі пружин та прагне мінімізувати енергію системи. Незважаючи на високу якість отримуваних візуалізацій, цей алгоритм має значну обчислювальну складність, що обмежує його застосування для великих графів. У зв’язку з цим актуальними є підходи до прискорення оптимізації, зокрема із використанням методів машинного навчання, таких як алгоритм NeuLay [2], запропонований Альбертом Барабаші, що застосовує нейромережеві методи для побудови розкладок графів.

У бакалаврській роботі [3] було запропоновано алгоритм KamadaNN, що поєднує переваги класичного методу Камада-Каваї та нейромережевого прискорення. Однак питання ефективності відображення спільнотної структури цим алгоритмом потребує подальшого дослідження та порівняння із сучасними підходами, зокрема NeuLay-2. Актуальність роботи зумовлена необхідністю створення методів візуалізації графів, що забезпечують баланс між якістю відображення структурних властивостей та обчислювальною ефективністю.

Важливою проблемою експериментальних досліджень є обмежена кількість генераторів графів із керованою спільнотною структурою. Більшість інструментів, зокрема в бібліотеці NetworkX, не дозволяють гнучко задавати характеристики спільнот або відтворювати необхідні сценарії. Тому в роботі розроблено власні генератори графів зі спільнотами, які забезпечують контроль над їхньою структурою та дозволяють більш точно оцінювати якість їх відображення.

ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1. Базові поняття [4]

Задача укладки графа в математиці та комп'ютерних науках полягає у визначенні оптимального розташування вершин і ребр графа на площині з метою задоволення певних критеріїв. Оптимальність у даному контексті може означати мінімізацію перетинів ребр, збереження пропорцій відстаней між вершинами, естетичний вигляд графічного представлення та інші параметри відповідно до конкретних вимог і контексту застосування.

Проблема інформативного зображення графів виникає у багатьох областях, включаючи комп'ютерну науку, телекомунікації, біологію, соціологію та інші. Наприклад, в інформаційних технологіях, укладка графів може бути потрібна для представлення мережі зв'язків між комп'ютерами або веб-сторінками. У біології граф може відображати взаємодії між різними видами організмів. Згадаємо визначення термінів, які нам будуть потрібні:

Вершина v – довільна точка в просторі.

Ребро $e = \{u, v\}$ – відрізок, який з'єднує пару вершин.

Граф $G = (V, E)$ складається з множини $V \neq \emptyset$, кожен елемент якої є вершиною, а також з множини E, кожен елемент якої є ребром.

Зважений граф – граф, де у кожного ребра є вага, яка виражається числом.

Матриця суміжності графу G це матриця $A = ||a||$ розміру $(n \times n)$, де $a_{ij} = a_{ji} = 1$, якщо між вершинами i та j є ребро, та $a_{ij} = a_{ji} = 0$, якщо ребра немає.

Шлях – послідовність вершин, в якій кожна наступна вершина з'єднана з попередньою ребром.

Зв'язний граф – граф, в якому між будь-якою парою вершин є шлях.

Спільнота – це підмножина вершин графа, для якої характерні щільні зв'язки в середині неї та відносно рідкі зв'язки з іншими вершинами графа.

Пошук спільнот – це задача знаходження розбиття вершин графа на групи (спільноти), такі що внутрішні зв'язки максимально щільні, міжгрупові – мінімальні.

Медоїд – це такий елемент спільноти, який має мінімальну сумарну відстань до всіх інших елементів цієї спільноти.

Також варто згадати декілька понять з машинного навчання [5]:

Нейрон – основна будівельна одиниця нейронних мереж, яка моделює нейрон в мозку. Вона приймає вхідні сигнали, зважує їх і використовує активаційну функцію для генерування вихідного сигналу.

Функція активації – функція, яка використовується в кожному штучному нейроні для обчислення вихідного значення на основі зважених вхідних сигналів.

Навчання – процес, під час якого модель нейронної мережі вдосконалюється шляхом зміни внутрішніх параметрів за допомогою тренувальних даних.

Функція втрат – це функція, яка вимірює різницю між прогнозованими значеннями моделі та фактичними значеннями в тренувальних даних.

Оптимізація – алгоритм, який використовується для налаштування параметрів моделі з метою мінімізації функції втрат.

2.2. Основні алгоритми

2.2.1. Алгоритм KamadaNN

У даній роботі використовується модифікація класичного алгоритму укладання графів Kamada–Kawai, розширена за допомогою нейронних мереж, що дістала назву KamadaNN [3]. Ідея і програмний код оптимізації за

допомогою нейронних мереж були взяті зі статті Барабаші Альберта [2], в якій він таким методом оптимізував алгоритм FDL [6].

В запропонованому раніше в роботі [3] алгоритмі KamadaNN поєднується фізична модель пружинної системи з можливостями графових нейронних мереж для ефективного відображення структури графа у евклідовому просторі.

Нехай задано неорієнтований граф $G = (V, E)$ з множиною вершин V та ребер E. Метою є знайти таке відображення вершин у d-вимірному просторі:

$$X = \{x_i \in \mathbb{R}^d \mid i \in V\},$$

що геометричні відстані між вершинами узгоджуються з їх топологічними відстанями у графі.

Алгоритм базується на мінімізації функції енергії:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{ij} \left(\left| p_i - p_j \right| - l_{ij} \right)^2. \quad (1)$$

В цій формулі d_{ij} – довжина мінімального шляху між вузлами i та j, $k_{ij} = \frac{K}{d_{ij}^2}$ – сила натяжіння пружини між вузлами, якщо вони з'єднані, або ж сила відштовхування, якщо ні (K – константа для масштабування зображення системи, я дала їй значення 1), $L = L_0 / \max_{i<j} d_{ij}$ (L_0 – довжина горизонталі екрану, її я також встановила одиницею) – оптимальна довжина одного ребра, $l_{ij} = L \times d_{ij}$ – «ідеальна» відстань між вузлами на зображенні, $\left| p_i - p_j \right|$ – поточна відстань між вузлами на зображенні (p_i та p_j мають координати, за допомогою яких рахується ця величина). Таким чином, кожна пара вершин моделюється як пружина, яка прагне до рівноважної довжини.

Для мінімізації загальної енергії системи використовується дві архітектури: NodeMLP та NeuLay-2.

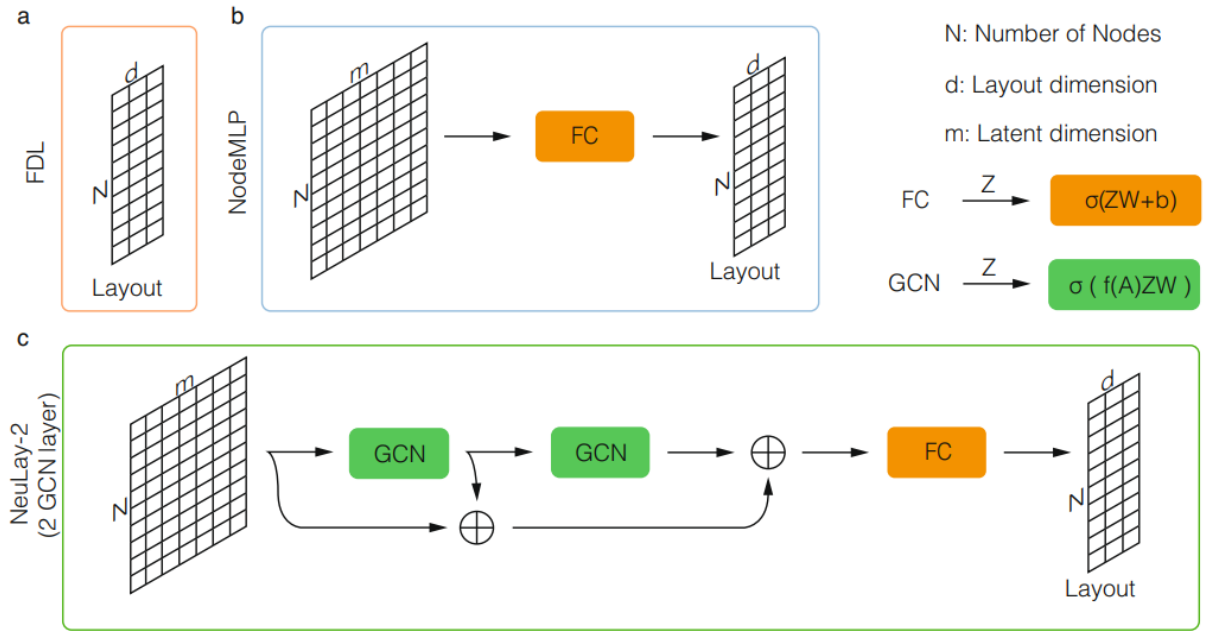
1) NodeMLP починає з випадкової Z – вбудови вершин в простір

$N \times h$ (де $N -$ кількість вершин). Далі алгоритм шукає оптимальну проекцію цього розташування в тривимірний простір $(N \times d, d = 3)$. Обчислення відбувається наступним чином: $X = \sigma(ZW + b)$, де σ – нелінійна функція, Z, W, b – параметри нейронної мережі, які треба натренувати.

2) NeuLay використовує GNN (Graph Neural Network), яка починає з випадкової Z–вбудови вершин в простір $N \times h$ (де $N -$ кількість вершин) і застосовує GCN (Graph Convolutional Network) шар для обчислення:

$$G_1 = \sigma\left(f(A)ZW^{(1)}\right), f(A) = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}, \tilde{A} = A + I, \tilde{D}_{ii} = \sum_j \tilde{A}_{ij} \qquad (2)$$

$\tilde{D}_{ij}$ – матриця ступенів, I – одинична матриця. G_1 є новим розташуванням вузлів у h_1 – вимірному просторі та має розміри $N \times h_1$.



(Рис. 1)

На рисунку 1 автори архітектур схематично зображують простори, з якими працюють алгоритми FDL, NodeMLP (є частиною NeuLay-2) та NeuLay-2, та показують, що застосовується при переході від однієї до іншої розмірності.

NeuLay-2 додатково застосовує ще один шар GCN, отримує розташування $G_2 = \sigma\big(f(A)ZW^{(2)}\big)$ розміру $N \times h_2$ у h_2–вимірному просторі. Далі автори пропонують об’єднати всі шари наступним чином: $G = (Z|G_1|G_2)$, отримуючи укладку в $(h + h_1 + h_2)$–вимірному просторі.

В кінці NodeMLP застосовується для отриманої $(h + h_1 + h_2)$–вимірної укладки графу, щоб отримати проекцію G у тривимірний простір – результат. Множина параметрів для тренування у NeuLay-2:

$$\theta = \{Z, W^{(1)}, W^{(2)}, W, b\}. \quad (3)$$

Нижче наведено схему роботи алгоритму KamadaNN:

Ініціалізація моделі в багато-вимірному просторі: 1. створення матриці суміжності 2. випадкове розміщення графу в просторі 3. створення матриці відстаней методом Флойда 4. створення матриці обернених квадратичних відстаней 5. масштабування
↓
Запуск першого GCN з функцією втрат Камади-Каваї
↓
Запуск другого GCN з функцією втрат Камади-Каваї
↓
Злиття трьох отриманих шарів графу в один шар за допомогою методу NodeMLP
↓
Проекція отриманого розташування з багатовимірного простору у дво/тривимірний
↓
Відображення укладки графу залежно від його розмірності

2.2.2. Алгоритми пошуку спільнот

Для розбиття графу на спільноти було використано 4 найвідоміші

алгоритми, кожен з яких працює зі структурою графу, а не його розташуванням в просторі.

2.2.2.1. Жадібна оптимізація модулярності

Жадібний алгоритм максимізації модулярності [7] ґрунтується на поетапному об'єднанні вершин у спільноти з метою максимізації функції модулярності.

Модулярність визначається як:

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j), \text{ де:} \quad (4)$$

- A_{ij} – елемент матриці суміжності,
- k_i, k_j – степені вершин,
- m – кількість ребер у графі,
- c_i – номер спільноти вершини i,
- $\delta(c_i, c_j)$ – індикатор належності до однієї спільноти.

Жадібний алгоритм починає з того, що кожна вершина утворює окрему спільноту. На кожному кроці виконується об'єднання двох спільнот, яке дає найбільше збільшення модулярності ΔQ. Процес триває, поки $\Delta Q > 0$.

2.2.2.2. Алгоритм поширення міток (Label Propagation)

Алгоритм поширення міток [8] є ітеративним методом, який не потребує попереднього задання кількості спільнот. Нехай $c_i^{(t)}$ – мітка вершини i на ітерації t. Оновлення мітки відбувається за правилом:

$$c_i^{(t+1)} = \arg\max_c \left|\{j \in \mathcal{N}(i): c_j^{(t)} = c\}\right|, \quad (5)$$

де $\mathcal{N}(i)$ – множина сусідів вершини i. Тобто вершина приймає мітку, яка найчастіше зустрічається серед її сусідів. Процес повторюється до досягнення стаціонарного стану, коли мітки перестають змінюватися.

2.2.2.3. Лувенський алгоритм (Louvain)

Лувенський алгоритм [9] є ітеративним методом максимізації модулярності, що працює у два етапи. На першому етапі для кожної вершини оцінюється зміна модулярності при перенесенні вершини iдо спільноти C:

$$\Delta Q = \left[\frac{\sum_{in}+2k_{i,in}}{2m} - \left(\frac{\sum_{tot}+k_i}{2m}\right)^2\right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right], \text{ де:} \quad (6)$$

- $\sum_{in}$ – сумарна вага ребер всередині спільноти,
- $\sum_{tot}$ – сумарна вага ребер, інцидентних вершинам спільноти,
- k_i – степінь вершини i,
- $k_{i,in}$ – сумарна вага ребер між i і спільнотою C.

На другому етапі будується новий граф, у якому вершини відповідають знайденим спільнотам, і процес повторюється.

2.2.2.4. Алгоритм Гірван-Н'юмена

Алгоритм Girvan–Newman [10] базується на послідовному видаленні ребер з найбільшою посередницькою центральністю. Посередницька центральність ребра визначається як:

$$g(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}, \text{ де:} \quad (7)$$

- σ_{st} – кількість найкоротших шляхів між вершинами sі t,
- $\sigma_{st}(e)$ – кількість таких шляхів, що проходять через ребро e.

На кожному кроці:

1. обчислюється $g(e)$для всіх ребер
2. видаляється ребро з максимальним значенням
3. процес повторюється.

В результаті граф розпадається на компоненти зв'язності, які

інтерпретуються як спільноти.

2.3. Метрики для оцінки якості візуалізації спільнот

Для оцінки якості відображення спільнот було використано 6 різних метрик. Слід зазначити, що деякі розглянуті метрики (індекс Девіса–Болдіна [11], коефіцієнт силуету [12], компактність та сепарація) за своєю суттю є метриками якості кластеризації у випадку, коли еталонне розбиття на кластери відсутнє. Кожен кластер містить точки, що відповідають вершинам однієї спільноти, а метрики дозволяють оцінити їх компактність та ступінь відокремленості від інших кластерів. Також в роботі було використано дві геометричні метрики: перекриття опуклих оболонок [13] та узгодженість найближчих сусідів [14]. Таким чином, оцінювання якості виконується не безпосередньо на структурі графа, а на геометричному розташуванні вершин у просторі.

Нехай задано:

- $G = (V, E)$ – неорієнтований граф;
- $n = |V|$ – кількість вершин;
- K – кількість спільнот;
- $x_i \in \mathbb{R}^d$— координати вершини iу просторі;
- $X = \{x_1, x_2, \dots, x_n\}$;
- $C_1, C_2, \dots, C_K \subset V$ – спільноти;
- $C(i) \in \{1, \dots, K\}$ – індекс спільноти вершини i;
- $n_k = |C_k|$ – кількість вершин у спільноті C_K;

У всіх метриках використовується тільки евклідова відстань у просторі координат:

$$d_E(i,j) = \| x_i - x_j \|_2 \quad (8)$$

Також ми будемо використовувати центри спільнот:

$$\mu_k = \frac{1}{n_k} \sum_{i \in C_k} x_i, \tag{9}$$

внутрішню розсіюваність спільноти:

$$\sigma_k = \frac{1}{n_k} \sum_{i \in C_k} d_E(i, \mu_k), \text{ де} \tag{10}$$

$$d_E(i, \mu_k) = \|x_i - \mu_k\|_2, \tag{11}$$

та відстані між спільнотами:

$$\delta_{kl} = \|\mu_k - \mu_l\|_2. \tag{12}$$

Знайдемо максимальну відстань між вершинами графа для нормалізації власних метрик, щоб масштабування графу не впливало на результат:

$$diameter = \max_{i \neq j} \left(d_E(i, j)\right). \tag{13}$$

1) Індекс Девіса–Болдіна [11]

$$DB = \frac{1}{K} \sum_{k=1}^{K} \max_{\ell \neq k} \left(\frac{\sigma_k + \sigma_\ell}{\delta_{k\ell}}\right) \tag{14}$$

Для кожної спільноти шукаємо найбільш схожу (ту, що перекривається найбільше) іншу спільноту, потім усереднюємо цей “найгірший випадок”.

Інтерпретація:

- $DB \to 0$ – дуже добре,
- $DB \approx 1$ – нормально,
- $DB > 1.5$ – погано.

2) Silhouette Score [12]

Допоміжні величини для вершини *i*:

- $a(i) = \frac{1}{n_{C(i)}-1} \sum_{j \in C(i), j \neq i} d_E(i, j)$ – середня відстань до своєї спільноти;
- $b(i) = \min_{\ell \neq C(i)} \left(\frac{1}{n_\ell} \sum_{j \in C_\ell} d_E(i, j)\right)$ – середня відстань до іншої спільноти;

- $s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}$ – формула силуету.

Загальний показник для укладки:

$$S = \frac{1}{n}\sum_{i=1}^{n} s(i) \tag{15}$$

Таким чином ми для кожної вершини перевіряємо чи вона ближче до своєї спільноти, чи до іншої.

Чим більше значення S і ближче до 1 тим краще.

3) Compactness (нормалізована)

$$Compactness = \frac{\frac{1}{K}\sum_{k=1}^{K}\sigma_k}{diameter} \tag{16}$$

Нормалізована компактність показує наскільки спільноти "розмиті" відносно масштабу всієї укладки. Чим менше значення $Compactness$, тим компактніші спільноти.

4) Separation (нормалізована)

$$Separation = \frac{\frac{2}{K(K-1)}\sum_{i<j} d_E(i,j)}{diameter}. \tag{17}$$

Нормалізована сепарація показує наскільки далеко один від одного розташовані спільноти. Чим більше значення $Separation$, тим далі спільноти одна від одної.

5) Перекриття випуклих оболонок (Convex Hull Overlap)

Для оцінювання просторового розділення спільнот у побудованій візуалізації введено геометричну метрику, що базується на випуклих оболонках спільнот [13]. Для кожної спільноти C_k будується її випукла

оболонка у тривимірному просторі на основі координат вершин. Випукла оболонка є мінімальним опуклим багатогранником, що містить усі вершини спільноти. Далі для кожної пари спільнот (C_i, C_j) оцінюється об'єм їх перетину: $\mathrm{Vol}(H_i \cap H_j)$, де H_i, H_j— випуклі оболонки відповідних спільнот. Оскільки точне обчислення об'єму перетину в тривимірному просторі є обчислювально складним, у роботі використовується наближений метод Монте-Карло [15]. Для цього генерується велика кількість випадкових точок у прямокутному паралелепіпеді, що обмежує всю укладку, і визначається частка точок, які належать одночасно двом оболонкам. Загальна міра перекриття визначається як:

$$\mathrm{Overlap} = \frac{\sum_{i<j} \mathrm{Vol}(H_i \cap H_j)}{\mathrm{Vol}(H_{\mathrm{layout}})}, \text{ де } H_{\mathrm{layout}} \text{ – випукла оболонка всієї укладки.} \quad (18)$$

Чим менше і ближче до 0 значення Overlap тим краще.

6) Метрика узгодженості найближчих сусідів (kNN Same-Community Ratio)

Для оцінки локальної структури укладки введено метрику, що базується на аналізі найближчих сусідів [14]. Для кожної вершини v_iвизначається множина її kнайближчих сусідів у просторі укладки $\mathcal{N}_k(i)$. Далі обчислюється частка сусідів, що належать до тієї ж спільноти:

$$r_i = \frac{|\{j \in \mathcal{N}_k(i) : c_j = c_i\}|}{k} \quad (19)$$

Загальне значення метрики визначається як середнє по всіх вершинах:

$$\mathrm{kNN\_ratio} = \frac{1}{N}\sum_{i=1}^{N} r_i \quad (20)$$

Чим більше і ближче значення kNN_ratio до 1 тим краще.

ЕКСПЕРИМЕНТАЛЬНІ ГРАФИ ТА СХЕМА ЕКСПЕРИМЕНТУ

3.1. Графи зі структурою спільнот

3.1.1. Відомі графи зі структурою спільнот

Для тестування алгоритмів та проведення експериментів в роботі було використано декілька графів з Python бібліотеки NetworkX [16]. Нижче наведено їх короткий опис:

1) Karate Club graph

Граф карате-клубу [17] є класичним прикладом реального соціального графа, що відображає взаємодії між членами невеликої спільноти. Він містить 34 вершини та має природний поділ на дві спільноти, що виникли внаслідок конфлікту. Часто використовується як еталон для задач виявлення спільнот і тестування алгоритмів.

2) Stochastic block model graph

Граф стохастичної блочної моделі (SBM) [18] генерується випадковим чином із заданою структурою спільнот. Ймовірність появи ребра залежить від належності вершин до тих самих або різних блоків. Є однією з основних моделей для дослідження алгоритмів виявлення спільнот. Користувач має вказати кількість вершин в блоках та матрицю ймовірностей появи ребр між вершинами, які належать різним блокам.

3) Caveman graph

Граф печерної людини [19] складається з кількох повних підграфів (печер), які слабо або взагалі не з’єднані між собою. Така структура моделює чітко виражені спільноти з високою внутрішньою щільністю та мінімальними зв’язками між ними. Цей граф є окремим випадком графу стохастичної блочної моделі (якщо при генерації SBM вказати ймовірність появи ребр між вершинами одного блоку 1). Користувач має вказати кількість печер, кількість

вершин в печерах та ймовірності появи ребр між вершинами, які належать різним печерам.

3.1.2. Генератори для створення графів зі спільнотами

Для більшої різноманітності експериментів було створено два загальних генератори графів зі спільнотами.

Метод graph_lifting

Даний метод приймає на вхід граф G, параметри n та m, а також ймовірності p_{in}, p_{out} і p_{global}, і будує новий граф за наступним принципом:

- Кожна вершина початкового графа Gзамінюється на спільноту, що складається з випадкової кількості вершин k, де $k \in [n, m]$. Усередині кожної спільноти вершини з'єднуються між собою з імовірністю p_{in}.
- Для вершин, що належать до різних спільнот, ребра додаються залежно від структури графа G: якщо відповідні спільноти відповідають сусіднім вершинам у графі G, то ребра між ними створюються з імовірністю p_{out}, а також обов'язково зв'язуються медоїди цих спільнот. В іншому випадку ребра додаються з імовірністю p_{global}.

Таким чином, побудований граф зберігає загальну структуру початкового графа G, але кожна його вершина замінюється на підграф (спільноту). В такому випадку ми говоримо, що граф G розтягнутий спільнотами.

Якщо ймовірності $p_{out} = 0$ та $p_{out} = 0$, граф все одно буде зв'язним, бо медоїди всіх спільнот з'єднані відповідно до структури початково графа.

В експериментах цей метод було застосовано до симетричних графів, щоб отримати структурно симетричні графи з чіткими спільнотами. Нижче наведено перелік використаних симетричних графів:

1) Grid graph

Граф решітки [20] є регулярним графом, вершини якого утворюють решітку в просторі довільної розмірності. Кожна вершина з'єднана зі своїми сусідами вздовж координатних осей. Така структура є узагальненням двовимірної решітки на багатовимірний випадок і не має вираженої спільнотної структури. Користувач має вказати розмірність решітки.

2) Star graph

Граф зірки [21] – це граф із однією центральною вершиною, яка з'єднана з усіма іншими вершинами. Інші вершини не мають зв'язків між собою. Така структура є прикладом сильно централізованої мережі та не містить спільнот. Користувач має вказати кількість вершин, з якими буде з'єднана центральна.

3) Icosahedral graph

Граф ікосаедру [22] – це регулярний граф, що відповідає структурі правильного ікосаедра. Він характеризується високою симетрією та однаковими степенями всіх вершин. Використовується для тестування алгоритмів укладання графів завдяки своїй геометричній регулярності.

Метод graph_lifting_with_template

Цей метод працює за схожим з попереднім методом принципом, але в ньому вершини одного графа замінюються на інший граф.

Даний метод приймає на вхід графи $G1$, $G2$, а також ймовірності p_{out}, p_{global}. Фінальний граф будується за наступним принципом:

- Кожна вершина початкового графа $G1$ замінюється на граф $G2$.
- Для вершин, що належать до різних копій графу $G2$, ребра додаються залежно від структури графа $G1$: якщо відповідні копії відповідають сусіднім вершинам у графі $G1$, то ребра між ними створюються з імовірністю p_{out}. В іншому випадку ребра додаються з імовірністю p_{global}.

Таким чином, побудований граф зберігає загальну структуру початкового графа $G1$, але кожна його вершина замінюється на підграф $G2$. В такому випадку ми говоримо, що граф $G1$ розтягнутий копіями графа $G2$.

В роботі цей метод було застосовано для побудови графу карате клубу, кожна вершина якого є карате клубом.

Генератор football_communities_graph

Цей метод є окремим випадком методу graph_lifting, якщо в той надати граф усіченого ікосаедру (футбольного м'яча) та $p_{global} = 0$.

Метод приймає на вхід граф параметри n та m, а також ймовірності p_{in} та p_{out}, і будує новий граф за наступним принципом:

- Кожна вершина графа усіченого ікосаедру замінюється на спільноту, що складається з випадкової кількості вершин k, де $k \in [n, m]$. Усередині кожної спільноти вершини з'єднуються між собою з імовірністю p_{in}.
- Для вершин, що належать до різних спільнот, ребра додаються залежно відповідно до структури футбольного м'яча: якщо відповідні спільноти відповідають сусіднім вершинам у графі, то ребра між ними створюються з імовірністю p_{out}. В іншому випадку ребра не додаються.

Даний генератор планується додавати в бібліотеки генерації графів.

Також в роботі були використані графи, які не мають чітких спільнот:

1) Random geometry graph

Граф випадкової геометрії [23] будується шляхом випадкового розміщення вершин у просторі та з'єднання тих пар, відстань між якими менша за заданий поріг. Такий граф відображає просторові мережі та може мати локально щільні області, що іноді інтерпретуються як спільноти. Користувач має вказати кількість вершин та поріг для з'єднання вершин.

2) Watts-Strogatz graph

Граф Воттса-Строгаца [24] – це модель генерації випадкових мереж типу «малого світу» (small-world networks), яка поєднує властивості регулярних та випадкових графів. Вона була запропонована для моделювання реальних систем, у яких одночасно спостерігаються високий рівень локальної кластеризації та малі середні відстані між вершинами.

Побудова графа відбувається у кілька етапів:

1. Початковий регулярний граф

   Створюється кільцевий граф з n вершинами, де кожна вершина з'єднана з k найближчими сусідами (зазвичай $k/2$ зліва і $k/2$ справа). Таким чином формується регулярна структура з високою кластеризацією.

2. Перепідключення ребер (rewiring)

   Кожне ребро графа з певною ймовірністю p перепідключається: один його кінець залишається фіксованим, а інший приєднується до випадково вибраної вершини графа. При цьому зазвичай уникають петель і кратних ребер.

   Параметри моделі:

   - n – кількість вершин графа;
   - k – кількість сусідів кожної вершини у початковому регулярному графі (має бути парним);
   - p – ймовірність перепідключення кожного ребра.

3) Barabasi Albert graph

Модель графа Барабаші [25] (Barabasi – Albert model) — це алгоритм генерації випадкових безмасштабних мереж (scale-free networks), який базується на двох основних механізмах: зростанні та переважному приєднанні. Ця модель часто використовується для моделювання соціальних мереж, інтернет-топологій та інших систем, де нові елементи переважно

приєднуються до вже популярних вузлів. Користувач задає загальну кількість вершин графу – n, а також параметр m – при створені кожної нової вершини графу, вона з'єднується з m вже створеними вершинами.

Також за допомогою вище наведених методів було створено наступний граф: граф Barabasi, розтягнутий графом Watts-Strogatz, додатково розтягнутий маленькими спільнотами.

4) Random lobster graph

Граф лобстера [26] це такий граф, який після видалення всіх листків (вершин степеня 1) він перетворюється на гусеничний граф, тобто такий, у якому після повторного видалення листків залишається простий шлях.

У програмній реалізації побудова графа відбувається ітеративно з використанням заданих користувачем імовірностей p_1 та p_2, які визначають додавання ребер відповідно на першому та другому рівнях до хребта заданої довжини n. Таким чином, граф формується поступово з локальною випадковістю на кожному рівні, а не вибирається рівноймовірно з множини всіх можливих графів-лобстерів.

3.2. Опис доданих методів візуалізації

У рамках даної роботи було розширено набір інструментів укладання графів (на основі алгоритму KamadaNN або іншого), для забезпечення можливості аналізу та візуалізації спільнотної структури.

На першому етапі для заданого графа будується укладка за допомогою алгоритму KamadaNN, що є поєднанням алгоритму Камада–Каваї та нейромережевого прискорення. Результатом цього етапу є координати вершин у тривимірному просторі.

Після побудови укладки виконується пошук спільнот у графі (не за розташуванням вершин, а за структурою графу). Для цього застосовуються

чотири різні алгоритми:

- Лувенський алгоритм;
- жадібна оптимізація модулярності;
- алгоритм поширення міток;
- алгоритм Гірван–Н’юмана.

Кожен із цих методів формує власне розбиття графа на спільноти, яке задається масивом міток вершин.

Для забезпечення відтворюваності експериментів результати обчислень зберігаються у файлах формату JSON:

- окремо зберігається укладка графа (координати вершин та структура ребер);
- окремо зберігаються мітки вершин для кожного методу розбиття на спільноти.

Після цього для кожної пари “укладка – розбиття на спільноти” обчислюються метрики якості. У роботі використовуються як класичні метрики кластеризації, так і запропоновані геометричні метрики:

- індекс Девіса–Болдіна;
- коефіцієнт силуету;
- нормалізована компактність;
- нормалізована сепарованість;
- перекриття випуклих оболонок (обчислене методом Монте-Карло [15]);
- коефіцієнт найближчих сусідів (kNN).

На завершальному етапі використовується модифікований метод візуалізації, який відображає результати кластеризації у просторі укладки. Вершини графа фарбуються у різні кольори відповідно до їх належності до спільнот, визначених обраним алгоритмом. Медоїди спільнот обчислюються

на основі розташування вершин в просторі, додатково виділяються розміром, що дозволяє візуально ідентифікувати центральні вершини кожної спільноти. Окрім графічного відображення, для кожного методу розбиття на спільноти на екран виводяться:

- кількість знайдених спільнот;
- значення всіх обчислених метрик якості.

Таким чином, запропоновані інструменти дозволяють поєднати геометричну візуалізацію графа з аналізом його спільнотної структури та забезпечує зручний інструмент для порівняння різних алгоритмів укладання.

3.3. Схема роботи програми

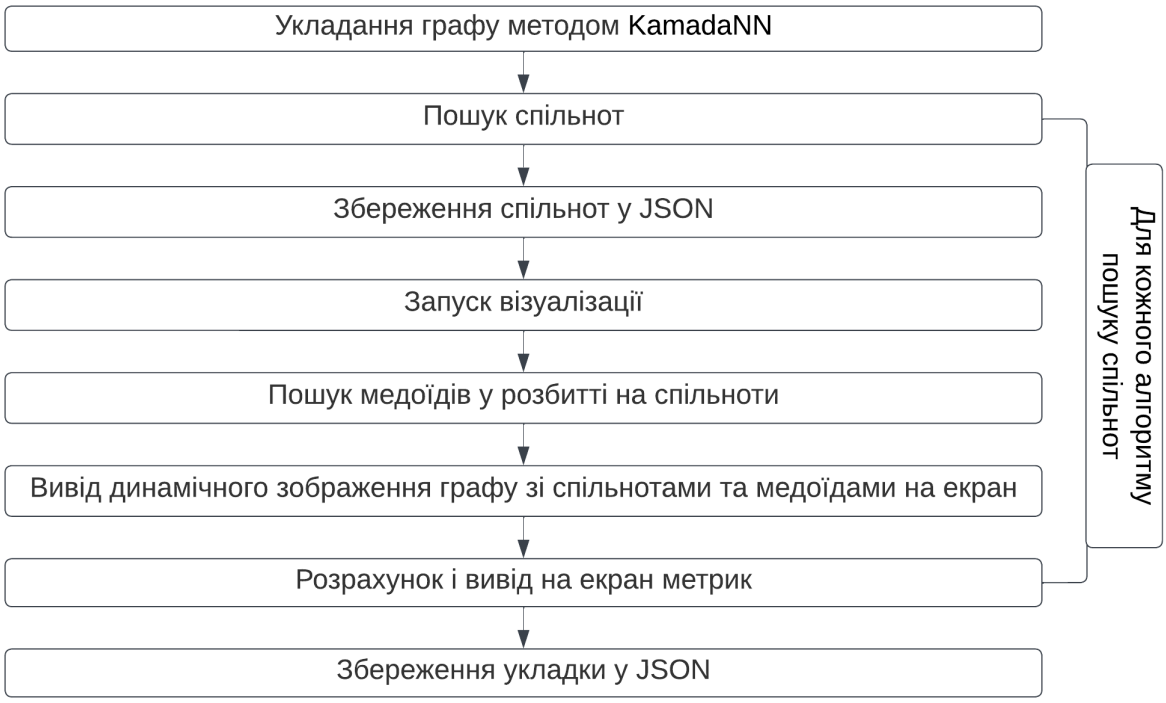


Рис. 2: Схема роботи доданих інструментів

На рис. 2 зображено схему роботи доданих ісntсрументів.

ЕКСПЕРИМЕНТ

Основною задачею дослідження було порівняти алгоритми KamadaNN та NeuLay-2 на різноманітних графах, використовуючи різноманітні алгоритми пошуку спільнот та метрики для оцінки якості зображення спільнот. Для зручності графи, на яких проводились експерименти, було поділено на три сімейства:

1. симетричні графи з чіткою структурою спільнот;
2. несиметричні графи з чіткою структурою спільнот;
3. графи без чіткої структури спільнот.

Нижче наведено результати експериментів для кожного сімейства графів.

4.1. Симетричні графи з чіткою структурою спільнот

Серед симетричних графів з чіткою структурою спільнот було розглянуто наступні графи:

- футбольний м’яч, розтягнутий спільнотами;
- 3D-решітка, розтягнута спільнотами;
- зірка, розтягнута спільнотами;
- ікосаедр, розтягнутий спільнотами.

1) Football with communities/FBCom20

Граф футбольного м’яча (урізаного ікосаедру), розтягнутого спільнотами, було побудовано за допомогою football_communities_graph та graph_lifting генераторів, про які було детально розказано в підрозділі 3.1.2. Спочатку було створено граф футбольного м’яча, викликавши football_communities_graph з параметрами $n = 1, p_{in} = 1, p_{out} = 1$. Потім цей граф було розтягнуто за допомогою graph_lifting з параметрами $n = 20, m = 20, p_{in} = 0.35, p_{out} = 0.01, p_{global} = 0$. Саме такий спосіб створення графу

було використано для того, щоб медоїди спільнот були зв'язані між собою (метод football_communities_graph цього не гарантує).

Таким чином було отримано граф футбольного м’яча з чітко вираженими спільнотами.

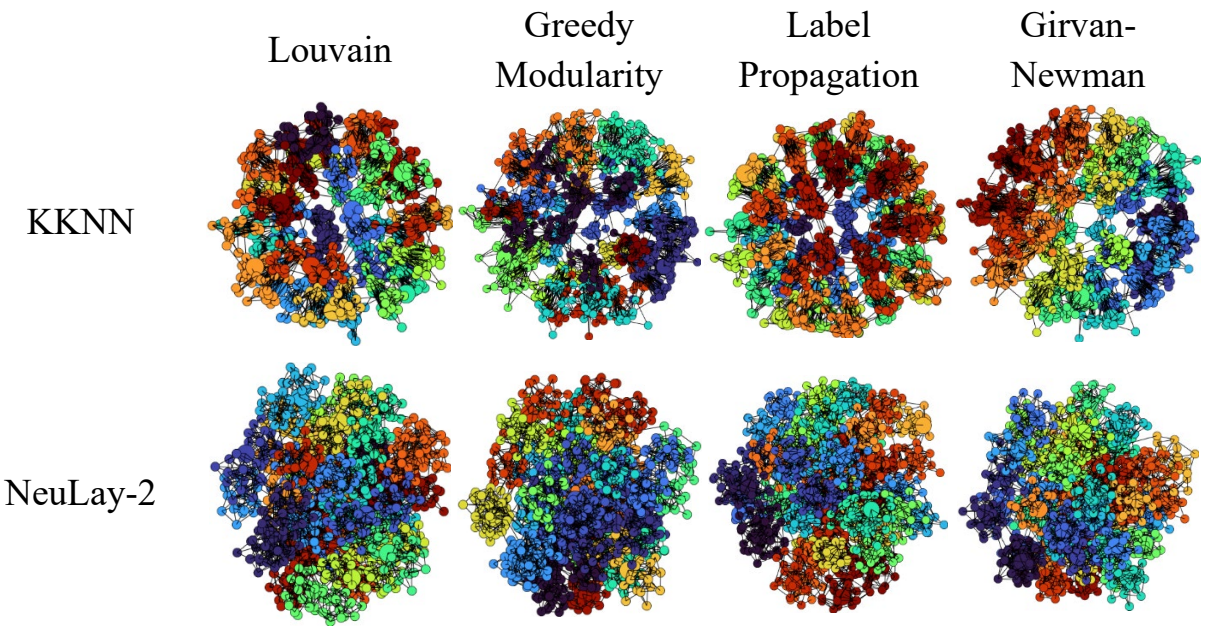


Рис. 3: укладання графа FBCom20 алгоритмами KamadaNN та NeuLay-2

На рис. 3 зображено порівняння укладок графа урізаного ікосаедра зі спільнотами алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально видно, що алгоритм NeuLay-2 погано передає структуру м'яча, змішуючи спільноти, в той час як KamadaNN гарно відокремлює спільноти і розташовує їх в просторі з мінімаліньми перетинами.

2) 3D-grid with communities/Cube4Com

Цей граф було отримано за допомогою розтягування 3D-решітки $4 \times 4 \times 4$ методом graph_lifting з параметрами $n = 20, m = 20, p_{in} = 0.35, p_{out} = 0.005, p_{global} = 0$.

3D-решітка була побудовано за допомогою генератора grid_graph з

Python-бібліотеки Networx [27], про який детально було розказано у підрозділі 3.1.2. Таким чином було отримано граф куба $4 \times 4 \times 4$ з чітко вираженими спільнотами.

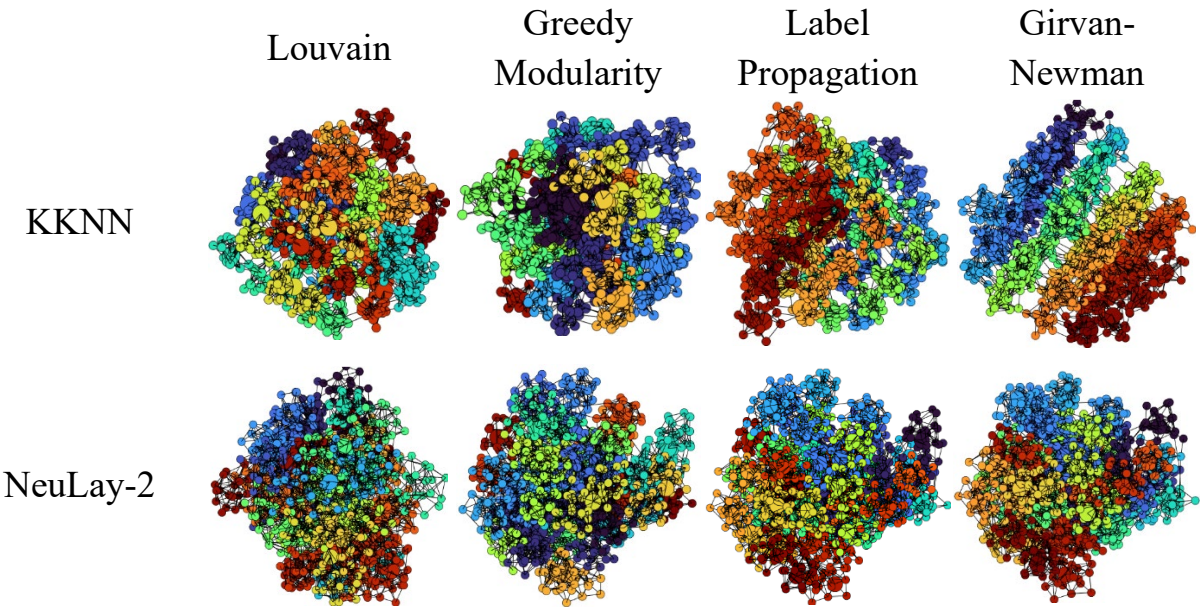


Рис. 4: укладання графа Cube4Com алгоритмами KamadaNN та NeuLay-2

На рис. 4 зображено порівняння укладок графа куба зі спільнотами алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально видно, що алгоритм NeuLay-2 погано передає структуру куба, змішуючи спільноти, в той час як KamadaNN гарно відокремлює спільноти і розташовує їх в просторі з мінімаліньми перетинами.

3) Star with communities/Star10Com

Цей граф було отримано за допомогою розтягування графа зірки з однією центральною вершиною і 10-ма доєднаними до неї вершинами, методом graph_lifting з заданими параметрами $n = 20, m = 50, p_{in} = 0.35, p_{out} = 0.005, p_{global} = 0$.

Зірка з 10-ти променів була побудована за допомогою генератора star_graph з Python-бібліотеки Networx [28], про який детально було розказано

у підрозділі 3.1.2.

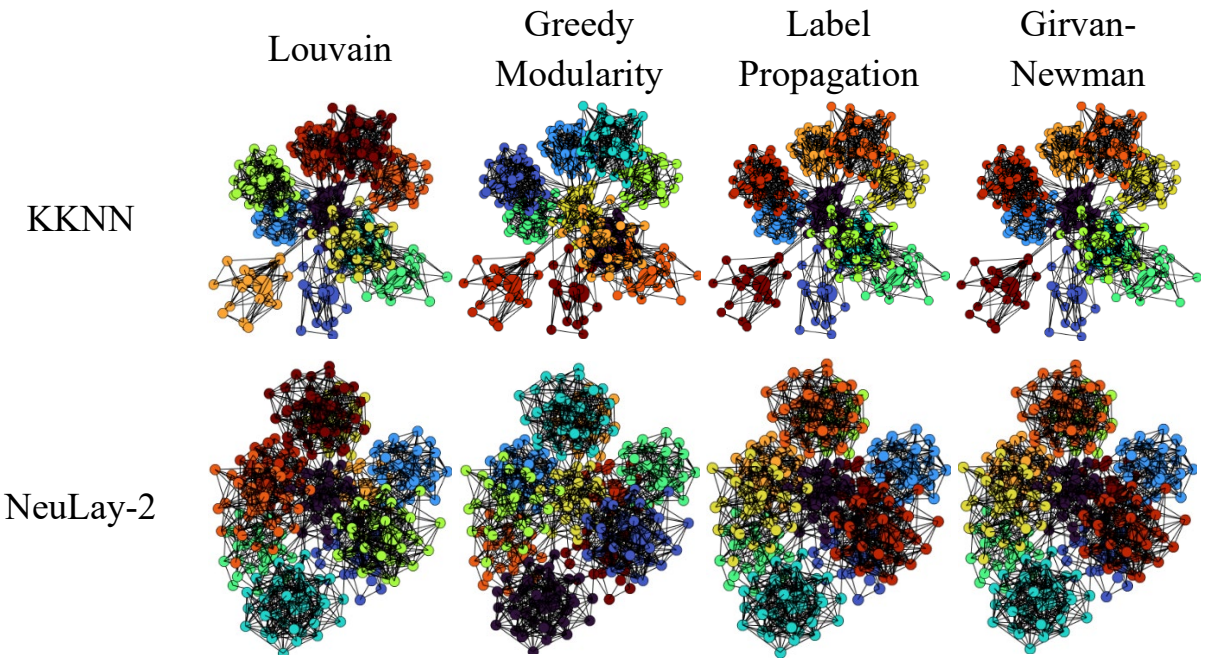


Рис. 5: укладання графа Star10Com алгоритмами KamadaNN та NeuLay-2

На рис. 5 зображено порівняння укладок графа зірки зі спільнотами алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально видно, що алгоритм NeuLay-2 погано передає структуру зірки, розташовуючи вершини в спільнотах не дуже компактно та погано сепаруючи спільноти, в той час як KamadaNN гарно відокремлює спільноти і розташовує їх в просторі без перетинів.

4) IcosahedralCom2050

Цей граф було отримано за допомогою розтягування графа ікосаедра методом graph_lifting з параметрами $n = 20, m = 50, p_{in} = 0.35, p_{out} = 0.005, p_{global} = 0$.

Граф ікосаедру було побудовано за допомогою генератора icosahedral_graph з Python-бібліотеки Networx [29], про який детально було розказано у підрозділі 3.1.2.

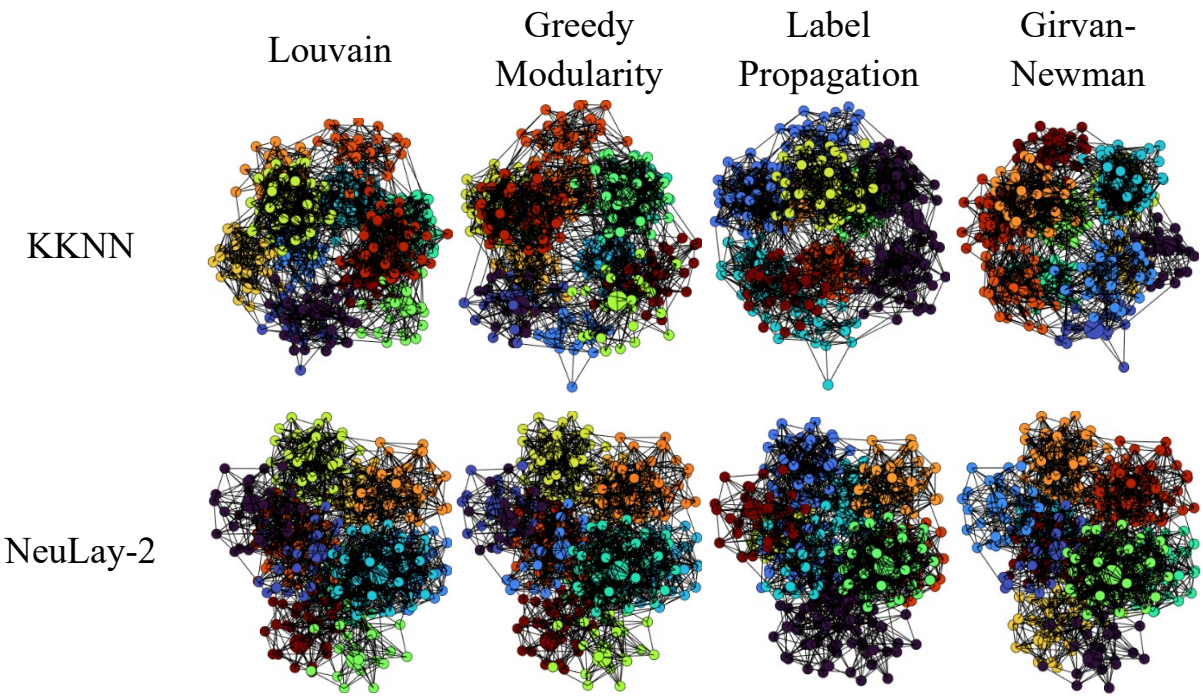


Рис. 6: укладання графа IcosahedralCom2050 алгоритмами KamadaNN та NeuLay-2

На рис. 6 зображено порівняння укладок графа ікосаедра зі спільнотами алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально видно, що алгоритм NeuLay-2 погано передає структуру ікосаедру, розташовуючи вершини в спільнотах не дуже компактно та погано сепаруючи спільноти, в той час як KamadaNN гарно відокремлює спільноти і розташовує їх в просторі з мінімальними перетинами.

Для всіх перерахованих графів було пораховано наступні метрики:

- індекс Девіса-Болдіна;
- коефіцієнт силуету;
- нормалізована компактність спільнот;
- нормалізована сепарованість спільнот;
- коефіцієнт перекриття випуклих оболонок;
- узгодженість найближчих сусідів.

	Community Number							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	Real	Received	Real	Received	Real	Received	Real	Received
FBCom20	60	29	60	21	60	70	60	60
Cube4Com	64	31	64	23	64	71	64	64
Star10Com	11	11	11	11	11	11	11	11
IcosahedralCom2050	12	12	12	8	12	13	12	12
	Davis-Boldin index							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
FBCom20	1.255	2.875	1.397	3.295	1.579	1.639	1.579	1.639
Cube4Com	1.325	3.352	1.571	6.063	1.575	1.729	1.575	1.729
Star10Com	0.951	1.049	0.951	1.049	0.951	1.049	0.951	1.049
IcosahedralCom2050	1.075	1.452	1.362	2.295	2.409	1.704	2.409	1.704
	Silhouette							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
FBCom20	0.337	0.096	0.247	0.025	0.355	0.245	0.407	0.292
Cube4Com	0.317	0.071	0.14	-0.055	0.401	0.234	0.447	0.262
Star10Com	0.51	0.421	0.51	0.421	0.51	0.421	0.51	0.421
IcosahedralCom2050	0.419	0.307	0.296	0.21	0.387	0.289	0.419	0.307
	Normalised Compactness							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
FBCom20	0.126	0.169	0.148	0.202	0.077	0.093	0.082	0.098
Cube4Com	0.116	0.167	0.136	0.186	0.069	0.097	0.07	0.099
Star10Com	0.147	0.164	0.147	0.164	0.147	0.164	0.147	0.164
IcosahedralCom2050	0.163	0.181	0.201	0.226	0.163	0.179	0.163	0.181
	Normalised Separation							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
FBCom20	0.496	0.407	0.497	0.406	0.499	0.44	0.499	0.429
Cube4Com	0.421	0.389	0.433	0.392	0.426	0.421	0.429	0.415
Star10Com	0.469	0.447	0.469	0.447	0.469	0.447	0.469	0.447
IcosahedralCom2050	0.469	0.425	0.448	0.403	0.463	0.425	0.469	0.425
	Convex Hull Overlap							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
FBCom20	0	0.053	0	0.241	0	0.001	0	0
Cube4Com	0	0.099	0.005	0.323	0.002	0.004	0	0.001
Star10Com	0	0	0	0	0	0	0	0
IcosahedralCom2050	0	0	0.001	0.055	0.011	0.007	0	0
	K Near Neighbours							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
FBCom20	0.982	0.867	0.978	0.862	0.911	0.798	0.96	0.837
Cube4Com	0.969	0.814	0.969	0.822	0.899	0.755	0.945	0.785
Star10Com	0.98	0.975	0.98	0.975	0.98	0.975	0.98	0.975
IcosahedralCom2050	0.95	0.886	0.962	0.893	0.912	0.862	0.95	0.886

Таблиця 1: результати експериментів для сімейства симетричних графів з чіткими спільнотами

У таблиці 1 наведено результати обчислення сукупності метрик якості для графів із сімейства симетричних структур із чітко вираженими спільнотами. Аналіз отриманих значень демонструє стійку та узгоджену перевагу запропонованого алгоритму KamadaNN порівняно з методом NeuLay-2. Зокрема, для всіх розглянутих алгоритмів виявлення спільнот (Louvain, greedy modularity, label propagation та Girvan-Newman) спостерігається покращення відповідних показників якості у випадку використання KamadaNN -укладки.

Отримані результати свідчать про те, що геометричне представлення графа, сформоване за допомогою KamadaNN, більш якісно відображає внутрішню структуру спільнот, забезпечуючи їх кращу просторову відокремленість та компактність. Це, у свою чергу, позитивно впливає на значення як внутрішньоспільнотих, так і міжспільнотних метрик (зокрема, компактності, сепарованості, силуетного коефіцієнта та інших похідних характеристик) і підтверджує припущення, зроблені під час візуальної оцінки укладок графів (див. підрозділ 4.1), побудованих алгоритмами KamadaNN та NeuLay-2.

Важливо підкреслити, що перевага KamadaNN має системний характер: вона спостерігається не для окремих випадків чи окремих метрик, а є узгодженою для всіх досліджуваних конфігурацій та методів пошуку спільнот. Така стабільність результатів дозволяє зробити висновок про вищу здатність алгоритму KamadaNN зберігати та підсилювати структурні особливості симетричних графів із вираженою спільнотною організацією.

Отже, проведене експериментальне дослідження підтверджує, що використання KamadaNN є більш ефективним підходом до побудови вкладень для симетричних графів зі спільнотами порівняно з NeuLay-2, що відкриває перспективи його подальшого застосування в задачах аналізу складних мереж

та візуалізації структур із чіткою модульною організацією.

4.2. Несиметричні графи з чіткою структурою спільнот

Серед несиметричних графів з чіткою структурою спільнот було розглянуто наступні графи:

- граф карате клубу;
- граф печерної людини;
- граф стохастичної блочної моделі;
- випадковий граф лобстера;
- граф карате клубу, розтягнутий графом карате клубу.

1) Karate club graph/KarateClub

Граф карате клубу було побудовано за допомогою генератора karate_club_graph з Python-бібліотеки Networx [30], про який детально було розказано у підрозділі 3.1.1. Він не приймає жодних параметрів, має дві природні спільноти.

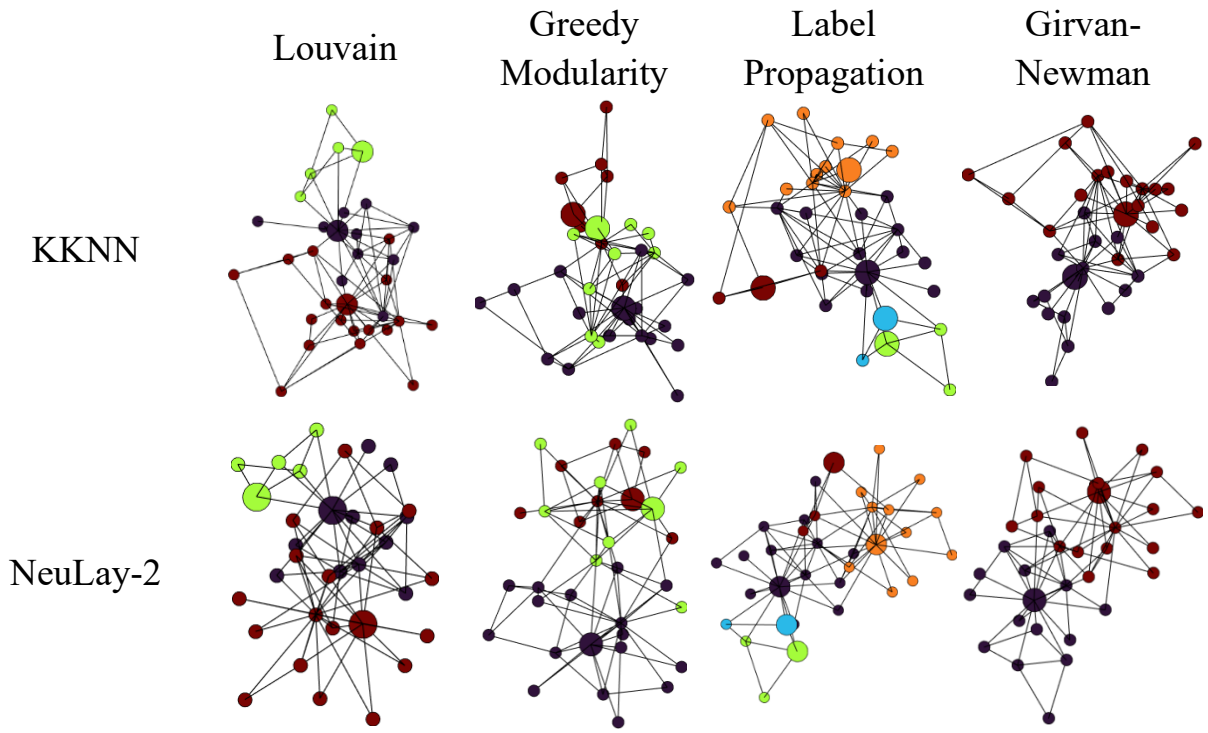


Рис. 7: укладання графа KarateClub алгоритмами KamadaNN та NeuLay-2

На рис. 7 зображено порівняння укладок графа карате клубу алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально оцінити якість розташування спільнот в цьому графі важко через їх змішаність. Можна побачити, що лише алгоритм Girvan-Newman правильно розбиває цей граф на спільноти.

2) Caveman graph/Caveman2020

Граф печерної людини було побудовано за допомогою генератора relaxed_caveman_graph з Python-бібліотеки Networx [31], про який детально було розказано у підрозділі 3.1.1. В якості параметрів було взято кількість печер $l = 20$, розмір печер $k = 20$, ймовірність перерозташування ребер між печерами $p = 0.01$.

Таким чином було отримано граф ікосаедру з чітко вираженими спільнотами.

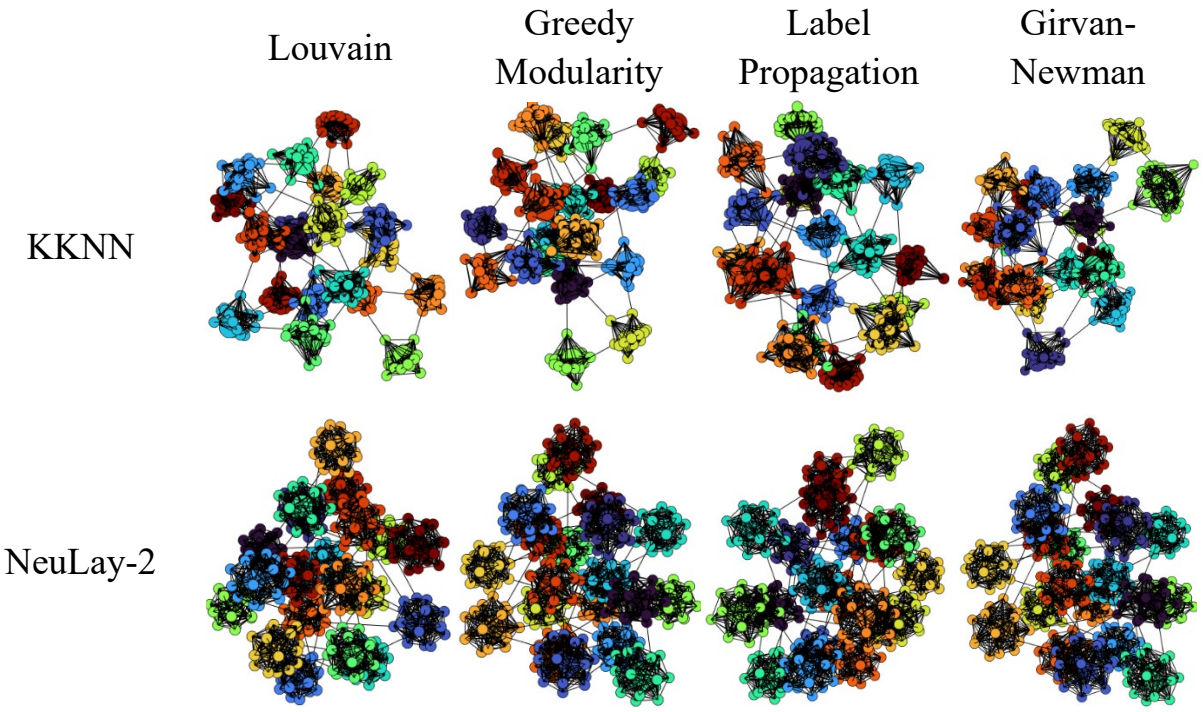


Рис. 8: укладання графа Caveman2020 алгоритмами KamadaNN та NeuLay-2

На рис. 8 зображено порівняння укладок графа печерної людини зі спільнотами алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

З візуальної точки зору обидва алгоритми гарно розташовують спільноти, відображаючи структуру графу. Також можна помітити, що спільноти у графі, вкладеному алгоритмом KamadaNN, краще сепаровані та більш компактні ніж в графі, вкладеному алгоритмом NeuLay-2. Обидва методи розташовують печери без накладань.

3) Stochastic block model/SBM10

Граф стохастичної блочної моделі було побудовано за допомогою генератора stochastic_block_model з Python-бібліотеки Networx [32], про який детально було розказано у підрозділі 3.1.1. В якості параметрів було взято 10 блоків з випадковими розмірами від 30 до 50 вершин, $p_{in} = 0.35, p_{out} = 0.005$.

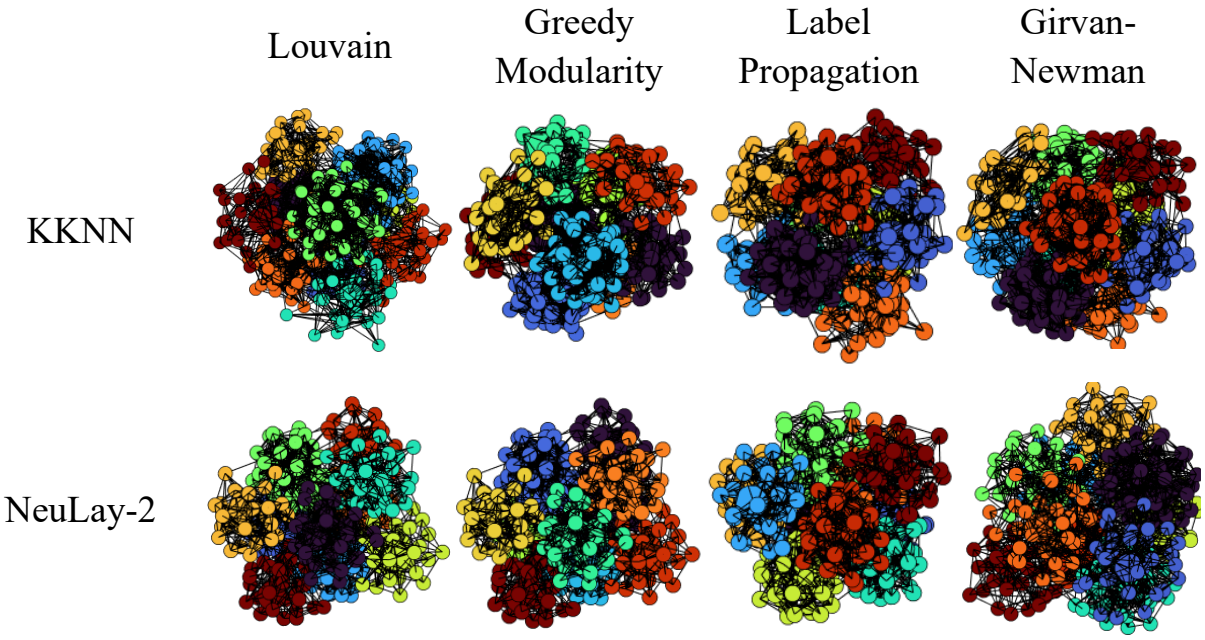


Рис. 9: укладання графа SBM10 алгоритмами KamadaNN та NeuLay-2

На рис. 9 зображено порівняння укладок графа стохастичної блочної

моделі алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

З візуальної точки зору обидва алгоритми логічно розташовують спільноти, відображаючи структуру графу.

Через велику кількість ребр між блоками, спільноти не такі відокремлені, як в укладці попереднього графу, тому оцінити якими будуть метричні результати за цими зображеннями важко.

Помітно, що алгоритм KamadaNN розташовує вершини у спільнотах відповідно до загальної структури графу, розплющуючи таким чином деякі спільноти і тягнучи вершини спільнот одну до одної, в той час як NeuLay-2 робить спільноти більш округлими, фокусується на ребрах в середині спільнот при цьому втрачаючи важливість ребр між спільнотами.

4) Random lobster graph lifted by communities/Lobster25Com20

Граф лобстера зі спільнотами було побудовано за допомогою генератора random_lobster з Python-бібліотеки Networx [33], про який детально було розказано у підрозділі 3.1.2. В якості параметрів було взято кількість вершин у хребті $n = 15$, ймовірність появи лапки $p_1 = 0.75$, ймовірність появи вусика $p_2 = 0.5$. Граф генерувався допоки не було отримано загальну кількість вершин від 25 до 30 (маленька кількість вершин потрібна для того, щоб при розтягуванні не отримати занадто великий граф).

Після цього граф було розтягнуто методом graph_lifting, про який було розказано в підрозділі 3.1.2, з такими параметрами: $n = 20, m = 20, p_{in} = 0.35, p_{out} = 0.005, p_{global} = 0$.

Таким чином ми отримали граф з чіткими спільнотами, скелетом якого є граф лобстера.

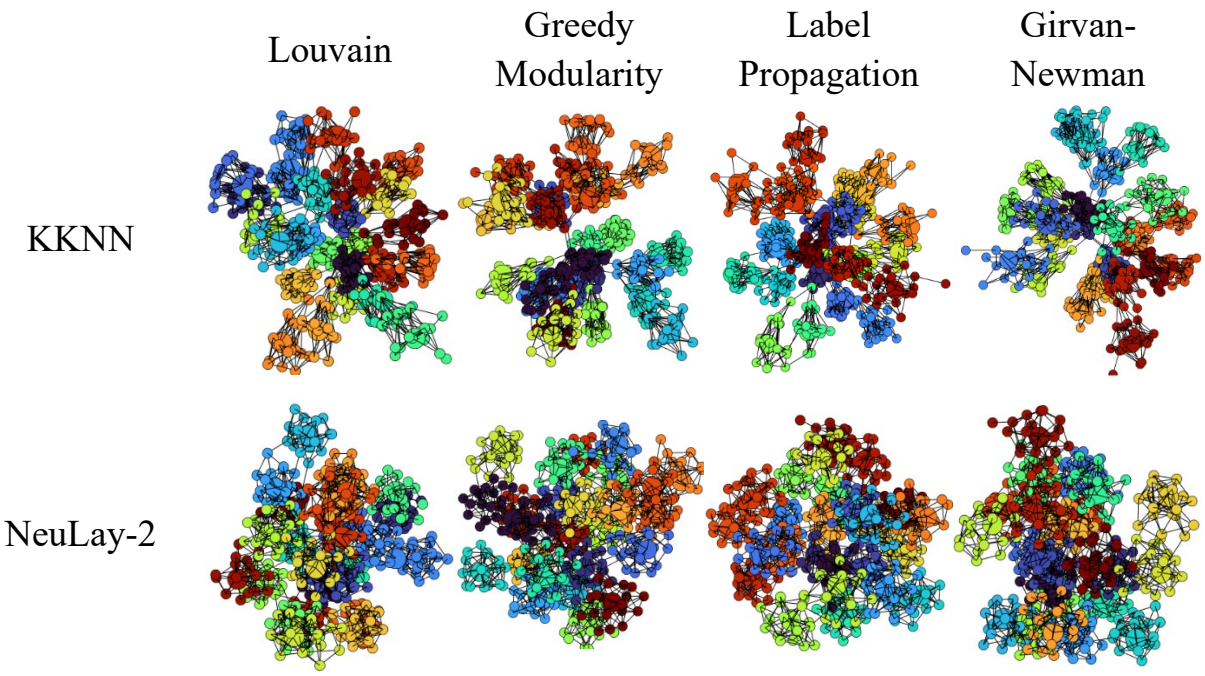


Рис. 10: укладання графа Lobster25Com20 алгоритмами KamadaNN та NeuLay-2

На рис. 10 зображено порівняння укладок графа випадкового лобстера зі спільнотами алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

З візуальної точки зору обидва алгоритми гарно розташовують спільноти, відображаючи структуру графу.

Також можна помітити, що спільноти у графі, вкладеному алгоритмом KamadaNN, краще сепаровані та більш компактні ніж в графі, вкладеному алгоритмом NeuLay-2. Обидва методи розташовують спільноти без накладань.

5) Karate club graph lifted by Karate club graph/KarateKarate

Граф карате клубу, розтягнутий графом карате клубу було побудовано за допомогою генератора karate_club_graph з Python-бібліотеки Networx [30], про який детально було розказано у підрозділі 3.1.1, а також методу graph_lifting_with_template з параметрами $p_{out} = 0.005, p_{global} = 0$.

Таким чином ми отримали граф з 34-х чітких спільнот, кожна з яких має 2 менш чіткі підспільноти.

За результатами запусків алгоритмів пошуку спільнот Louvain, Greedy Modularity та Label Propagation, було вирішено шукати 68 спільнот алгоритмом Girvan-Newman, адже це число є логічним з точки зору структури графу і є близьким до середнього між можливими кількостями спільнот, отриманими цими алгоритмами.

Також було вирішено розглядати число 68 як еталонну кількість спільнот в такому графі.

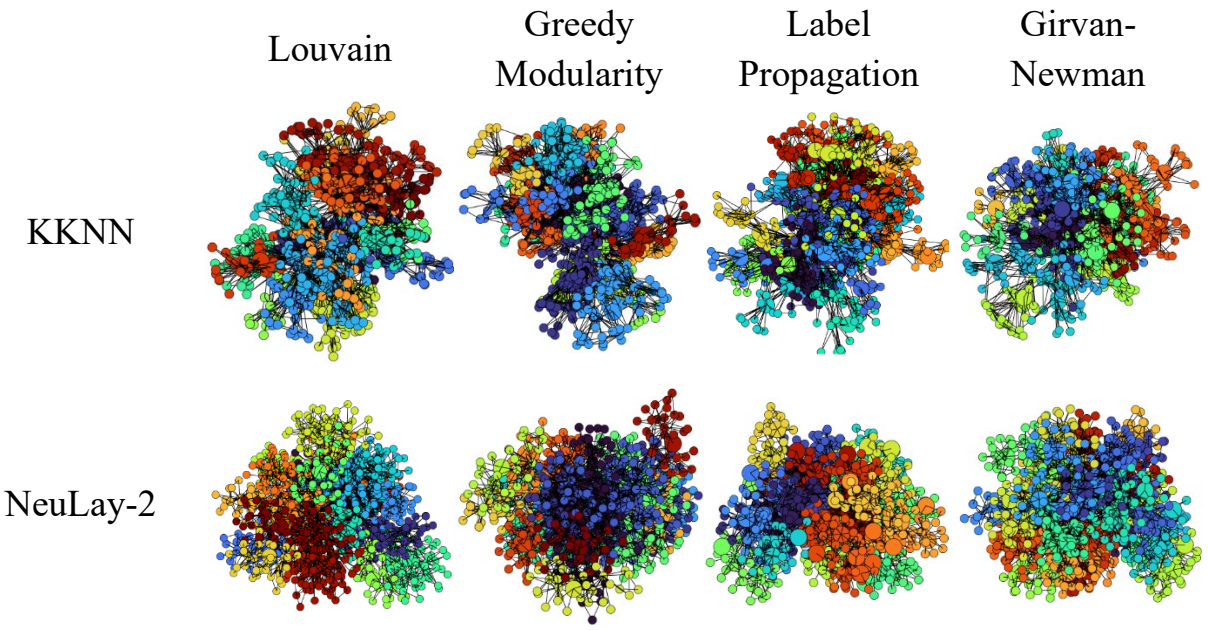


Рис. 11: укладання графа KarateKarate алгоритмами KamadaNN та NeuLay-2

На рис. 11 зображено порівняння укладок графа карате клубу, розтягнутого графом карате клубу алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально важко оцінити якість відображення спільнот в такому графі. Обидва методи укладки графів значно змішують спільноти. Можна помітити, що алгоритм KamadaNN вкладає граф багаторівнево, в той час як NeuLay-2 фокусується на рівномірному розміщенні вершин в середині спільнот.

	Community Number							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	Real	Received	Real	Received	Real	Received	Real	Received
KarateClub	2	3	2	3	2	5	2	2
Caveman2020	20	20	20	20	20	20	20	20
SBM10	10	10	10	9	10	10	10	10
Lobster25Com20	25	24	25	24	25	29	25	25
KarateKarate	68	22	68	20	68	99	68	68
	Davis-Boldin index							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
KarateClub	2.101	1.52	2.516	1.901	1.454	1.293	1.695	1.499
Caveman2020	0.819	0.893	0.819	0.893	0.819	0.893	0.819	0.893
SBM10	1.254	1.262	1.798	1.303	1.254	1.262	1.254	1.262
Lobster25Com20	1.082	1.122	1.084	1.115	1.401	1.519	1.073	1.097
KarateKarate	2.08	2.628	2.111	2.937	2.877	2.204	2.469	2.192
	Silhouette							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
KarateClub	0.136	0.257	0.064	0.212	0.141	0.155	0.274	0.333
Caveman2020	0.566	0.486	0.566	0.486	0.566	0.486	0.566	0.486
SBM10	0.334	0.329	0.263	0.302	0.334	0.329	0.334	0.329
Lobster25Com20	0.431	0.376	0.428	0.374	0.38	0.337	0.437	0.388
KarateKarate	0.121	0.095	0.092	0.023	-0.061	-0.019	0.034	0.111
	Normalised Compactness							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
KarateClub	0.256	0.278	0.283	0.305	0.182	0.201	0.299	0.336
Caveman2020	0.08	0.106	0.08	0.106	0.08	0.106	0.08	0.106
SBM10	0.198	0.194	0.217	0.204	0.198	0.194	0.198	0.194
Lobster25Com20	0.093	0.121	0.093	0.121	0.086	0.114	0.09	0.118
KarateKarate	0.161	0.167	0.163	0.178	0.092	0.086	0.109	0.105
	Normalised Separation							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
KarateClub	0.357	0.44	0.297	0.396	0.406	0.456	0.353	0.449
Caveman2020	0.446	0.425	0.446	0.425	0.446	0.425	0.446	0.425
SBM10	0.464	0.446	0.451	0.438	0.464	0.446	0.464	0.446
Lobster25Com20	0.423	0.446	0.43	0.449	0.42	0.431	0.424	0.447
KarateKarate	0.386	0.383	0.393	0.384	0.367	0.385	0.372	0.384
	Convex Hull Overlap							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
KarateClub	0.022	0	0.024	0	0	0	0.007	0
Caveman2020	0	0	0	0	0	0	0	0
SBM10	0.003	0	0.024	0	0.003	0	0.003	0
Lobster25Com20	0	0	0	0	0	0.001	0	0
KarateKarate	0.141	0.182	0.108	0.341	0.062	0.035	0.07	0.036
	K Near Neighbours							
	Louvain		Greedy Modularity		Label Propagation		Girvan-Newman	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
KarateClub	0.7	0.794	0.612	0.706	0.665	0.665	0.829	0.894
Caveman2020	0.979	1	0.979	1	0.979	1	0.979	1
SBM10	0.853	0.927	0.853	0.931	0.853	0.927	0.853	0.927
Lobster25Com20	0.936	0.948	0.928	0.946	0.875	0.894	0.931	0.945
KarateKarate	0.684	0.744	0.713	0.739	0.44	0.555	0.502	0.61

Таблиця 2: результати експериментів для сімейства несиметричних графів з чіткими спільнотами

У таблиці 2 наведено результати обчислення метрик якості для графів із сімейства несиметричних структур зі спільнотами. На відміну від випадку симетричних графів, отримані результати не демонструють однозначної переваги одного з алгоритмів розкладки. Зокрема, як алгоритм KamadaNN, так і NeuLay-2 показують варіативну ефективність залежно від конкретної структури графа: для частини графів обидва підходи забезпечують якісні вкладення, тоді як для інших – спостерігається погіршення значень окремих метрик.

Разом із тим, аналіз результатів дозволяє виділити певні стійкі закономірності. Зокрема, для всіх розглянутих графів даного сімейства алгоритм KKNN забезпечує кращі показники компактності спільнот, що свідчить про його здатність формувати більш щільні та локалізовані кластери у просторі вкладення.

У свою чергу, метод NeuLay-2 демонструє перевагу за метрикою найближчих сусідів, що вказує на більш точне збереження локальної структури графа та відносин між найближчими вершинами.

Окремої уваги заслуговує аналіз ефективності алгоритмів виявлення спільнот. Було встановлено, що найбільш коректне відновлення структури спільнот у даному сімействі графів забезпечує алгоритм Girvan–Newman, що пояснюється можливістю задання наперед відомої кількості спільнот. Це дозволяє алгоритму більш точно відтворювати цільову спільнотну структуру.

Водночас, за сукупністю кількісних метрик якості найкращі результати демонструє метод Label Propagation, який, попри свою евристичну природу, забезпечує високу узгодженість із геометричним представленням графа.

Таким чином, для несиметричних графів зі спільнотами спостерігається більш складна картина взаємодії між методом побудови вкладення та алгоритмами пошуку спільнот.

Жоден із розглянутих підходів не має універсальної переваги, однак кожен із них підкреслює різні аспекти структури графа: KamadaNN – глобальну компактність спільнот, NeuLay-2 – локальну узгодженість сусідств, тоді як вибір алгоритму виявлення спільнот суттєво впливає на фінальну якість картини спільнот. Це свідчить про доцільність комбінованого підходу та подальшого дослідження узгодження методів вкладення і пошуку спільнот для складних несиметричних структур.

4.3. Графи без чіткої структури спільнот

Серед графів, які не мають чітких спільнот було розглянуто наступні графи:

- граф випадкової тривимірної геометрії;
- граф Барабаші;
- граф Воттса-Строгатца;
- граф Барабаші розтягнутий графом Воттса-Строгатца та спільнотами.

1) 3D random geometry graph/RandomGeom3D500

Граф випадкової геометрії в тривимірному просторі було побудовано за допомогою генератора random_geometric_graph з Python-бібліотеки Networx [34], про який детально було розказано у підрозділі 3.1.2. В якості параметрів було взято кількість вершин $n = 500, radius = 0.2$, розмірність простору $dim = 3$.

Таким чином, було отримано граф, який візуально має виражені ознаки спільнотної організації, хоча зазвичай подібні графи не входять у відповідний клас. Це дозволяє розглядати його як приклад прихованої спільнотної структури.

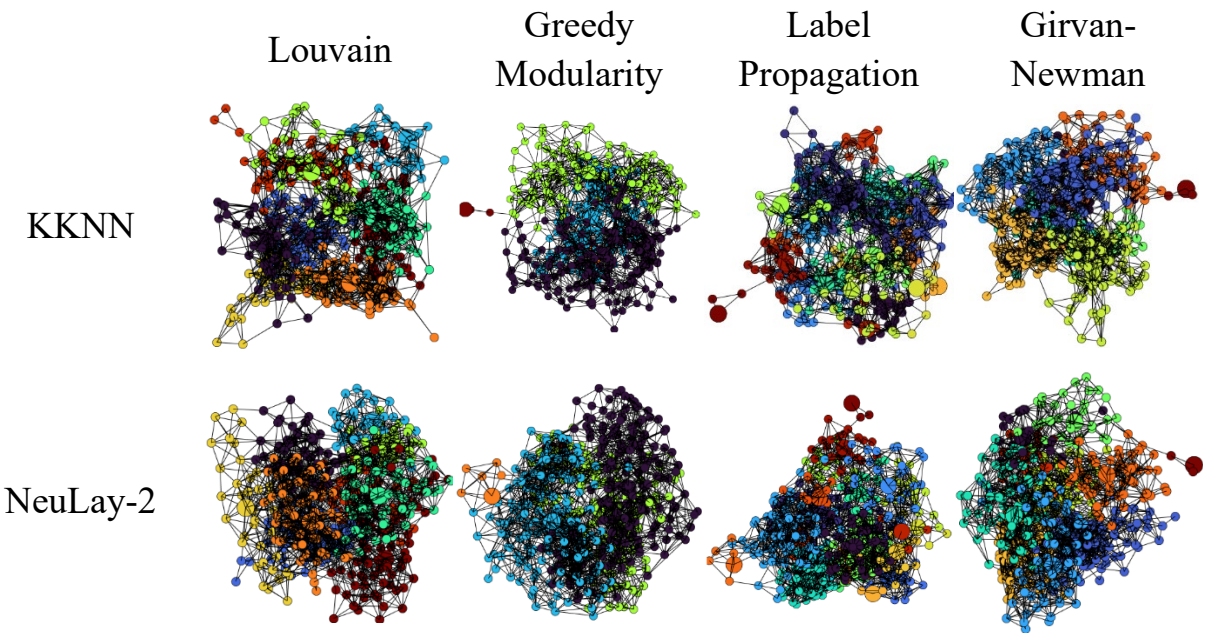


Рис. 12: укладання графа RandomGeom3D500 алгоритмами KamadaNN та NeuLay-2

На рис. 12 зображено порівняння укладок графа випадкової геометрії у тривимірному просторі алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

2) Barabasi Albert graph/BA500

Граф Барабаші Альберта було побудовано за допомогою генератора barabasi_albert_graph з Python-бібліотеки Networx [35], про який детально було розказано у підрозділі 3.1.2. В якості параметрів було взято кількість вершин $n = 500$, кількість створених вершин, з якими з'єднується кожна нова вершина, $m = 1$.

Таким чином, було отримано граф, який є безмасштабною мережею і може розглядатись як граф з нечітко вираженими спільнотами.

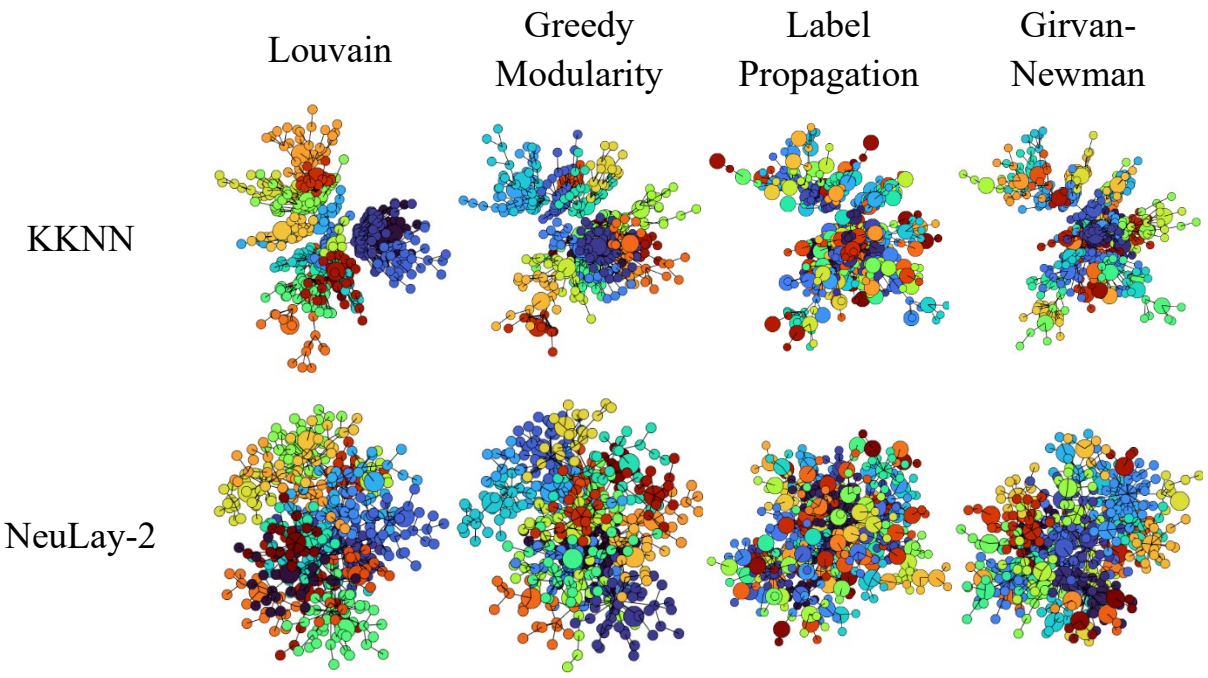


Рис. 13: укладання графа BA500 алгоритмами KamadaNN та NeuLay-2

На рис. 13 зображено порівняння укладок графа безмасштабної мережі Барабаші Альберта алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально можна побачити, що знайдені спільноти не перекриваються у випадку алгоритму візуалізації KamadaNN і трохи перекриваються при використанні алгоритму NeuLay-2. Також можна помітити, що алгоритм KamadaNN краще сепарує знайдені спільноти і робить їх компактнішими.

3) Watts-Strogatz graph/WS500

Граф Воттса-Строгаца було побудовано за допомогою генератора watts_strogatz_graph з Python-бібліотеки Networx [36], про який детально було розказано у підрозділі 3.1.2. В якості параметрів було взято кількість вершин $n = 500$, кількість вершин у топології кільця, з якими початково з'єднується кожна вершина, $m = 10$, ймовірність переадресації ребра $p = 0.05$.

Таким чином, було отримано граф, який є моделлю маленького світу і

може розглядатись як граф з нечітко вираженими спільнотами.

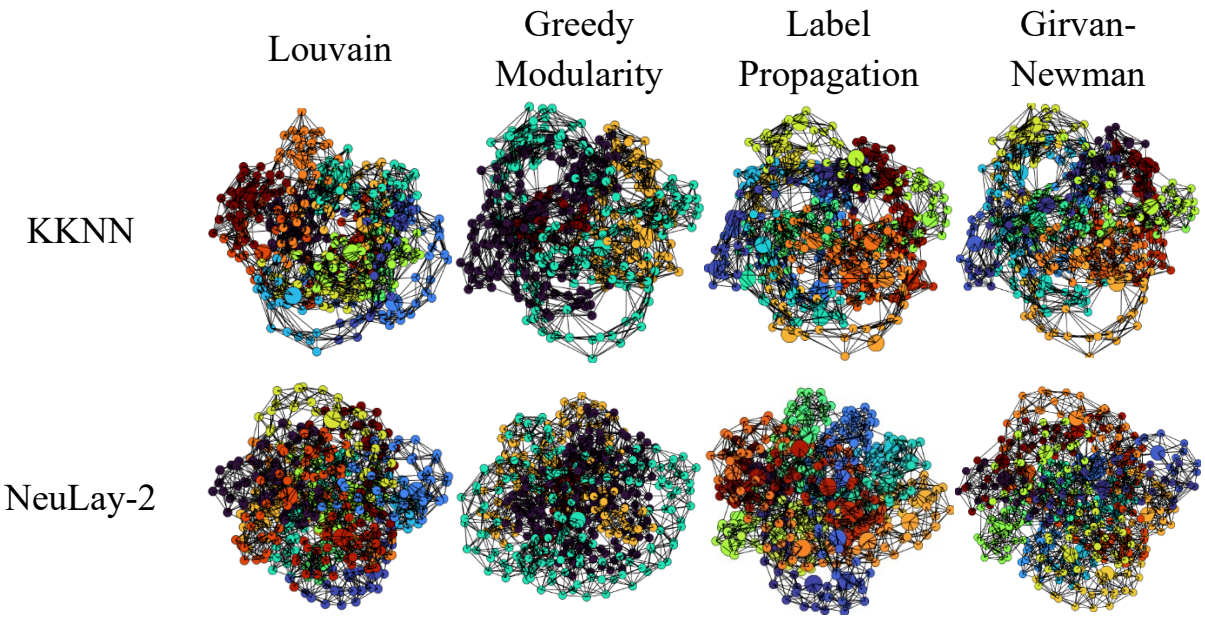


Рис. 14: укладання графа WS500 алгоритмами KamadaNN та NeuLay-2

На рис. 14 зображено порівняння укладок графа маленького світу Воттса-Строгаца алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально оцінити метрики якості візуалізації спільнот важко, адже структура даного графу доволі складна.

4) Barabasi graph lifted by Watts-Strogatz graph and lifted by small communities/BA50WS10Com

Даний граф було побудовано за допомогою генераторів barabasi_albert_graph та watts_strogatz_graph з Python-бібліотеки Networx [35], [36] (див. підрозділ 3.1.2), а також за допомогою методів graph_lifting та graph_lifting_with_template (див. підрозділ 3.1.2). Процес побудови графу відбувався наступним чином:

1. Будується граф Barabasi з 50 вершин, де кожна нова вершина з’єднується з 1 попередньою.

2. Будується граф Watts-Strogatz з 10 вершин, де кожна вершина з'єднана з 4-ма найближчими по топології кільця, ймовірність переорієнтації кожного ребра 0.05.
3. Побудований граф Барабаші розтягується побудованим графом Воттса-Строгаца методом graph_lifting_with_template з параметром $p_{out} = 0.05$.
4. Отриманий граф додатково розтягується методом graph_lifting з параметрами $n = 1, m = 4, p_{in} = 1, p_{out} = 0.05$.

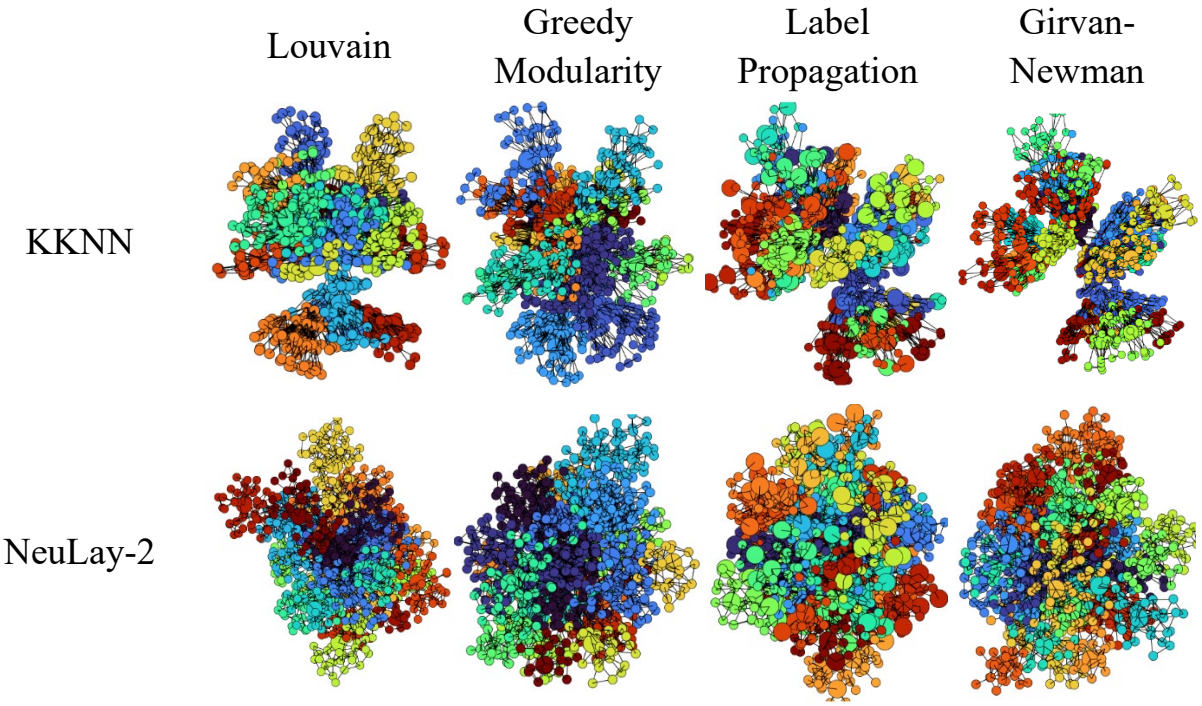


Рис. 15: укладання графа BA50WS10Com алгоритмами KKNN та NeuLay-2

На рис. 15 зображено порівняння укладок графа Барабаші Альберта, розтягнутого графом Воттса-Строгаца і розтягнутого маленькими спільнотами, алгоритмами KamadaNN та NeuLay-2 з різними методами пошуку спільнот.

Візуально можна побачити, що знайдені спільноти менше перекриваються у випадку алгоритму візуалізації KamadaNN ніж у NeuLay-2.

	Community Number							
	Louvain		**Greedy Modularity**		**Label Propagation**		**Girvan-Newman**	
	Real	Received	Real	Received	Real	Received	Real	Received
RandomGeom3D500		9		5		23		10
BA500		18		19		133		50
WS500		13		4		44		20
BA50WS10Com		25		20		239		50
	Davis-Boldin index							
	Louvain		**Greedy Modularity**		**Label Propagation**		**Girvan-Newman**	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
RandomGeom3D500	1.504	2.081	1.533	1.99	1.363	1.644	1.381	1.853
BA500	2.097	2.288	2.109	2.246	1.792	1.233	1.867	1.707
WS500	3.067	3.054	9.393	4.338	2.407	1.581	2.64	2.205
BA50WS10Com	2.495	2.219	2.586	2.218	2.507	1.741	2.697	2.26
	Silhouette							
	Louvain		**Greedy Modularity**		**Label Propagation**		**Girvan-Newman**	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
RandomGeom3D500	0.265	0.2	0.156	0.071	0.203	0.119	0.26	0.172
BA500	0.219	0.156	0.212	0.159	0.039	0.115	0.146	0.118
WS500	0.065	0.068	-0.03	-0.017	0.008	0.181	0.052	0.115
BA50WS10Com	0.134	0.086	0.115	0.033	-0.077	0.038	0.112	0.149
	Normalised Compactness							
	Louvain		**Greedy Modularity**		**Label Propagation**		**Girvan-Newman**	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
RandomGeom3D500	0.198	0.225	0.211	0.24	0.118	0.143	0.178	0.206
BA500	0.119	0.166	0.116	0.161	0.045	0.53	0.076	0.098
WS500	0.268	0.252	0.386	0.374	0.135	0.121	0.221	0.197
BA50WS10Com	0.131	0.143	0.136	0.147	0.053	0.051	0.106	0.109
	Normalised Separation							
	Louvain		**Greedy Modularity**		**Label Propagation**		**Girvan-Newman**	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
RandomGeom3D500	0.393	0.401	0.487	0.499	0.442	0.448	0.429	0.432
BA500	0.333	0.375	0.337	0.377	0.346	0.405	0.335	0.406
WS500	0.412	0.394	0.251	0.253	0.474	0.445	0.433	0.416
BA50WS10Com	0.357	0.363	0.355	0.369	0.376	0.373	0.362	0.357
	Convex Hull Overlap							
	Louvain		**Greedy Modularity**		**Label Propagation**		**Girvan-Newman**	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
RandomGeom3D500	0.001	0.055	0.047	0.238	0	0.008	0.001	0.058
BA500	0.002	0.047	0.001	0.043	0	0	0.001	0
WS500	0.202	0.074	0.525	0.676	0.008	0.001	0.067	0.01
BA50WS10Com	0.046	0.217	0.042	0.256	0.002	0.006	0.037	0.033
	K Near Neighbours							
	Louvain		**Greedy Modularity**		**Label Propagation**		**Girvan-Newman**	
	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2	KKNN	NeuLay-2
RandomGeom3D500	0.895	0.844	0.931	0.878	0.834	0.755	0.886	0.837
BA500	0.876	0.778	0.872	0.776	0.4	0.466	0.659	0.678
WS500	0.694	0.85	0.765	0.902	0.551	0.705	0.63	0.819
BA50WS10Com	0.738	0.782	0.8	0.782	0.305	0.436	0.616	0.711

Таблиця 3: результати експериментів для сімейства графів без чітких спільнот

У таблиці 3 наведено результати обчислення метрик якості для графів із сімейства без чітко вираженої спільнотної структури. Аналіз отриманих даних свідчить про відсутність однозначної переваги одного з алгоритмів, однак дозволяє виділити певні закономірності.

Зокрема, алгоритм KamadaNN демонструє кращі результати за коефіцієнтом силуету, нормалізованою компактністю та метрикою перекриття випуклих оболонок, що вказує на його здатність формувати більш згруповані та просторово узгоджені кластери навіть за відсутності чіткої спільнотної структури. Водночас метод NeuLay-2 забезпечує кращі значення індексу Девіса–Болдіна, що свідчить про більш сприятливе співвідношення між внутрішньокластерною компактністю та міжкластерною віддаленістю для окремих конфігурацій.

Що стосується метрик найближчих сусідів та сепарованості, то для більшості випадків спостерігається порівнянна ефективність обох підходів. Водночас для окремих алгоритмів виявлення спільнот перевага за цими показниками переходить до NeuLay-2, що може свідчити про краще збереження локальної структури графа.

Таким чином, для графів без чітко виражених спільнот обидва алгоритми підкреслюють різні аспекти структури: KamadaNN – глобальну компактність і узгодженість кластерів, тоді як NeuLay-2 у ряді випадків краще відображає локальні взаємозв’язки між вершинами. Це підтверджує доцільність використання комплексного підходу до оцінювання якості вкладень для графів із розмитою спільнотною організацією.

4.4. Підсумок

Для глобальної оцінки якості зображення спільнот алгоритмами KamadaNN та NeuLay-2, результати було згруповано по обраним сімействам

графів і для кожного методу пошуку спільнот та для кожної метрики було визначено який з алгоритмів дає кращий результат.

Louvain	DB	Silhouette	Compactness	Separation	ConvexHullOverlap	NearNeighbours
ComSymGraphs	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**
ComGraphs	**KKNN**	**KKNN**	**KKNN**	Draw	Draw	NeuLay-2
NoComGraphs	Draw	**KKNN**	**KKNN**	Draw	**KKNN**	Draw
Greedy	DB	Silhouette	Compactness	Separation	ConvexHullOverlap	NearNeighbours
ComSymGraphs	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**
ComGraphs	Draw	Draw	**KKNN**	Draw	Draw	NeuLay-2
NoComGraphs	Draw	**KKNN**	Draw	NeuLay-2	**KKNN**	Draw
Label propagation	DB	Silhouette	Compactness	Separation	ConvexHullOverlap	NearNeighbours
ComSymGraphs	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**
ComGraphs	Draw	Draw	**KKNN**	**KKNN**	Draw	NeuLay-2
NoComGraphs	NeuLay-2	Draw	Draw	Draw	Draw	NeuLay-2
Girvan-Newman		Silhouette	Compactness	Separation	ConvexHullOverlap	NearNeighbours
ComSymGraphs	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**	**KKNN**
ComGraphs	Draw	Draw	**KKNN**	Draw	NeuLay-2	NeuLay-2
NoComGraphs	NeuLay-2	Draw	**KKNN**	Draw	Draw	Draw

Таблиця 4: результати експериментів згруповані по сімействам графів

Узагальнені результати, наведені в підсумковій таблиці 4, дозволяють комплексно порівняти ефективність алгоритмів KamadaNN та NeuLay-2 для різних сімейств графів з урахуванням використаних метрик якості та методів пошуку спільнот. В клітинках таблиці позначка KKNN означає, що на відповідному сімействі графів, при використанні відповідного алгоритму кращу метрику дає алгоритм KamadaNN, NeuLay-2 означає, що краща метрика у NeuLay-2, Draw означає, що алгоритми дають співставні значення метрики.

Аналіз показує, що для графів із чітко вираженою спільнотною структурою алгоритм KamadaNN демонструє стабільну перевагу за всіма розглянутими метриками незалежно від обраного методу кластеризації. Для несиметричних графів зі спільнотами спостерігається більш варіативна картина: обидва алгоритми виявляються ефективними залежно від конкретної метрики, однак KamadaNN переважає за показниками компактності, тоді як NeuLay-2 частіше забезпечує кращі результати за локальними

характеристиками, зокрема метрикою найближчих сусідів. У випадку графів без чіткої спільнотної структури також відсутня однозначна перевага: KamadaNN краще відображає глобальну узгодженість і компактність кластерів, тоді як NeuLay-2 у ряді випадків демонструє кращі значення індексу Девіса–Болдіна та окремих локальних метрик.

Таким чином, узагальнені результати підтверджують, що KamadaNN є більш ефективним для графів із вираженою або частково вираженою спільнотною структурою, забезпечуючи кращу глобальну організацію вкладення. Водночас NeuLay-2 у ряді випадків краще зберігає локальні властивості графа. Це свідчить про те, що вибір методу вкладення доцільно здійснювати з урахуванням структурних особливостей графа та цільових метрик оцінювання.

ВИСНОВКИ

У роботі досліджено задачу візуалізації графів із акцентом на коректне відображення їхньої спільнотної структури. Запропонований підхід KKNN, що поєднує алгоритм Камада–Каваї з нейромережевим прискоренням, продемонстрував високу ефективність у побудові вкладень, які зберігають як глобальні, так і локальні властивості графів.

Проведене експериментальне дослідження на різних сімействах графів показало, що для симетричних графів із чітко вираженими спільнотами алгоритм KKNN має стабільну перевагу над NeuLay-2 за всіма розглянутими метриками. Для несиметричних графів зі спільнотами та графів без чіткої спільнотної структури обидва підходи демонструють порівнянні результати, однак підкреслюють різні аспекти структури: KKNN забезпечує кращу компактність і узгодженість кластерів, тоді як NeuLay-2 у ряді випадків ефективніше зберігає локальні зв’язки між вершинами.

Також встановлено, що вибір алгоритму виявлення спільнот суттєво впливає на якість отриманих результатів: у ряді випадків найбільш коректне відтворення структури забезпечує метод Girvan–Newman, тоді як найкращі значення метрик часто демонструє Label Propagation.

Отже, отримані результати підтверджують доцільність використання KKNN для задач візуалізації графів зі спільнотною структурою, а також вказують на важливість комплексного підходу, що враховує як метод побудови вкладення, так і алгоритм кластеризації. Перспективними напрямами подальших досліджень є розробка адаптивних методів поєднання глобальних і локальних критеріїв якості, а також застосування запропонованого підходу до реальних мереж складної структури.

Код програми та результати експериментів наведено у [37].

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

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

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

[3] Linnyk O., Polyakova L., Zaretska I. On Kamada-Kawai graph layout with graph neural networks. *Radioelectronic and Computer Systems*. 2025. No. 3. DOI: 10.32620/reks.2025.3.07.

[4] Bondy J. A., Murty U. S. R. *Graph Theory with Applications*. New York: North-Holland, 1976.

[5] McCulloch W. S., Pitts W. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*. 1943. Vol. 5. P. 115–133. DOI: 10.1007/BF02478259.

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

[7] Clauset A., Newman M. E. J., Moore C. Finding community structure in very large networks. *Physical Review E*. 2004. Vol. 70. Article 066111. DOI: 10.1103/PhysRevE.70.066111.

[8] Raghavan U. N., Albert R., Kumara S. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*. 2007. Vol. 76. Article 036106. DOI: 10.1103/PhysRevE.76.036106.

[9] Blondel V. D., Guillaume J.-L., Lambiotte R., Lefebvre E. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and*

Experiment. 2008. P10008. DOI: 10.1088/1742-5468/2008/10/P10008.

[10] Girvan M., Newman M. E. J. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*. 2002. Vol. 99, No. 12. P. 7821–7826. DOI: 10.1073/pnas.122653799.

[11] Davies D. L., Bouldin D. W. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1979. Vol. PAMI-1, No. 2. P. 224–227. DOI: 10.1109/TPAMI.1979.4766909.

[12] Rousseeuw P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*. 1987. Vol. 20. P. 53–65. DOI: 10.1016/0377-0427(87)90125-7.

[13] Barber C. B., Dobkin D. P., Huhdanpaa H. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*. 1996. Vol. 22, No. 4. P. 469–483. DOI: 10.1145/235815.235821.

[14] Cover T. M., Hart P. E. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*. 1967. Vol. 13, No. 1. P. 21–27. DOI: 10.1109/TIT.1967.1053964.

[15] Metropolis N., Ulam S. The Monte Carlo method // Journal of the American Statistical Association. – 1949. – Vol. 44, No. 247. – P. 335–341. – DOI: 10.1080/01621459.1949.10483310.

[16] Hagberg A. A., Schult D. A., Swart P. J. Exploring network structure, dynamics, and function using NetworkX. In: *Proceedings of the 7th Python in Science Conference (SciPy 2008)*. 2008. P. 11–15.

[17] Zachary W. W. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*. 1977. Vol. 33, No. 4. P. 452–473.

[18] Holland P. W., Laskey K. B., Leinhardt S. Stochastic blockmodels: first steps. *Social Networks*. 1983. Vol. 5, No. 2. P. 109–137. DOI: 10.1016/0378-8733(83)90021-7.

[19] Watts D. J. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton: Princeton University Press, 1999.

[20] Harary F. *Graph Theory*. Reading, MA: Addison-Wesley, 1969.

[21] Beineke L. W., Wilson R. J. *Topics in Algebraic Graph Theory*. Cambridge University Press, 2004.

[22] Tutte W. T. Graph theory of polyhedra. *Proceedings of the London Mathematical Society*. 1963.

[23] Gilbert E. N. Random plane networks. *Journal of the Society for Industrial and Applied Mathematics*. 1961. Vol. 9, No. 4. P. 533–543. DOI: 10.1137/0109045.

[24] Watts D. J., Strogatz S. H. Collective dynamics of "small-world" networks. *Nature*. 1998. Vol. 393. P. 440–442. DOI: 10.1038/30918.

[25] Barabási A.-L., Albert R. Emergence of scaling in random networks. *Science*. 1999. Vol. 286, No. 5439. P. 509–512. DOI: 10.1126/science.286.5439.509.

[26] Bermond J.-C., Schönheim J. On the structure of trees. *Discrete Mathematics*. 1974.

[27] Генератор графу решітки: grid_graph – NetworkX 3.6.1 documentation. URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.lattice.grid_graph.html (дата звернення: 05.05.2026).

[28] Генератор графу зірки: star_graph – NetworkX 3.6.1 documentation. URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.classic.star_graph.html (дата звернення: 05.05.2026).

[29] Генератор графу ікосаедру: icosahedral_graph – NetworkX 3.6.1 documentation.
URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.small.icosahedral_graph.html (дата звернення: 05.05.2026).

[30] Генератор графу карате клубу: karate_club_graph – NetworkX 3.6.1 documentation.

URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.social.karate_club_graph.html (дата звернення: 05.05.2026).

[31] Генератор графу печерної людини: caveman_graph – NetworkX 3.6.1 documentation.

URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.caveman_graph.html (дата звернення: 05.05.2026).

[32] Генератор графу стохастичної блочної моделі: stochastic_block_model – NetworkX 3.6.1 documentation.

URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.community.stochastic_block_model.html (дата звернення: 05.05.2026).

[33] Генератор графу лобстера: random_lobster_graph – NetworkX 3.7rc0.dev0 documentation.

URL: https://networkx.org/documentation/latest/reference/generated/networkx.generators.random_graphs.random_lobster_graph.html (дата звернення: 05.05.2026).

[34] Генератор графу випадкової геометрії: random_geometric_graph – NetworkX 3.6.1 documentation.

URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.geometric.random_geometric_graph.html (дата звернення: 05.05.2026).

[35] Генератор графу Барабаші: barabasi_albert_graph – NetworkX 3.6.1 documentation.

URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.barabasi_albert_graph.html (дата звернення: 05.05.2026).

[36] Генератор графу Воттса-Строгаца: watts_strogatz_graph – NetworkX 3.6.1 documentation.

URL: https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.watts_strogatz_graph.html (дата звернення: 05.05.2026).

[37] Код програми та результати експериментів: GitHub -

OlenaLinnyk/KamadaNN: This is the code for my diploma thesis. *GitHub*. URL: https://github.com/OlenaLinnyk/KamadaNN (дата звернення: 05.05.2026).