ВИКОРИСТАННЯ ГРАФОВИХ НЕЙРОМЕРЕЖ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ ПОШУКУ ОПТИМАЛЬНОГО ШЛЯХУ

Погорєлова Л.А., Сердюк Н.М.

Кафедра комп'ютерних інтелектуальних технологій та систем,
Харківський національний університет радіоелектроніки,
Україна

E-mail: liliia.pohorielova@nure.ua,
nataliya.serdyuk@nure.ua

Abstract

The article examines the application of Message Passing Graph Neural Networks (MPGNN) to the classical shortest path problem in graph theory. The proposed approach utilizes the IGNNITION framework, which enables the declarative design and training of MPGNN architectures without low-level coding. The network learns to classify nodes belonging to the shortest path by aggregating information from neighboring nodes through iterative message passing. Experimental results demonstrate that the developed model achieves high accuracy on small graphs and provides a flexible foundation for scaling to more complex structures. The research results can be used to improve route optimization systems in transport logistics and other domains dealing with graph-structured data.

Сучасні інформаційні технології стрімко розвиваються, створюючи попит на ефективні методи обробки складних структур даних, зокрема графів. Графи є універсальною моделлю для представлення зв'язків між об'єктами в таких галузях, як транспортна логістика, соціальні мережі, телекомунікації, біоінформатика та системи рекомендацій [1]. Різноманіття графових структур забезпечує можливість відтворення об'єктів різної природи, зокрема зображень, текстів, молекулярних сполук, програмних залежностей та дорожніх мереж.

Однією з ключових задач на графах є пошук найкоротшого шляху, який має широке практичне застосування: від оптимізації маршрутів доставки до аналізу зв'язності мереж і планування телекомунікаційних систем. Традиційні алгоритми, такі як алгоритм Дейкстри, Беллмана-Форда чи A*, забезпечують точне розв'язання задачі пошуку найкоротшого шляху, але їхня обчислювальна ефективність може бути обмеженою для великих графів або в умовах складних залежностей між вузлами, де потрібне врахування контекстної інформації чи адаптація до динамічних змін.

Графові нейронні мережі (Graph Neural Networks, GNN) є сучасним інструментом, який дозволяє обробляти структури графів, враховуючи їхню топологію та атрибути вузлів і ребер. GNN базуються на ідеї агрегації інформації від сусідніх вузлів через механізми передачі повідомлень, що дає змогу моделі навчатися на складних зв'язках у графі.

Одним із різновидів GNN є Message Passing Graph Neural Network (MPGNN). Ця архітектура базується на ітеративному процесі обміну повідомленнями між вузлами, що дозволяє формувати їхні представлення на основі локальної структури графа. Завдяки своїй гнучкості MPGNN здатна працювати з графами різної топології та розміру, що робить її придатною для задач, пов'язаних із передбаченням оптимальних дій, наприклад вибору наступного вузла в маршруті.

MPGNN – це клас графових нейронних мереж, де прихований стан кожного вузла оновлюється шляхом агрегації повідомлень від сусідів та комбінування з власним поточним станом. Цей процес дозволяє моделі вивчати як локальні, так і, з часом, більш глобальні залежності. Вхідними даними для MPGNN є граф $G = (V, E)$, який складається з множини вузлів V та множини ребер E, що їх з'єднують (рис. 1). Кожен вузол $v \in V$ асоціюється з вектором ознак x_v певного розміру, який використовується для ініціалізації його прихованого стану h_v. Ребра $e_{uv} \in E$ також можуть мати свої ознаки, що несуть додаткову інформацію про зв'язок між вузлами u та v. Прихований стан вузла h_v – це вектор

фіксованої довжини, який динамічно оновлюється протягом обчислювального процесу і слугує представленням вузла, що містить інформацію про його властивості та контекст у графі.

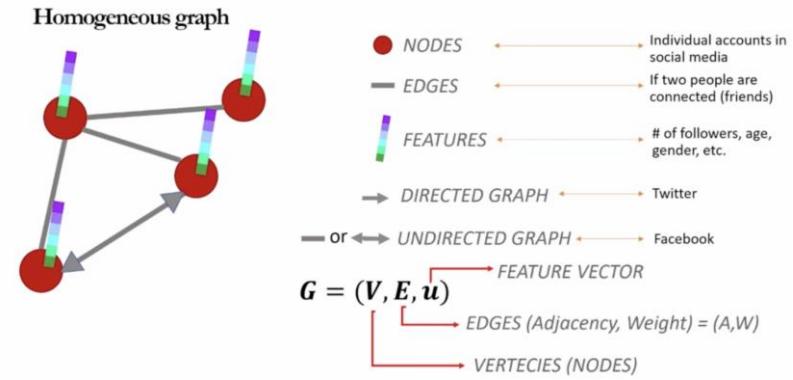

Рис. 1. Графічне представлення графу

Основу роботи MPGNN становить ітеративний алгоритм передачі повідомлень, який складається з трьох ключових етапів: формування повідомлень, агрегації та оновлення [2]. Ці етапи виконуються послідовно протягом заданої кількості ітерацій. Після завершення T ітерацій передачі повідомлень, кінцеві приховані стани вузлів h_v^T використовуються для генерації вихідних даних моделі за допомогою функції зчитування $R(\cdot)$.

Однією з головних переваг MPGNN є гнучкість функцій M, U та R. Зазвичай вони реалізуються окремими (часто невеликими) нейронними мережами з параметрами, спільними для всіх вузлів і/або ребер. Ці параметри одночасно налаштовуються в процесі навчання, що дозволяє моделі вивчати універсальні шаблони обробки інформації в графах та узагальнювати набуті знання на нові, раніше не бачені графи з різною структурою та розміром. У табл. 1 наведена систематизація всіх фаз обчислювального процесу.

Таблиця 1. Фази обчислювального процесу в MPGNN

Фаза	Опис	Математичне представлення
Повідомлення	Кожен вузол v надсилає повідомлення своїм сусідам w, використовуючи їхні приховані стани та ознаки ребра.	$m_{vw}^{t+1} = M(h_v^t, h_w^t, e_{v,w})$
Агрегація	Кожен вузол v агрегує повідомлення, отримані від усіх своїх сусідів $w \in N(v)$, в єдиний вектор.	$m_v^{t+1} = \sum_{w \in N(v)} m_{vw}^{t+1}$
Оновлення	Кожен вузол v оновлює свій прихований стан, комбінуючи поточний стан з агрегованим повідомленням.	$h_v^{t+1} = U(h_v^t, m_v^{t+1})$
Зчитування	Після T ітерацій, кінцеві приховані стани вузлів h_v^T використовуються для генерації вихідних прогнозів моделі $\hat{y}$.	$\hat{y} = R(h_v^T \forall v \in V)$

Задача пошуку найкоротшого шляху є класичною проблемою в теорії графів. У класичному вигляді, задача полягає у знаходженні шляху між двома заданими вузлами (початковим та кінцевим) у графі таким чином, щоб сума ваг ребер, що складають цей шлях, була мінімальною.

Для застосування моделей MPGNN цю задачу часто трансформують. Оскільки найкоротші шляхи можуть мати різну довжину, пряме прогнозування послідовності вузлів може бути складним. Натомість, популярним підходом є переформулювання задачі як бінарної класифікації вузлів: для кожної заданої пари “джерело-призначення”, кожен вузол графа класифікується на предмет того, чи належить він до найкоротшого шляху між цією парою. Альтернативно, можна класифікувати ребра. Таке перетворення дозволяє використовувати стандартні архітектури MPGNN, які генерують прогнози на рівні вузлів (або ребер) та стандартні функції втрат для задач класифікації.

Навчання MPGNN для задачі пошуку найкоротшого шляху (сформульованої як класифікація вузлів) відбувається за стандартною схемою [3]:

1. Пряме поширення: На вхід моделі подається граф з ознаками src_tgt, які приймають значення 1 для вузлів, що є початковими або кінцевими точками шуканого шляху, та 0 – в інших випадках. Модель обробляє ці дані та генерує прогноз для кожного вузла.

2. Розрахунок функції втрат: Отримані прогнози порівнюються з істинними мітками, які вказують, чи належить вузол до найкоротшого шляху. Для задачі бінарної класифікації зазвичай використовується функція втрат Binary Cross-Entropy.

3. Зворотне поширення помилки та оновлення ваг: Градієнт функції втрат обчислюється та поширюється назад по мережі, а оптимізатор (наприклад, Adam) оновлює ваги моделі.

Лідером в проектуванні графових нейронних мереж є фреймворк IGNNITION. Його популярність зумовлена декларативним підходом до опису архітектури, автоматичною генерацією PyTorch-коду та сумісністю з основними бібліотеками машинного навчання, що суттєво спрощує й пришвидшує процес розробки [4].

IGNNITION дозволяє визначати архітектуру GNN за допомогою декларативного YAML-файлу, уникаючи потреби писати низькорівневий код, наприклад, на TensorFlow. Такий підхід особливо зручний для швидкого створення та тестування графових моделей у прикладних задачах.

Для побудови моделі дослідник задає параметри конфігурації у файлі model_description.yaml, де описуються всі ключові компоненти архітектури. Цей процес передбачає такі основні етапи:

1. Ініціалізація прихованого стану;
2. Опис передачі повідомлень:
   2.1. Фаза повідомлення;
   2.2. Фаза агрегації та оновлення;
3. Опис фази зчитування.

Розглянемо приклад програмної реалізації всіх етапів, що наведений в офіційній документації фреймворку IGNNITION [5]. Під час ініціалізації прихованого стану було задано розмірність вектора 16, оскільки тестовий датасет містить графи з кількістю вузлів від 5 до 15 та вагами в діапазоні від 0 до 10. Такої розмірності достатньо, щоб закодувати інформацію про вузол, його локальний контекст і необхідні ознаки для задачі класифікації вузлів. Для складніших графів розмірність доцільно збільшувати (наприклад, до 32, 64 або 128).

Наступний етап – опис передачі повідомлень – визначає, як вузли графа ітеративно обмінюються інформацією для оновлення своїх станів. Наведена у прикладі конфігурація передбачає 4 ітерації обміну повідомленнями між вузлами типу node. Функція для генерації повідомлень (message) реалізована за допомогою нейронної мережі message_function. Вона використовує стани вузлів-джерел/призначень та вагу ребра як вхідні дані. Архітектура message_function, визначена окремо в секції neural_networks, є двошаровою повнозв'язною мережею з функцією активації ReLU (Rectified Linear Unit).

Після генерації повідомлень від сусідніх вузлів відбувається їх агрегація. Цей етап має на меті звести всі отримані повідомлення в єдиний вектор фіксованого розміру. У наведеному прикладі для агрегації використовується операція min, що є ефективною для задачі пошуку найкоротшого шляху. Отриманий агрегований вектор застосовується для оновлення прихованого стану вузла. Цю операцію виконує функція update, реалізована як окрема нейронна мережа update_function. В якості архітектури для цієї мережі обрано рекурентний блок GRU (Gated Recurrent Unit), який приймає поточний стан вузла та агреговане повідомлення для обчислення нового, оновленого стану.

Завершальним етапом роботи GNN є фаза зчитування (Readout). На цьому етапі кінцеві приховані стани вузлів, отримані після завершення ітерацій передачі повідомлень, використовуються для генерації вихідних даних моделі. Архітектура мережі readout_function визначена як тришарова повнозв'язна мережа. Останній шар містить лише один нейрон з функцією активації sigmoid, що є типовим для задач бінарної класифікації, оскільки він видає на виході значення в діапазоні від 0 до 1, яке розглядається як імовірність.

Процес навчання нейромережі описується у окремому файлі train_options.yaml. Цей файл містить шляхи до тренувального, валідаційного та тестового наборів даних, а також задає параметри оптимізації та керування навчальним процесом. Розглянута модель використовує функцію втрат Binary Cross-Entropy, оптимізатор Adam із швидкістю 0.001 та метрики BinaryAccuracy, Precision і Recall для оцінювання якості класифікації. Навчання відбувається протягом 60 епох із розміром батчу 1 та частотою валідації кожну епоху.

Після завершення етапів опису архітектури та навчання нейромережі наступним кроком є тестування моделі. На вхід необхідно подати граф у форматі JSON із зазначенням вихідного та цільового вузлів. Після запуску та прогнозування програма здійснює відповідну візуалізацію результатів (рис. 2), що включає три графи. Перший граф відображає оригінальну вхідну структуру. Другий граф ілюструє вихідні дані моделі MPGNN, де кольорова градієнтна шкала позначає прогнозовану ймовірність належності кожного вузла до найкоротшого шляху (синій колір – ймовірність належності вузла до найкоротшого шляху дорівнює 0, червоний колір – 1). Третій граф демонструє остаточний прогнозований шлях, сформований з вузлів, для яких ця ймовірність перевищила порогове значення 0.5.

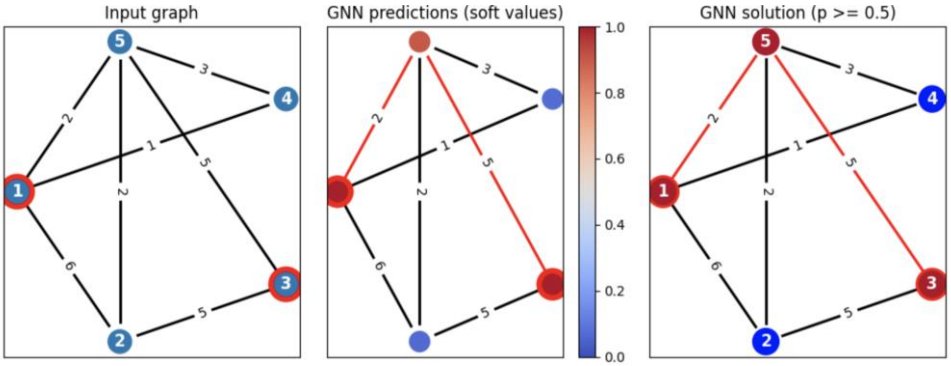

Рис. 2. Результати прогнозування найкоротшого шляху на графі

Використання MPGNN у задачі пошуку найкоротшого шляху продемонструвало достатньо високу точність для графів малого розміру, де еталонний маршрут складався з 3–4 вузлів. Водночас у більших графах модель іноді формувала неповні або надлишкові шляхи – пропускала вузли, що мали належати до найкоротшого маршруту, або включала зайві. Це свідчить про потребу адаптації графових нейронних мереж до складніших структур. Підвищити якість результатів можна за рахунок збільшення розмірності прихованих векторів, кількості ітерацій передачі повідомлень, а також глибини шарів нейронної мережі прямого поширення.

Крім архітектурних змін, важливо масштабувати процес навчання та вдосконалити якість даних. Зокрема, варто розширити навчальний набір, включивши до нього графи зі складнішою топологією, що містять 30–50 і більше вузлів. Для ефективного навчання на такому наборі даних слід також збільшити кількість епох і кроків у кожній епосі, забезпечивши стабільну збіжність моделі.

Загалом, графові нейронні мережі мають значний потенціал, адже дозволяють зберігати на ребрах не лише вагові коефіцієнти, а й додаткові атрибути. У контексті транспортної логістики це можуть бути середня швидкість руху, рівень завантаженості дороги або стан дорожнього покриття. За таких умов задача трансформується з пошуку найкоротшого шляху у задачу визначення оптимального маршруту за сукупністю критеріїв. Фреймворк IGNNITION істотно спрощує реалізацію таких моделей, забезпечуючи гнучкість у створенні нейронних архітектур різного рівня складності.

Література

1. Graph Neural Networks: Foundations, Frontiers, and Applications / ред.: L. Wu та ін. Singapore : Springer Singapore, 2022. 701 p. **DOI**: https://doi.org/10.1007/978-981-16-6054-2.

2. Labonne M. Hands-On Graph Neural Networks Using Python: Practical Techniques and Architectures for Building Powerful Graph and Deep Learning Apps with Pytorch. Packt Publishing, Limited, 2023. 354 p.

3. Broadwater K., Stillman N. Graph Neural Networks in Action. Manning Publications Co. LLC, 2023. 394 p.

4. Pujol-Perich D., Suarez-Varela J., Ferriol M., Xiao S., Wu B., Cabellos-Aparicio A., Barlet-Ros P. IGNNITION: Bridging the Gap between Graph Neural Networks and Networking Systems // IEEE Network. 2021. Vol. 35, No. 6. P. 171–177. **DOI**: https://doi.org/10.48550/arXiv.2109.06715.

5. Computing the shortest path with Graph Neural Networks (GNN): a hands-on Introduction to IGNNITION // Medium. URL: https://medium.com/@bnn_upc/computing-the-shortest-path-with-graph-neural-networks-gnn-a-hands-on-introduction-to-ignnition-bea531b3b5b2 (дата звернення: 11.11.2025).