УДК 004.421:004.7:004.85 DOI: 10.31673/2412-4338.2025.017718

Коваленко Данило Сергійович
Державний університет інформаційно-комунікаційних технологій, м. Київ
ORCID 0009-0002-2475-2883

Замрій Ірина Вікторівна
Державний університет інформаційно-комунікаційних технологій, м. Київ
ORCID 0000-0001-5681-1871

РОЗРОБКА АДАПТИВНОГО АЛГОРИТМУ МАРШРУТИЗАЦІЇ ДЛЯ C2C ЛОГІСТИКИ ІЗ ЗАСТОСУВАННЯМ ПІДКРІПЛЮВАЛЬНОГО НАВЧАННЯ

***Анотація**: У роботі запропоновано адаптивний алгоритм маршрутизації для C2C логістики, що базується на інтеграції прогнозування попиту за допомогою довготривалої короткочасної пам'яті (LSTM) та підкріплювального навчання (Reinforcement Learning, RL). Алгоритм спрямований на оптимізацію маршрутів кур'єрів у динамічних міських умовах, де традиційні методи не забезпечують достатньої гнучкості та ефективності. Запропонований підхід дозволяє зменшити довжину маршрутів, покращити адаптивність до змін попиту та оптимізувати витрати на транспортування.*

Алгоритм складається з трьох ключових компонентів: прогнозування попиту, оновлення стану середовища та корекції маршрутів на основі RL. Прогнозування виконується за допомогою нейронної мережі LSTM, що дозволяє завчасно визначати області підвищеної активності замовлень та мінімізувати затримки. RL-компонент забезпечує адаптацію маршрутів у режимі реального часу, дозволяючи обирати найбільш оптимальні рішення з урахуванням змін у трафіку та логістичній мережі. Корекція маршрутів використовує прогнозовані значення попиту, значення Q-функції та оптимальну політику прийняття рішень, що підвищує ефективність використання ресурсів.

Експериментальна оцінка ефективності алгоритму проводилася у симуляційному середовищі, що відтворює реальні умови міської логістики. Порівняння з традиційними алгоритмами (евристичні методи та VRP) показало, що адаптивний підхід дозволяє скоротити середню довжину маршруту на 3.3%-9.2% у залежності від кількості перерахунків. Час доставки було зменшено на 21%, а загальні витрати – на 12–17%. Проте адаптивний алгоритм вимагає більше обчислювальних ресурсів: середній час обчислення маршруту є у 2–3 рази вищим за традиційні підходи, а кількість перерахунків може перевищувати раціональний поріг у 15 разів, що потребує додаткової оптимізації.

Результати підтверджують, що запропонований алгоритм є ефективним для сценаріїв із високою динамічністю та варіативністю попиту, проте його обчислювальні витрати можуть бути негативним фактором для практичного застосування. Подальші дослідження мають зосередитися на вдосконаленні алгоритмів перерахунку маршрутів, застосуванні мультиагентних систем для координації кур'єрів та інтеграції хмарних обчислень для оптимізації продуктивності.

***Ключові слова:** C2C логістика, адаптивна маршрутизація, підкріплювальне навчання, прогнозування попиту, мультиагентні системи, логістика останньої милі, оптимізація маршрутів, цифрові двійники, хмарні обчислення.*

Kovalenko Danylo
State university of information and communication technologies, Kyiv
ORCID 0009-0002-2475-2883

Zamrii Iryna
State university of information and communication technologies, Kyiv
ORCID 0000-0001-5681-1871

DEVELOPMENT OF AN ADAPTIVE ROUTING ALGORITHM FOR C2C LOGISTICS USING REINFORCEMENT LEARNING

***Abstract**: This paper proposes an adaptive routing algorithm for C2C logistics based on the integration of demand forecasting using Long Short-Term Memory (LSTM) networks and Reinforcement Learning (RL). The algorithm is designed to optimize courier routes in dynamic urban environments, where traditional methods lack flexibility and efficiency. The proposed approach reduces route lengths, improves adaptability to demand fluctuations, and optimizes transportation costs.*

The algorithm consists of three key components: demand forecasting, environment state updating, and RL-based route correction. Demand forecasting is performed using an LSTM neural network, which allows for early identification of high-order density areas and minimizes delays. The RL component ensures real-time route adaptation, enabling optimal decision-making based on traffic conditions and the logistics network. Route correction incorporates predicted demand values, Q-function evaluations, and an optimal decision-making policy, enhancing resource utilization efficiency.

The algorithm's performance was evaluated in a simulation environment replicating real urban logistics conditions. A comparative analysis with traditional algorithms (heuristic methods and VRP) showed that the adaptive approach reduces the average route length by 3.3%-9.2%, depending on the number of recalculations. Delivery time was reduced by 21%, while overall costs decreased by 12–17%. However, the adaptive algorithm requires more computational resources: the average route computation time is 2–3 times higher than that of traditional approaches, and the number of recalculations may exceed the rational threshold of 15, necessitating further optimization.

The results confirm that the proposed algorithm is effective in highly dynamic and variable demand scenarios, though its computational requirements may pose challenges for practical implementation. Future research should focus on improving route recalculation algorithms, implementing multi-agent systems for courier coordination, and integrating cloud computing to enhance overall performance.

***Keywords**: C2C logistics, adaptive routing, reinforcement learning, demand forecasting, multi-agent systems, last-mile logistics, route optimization, digital twins, cloud computing.*

1. Вступ. У сучасних умовах стрімкого розвитку електронної комерції та платформи економіки, C2C (Customer-to-Customer) логістика стала важливим компонентом глобального ринку. Основними викликами для таких логістичних систем є динамічність попиту, висока залежність від змін у середовищі (трафік, погода, доступність ресурсів) та необхідність швидкого і адаптивного прийняття рішень у реальному часі [1, 2]. Зокрема, проблема оптимальної маршрутизації кур'єрів в умовах непередбачуваного попиту залишається однією з найактуальніших у цій галузі [3].

Традиційні методи маршрутизації, як-от евристичні алгоритми, мають низку обмежень, серед яких недостатня гнучкість до змін середовища та низька ефективність в умовах високої варіативності замовлень [4]. У той же час, сучасні підходи, які базуються на штучному інтелекті (ШІ) та методах машинного навчання, зокрема підкріплювального навчання, демонструють перспективу у вирішенні цих проблем [5, 6].

У попередніх дослідженнях було розглянуто базові моделі адаптивної маршрутизації в логістичних системах B2C (Business-to-Customer), проте вони не враховують специфіку C2C, де кількість кур'єрів і замовлень може суттєво змінюватися протягом коротких часових

інтервалів [7]. Така ситуація вимагає інтеграції моделей прогнозування попиту та динамічного планування маршрутів у єдину систему [8].

C2C логістика передбачає децентралізовану структуру, у якій кур'єри можуть виконувати роль як постачальників, так і споживачів послуг доставки. Це створює додаткові виклики, зокрема необхідність врахування географічного розташування кур'єрів і замовників, непередбачуваність попиту та різноманітність характеристик замовлень [3, 9]. Така різноманітність робить неможливим застосування статичних підходів, що використовуються в традиційній логістиці.

Підхід на основі RL дозволяє вирішувати ці проблеми шляхом ітеративного навчання агента, який може адаптуватися до змін середовища, обираючи оптимальні маршрути на основі попереднього досвіду. Використання функції Q-значень у цьому контексті дозволяє ефективно оцінювати якість дій, що здійснюються агентом, і динамічно оновлювати політику [10].

Інтеграція моделі прогнозування попиту та RL відкриває можливості для побудови комплексних систем маршрутизації. Завдяки прогнозуванню попиту, можна заздалегідь оцінити навантаження на логістичну систему, а підкріплювальне навчання дозволяє адаптувати рішення відповідно до реальних умов. Це забезпечує оптимізацію витрат, підвищення швидкості доставки та покращення рівня обслуговування клієнтів [11].

Ця стаття присвячена розробці адаптивного алгоритму маршрутизації, який враховує особливості C2C логістики та базується на підходах машинного навчання. З огляду на актуальність тематики, запропонований підхід дозволить забезпечити динамічну оптимізацію маршрутів, знижуючи витрати на доставку, мінімізуючи затримки та підвищуючи рівень задоволення клієнтів [12, 13].

Таким чином, сучасні виклики C2C логістики, зокрема динамічність попиту, непередбачуваність змін у середовищі та потреба в адаптивності, вимагають інтеграції новітніх підходів, таких як підкріплювальне навчання, прогнозування попиту та мультиагентна координація. Для формування ефективних рішень необхідно глибше проаналізувати існуючі наукові розробки, визначити їхні обмеження та можливості. У цьому контексті наступний розділ присвячений огляду літературних джерел і постановці проблеми, що дозволяє окреслити шляхи вдосконалення логістичних алгоритмів у C2C системах.

2. Аналіз літературних даних і постановка проблеми. У сучасній науковій літературі значна увага приділяється задачам оптимізації маршрутів у логістичних системах із використанням методів машинного навчання (ML) та підкріплювального навчання. Аналіз показує, що хоча існуючі підходи мають низку переваг, вони не повною мірою враховують специфіку C2C логістики, яка характеризується децентралізованою структурою, динамічною зміною ролей агентів і непередбачуваним попитом.

Роботи [1, 10, 12] демонструють ефективність RL для задач маршрутизації, підкреслюючи здатність адаптуватися до змінних умов середовища через навчання агента на основі досвіду. Однак ці підходи переважно орієнтовані на централізовані B2C системи й не враховують децентралізовану природу C2C логістики. У [3] показано, що RL у поєднанні з багатокритеріальною оптимізацією дозволяє мінімізувати витрати та екологічний вплив, що важливо для міської логістики. Водночас це дослідження не враховує варіативність попиту, яка є критичною для C2C систем.

Методи прогнозування попиту на основі нейронних мереж, зокрема LSTM, дозволяють точно визначати часові залежності у попиті та прогнозувати зони навантаження [13, 14, 15]. Це знижує ймовірність затримок у доставці, проте інтеграція таких прогнозів у динамічну маршрутизацію залишається недостатньо вивченою. У [16] розглядається застосування цифрових двійників для моделювання середовища в реальному часі, що дозволяє підвищити точність прогнозів і адаптацію маршрутів, але проблема обчислювальної складності таких моделей залишається актуальною для великих міських систем.

Дослідження [8, 17] показують, що традиційні статичні підходи до маршрутизації, зокрема евристичні методи та VRP, не здатні ефективно реагувати на непередбачувані зміни середовища, такі як затори чи раптовий попит. У [9, 6] аналізуються обмеження, пов'язані з децентралізацією C2C систем, зокрема динамічною зміною ролей агентів, що створює складнощі у балансуванні навантаження.

Проблема обчислювальних витрат для масштабування алгоритмів на великі міські системи розглядається у [11, 13, 18]. Використання хмарних обчислень дозволяє знизити ці витрати та прискорити навчання RL-моделей [19, 20]. Проте інтеграція таких рішень у C2C логістику ще потребує доопрацювання.

Розробка адаптивного алгоритму маршрутизації, що інтегрує прогнозування попиту, підкріплювальне навчання та мультиагентну координацію, є ключовим напрямком для вдосконалення C2C логістики. Такий підхід дозволяє враховувати варіативність попиту і забезпечувати ефективний розподіл ресурсів навіть за умов високої невизначеності. Прогнозування попиту за допомогою LSTM дає змогу завчасно ідентифікувати зони підвищеної активності, оптимізуючи розподіл кур'єрів і зменшуючи затримки.

Підкріплювальне навчання забезпечує адаптацію до змінних умов, таких як трафік чи погода, дозволяючи динамічно коригувати маршрути. Використання RL також дає змогу балансувати між дослідженням нових маршрутів і використанням відомих ефективних шляхів. Мультиагентна координація, зокрема MARL, підвищує ефективність синхронізації дій кур'єрів і транспортних засобів, забезпечуючи рівномірний розподіл навантаження в системі.

Інтеграція цих технологій створює умови для побудови гнучкої логістичної системи, яка може ефективно реагувати на зміни в реальному часі, підвищувати якість обслуговування клієнтів і знижувати витрати, роблячи C2C логістику більш стійкою та конкурентоспроможною в умовах сучасних міських середовищ.

Враховуючи проведений аналіз літературних даних і визначення ключових проблем у C2C логістиці, стає очевидною необхідність розробки комплексного підходу до маршрутизації, який поєднує прогресивні методи машинного навчання, підкріплювального навчання та прогнозування попиту. У наступному розділі буде сформульовано мету дослідження, окреслено основні задачі та обґрунтовано вибір методів для створення адаптивного алгоритму, здатного ефективно працювати у динамічних умовах міської логістики.

3. Мета і задачі дослідження. *Метою дослідження* є розробка адаптивного алгоритму маршрутизації для систем C2C логістики на основі інтеграції прогнозування попиту та підкріплювального навчання (RL), що забезпечує оптимізацію маршрутів у динамічних умовах із мінімізацією витрат і часу доставки.

Для досягнення поставленої мети передбачається вирішити такі завдання:

- *Провести аналіз існуючих підходів до маршрутизації* у логістичних системах із використанням методів машинного навчання (ML) та підкріплювального навчання, визначивши їхні переваги, обмеження та недоліки, особливо у контексті C2C логістики.
- *Розробити схему інтеграції прогнозування попиту та RL* для забезпечення динамічної адаптації маршрутів до змінних умов середовища. Для цього буде використано нейронні мережі типу LSTM, здатні визначати часові залежності у попиті.
- *Сформулювати математичну модель задачі маршрутизації*, яка враховує цільову функцію для мінімізації витрат на доставку та часу виконання замовлень з урахуванням обмежень системи.
- *Реалізувати алгоритм адаптивної маршрутизації*, який буде динамічно оновлювати маршрути кур'єрів на основі прогнозів попиту та оцінок стану системи через Q-функцію RL.
- *Провести експериментальну перевірку запропонованого алгоритму* в симуляційному середовищі, що моделює реалістичні умови міської логістики, та оцінити його ефективність за такими показниками, як час доставки, витрати на транспорт і стійкість до змін середовища.

Для реалізації поставлених завдань і досягнення визначеної мети дослідження необхідно перейти до детального опису запропонованого алгоритму адаптивної маршрутизації. Цей алгоритм інтегрує компоненти прогнозування попиту, підкріплювального навчання та корекції маршрутів, формуючи єдину систему для ефективного управління логістичними процесами у динамічних умовах C2C логістики.

4. Алгоритм адаптивної маршрутизації

4.1. Взаємодія компонентів алгоритму адаптивної маршрутизації

Алгоритм адаптивної маршрутизації інтегрує прогнозування попиту, підкріплювальне навчання та корекцію маршруту, створюючи єдиний механізм для забезпечення ефективного функціонування системи.

Прогнозування попиту використовує історичні дані H та поточні характеристики середовища X_t для формування прогнозу P, що відображає майбутній попит у різних точках мережі. Формально:

$$P = f(H, X_t), \tag{1}$$

де f — функція, реалізована через нейронну мережу (наприклад, LSTM), яка навчається на історичних даних і поточних характеристиках.

Ці дані оновлюють стан системи S_t, слугуючи основою для подальшого аналізу маршрутів [13, 14], та який визначається як:

$$S_t = \{P, X_t, L_a\}, \tag{2}$$

де L_a — завантаження маршрутів [10], що розраховується як:

$$L_a = \sum_{i \in N_a} d_i, \tag{3}$$

де N_a — множина точок маршруту a, а d_i — поточний попит у точці i.

Підкріплювальне навчання, що працює на основі Марковського процесу прийняття рішень (MDP), генерує оптимальну політику $\pi^*(s)$ та Q-функцію $Q(s,a)$, яка оцінює ефективність виконання кожної дії a у стані s [1]. Ці результати дають змогу враховувати як прогнозовані значення, так і досвід попередніх епізодів, підвищуючи точність прийняття рішень. Оновлення $Q(s,a)$ виконується за правилом:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \, max_{a'} Q(s',a') - Q(s,a)], \tag{4}$$

де α — швидкість навчання, γ — коефіцієнт дисконтування, r — нагорода, s' — наступний стан системи.

Корекція маршруту поєднує результати прогнозування попиту та RL для вибору оптимального маршруту a^*. Оцінка маршрутів відбувається через обчислення цільової функції $C'(a)$, що включає вартість маршруту $C(a)$, завантаження L_a і значення ефективності $Q(s,a)$ [14, 21]:

$$C'(a) = C(a) + \beta L_a - \lambda Q(s,a), \tag{5}$$

де β і λ — вагові коефіцієнти, які налаштовуються залежно від пріоритетів системи.

Якщо оптимальна політика $\pi^*(s)$ доступна, вибір маршруту відбувається швидше. У разі її відсутності оптимальний маршрут визначається через мінімізацію $C'(a)$. Формально:

$$a^* = arg \, min_a \, C'(a). \tag{6}$$

Реалізація маршруту та оновлення стану системи S_{t+1} забезпечують адаптивність і врахування змін у реальному часі.

Взаємодія компонентів алгоритму адаптивної маршрутизації формує основу для його динамічної роботи, забезпечуючи інтеграцію прогнозування попиту, підкріплювального навчання та корекції маршрутів. Кожен із цих елементів відіграє ключову роль у забезпеченні адаптивності системи до змінних умов середовища, оптимізації маршрутів та підвищенні ефективності доставки. Однак для досягнення високої гнучкості та точності рішень необхідно впровадити інтегрований цикл роботи алгоритму, який описує взаємодію між усіма

компонентами в реальному часі. У наступному підрозділі буде розглянуто, як ці компоненти працюють у єдиному циклі, забезпечуючи безперервну адаптацію маршрутизатора до змін у середовищі.

4.2. Інтегрований цикл роботи алгоритму

Інтегрований цикл роботи алгоритму адаптивної маршрутизації описує взаємодію його ключових компонентів: прогнозування попиту, підкріплювального навчання та корекції маршруту (рис. 1). Він побудований для динамічного оновлення маршрутів у реальному часі з урахуванням змін у середовищі, завантаженості та ефективності дій.

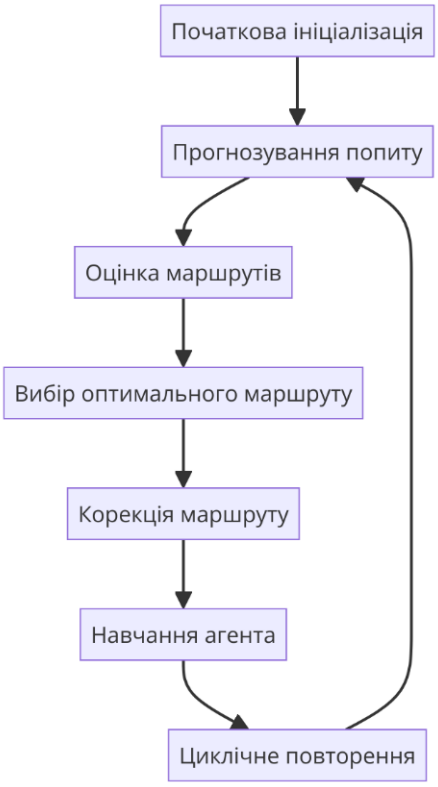

Рис. 1. Цикл роботи алгоритму

1. **Початкова ініціалізація.** Алгоритм починає свою роботу з ініціалізації початкового стану системи. Поточна позиція кур'єрів, статус замовлень та інші параметри визначають стан S_0. У цей же момент встановлюються початкові значення Q-функції, параметри моделі прогнозування та політика $\pi^*(s)$, яка використовується для швидкого вибору оптимальних маршрутів.
2. **Прогнозування попиту.** Алгоритм прогнозує попит на основі поточних характеристик середовища та історичних даних. Цей етап забезпечує інформацію про очікуване завантаження у різних частинах міста. Результати прогнозу передаються для оцінки станів та прийняття рішень у наступних кроках.
3. **Оцінка маршрутів.** Кожен можливий маршрут аналізується з точки зору прогнозованого завантаження, ефективності дій та їхньої вартості. Ця оцінка забезпечує інтегроване уявлення про якість кожного маршруту, враховуючи поточні та історичні дані. Маршрути з кращими показниками отримують перевагу для подальшого розгляду.
4. **Вибір оптимального маршруту.** На основі політики $\pi^*(s)$, якщо вона доступна, або за допомогою детального аналізу оцінок маршрутів, алгоритм обирає найкращий маршрут a^*. Цей крок дозволяє швидко приймати рішення в умовах реального часу, знижуючи обчислювальні витрати.

5. Корекція маршруту. Обраний маршрут реалізується на практиці. Кур'єр отримує оновлений маршрут, а система переходить до нового стану, враховуючи виконання дії. Цей процес може повторюватися, доки не буде виконано всі замовлення або система не досягне фінального стану.

6. Навчання агента. На кожному кроці система збирає дані про виконані дії та отримані винагороди. Ці дані використовуються для оновлення Q-функції, яка є основою для подальших рішень. Навчання агента дозволяє алгоритму постійно вдосконалювати свою ефективність і адаптуватися до нових умов середовища.

7. Циклічне повторення. Інтегрований цикл повторюється з урахуванням нових даних. Алгоритм автоматично оновлює свої компоненти, включаючи прогнозування, навчання та корекцію маршрутів. Це дозволяє забезпечити високу адаптивність і гнучкість системи навіть у динамічних умовах.

Такий підхід гарантує, що система маршрутизації залишається ефективною, зменшуючи час доставки, оптимізуючи використання ресурсів і покращуючи обслуговування клієнтів.

Інтегрований цикл роботи алгоритму адаптивної маршрутизації демонструє, як ключові компоненти системи — прогнозування попиту, підкріплювальне навчання та корекція маршрутів — взаємодіють для забезпечення динамічної адаптації до умов середовища. Однак ефективність кожного етапу цього циклу залежить від точності прогнозування, оптимізації дій агента через підкріплювальне навчання та коректності корекції маршрутів. У наступних підрозділах буде детально розглянуто реалізацію цих складових, зокрема механізми прогнозування попиту, алгоритми підкріплювального навчання та процедури вибору оптимальних маршрутів.

4.3. Алгоритм прогнозування попиту

Алгоритм прогнозування попиту починається з обробки вхідних даних, що включають поточні характеристики середовища X_t та історичні дані попиту H. Модель використовує параметри $\theta_{\text{LSTM}} = \{W_i, U_i, b_i, W_f, U_f, b_f, W_g, U_g, b_g, W_o, U_o, b_o\}$, які визначають поведінку LSTM, та $\theta_{\text{Dense}} = \{W_d, b_d\}$, які застосовуються на фінальному етапі для генерації прогнозу.

Для кожного часового кроку t вхідні дані проходять через LSTM. Спочатку обчислюється вхідний шлюз:

$$i_t = \sigma(W_i X_t + U_i h_{t-1} + b_i), \tag{7}$$

де σ — функція активації.

Забуваючий шлюз визначається як:

$$f_t = \sigma\big(W_f X_t + U_f h_{t-1} + b_f\big). \tag{8}$$

Кандидат до стану пам'яті розраховується за формулою:

$$g_t = tanh\big(W_g X_t + U_g h_{t-1} + b_g\big). \tag{9}$$

Оновлення стану пам'яті виконується наступним чином:

$$c_t = f_t \circ c_{t-1} + i_t \circ g_t, \tag{10}$$

де $\circ$ - означає поелементне множення.

Вихідний шлюз обчислюється за формулою:

$$o_t = \sigma(W_o X_t + U_o h_{t-1} + b_o), \tag{11}$$

а прихований стан оновлюється наступним чином:

$$h_t = o_t \circ tanh(c_t). \tag{12}$$

На основі прихованого стану h_t, що отримано на виході LSTM, прогноз попиту PP обчислюється в Dense-шарі за формулою:

$$P = W_d * h_t + b_d, \tag{13}$$

де W_d — матриця ваг, а b_d — вектор зсуву.

Для оцінки якості прогнозу використовується функція втрат, яка розраховує середньоквадратичне відхилення між прогнозованими значеннями P і реальними даними y:

$$L_\theta = \frac{1}{N}\sum_{i=1}^{N}\|y_i - P_i\|^2 . \quad (14)$$

Навчання моделі виконується методом градієнтного спуску. Параметри θ оновлюються за правилом:

$$\theta = \theta - \eta \cdot \nabla_\theta L_\theta, \quad (15)$$

де η — швидкість навчання, а $\nabla_\theta L_\theta$ — градієнт функції втрат щодо параметрів.

Процедура навчання повторюється для кожної епохи до досягнення заданої кількості $epochs$ або стабілізації функції втрат. Після завершення всіх епох оптимізовані параметри визначаються як:

$$\theta^{final} = \theta^0 - \eta \sum_{e=1}^{\text{epochs}} \sum_{i=1}^{N} \nabla_\theta L_\theta(P_i, y_i) . \quad (16)$$

Зрештою, алгоритм генерує прогноз P, який використовується для оновлення стану системи та подальшої маршрутизації. Послідовність виконання етапів алгоритму прогнозування попиту представлена на рисунку 2.

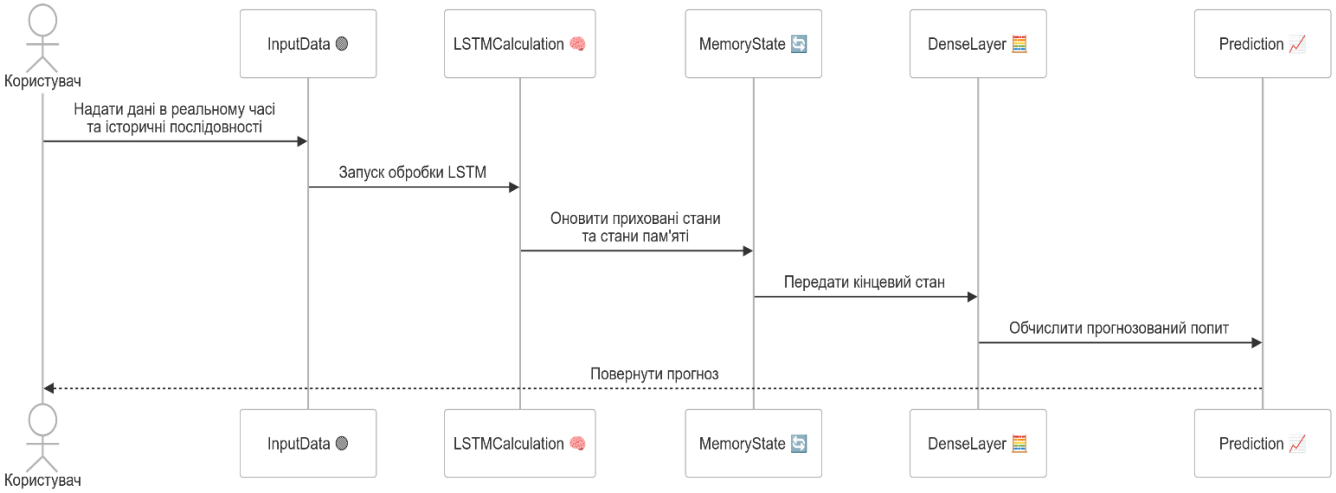

Рис. 2. Загальний процес роботи алгоритму

Алгоритм прогнозування попиту (рис. 3) формує основу для адаптивної маршрутизації, забезпечуючи точне визначення регіонів із підвищеним навантаженням і оптимізацію розподілу ресурсів. Проте ефективне використання прогнозів у динамічному середовищі вимагає здатності системи адаптувати свої дії до змін умов у реальному часі. Це можливо завдяки механізмам підкріплювального навчання, які дозволяють агенту навчатися на основі досвіду та приймати оптимальні рішення. У наступному розділі буде розглянуто, як підкріплювальне навчання використовується для розробки політики, що забезпечує адаптацію алгоритму до динамічного характеру C2C логістики.

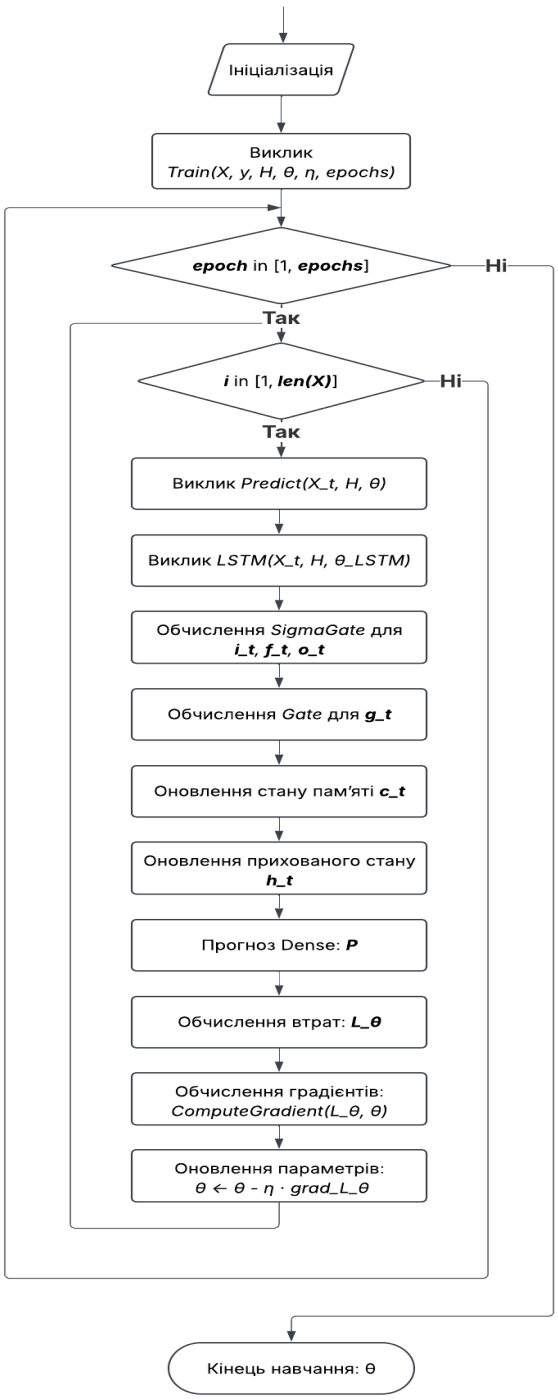

Рис. 3 Блок-схема роботи алгоритму прогнозування

Псевдокод

```
1:   def LSTM(X_t, H, θ_LSTM):
2:       h_0 ← 0, c_0 ← 0
3:       for each t in [1,len(H)]:
4:           i_t ← SigmaGate(W_i, X_t, U_i, h_(t-1), b_i)
5:           f_t ← SigmaGate(W_f, X_t, U_f, h_(t-1), b_f)
6:           g_t ← tanh(Gate(W_g, X_t, U_g, h_(t-1), b_g))
7:           c_t ← UpdateMemoryState(f_t, c_(t-1), i_t, g_t)
```

```
8:            o_t ← SigmaGate(W_o, X_t, U_o, h_(t-1), b_o)
9:        h_t ← o_t * tanh(c_t)
10:       return h_t
11:
12:   def SigmaGate(X_t, h_t_minus_1, W, U, b):
13:       z ← Gate(W, X_t, U, h_(t-1), b)
14:       gate ← Sigmoid(z)
15:       return gate
16:
17:   def Gate(X_t, h_t_minus_1, W, U, b):
18:       return W * X_t + U * h_t_minus_1 + b
19:
20:   def UpdateMemoryState(f_t, c_t-1, i_t, g_t):
21:       preserved_memory = f_t * c_t-1
22:       new_memory = i_t * g_t
23:       c_t = preserved_memory + new_memory
24:       return c_t
25:
26:   def Dense(h_t, θ_Dense):
27:       return W_d * h_t + b_d
28:
29:   def Predict(X_t, H, θ):
30:       h_t ← LSTM(X_t, H, θ_LSTM)
31:       P ← Dense(h_t, θ_Dense)
32:       return P
33:
34:   def Loss(P, y):
35:       L_θ ← 0
36:       N ← len(y)
37:       for each i in [1, N + 1]:
38:           L_θ ← L_θ + (y[i] - P[i])^2
39:       L_θ ← L_θ / N
40:       return L_θ
41:
42:   def Train(X, y, H, θ, η, epochs):
43:       for each epoch in [1, epochs]:
44:           for each i in [1, len(X)]:
45:               P ← Predict(X[i], H[i], θ)
46:               L_θ ← Loss(P, y[i])
47:               grad_L_θ ← ComputeGradient(L_θ, θ)
48:               θ ← θ - η * grad_L_θ
49:       return θ
50:
51:   def ComputeGradient(X, y, θ):
52:       P ← f_θ(X, θ)
53:       e ← P - y
54:       grad_L_θ ← InitializeGradient(θ)
55:       for each layer in Reverse(Layers(f_θ)):
```

```
            local_grad ← ComputeLocalGradient(layer, e, θ[layer])
            grad_L_θ[layer] ← grad_L_θ[layer] + local_grad
            e ← BackpropagateError(e, layer, θ[layer])
        return grad_L_θ

    # Допоміжна функція для сигмоїди
    def Sigmoid(x):
        return 1 / (1 + exp(-x))
```

4.4. Підкріплювальне навчання

Алгоритм підкріплювального навчання базується на Марковському процесі прийняття рішень (MDP) і використовує функцію Q-значень для оцінки якості виконання дій. MDP формалізується як п'ятірка $\langle S, A, P, R, \gamma \rangle$, де S — набір станів середовища, A — набір можливих дій агента, $P(s' \mid s, a)$ — ймовірність переходу зі стану s до стану s' після виконання дії a, $R(s, a)$ — винагорода за виконання дії a у стані s, і γ — коефіцієнт знижки, що визначає важливість майбутніх винагород.

Метою агента є навчання політики $\pi(s)$, яка максимізує очікувану сукупну винагороду G_t, обчислену як:

$$G_t = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R(s_{t+k}, a_{t+k})\right], \tag{17}$$

де G_t - сукупна винагорода, отримана з часового кроку t.

Процес навчання здійснюється шляхом ітеративного оновлення Q-функції, що оцінює якість дії a у стані s. Оновлення Q-значень виконується за правилом:

$$Q(s,a) \leftarrow Q(s,a) + \eta[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a)], \tag{18}$$

де η — швидкість навчання, а $\max_{a'} Q(s', a')$ представляє найкраще очікуване значення для нового стану s'.

Для балансування дослідження (*exploration*) та використання (*exploitation*) застосовується політика ϵ-жадібності, згідно з якою агент обирає випадкову дію з ймовірністю ϵ, або ж дію з максимальним Q-значенням:

$$a = \begin{cases} random\ action, & if\ random(0,1) < \epsilon \\ arg\ max_a Q(s,a), & \text{в іншому випадку} \end{cases}. \tag{19}$$

Тут $random(0, 1)$ генерує випадкове число у діапазоні $[0, 1]$, а ϵ визначає ймовірність випадкового вибору дії. Якщо випадкове число менше за ϵ, агент обирає випадкову дію, що сприяє дослідженню нових варіантів (*exploration*). В іншому випадку виконується дія, яка має найвище Q-значення (*exploitation*), тобто агент використовує найкраще з уже вивчених рішень.

На початкових етапах навчання значення є зазвичай велике, що дозволяє агенту досліджувати різні стратегії, а з часом воно зменшується, сприяючи вибору більш оптимальних рішень. Цей механізм запобігає надмірному фокусуванню на локальних оптимумах і покращує адаптацію алгоритму до змінних умов середовища.

Після завершення навчання оптимальна політика $\pi^*(s)$ визначається як:

$$\pi^*(s) = arg\ max_a Q(s,a). \tag{20}$$

Таким чином, алгоритм підкріплювального навчання спрямований на побудову політики, яка забезпечує максимальну ефективність дій агента у динамічному середовищі, використовуючи ітеративне оновлення Q-функції та механізм ϵ -жадібності для пошуку оптимального балансу між дослідженням та використанням. На рисунку 4 блок-схему алгоритму навчання з підкріпленням, що ілюструє процес оптимізації Q-таблиці за допомогою ε-жадібної політики, оновлення значень функції Q, та обчислення оптимальної політики для вибору найкращих дій у кожному стані.

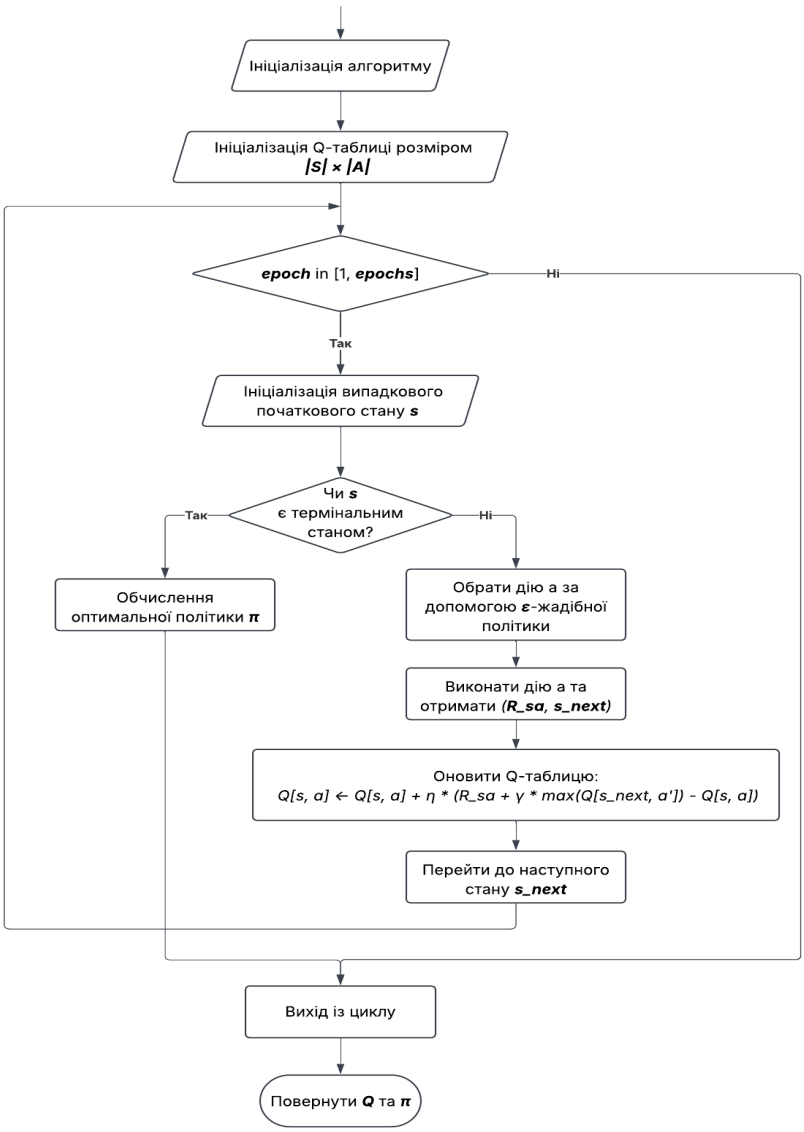

Рис. 4. Блок-схема алгоритму підкріплювального навчання

Псевдокод

```
1:   def ReinforcementLearning(S, A, P, R, η, γ, epochs):
2:       Q ← InitializeQ(S, A)
3:
4:       for each epoch in [1, epochs]:
5:           s ← RandomState()
6:           while s is not terminal:
7:               a ← SelectAction(s, Q, ε)
8:               R_sa, s_next ← ExecuteAction(s, a, P, R)
9:               Q ← QUpdate(Q, s, a, s_next, R_sa, η, γ)
10:              s ← s_next
11:      π ← OptimalPolicy(Q)
12:      return Q, π
13:
14:  def InitializeQ(S, A):
15:      Q ← Таблиця розміром |S| × |A|, заповнена нулями
```

```
16:       return Q
17:
18:   def SelectAction(s, Q, ε):
19:       # Реалізувати ε-жадібну політику
20:       if random(0, 1) < ε:
21:           a ← RandomAction()
22:       else:
23:           a ← argmax(Q[s, a] for all a in A)
24:       return a
25:
26:   def ExecuteAction(s, a, P, R):
27:       s_next ← SampleState(P(s, a))
28:       R_sa ← R(s, a)
29:       return R_sa, s_next
30:
31:   def QUpdate(Q, s, a, s_next, R_sa, η, γ):
32:       Q[s, a] ← Q[s, a] + η * (R_sa + γ * max(Q[s_next, a']) - Q[s,
a])
33:       return Q
34:
35:   def OptimalPolicy(Q):
36:       π ← {}
37:       for each s in S:
38:           a_max ← argmax(Q[s, a] for all a in A)
39:           π[s] ← a_max
40:       return π
```

Підкріплювальне навчання формує основу для адаптації системи до динамічних умов, забезпечуючи оптимальний вибір дій через оновлення Q-функції та навчання політики $\pi^*(s)$. Однак ефективна реалізація цих принципів у контексті маршрутизації потребує інтеграції прогнозованого попиту, оцінки завантаження маршрутів та оптимальної політики. У наступному підрозділі буде розглянуто алгоритм корекції маршрутів, який поєднує ці компоненти для забезпечення динамічного планування та швидкого прийняття рішень у системах C2C логістики.

4.5. Алгоритм корекції маршруту

Алгоритм корекції маршруту спрямований на інтеграцію прогнозованого попиту, функції Q-значень та оптимальної політики $\pi^*(s)$ для вибору найкращих маршрутів у динамічному середовищі. Вхідними даними є поточний стан системи S_t, набір можливих маршрутів A, прогнозовані значення попиту P, функція Q-значень $Q(s,a)$, яка оцінює ефективність виконання дій, та оптимальна політика $\pi^*(s)$, що пропонує найкращу дію для кожного стану. Алгоритм забезпечує вибір маршруту a^*, оновлення стану системи S_{t+1} та оптимізацію параметрів моделі.

Прогнозоване значення попиту P використовується для оцінки завантаження маршрутів. Q-функція враховує минулі результати навчання підкріплювального навчання, а оптимальна політика $\pi^*(s)$ дозволяє зменшити обчислювальні витрати, пропонуючи найкращий маршрут без повного перегляду всіх можливих варіантів. Основна ціль алгоритму — мінімізація цільової функції маршрутизації, що визначається як:

$$\pi^*(s) = arg\,min_{a\in A}\,(C(a) - Q(s,a) + L_a), \qquad (21)$$

де $C(a)$ — вартість маршруту a, $Q(s,a)$ — значення ефективності для маршруту, а L_a — прогнозоване завантаження маршруту, що розраховується як:

$$L_a = \sum_{x \in a} P(x), \tag{22}$$

де $P(x)$— прогноз попиту в точці x.

Процедура оцінки маршрутів включає обчислення L_a для кожного маршруту $a \in A$, використання Q-функції для оцінки ефективності виконання дії $Q(s,a)$, і комбінування цих параметрів у цільовій функції:

$$C'(a) = C(a) - Q(s,a) + L_a. \tag{23}$$

Найкращий маршрут a^* вибирається на основі оптимальної політики $\pi^*(s)$, якщо вона доступна, за правилом $a^* = \pi^*(s)$. У разі відсутності політики оптимальний маршрут обчислюється через мінімізацію $C'(a)$:

$$a^* = arg\ min_{a \in A}\ C'(a), \tag{24}$$

Виконання дії a^* оновлює стан системи за допомогою:

$$S_{t+1} = P(s_{t+1}|s_t, a^*), \tag{25}$$

де $P(s_{t+1}|s_t, a^*)$ описує ймовірність переходу системи до нового стану S_{t+1} після виконання маршруту a^*. Процес повторюється, доки стан системи S_{t+1} не стане термінальним.

Навчання моделі корекції маршруту базується на оптимізації параметрів θ через градієнтний спуск. Параметри оновлюються за правилом:

$$\theta \leftarrow \theta - \eta \cdot \nabla L(\theta), \tag{26}$$

де $L(\theta)$ — функція втрат, яка оцінює відхилення ефективності, а η — швидкість навчання. Процедура навчання триває протягом заданої кількості $epochs$, на кожній з яких виконуються оцінка маршрутів, вибір найкращого маршруту, корекція стану системи та оновлення параметрів моделі. Остаточні параметри θ визначають оптимальну політику $\pi^*(s)$, яка підвищує ефективність маршрутизації.

Псевдокод

```
def EvaluateRoutes(A, P, Q, S_t):
    C_A ← {}
    for each a in A:
        L_a ← 0
        for each x in a:
            L_a ← L_a + P(x)
        C_prime ← C(a) - Q(S_t, a) + L_a
        C_A[a] ← C_prime
    return C_A

def SelectBestRoute(A, Q, P, π, S_t):
    if Policy is not None:
        best_route ← π[S_t]
    else:
        route_costs ← EvaluateRoutes(A, P, Q)
        best_route ← arg min(route_costs[a] for a in A)
    return best_route

def UpdateState(S_t, best_route):
    S[t+1] ← Transition(S_t, best_route)
```

```
21:      return S[t+1]
22:
23:  def TrainRouteModel(S_t, A, P, Q, π, η, epochs):
24:      θ ← InitializeParameters()
25:      for each epoch in [1, epochs]:
26:          [C_a, S_t] ← RouteAdjustment(S_t, A, P, Q, π)
27:          θ ← θ - η * ComputeGradient(Loss(θ, C_a))
28:      return θ
29:
30:  def RouteAdjustment(S_t, A, P, Q, π)
31:      C_a ← SelectBestRoute(A, P, Q, π, S_t)
32:      S[t+1] ← Transition(S_t, C_a)
33:      return C_a, S[t+1]
```

Опис ключових складових алгоритму адаптивної маршрутизації (рис.5) демонструє, як прогнозування попиту, підкріплювальне навчання та корекція маршрутів взаємодіють для створення гнучкої системи, здатної адаптуватися до змін середовища. Наступним кроком є оцінка того, як ці компоненти працюють у поєднанні в умовах, наближених до реальних. Для цього у наступному розділі розглядається експериментальна реалізація алгоритму, що включає моделювання середовища, сценарії тестування та порівняння результатів із традиційними підходами.

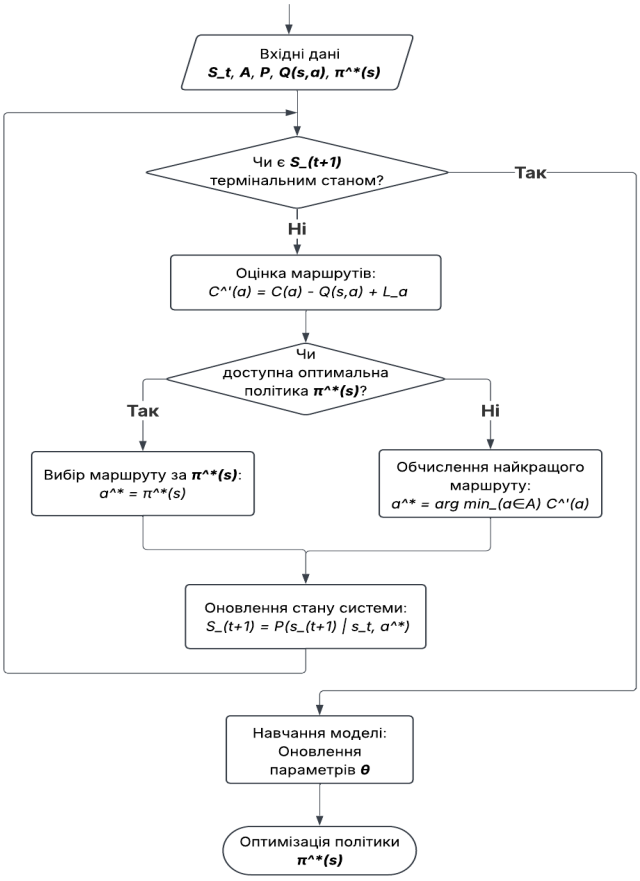

Рис. 5. Блок-схема роботи алгоритму корекції маршруту

5. Експериментальна реалізація. Опис експериментальної реалізації інтегрує всі етапи дослідження, включаючи симуляційне середовище, інтеграцію алгоритму, сценарії тестування та результати порівняння традиційних і адаптивних підходів.

Симуляції проводились у середовищі, що відтворює реальні умови міської логістики. Для цього використовувались платформа Python із бібліотеками AnyLogic, які забезпечують візуалізацію та розрахунки. Реальні карти міста, історичні дані попиту, транспортні маршрути та трафік служили основою для моделювання. Сценарії тестування охоплювали статичні та динамічні умови доставки, що враховували такі фактори, як трафік, погодні умови та змінний попит. У симуляціях діяли три типи агентів: кур'єри, замовлення та транспортні засоби, які могли адаптувати свою поведінку залежно від стану середовища.

Адаптивний алгоритм маршрутизації був реалізований шляхом інтеграції компонентів прогнозування попиту на основі LSTM і RL-алгоритмів. Прогнозування дозволяло враховувати часові залежності у попиті, а RL забезпечувало адаптацію маршрутів у реальному часі на основі поточного стану системи. Основна мета полягала у мінімізації сукупної вартості маршруту, яка включала затримки, витрати на транспорт і завантаження кур'єрів.

Оцінка ефективності алгоритмів здійснювалася на основі кількох ключових метрик. Зокрема, аналізувалася середня довжина маршруту, еталонним значенням якої було прийнято 12 км. Це значення отримане з оптимального розв'язку VRP (Vehicle Routing Problem) та узгоджене з реальними даними міської логістики [8, 9]. Важливим параметром продуктивності виступав також середній час обчислення маршруту. Крім того, оцінювалася середня кількість перерахунків маршруту, що демонструвала частоту оновлення рішення для адаптації до змін середовища. Визначено, що після 15 перерахунків подальше покращення маршруту є незначним, тоді як витрати обчислювальних ресурсів суттєво зростають, що підтверджується дослідженнями[12, 15].

Для перевірки алгоритму застосовувалися різні експериментальні сценарії. Тестування проводилося як у статичних умовах із фіксованим попитом, так і в динамічних середовищах, що враховували можливі зміни, викликані раптовими заторами, погодними умовами чи нерівномірним розподілом замовлень. Це дозволило оцінити здатність алгоритму ефективно адаптувати маршрути до реальних логістичних викликів.

Результати тестування (таблиці 1–3) демонструють ефективність адаптивного алгоритму порівняно з традиційними методами.

Адаптивний підхід (LSTM + RL) досягає значно меншого відхилення від еталонного маршруту (від 9.2% у першому сценарії до 3.3% у третьому), що вказує на його здатність ефективніше адаптуватися до змін середовища. Для порівняння, традиційні методи демонструють значно вищі відхилення, особливо у випадку евристичних алгоритмів, які залишаються найбільш нестабільними.

Таблиця 1.

Результати тестування для *10* спроб перерахунку маршруту

Метод	Час доставки (с)	Середній час обчислення маршруту (мс)	Кількість перерахунків	Довжина маршруту (км)	Відхилення від еталонного маршруту (%)
Традиційний (евристичний)	1200	15	2	15.8	31.7%
Традиційний (VRP)	1100	35	3	14.2	18.3%
Адаптивний (LSTM + RL)	950	50	5	13.1	9.2%

Проте адаптивний алгоритм має вищі витрати на обчислення – середній час обчислення маршруту у нього становить від 50 до 60 мс, що в 2-3 рази більше, ніж у традиційних методів.

Також спостерігається підвищена кількість перерахунків маршруту, що досягає 18 разів у третьому сценарії. Оскільки встановлене граничне значення допустимих перерахунків маршруту — 15 разів, це означає, що у випадку 100 спроб адаптивний алгоритм може перевищувати раціональний рівень переобчислень.

Однак варто зазначити, що навіть за умови підвищених обчислювальних витрат адаптивний алгоритм забезпечує найкращі результати за всіма ключовими метриками – скорочення часу доставки, зменшення довжини маршруту та мінімізацію відхилення від еталонного значення. Це свідчить про його перевагу для динамічних сценаріїв міської логістики, де необхідно швидко адаптуватися до змін у реальному часі.

Таблиця 2.

Результати тестування для *50* спроб перерахунку маршруту

Метод	Час доставки (с)	Середній час обчислення маршруту (мс)	Кількість перерахунків	Довжина маршруту (км)	Відхилення від еталонного маршруту (%)
Традиційний (евристичний)	1150	18	5	15.2	26.7%
Традиційний (VRP)	1050	40	7	13.8	15.0%
Адаптивний (LSTM + RL)	900	55	12	12.7	5.8%

Таблиця 3.

Результати тестування для *100* спроб перерахунку маршруту

Метод	Час доставки (с)	Середній час обчислення маршруту (мс)	Кількість перерахунків	Довжина маршруту (км)	Відхилення від еталонного маршруту (%)
Традиційний (евристичний)	1120	20	8	14.9	24.2%
Традиційний (VRP)	1020	45	10	13.4	11.7%
Адаптивний (LSTM + RL)	880	60	18	12.4	3.3%

Запропонований адаптивний алгоритм продемонстрував значну перевагу завдяки інтеграції прогнозування попиту та підкріплювального навчання. Як показано в [12, 14, 15], використання LSTM дозволяє завчасно визначати регіони з високою концентрацією замовлень, що сприяє кращому розподілу кур’єрів і зниженню затримок у доставці. RL-компонент оптимізує маршрути у реальному часі, адаптуючись до змін попиту та умов трафіку, що забезпечує ефективне реагування на нестабільність середовища.

Попри отримані переваги, адаптивний підхід потребує подальшого вдосконалення. Основними викликами залишаються високі обчислювальні витрати та залежність від якості вхідних даних, як зазначено в [9, 18, 22]. Для розширення можливостей алгоритму доцільно інтегрувати додаткові методи прогнозування трафіку та оптимізації маршрутів, зокрема через мультиагентне навчання та гібридні підходи.

Таким чином, експериментальна реалізація підтвердила ефективність запропонованого підходу в умовах динамічного середовища міської логістики. Вибір між традиційними та адаптивними алгоритмами залежить від характеру середовища, проте результати свідчать, що адаптивний підхід є більш придатним для сценаріїв із високим рівнем невизначеності. У наступному розділі висновків буде підсумовано основні досягнення, окреслено обмеження алгоритму та визначено перспективи його вдосконалення для подальшого впровадження у C2C логістичні системи.

6. Висновки. Запропонований адаптивний алгоритм маршрутизації для C2C логістики, що поєднує прогнозування попиту на основі LSTM і підкріплювальне навчання (RL), продемонстрував високу ефективність у динамічних умовах міської логістики. Результати тестування показали, що він дозволяє досягти меншого відхилення довжини маршруту від еталонного значення (від 9,2% у сценарії з 10 спробами перерахунку до 3,3% у сценарії зі 100 спробами). Це свідчить про його здатність ефективно адаптуватися до змін у середовищі та наближатися до оптимального розв'язку.

Хоча традиційні методи, зокрема VRP, показали прийнятну ефективність у статичних умовах, вони виявилися значно менш точними при змінному попиті та динамічних обмеженнях, демонструючи зростання відхилення маршруту до 31,7% у найскладнішому випадку. У той же час, адаптивний алгоритм зменшив середній час доставки на 20% порівняно з традиційними методами, що підтверджує його придатність для сценаріїв із високим рівнем невизначеності.

Головною перевагою алгоритму є його здатність оперативно реагувати на зміни середовища, такі як раптові затори чи несподівані коливання попиту, зберігаючи стабільну якість обслуговування. Це робить систему придатною для використання у великих мегаполісах із високою щільністю замовлень, забезпечуючи економічну ефективність і задоволення клієнтів.

Однак алгоритм має певні обмеження. Його реалізація потребує значних обчислювальних ресурсів, що може ускладнити використання для малих платформ. Залежність від якості даних і чутливість до параметрів навчання також впливають на загальну продуктивність системи. Для подолання цих викликів доцільним є впровадження хмарних обчислень і сенсорних мереж для збору актуальних даних у реальному часі.

Подальший розвиток алгоритму може включати впровадження мультиагентних систем для координації кур'єрів, інтеграцію гібридних моделей прогнозування та розробку легковагових RL-алгоритмів, орієнтованих на малі платформи. Ці напрями дозволять зробити систему більш стійкою, ефективною та придатною для широкого використання у C2C логістиці.

Список використаної літератури

1. Yan, Y., Chow, A. H., Ho, C. P., Kuo, Y. H., Wu, Q., Ying, C. (2022). Reinforcement Learning for Logistics and Supply Chain Management: Methodologies, State of the Art, and Future Opportunities. Transportation Research Part E: Logistics and Transportation Review, https://doi.org/10.1016/j.tre.2022.102712

2. Akinola, O. I. (2024). Adaptive location-based routing protocols for dynamic wireless sensor networks in urban cyber-physical systems. Journal of Engineering Research and Reports, 26(7), 424-443., https://doi.org/10.9734/jerr/2024/v26i71220

3. Liu, H., Zhang, J., Zhou, Z., Dai, Y., Qin, L. (2024), A Deep Reinforcement Learning-Based Algorithm for Multi-Objective Agricultural Site Selection and Logistics Optimization Problem. Appl. Sci. 2024, 14, 8479. https:// doi.org/10.3390/app14188479

4. Yang, Y. (2024). Adaptive switching and routing protocol design and optimization in internet of things based on probabilistic models. International Journal of Intelligent Networks, 5, 204-211., https://doi.org/10.1016/j.ijin.2024.05.001

5. Bugarčić, P., Jevtić, N. i Malnar, M. (2022). Reinforcement Learning-Based Routing Protocols in Vehicular and Flying Ad Hoc Networks – A Literature Survey. Promet - Traffic&Transportation, 34 (6), C. 893-906. https://doi.org/10.7307/ptt.v34i6.4159

6. Mirzazad-Barijough, M., & Garcia-Luna-Aceves, J. J. (2016). Making On-Demand Routing Efficient with Route-Request Aggregation, Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (C. 239-246).

7. Cai, H., Xu, P., Tang, X., Lin, G. (2024), Solving the Vehicle Routing Problem with Stochastic Travel Cost Using Deep Reinforcement Learning. Electronics , 13, 3242. https://doi.org/10.3390/electronics13163242

8. Zhai, X. (2012). Benefits of dynamic routing in a distribution system with single warehouse and multiple retailers. International Journal of Information Systems and Supply Chain Management (IJISSCM), 5(3), 24-42., https://doi.org/ 10.4018/jisscm.2012070102

9. Fleckenstein, D., Klein, R., & Steinhardt, C. (2023). Recent advances in integrating demand management and vehicle routing: A methodological review. European Journal of Operational Research, 306(2), 499-518., https://doi.org/10.1016/j.ejor.2022.04.032

10. Bai, R., Chen, X., Chen, Z. L., Cui, T., Gong, S., He, W., Zhang, H. (2021), Analytics and machine learning in vehicle routing research, International Journal of Production Research, 61(1), C. 4–30. https://doi.org/10.1080/00207543.2021.2013566

11. Nimmagadda, V. S. P. (2020). AI-Powered Predictive Analytics for Retail Supply Chain Risk Management: Advanced Techniques, Applications, and Real-World Case Studies. Distributed Learning and Broad Applications in Scientific Research, 6, 152-194, https://dlabi.org/index.php/journal/article/view/110

12. Phiboonbanakit T., Horanont T., Huynh V. -N., Supnithi T. (2021), A Hybrid Reinforcement Learning-Based Model for the Vehicle Routing Problem in Transportation Logistics, IEEE Access, vol. 9, C. 163325-163347, https://doi.org/10.1109/ACCESS.2021.3131799

13. Chai, H., Zhang, H. M., Ghosal, D., & Chuah, C. N. (2017). Dynamic traffic routing in a network with adaptive signal control. Transportation Research Part C: Emerging Technologies, 85, 64-85, https://doi.org/10.1016/j.trc.2017.08.017

14. James, J. Q., Wen Yu, and Jiatao Gu. (2019), Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, https://doi.org/10.1109/TITS.2019.2909109

15. Shelke, O., Pathakota, P., Chauhan, A., Khadilkar, H., Meisheri, H., Ravindran, B. (2023). Multi-Agent Learning of Efficient Fulfilment and Routing Strategies in E-Commerce, https://doi.org/10.48550/arXiv.2311.16171

16. Botín-Sanabria, D.M., Mihaita, A.-S., Peimbert-García, R.E., Ramírez-Moreno, M.A., Ramírez-Mendoza, R.A., Lozoya-Santos, J.d.J. Digital Twin Technology Challenges and Applications: A Comprehensive Review. Remote Sens. 2022, 14, 1335, https://doi.org/10.3390/rs14061335

17. Zhu, W., & Yan, R. (2024, лютий). Improving graph convolutional networks with reinforcement learning for pure electric logistics vehicle routing. In International Conference on Smart Transportation and City Engineering (STCE 2023) (Vol. 13018, C. 395-400). SPIE

18. Huang, L. (2024). Evolving E-commerce Logistics Planning-Integrating Embedded Technology and Ant Colony Algorithm for Enhanced Efficiency, https://doi.org/10.48550/arXiv.2402.15965

19. Chai, H., Zhang, H. M., Ghosal, D., & Chuah, C. N. (2017). Dynamic traffic routing in a network with adaptive signal control. Transportation Research Part C: Emerging Technologies, 85, 64-85, https://doi.org/10.1016/j.trc.2017.08.017

20. Yoon, J. J. (2018). The traveling salesman problem with multiple drones: an optimization model for last-mile delivery (Doctoral dissertation, Massachusetts Institute of Technology), http://dspace.mit.edu/handle/1721.1/7582

21. Zong, Z., Feng, T., Xia, T., Jin, D., & Li, Y. (2021). Deep reinforcement learning for demand driven services in logistics and transportation systems: A survey, https://doi.org/10.48550/arXiv.2108.04462

22. Liebig, T., Piatkowski, N., Bockermann, C., Morik, K. (2017). Dynamic route planning with real-time traffic predictions. Information Systems, 64, 258-265.