УДК 004.41 http://doi.org/10.15407/pp2025.02.040

Г.Б. Мороз

ЗАСТОСУВАННЯ ГЛИБОКОГО НАВЧАННЯ З ПІДКРІПЛЕННЯМ ДЛЯ ДОВГОСТРОКОВОЇ ДИНАМІЧНОЇ КОМПОЗИЦІЇ ВЕБСЕРВІСІВ

У сервісно-орієнтованій архітектурі програмне забезпечення та системи абстрагуються як вебсервіси, які можуть викликатися іншими системами. Композиція сервісу – це технологія, яка будує складну систему шляхом поєднання існуючих простих вебсервісів. Із розвитком сервісно-орієнтованих архітектур та технології вебсервісів починають з'являтися масові вебсервіси з однаковими функціями. Вони обслуговуються різними організаціями та мають різну якість обслуговування. Отож, як вибрати відповідні вебсервіси, щоб уся система забезпечувала найкращу загальну якість обслуговування, стало ключовою проблемою в дослідженні композиції вебсервісів. Крім того, через складність і динаміку мережевого середовища якість обслуговування з часом може змінюватися. Отже, як динамічно налаштувати систему композиції, щоб адаптуватися до мінливого середовища та забезпечити якість складеного вебсервісу є ще однією проблемою. Для вирішення вищезазначених проблем, пропонується підхід до композиції вебсервісів, заснований на прогнозуванні якості вебсервісів та навчанні з підкріпленням. Зокрема, ми використовуємо нейронну мережу довгої короткотривалої пам'яті для прогнозування якості обслуговування, а потім робимо динамічний вибір вебсервісів за допомогою навчання з підкріпленням. Цей підхід можна добре адаптувати до динамічного мережевого середовища.

Ключові слова: Сервісно-орієнтоване обчислення, якість обслуговування, композиція вебсервісу, композитний вебсервіс, глибоке навчання з підкріпленням.

H.B. Moroz

APPLICATION OF DEEP REINFORCED LEARNING FOR LONG-TERM DYNAMIC COMPOSITION OF WEB SERVICES

In the service oriented architecture, software and systems are abstracted as web services to be invoked by other systems. Service composition is a technology, which builds a complex system by combining existing simple services. With the development of service oriented architecture and web service technology, massive web services with the same function begin to spring up. These services are maintained by different organizations and have different quality of service. Thus, how to choose the appropriate service to make the whole system to deliver the best overall quality of service has become a key problem in service composition research. Furthermore, because of the complexity and dynamics of the network environment, quality of service may change over time. Therefore, how to dynamically configure the composition system to adapt to the changing environment and ensure the quality of the composed web service is another problem. To address the above challenges, we propose a service composition approach based on quality of web service rediction and reinforcement learning. Specifically, we use long short-term memory neural network to predict the quality of web service hrough reinforcement learning**.** This approach can be well adapted to a dynamic network environment.

Keywords: Service-oriented computing, quality of service, web service composition, composite web service, deep reinforcement learning.

Вступ

Мережеві технології мають великий вплив на розробку програмного забезпечення в багатьох сферах застосування. Все більше підприємств і організацій надають своє програмне забезпечення, системи, обчислювальні ресурси та ресурси зберігання тощо у формі вебсервісів для використання іншими користувачами. Тому питання про те, як ефективно інтегрувати різні вебсервіси, стало фундаментальною проблемою дослідження.

Композиція вебсервісів (*web services composition*, WSC) в основному вивчає, як створити програмну систему, яка може задовольнити складні функціональні вимоги користувачів шляхом поєднання існуючих

вебсервісів. При цьому потрібно враховувати, що вебсервіси з однаковою функціональністю можуть надаватися різними постачальниками і з різною якістю обслуговування (*quality of service*, QoS), яка в основному використовується для позначення нефункціональних атрибутів вебсервісів, як-то ціну, зручність використання, надійність, час відгуку, репутацію, пропускну здатність мереж і так далі. Через динамічну зміну мережевого середовища, коливання продуктивності самого вебсервісу та зміну шаблонів доступу користувачів QoS також можуть динамічно змінюватися з часом. Тож, методи WSC необхідно адаптувати до динамічно змінюваного середовища [1].

Наразі процес WSC, зазвичай, представляється як робочий. Кожний абстрактний вебсервіс описує його необхідні функціональні можливості і може мати кілька *реальних сервісів-кандидатів (компонентів)*, які відповідають функціональним вимогам. WSC на основі робочого процесу зосереджується на QoS. Для кожного абстрактного вебсервісу використовуючі QoS визначається кращий реальний вебсервіс, що дозволяє отримати оптимальний результат WSC, який буде якнайбільше задовольняти вимоги користувача.

Час перебування конкретних компонентів у складі композитного вебсервісу (*composite web service*, CWS**)** сильно залежать від їхньої здатності підтримувати довгострокові вимоги до QoS [2]. Розробка довгострокових CWS породжує проблему вибору компонентів, які задовольняють вимоги до QoS протягом тривалих періодів часу. Такий вибір передбачає прогнозування довгострокових трендів QoS (тобто QoS протягом тривалого періоду). Основна проблема, пов'язана з прогнозуванням довгострокової QoS, полягає в тому, що її значення можуть коливатися в майбутньому. Існує принаймні три переваги використання довгострокових тенденцій QoS під час WSC. *По-перше*, розробники покладаються на прогнозовану QoS, щоб точно вибрати найкращі вебсервіси, які відповідатимуть вимогам до CWS протягом тривалого періоду часу. Це забезпечує довготривале перебування компонентів у складі CWS. *По-друге,* реальні вебсервіси можуть зазнавати кількох змін протягом свого життєвого циклу (як-от, припинення роботи вебсервісу), що може призвести до розриву зв'язку між CWS та його компонентами. В подальшому розробники зможуть замінити проблемні компоненти новими, з кращими характеристиками QoS. *По-третє***,** постачальники вебсервісу (наприклад, хмарні постачальники) можуть покладатися на прогноз QoS для кращого управління ресурсами та балансування робочого навантаження. Наприклад, вони можуть використовувати більше ресурсів сервера (процесор і пам'ять) у періоди пік. Точне прогнозування QoS дозволяє провайдерам налаштовувати свої хмарні ресурси, щоб якомога точніше задовольняти вимоги користувачів у майбутньому. Це знижує вірогідність порушення зв'язку між довгостроковим CWS та його компонентами через зміни QoS.

Існуючі в літературних джерелах підходи до WSC на основі QoS здебільшого зосереджені на статичних значеннях QoS, які спостерігаються під час композиції [3,4,5]. Проте на практиці через динамічну зміну мережевого середовища та самого вебсервісу його QoS буде змінюватися з часом, а, отже, такі майбутні варіації QoS необхідно враховувати під час розробки CWS.

Запропонований підхід поєднує прогнозування значення QoS та навчання з підкріпленням, де перше використовується для оцінки атрибутів QoS і покращення точності вибору вебсервісів у динамічному середовищі, а останнє забезпечує самооптимізацію та адаптивну здатність для композиції вебсервісів. Це приводить до нового адаптивного методу WSC з урахуванням QoS, яка може адаптуватися до динамічного мережевого середовища.

Суміжні роботи

Враховуючи динаміку та складність мережевого середовища, QoS вебсервіса можуть постійно змінюватися, тому необхідний метод WSC, який дозволяє сприймати зміни середовища та автоматично на-

лаштовувати систему. Ця проблема привернула широку увагу, і було розроблено низку рішень. У роботі [6] автори використовують метод штучного інтелекту для планування складу вебсервісу. Але цьому методу бракує загального контролю над усім CWS. У [7,8] автори розглядають задачу оптимізації композиції вебсервісу як задачу цілочисельного програмування, а потім отримують оптимальний розв'язок за допомогою суміжної технології. У [9, 10] автори описують зв'язки між різними вебсервісами за допомогою карти плану, моделюють композицію вебсервісів за допомогою моделі діаграми плану, а потім шукають вебсервіси низької якості та замінюють їх, використовуючи жадібний алгоритм. Автори роботи [11] використовують цілочисельне програмування для глобальної оптимізації WSC. Напочатку, завдяки використанню технології горизонту (skyline), значно зменшується кількість сервісів-кандидатів. Отож, складність цілочисельного програмування також суттєво зменшується. Так само в [12] автори використовують технологію горизонту для фільтрації вебсервісів. Основна ідея наведених вище методів полягає у вирішенні задачі композиції вебсервісу за допомогою цілочисельного програмування або змішаного цілочисельного програмування. Але такі методи застосовні лише до невеликих проблем і не мають здатності до самоадаптації. Таким чином, вони не можуть вирішити проблему композиції вебсервісів у великомасштабному та динамічному середовищі. Еволюційні алгоритми також використовувалися для вибору та композиції вебсервіса. У [13] була запропонована оптимізація мурашиної колонії для підтримки глобальної оптимізації QoS композитного вебсервіса. Але цим методом важко позбутися від проблеми локальної оптимізації. У [14] розглядався баланс між дослідженням та експлуатацією для композиції хмарного вебсервісу. Для цього стратегія орла (eagle strategy) була об'єднана з алгоритмом оптимізації кита (whale optimization). Цей метод може уникнути локального оптимуму, але він не враховує зміну QoS з часом, що впливає на адаптивність.

В останні роки одним із основних напрямків досліджень у галузі інформаційних технологій стає машинне навчання. Деякі вчені намагалися застосувати навчання з підкріпленням (*reinforcement learning*, RL) до композиції вебсервісів. У роботі [15] використовано модель марковського процесу ухвалення рішень *(Markov decision processes,* MDP) та навчання з підкріпленням для виконання адаптивної WSC. У [16] автори використовують ієрархічний алгоритм RL для підвищення ефективності композиції вебсервісів. Автори робіт [17 – 19] для збільшення ефективності та якості результату WSC застосовують багатоагентну технологію. У роботі [20] була запропоновано трирівневу модель WSC із підтримкою довіри. Використовуючи переваги методу нечіткої комплексної оцінки, автори представили інтегровану модель довірчого управління. Ця модель може добре аналізувати переваги користувачів, щоб отримати кращу композицію вебсервісу. Проте наведені вище алгоритми, засновані на навчанні з підкріпленням, хоч і мають певну адаптивність, але вони ігнорують деякі особливі проблеми в композиції вебсервісів. Більш конкретно, вебсервіси часто розгортаються в мережевому середовищі. QoS часто змінюється з часом через динамічне мережеве середовище та можливі коливання продуктивності самого вебсервісу. Тому для отримання кращих результатів композиції вебсервісів необхідно ефективно передбачити змінену якість вебсервісу. Незважаючи на те, що традиційний метод RL має певну здатність до самоадаптації, він усе ще не може точно оцінити якість вебсервісу, що призводить до незадовільних результатів композиції вебсервісів у високодинамічному середовищі. Отже, необхідно певним чином передбачити зміну QoS, чого можна досягти за допомогою її прогнозування.

Останнім часом багато вчених вивчають проблему прогнозування QoS. Найпоширенішим дослідженням є підходи на основі спільної фільтрації [21–26]. Цей вид методів передбачає, що подібні користувачі можуть отримати подібну QoS від вебсервісу. Вони класифікують користувачів за деякими функціями та забезпечують якісне

прогнозування для інших подібних користувачів, використовуючи історичні дані, створені користувачем, який послуговувався вебсервісом. Прогноз QoS на основі спільної фільтрації більше стосується відмінностей QoS між різними користувачами. Але в композиції вебсервісів робочий процес усієї композиції вебсервісів розроблений відповідно до вимог певного користувача. Кожний вебсервіс викликається цим користувачем. Зміни якості обслуговування викликані не зміною розташування або стану абонента вебсервісу, а тим, що мережеве середовище на той момент змінилося. QoS змінюється з часом і може бути описана як дані часового ряду, подібні до вартості акцій. Зважаючи на це, деякі вчені намагаються застосувати технологію прогнозування часових рядів для прогнозування QoS. У [27] запропоновано прогнозувати часові ряди QoS на основі моделі авторегресійної інтегрованої ковзної середньої. Однак ця модель вимагає стабільності даних часових рядів, тому її не можна застосовувати до сценаріїв із дуже динамічними змінами. Останнім часом, з розвитком технології глибокого навчання, деякі дослідники вивчали методи прогнозування послідовності на основі *рекурентних нейронних мереж* (RNN), які досягли певного прогресу в інших сферах [28–30]. Наприклад, у [31] RNN використовувалися для багатокрокового прогнозування, яке може відповідати ширшому діапазону шаблонів даних. Загалом методи на основі глибокого навчання краще відповідають складним функціям порівняно з традиційними статистичними моделями і можуть забезпечити кращу ефективність прогнозування у роботі зі складними часовими рядами.

Попередні відомості

У цьому розділі представлено деякі попередні відомості, включаючи навчання з підкріпленням та прогнозування часових рядів, які мають безпосереднє відношення до предмету дослідження.

Прогнозування часових рядів

Прогноз часових рядів призначений для оцінки майбутніх результатів на основі минулих даних. У сфері вебсервісних обчислень через динамічну зміну мережевого середовища та коливання продуктивності самого вебсервісу його QoS часто змінюється з часом. Ця зміна є певною закономірністю. Таким чином QoS можна розглядати як дані часових рядів, і наша мета полягає в тому, щоб передбачити майбутню QoS на основі історичних даних. Припускається, що дані часового ряду x_t задовольняють $x_t = g(T)$, де $g(T)$ втілює основні правила зміни x_t з часом. Метод прогнозування часових рядів зазвичай виходить з того, що майбутні дані можна передбачити на основі минулих даних. Тобто існує функція f, така, що

$$g(t + 1) \approx f(x_t, x_{t-1}, x_{t-2}, \ldots, x_0).$$

Метою прогнозування часових рядів є знаходження функції f, щоб майбутні дані можна було оцінити на основі даних минулих часових рядів [32].

Класичні методи прогнозування часових рядів включають метод ковзного середнього, метод експоненційного згладжування, авторегресійну модель і модель авторегресійного підсумовування ковзного середнього [33]. Загальний принцип цих методів полягає в припущені, що f є *лінійною* функцією. Це дозволяє вибрати необхідну модель відповідно до характеристик історичних даних. Він встановлює форму $f(\theta)$ із параметром θ, потім визначає параметри функції, аналізуючи історичні дані, і використовує цю функцію для прогнозування майбутніх даних. Більшість цих методів базується на короткострокових даних і припускають, що цільова функція f є лінійною, а це добре впливає на короткострокове прогнозування або прогнозування простого часового ряду. Однак, коли часові ряди змінюються складним чином, який неможливо описати лінійними моделями, точність передбачення вищезазначених моделей буде відносно низькою.

Навчання з підкріпленням

RL займає проміжне місце між контрольованим та неконтрольованим навчаннями, оскільки вимагає скалярного сигналу винагороди за серію вихідних дій. Воно добре досягає деяких цілей у невідомому середовищі, постійно коригуючи поведінку

агента, як-от, керування роботом для знаходження виходу у лабіринті. RL моделює дослідницьку поведінку людини чи інших організмів у невідомому середовищі. У такому середовищі агент постійно з ним взаємодіє, отримує зворотний зв'язок, а потім коригує власну поведінку. Кінцевою метою є максимізація винагороди, отриманої від середовища [34]. Існує багато різновидів навчання з підкріпленням. На рис.1 показана базова структура навчання з підкріпленням. Агент виконує дію a_t. Після її виконання він отримує із середовища винагороду r_{t+1}, а потім переходить у новий стан s_{t+1}. Агент буде коригувати вибір дій відповідно до винагороди, і, нарешті, сформує власну стратегію вибору дій, яка отримує максимальне значення винагороди. Фактично агент і середовище утворюють систему зі зворотним зв'язком.

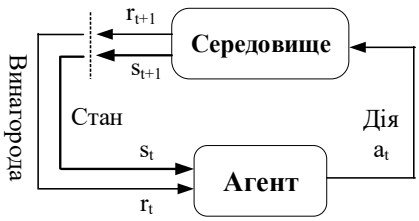

Рис. 1. Базова структура навчання з підкріпленням.

Вхідні дані алгоритму навчання з підкріпленням включають набір можливих станів середовища, допустимі дії в кожному стані, вартість винагороди та новий стан середовища, які будуть отримані в результаті виконання дії. Результатом є стратегія, яка вибирає дії на основі станів, котрі мають максимізувати очікування загальної вартості винагороди.

Зазвичай вхідними даними навчання з підкріпленням є *марковський процес ухвалення рішень* (MDP), який можна визначити як 4-кортеж MDP = ⟨S, A(·), P(·), R(·)⟩, де

S – скінченна множина станів середовища;

A(s) – скінченний набір дій, які можна виконати в стані $s \in$ S;

P($s'|s, a$) – функція розподілу ймовірностей переходу середовища від поточного стану s до наступного стану s' при виконанні дії $a \in$ A(s).

R(s, a) – функція негайної винагороди, якщо при поточному стані середовища s вибрана дії $a \in$ A(s).

Результатом навчання з підкріпленням є стратегія вибору дії, виражена як π: S → A, тобто вибір дії $a = \pi(s)$ у стані s. Найкраща стратегія вибору дій повинна максимізувати очікування загальної *цінності* винагороди. Загальна *цінність* винагороди агента (G_t), отримана за певною послідовністю *дій стану,* початковим станом якої є s_t, може бути визначена як:

$$G_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \qquad (1)$$

де r_{t+i} є винагорода, отримана при переході зі стану s до стану s' в момент часу $t+i$. Змінна γ ($0 \le \gamma \le 1$) є ставкою дисконтування, яка гарантує, що короткотривала винагорода є більшою, ніж довготривала, і вплив нещодавньої винагороди відіграє важливішу роль у виборі дії. Функція «стан- цінність» визначається як очікування випадкової величини G_t:

$$V^{\pi}(s_t) = E^{\pi}[G_t | s_t = s] = E^{\pi}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t = s\right] \qquad (2)$$

де $E^{\pi}(*)$ позначає очікуване значення випадкової величини за стратегією π. Величина $V^{\pi}(s_t)$ є очікуванням загальної винагороди від початкового стану s_t за стратегією π. Її рекурсивний вираз

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} p(s' | s, a)[r + \gamma V^{\pi}(s')] \qquad (3)$$

де p($s'|s, a$) представляє ймовірність переходу від стану s до наступного стану s'. Відповідно до формули (3) оптимальну стратегію вибору дії π^* можна виразити у вигляді:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} V^{\pi}(s), \quad \forall s \in S \qquad (4)$$

У поєднанні визначення MDP з формулами (3) та (4), кінцеву найкращу стратегію вибору дії можна також представити формулою

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\left(\sum_{s'} p(s' | s, \pi(s))[r + \gamma V^{\pi}(s')]\right) \qquad (5)$$

Як результат вищезазначеного обговорення, входом алгоритму навчання з

підкріпленням є MDP, а виходом - стратегія вибору дії $a = \pi(s)$ у стані s, яка може бути виражена формулою (5).

Q-навчання. Це широко використовуваний алгоритм RL [35]. Його основна ідея полягає в оцінці винагороди, отриманої за дію, за допомогою таблиці значень Q, де значення Q є оцінкою реальної винагороди. Якщо фактичне та оцінене значення відрізняються після виконання дії, таблиця значень Q оновлюється різницею між двома значеннями. Значення Q може поступово наближатися до реального [36]. Після певної кількості ітерацій таблиця значень Q може мати точнішу оцінку реальної вартості винагороди. Алгоритм Q-навчання використовує Q(*s, a*) для вказівки оціненої винагороди за дію *a* у стані *s*. Стратегія вибору дій задана формулою (6):

$$\pi(s) = arg\max_a Q(s,a), a \in \text{A(s)} \quad (6)$$

Відповідно до цієї стратегії реальна вартість винагороди за дію *a* на основі стану *s* може бути виражена формулою (7), де s′ представляє наступний стан, а нижній індекс «real» представляє реальну вартість винагороди, тоді як Q(*s, a*) представляє значення в таблиці значень Q для оцінки реального значення Q:

$$Q_{real}(s,a) = R(s'|s,a) + \gamma \max Q(s',a') \quad (7)$$

Значення в таблиці Q буде оновлено відповідно до формули (8).

$$Q(s,a) = Q(s,a) + \alpha\left(R(s'|s,a) + \gamma \max Q(s',a') - Q(s,a)\right) \quad (8)$$

В алгоритмі Q-навчання вибір теоретичної дії виконується за формулою (6). Але в реальному процесі навчання, якщо щоразу, коли ми вибираємо дію з найбільшим значенням Q, вона може легко впасти в локальний оптимум. Отже, у виборі дій фактичного алгоритму Q-навчання застосовується є − жадібна стратегія. Тобто вибір дії здійснюється за формулою (6) з імовірністю є та за допомогою випадкового відбору з імовірністю 1 − є. Ця стратегія може допомогти алгоритму навчання з підкріпленням досягти певного балансу між дослідженням і використанням досвіду, а також уникнути занадто ранього потрапляння в локальний оптимум. Алгоритм Q-навчання наведено в Алгоритмі 1.

Алгоритм 1: Q-навчання

Ініціалізуйте Q(*s, a*), параметри, введіть модель MDP

repeat

встановіть поточний стан s

repeat

Виберіть *a* в A(s) на основі

є − жадібної стратегії;

Виконайте *a*, отримайте винагороду $r = R(s' \mid s, a)$ і новий стан s′;

$Q(s,a) = Q(s,a) + \alpha(r + \gamma \max Q(s',a') - Q(s,a))$

$s \leftarrow s'$

until досягнення певного стану завершення

until умова закінчення алгоритму

Глибоке навчання

Глибоке навчання — це підмножина машинного навчання, в якому використовуються нейронні мережі для розпізнавання шаблонів у наданих даних для прогнозного моделювання на прихованих даних. Дані можуть бути табличними, текстовими, графічними або мовними. Існує багато типів архітектур нейронних мереж, використання яких залежить від особливостей проблеми, що розглядається. Далі коротко представлені три з них, які стосуються проблеми, що нас цікавить.

Нейронна мережа. Базова структура нейронної мережі показана на рис. 2. Вона містить вхідний, прихований та вихідний шари. Сила зв'язку між нейронами задається вагами [37].

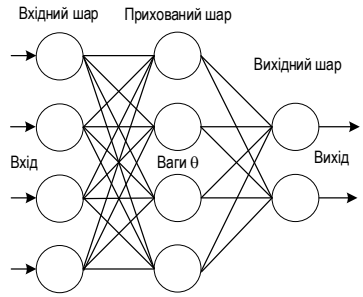

Рис. 2. Базова структура нейронної мережі.

Нейронну мережу можна представити у вигляді певної функції $y = f(x;\theta)$, де x — вхідні дані, y — вихідні дані, а θ — параметри нейронної мережі, зокрема, ваги. Вихідні дані ненавченої нейронної мережі

часто незадовільні, тобто існує помилка між виходом і очікуваним результатом. Зі зміною параметрів нейронної мережі помилка на певному наборі даних може ставати більшою або меншою. Величину такої помилки можна визначити за допомогою функції помилки L(θ). Отже, процес навчання нейронної мережі фактично полягає в знаходженні набору параметрів θ для мінімізації значення функції помилки. Для вирішення цієї проблеми часто використовують метод градієнтного спуску. Основний принцип якогополягає в тому, що напрямок градієнта є (локально) найбільш швидко змінюваним напрямком для функції, а отже, значення функції може бути зменшено через коригування незалежної змінної вздовж напрямку градієнта. Крок повторюється багато разів, поки значення функції не досягне найнижчої точки. Потім знаходиться локальне мінімальне значення. Відповідно до цього принципу оновлення параметрів нейронної мережі здійснюється за формулою:

$$\theta = \theta - \eta \frac{\partial L(\theta)}{\partial \theta} \qquad (9)$$

де η – швидкість навчання.

Рекурентні нейронні мережі. RNN є особливим типом нейронних мереж. Вони (на відміну від звичайних нейронних мереж) призначені для *обробки послідовних даних* шляхом підтримки прихованого стану, який фіксує інформацію про попередні вхідні дані [38,39]. Базова архітектура складається з вхідного шару, прихованого шару та вихідного шару. На відміну від нейронних мереж прямого зв'язку, RNN мають рекурентні з'єднання, як показано на рис. 3, що дозволяє інформації циклічно переміщатися в мережах.

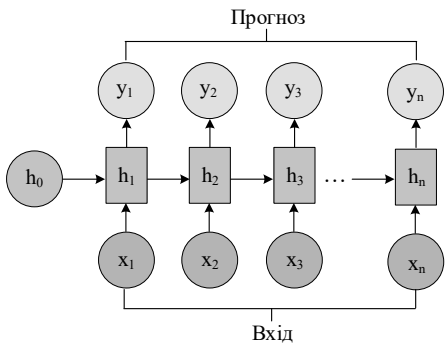

Рис. 3. Архітектура простої RNN

На кожному часовому кроці t RNN приймає вхідний вектор x_t та оновлює свій прихований стан h_t, використовуючи наступну формулу:

$$h_t = tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h),$$

де W_{xh} - вагова матриця між вхідним та прихованим шаром, W_{hh} - вагова матриця для рекурентного з'єднання, b_h - вектор зміщення, а tanh - функція гіперболічного тангенсна. Ця функція активації зводить вхідні значення до діапазону [−1, 1], роблячи її нульцентричною та придатною для моделювання послідовностей як з позитивними, так і з негативними значеннями.

Вихідний сигнал на кожному часовому кроці t визначається так:

$$y_t = \sigma(W_{hy}h_t + b_y),$$

де W_{hy} – вагова матриця між прихованим та вихідним шарами, b_y – вектор зміщення, а σ – сигмоїдальна функція активації для вихідного шару.

Нейронна мережа довгої короткотривалої пам'яті (*Long Short-Term Memory,* LSTM). Мережу LSTM було представлено Hochreiter і Schmidhuber [40] головним чином для вирішення проблеми зникнення градієнта, пов'язаної з простими RNN, що ускладнювало виявлення довготривалих залежностей у даних.

Ключовою інновацією в LSTM є використання механізмів стробування (*gating mechanisms*) для керування потоком інформації через мережу. Це дозволяє мережам LSTM підтримувати й оновлювати свій внутрішній стан протягом тривалих періодів, що робить їх ефективними для завдань, які вимагають моделювання довгострокових залежностей.

Типова внутрішня структура комірки нейронної мережі LSTM показана на рис. 4. Комірка містить три вентилі: вхідний, забувальний і вихідний, які регулюють стан комірки C_t і прихований стан h_t. Ці вентилі визначають, яку частину вхідних даних враховувати, яку частину попереднього стану потрібно забути, та яку частину стану комірки виводити.

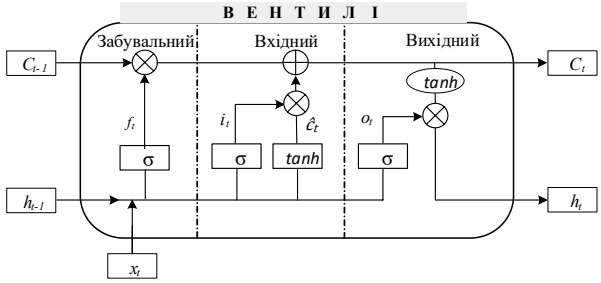

Рис. 4. Внутрішня структура комірки нейронної мережі LSTM.

Рівняння оновлення LSTM такі:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \quad (10)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (11)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (12)$$

$$\hat{c}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (13)$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes \hat{c}_t \quad (14)$$

$$h_t = o_t \otimes tanh\,(C_t) \quad (15)$$

де f_t - забувальний вентиль, i_t - вхідний вентиль, o_t - вихідний вентиль, C_t - стан комірки, $\hat{c}_t$ - стан комірки-кандидата на стан комірки, h_t - прихований стан, σ - сигмоїдна функція, tanh -функція гіперболічного тангенса, $\otimes$ - поелементний добуток, W_f, W_i, W_0, W_C - вагові матрці, а b_f, b_i, b_0, b_C − вектори зміщення.

У забувальному вентилі швидкість забувавання обчислюється за формулою (10), і знаходиться в діапазоні від 0 до 1.

Вхідний вентиль визначає, які дані потрібно додати до стану комірки. Для цього використовуються формули (11) та (13). Нарешті, новий стан комірки контролюється спільно результатами розрахунку забувального та вхідного вентилів за (14).

Вихід мережі LSTM визначається новим станом комірки, поточним входом і виходом прихованого *шару* в останній момент часу за формулою (15).

Додавши структуру з трьома вентилями до RNN, мережа LSTM може розрізняти та контролювати інформацію, запам’ятовувати важливу інформацію протягом тривалого часу та зберігати її в стані комірки. Отже, вона може бути використаною для прогнозування часових рядів QoS.

Глибоке навчання з підкріпленням (*Deep Reinforcement Learning,* DRL) виникло як нова техніка, яка поєднує методи навчання з підкріпленням та глибокого навчання. Метою DRL є вивчення оптимальних дій, які максимізують нашу винагороду за всі стани, в яких може перебувати наше середовище. Ми робимо це, взаємодіючи зі складними, багатовимірними середовищами, випробовуючи дії та навчаючись на зворотному зв'язку. Галузь глибокого навчання стосується апроксимації функцій у багатовимірних задачах; задачах, які настільки складні, що табличні методи більше не можуть знайти точні рішення [41]. Процес навчання DRL показаний на рис. 5, який розділений на три етапи.

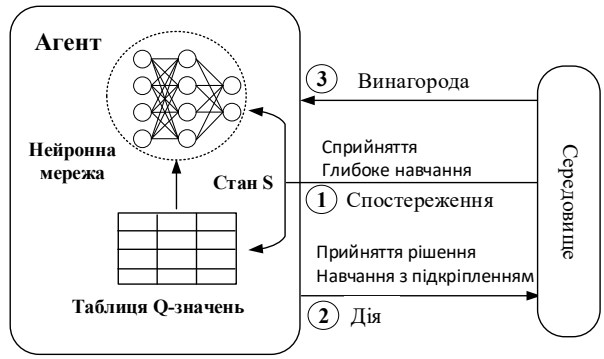

Рис. 5. Принцип глибокого навчання з підкріпленням

1. Агент отримує спостереження за допомогою взаємодії із навколишнім середовищем (через RL) і передає результати до нейронної мережі для вивчення абстрактних представлень;

2. Агент оцінює дію, виходячи з величини винагороди та зіставляє поточний стан з відповідною дією за допомогою стратегії поведінки;

3. Середовище реагує на дію та отримує наступне спостереження. Зрештою, з вивченням вищезгаданих процесів може бути досягнуто оптимальної стратегії.

Щоб RL досягло оптимального результату та конвергенції, потрібні дві важливі передумови: (1) доступна таблиця пошуку для зберігання значень Q; (2) Середовище має відповідати властивості Маркова. Перша передумова робить RL придатним лише для проблем невеликого масштабу через недостатню здатність до вираження та узагальнення. Тоді, як друга передумова передбачає, що агент чітко знає всю інформацію про навколишнє середовище, і наступний стан може бути визначений лише поточним станом і дією.

Довгострокова композиція вебсервісу на основі прогнозування QoS і DRL

У цьому розділі ми пропонуємо адаптивний метод WSC, що поєднує прогнозування QoS із навчанням з підкріпленням**.** Щоб допомогти зрозуміти проблему спочатку описується типовий сценарій композиції вебсервісу.

При WSC ми вибираємо відповідні сервіси-кандидати за допомогою RL, щоб максимізувати загальне значення QoS композитного вебсервісу. Враховуючи динаміку та складність мережевого середовища, QoS постійно змінюється, тому для її прогнозування пропонується метод на основі LSTM. Далі ми поєднуємо прогнозування QoS і навчання з підкріпленням для вирішення проблеми довгострокової композиції вебсервісу, що може покращити якість композитного вебсервісу в динамічному середовищі.

Приклад сценарію композиції вебсервісу. Типовий процес WSC показано на рис. 6, де BC_j, j=1,2,3,4 є абстрактні вебсервіси різного типу, як то готельний сервіс, авіасервіс, сервіс погоди тощо; a_i – вебсервіси-кандидати одного і того ж типу, але від різних постачальників. Для кожного абстрактного вебсервісу нам необхідно визначити його конкретні вебсервіси та сформулювати конкретний робочий процес композиції вебсервісу. Наприклад, ґрунтуючись на робочому процесі композиції вебсервісу на рис. 6, можна отримати 24 варіанти компонування вебсервісів, 2 з яких показано на рис. 7.

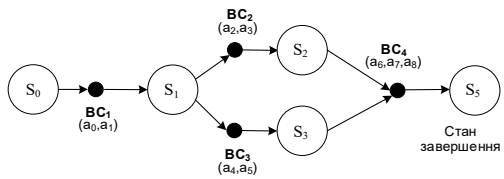

Рис. 6. Простий робочий процес створення композитного вебсервісу.

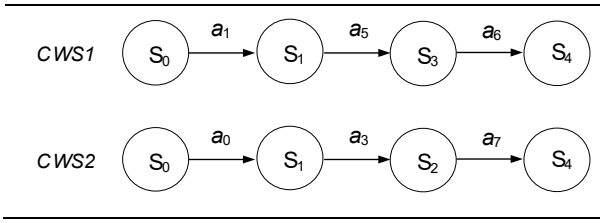

Рис. 7. Можливі результати композиції вебсервісу.

Легко побачити, що зі збільшенням кількості сервісів-кандидатів і ускладненням робочого процесу композиції вебсервісів кількість можливих результатів композиції вебсервісів різко збільшуватиметься.

Завдання композиції вебсервісу полягає в тому, щоб перевірити великий простір кандидатів і отримати оптимальний результат, який *максимізує цінність винагороди*.

Модель композиції вебсервісу в динамічному середовищі. Відповідно до проблеми, описаної в попередньому розділі, пропонується модель *динамічної композиції вебсервісу* (Dynamic Web Service Composition) на основі MDP, або коротко «модель DWSC». Ця модель являє собою 6-кортеж $DWSC=\langle S, S_0, S_r, A(\cdot), P(\cdot), R(\cdot)\rangle$, де

S – скінченна множина станів середовища;

S_0 – початковий стан, з якого починається процес композиції вебсервісу.

S_r –набір станів завершення: при досягненні будь-якого з них, процес композиції вебсервісів припиняється.

A(s) – скінченна множина вебсервісів, які можуть бути викликані в стані $s \in S$;

P(s′|*s,a*) – Імовірність переходу від поточного стану *s* до наступного стану *s′* коли викликається вебсервіс *a.*

R(s′|*s,a,t*) – функція негайної винагороди в результаті переходу середовища з поточного стану *s* до стану *s′* викликається дією *a,* де *t* представляє час виклику вебсервісу.

У складі вебсервісу цінність винагороди зазвичай визначається атрибутами QoS вебсервісу.

Для повного опису робочого процесу композиції вебсервісу можна використовувати модель DWSC. Розглянемо, як приклад, простий робочий процес на рис.6. Цей робочий процес можна описати за допомогою моделі DWSC, у якій набір станів $S = \{S_0, S_1, S_2, S_3, S_4\}$, початковий стан – S_0, набір станів завершення – $\{S_4\}$. Вебсервіси, які можна викликати в різних станах, включають $A(S_0) = \{a_0, a_1\}$ і $A(S_1) = \{a_2, a_3, a_4, a_5\}$ тощо. Приклади ймовірності переходу включають $P(S_1|S_0, a_0) = 1$, $P(S_2|S_1, a_2) = 1$ і т. д. Значення винагороди обчислюється за допомогою QoS, отриманої під час виклику вебсервісу.

Після визначення робочого процесу, процес WSC починається з початкового стану, обирається конкретний вебсервіс для кожного стану, а потім здійснюється перехід до нового стану. Коли досягнуто стану завершення, робочий процес композиції вебсервісу завершується. Цей робочий процес, який складається з кількох конкретних вебсервісів, є результатом WSC. Хороший результат композиції вебсервісів має зробити загальну цінність винагороди якомога більшою.

Функція винагороди в складі вебсервісу. Алгоритм навчання з підкріпленням видає найкращу стратегію вибору дій, використовуючи модель MDP, що робить його придатним для проблеми вибору вебсервісів. Для використання алгоритму навчання з підкріпленням, нам потрібно спочатку визначити функцію винагороди.

Оскільки під час композиції вебсервісу керуються, насамперед, вимогами користувачів до QoS, то логічно функцію винагороди визначати через атрибутами QoS. Проте нам спочатку потрібно нормалізувати ці атрибути та відобразити їх на [0, 1], так як різні атрибути QoS можуть мати різні діапазони значень. Крім того, враховуючи, що деякі атрибути QoS позитивно корельовані (наприклад, пропускна здатність), а деякі атрибути QoS корельовані негативно (наприклад, час відповіді), для визначенння винагороди будемо використовувати дві формули:

$$r = \frac{QoS - min}{max - min} \quad (15)$$

$$r = \frac{max - QoS}{max - min} \quad (16)$$

Формула (15) використовується для нормалізації атрибутів QoS, які позитивно корельовані, а формула (16) використовується для нормалізації атрибутів QoS, які корельовані негативно. Змінна *r* є результатом нормалізації значення атрибута. QoS представляє значення її атрибута після виклику вебсервісу, а max і min представляють максимальне та мінімальне значення атрибута QoS. Якщо вебсервіс має кілька атрибутів QoS, вартість винагороди буде розрахована за формулою (17).

$$R = \sum_{i=1}^{m} w_i * r_i \quad (17)$$

де R являє собою загальне значення винагороди, m - кількість розглянутих атрибутів QoS, r_i - нормалізоване значення і-го атрибута QoS, а w_i являє собою вагу і-го атрибута. Вага відображає важливість різних атрибутів. Як правило, вона налаштовується відповідно до переваг користувачів щодо різних атрибутів так, що $\sum_{i=1}^{m} w_i = 1$.

Прогнозування часових рядів на основі LSTM. У динамічному мережевому середовищі QoS часто змінюється з часом, і алгоритм навчання з підкріпленням не може оцінити тенденцію зміни якості обслуговування. У цьому розділі буде представлено метод прогнозування часових рядів на основі LSTM, щоб допомогти оцінити QoS у динамічному середовищі.

Основна ідея прогнозування часових рядів полягає в тому, щоб передбачити майбутню QoS, використовуючи минулі дані. У цій роботі використовується нейронна мережа LSTM для прогнозування

часових рядів, які можна представити таким чином:

$$x_t = LSTM(x_{t-1}, x_{t-2}, x_{t-3}, \dots, x_0; \theta), \quad (18)$$

де x_t є прогнозовані дані на наступному часовому інтервалі, а

$LSTM(x_{t-1}, x_{t-2}, x_{t-3}, \dots, x_0; \theta)$ представляє нейронну мережу LSTM із параметром θ.

Вхідними даними нейронної мережі є всі минулі дані. На практиці передбачити за всіма минулими даними часто важко. Тож, вхід нейронної мережі LSTM фактично є даними попередніх n часових інтервалів. Крім того, нейронна мережа LSTM може передбачати більше, ніж один часовий інтервал. Таким чином, передбачення нейронної мережі LSTM можна виразити формулою (19), в якій m є кількістю часових інтервалів передбачення, а n є кількістю часових інтервалів вхідних даних.

$$x_{x+m-1}, \dots, x_{t+1}, x_t = LSTM(x_{t-1}, x_{t-2}, \dots, x_{t-n}; \theta) \quad (19)$$

Після визначення входу та виходу нейронної мережі можна побудувати нейронну мережу LSTM, подану на рис. 8, де показано розширення нейронної мережі LSTM у часі. Тобто нейронна мережа, представлена на рис.8 як A, є тією самою нейронною мережею, і дані передаються зліва направо в послідовності часу. На фазі введення дані про n часових інтервалів по черзі вводяться в нейронну мережу LSTM.

Вихідні дані останньої нейронної мережі також будуть частиною поточних вхідних даних. На етапі виведення нейронна мережа LSTM спочатку виводить дані прогнозу x_t. Після цього нейронна мережа більше не має вхідних даних із зовнішніх даних (оскільки це тепер майбутній час, який більше не має фактичних даних). Вона приймає лише прогнозоване значення з попереднього моменту як вхід поточного моменту, як показано на рис. 8, і виводить прогнозоване значення на кожному кроці. Коли отримано m прогнозованих значень, завдання прогнозування завершується.

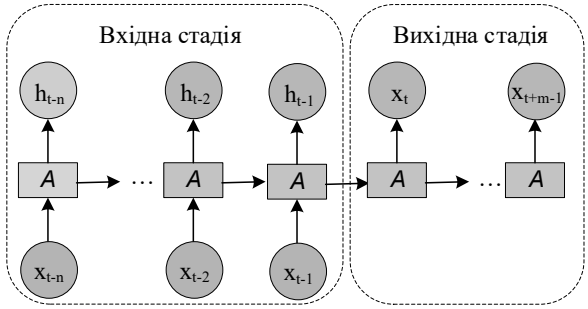

Рис. 8. Структура LSTM для прогнозування часових рядів.

Після побудови нейронної мережі LSTM для її навчання потрібна велика кількість історичних даних. Навчальний набір даних потрібно сегментувати, і кожні безперервні n+m точки даних використовуються як навчальні дані. Перші n даних є вхідними даними нейронної мережі, а дані останніх m часових кроків представляють очікувані результати прогнозування для обчислення помилки нейронної мережі. Помилка обчислюється за формулою (20), в якій x_t представляє фактичне значення моменту t, x'_t представляє прогнозоване значення в момент t, а N є кількістю точок даних у навчальному наборі. Використана функція похибки – це середня квадратична похибка:

$$L = \frac{1}{N}\sum_{t=0}^{N}(x'_t - x_t)^2 \quad (20)$$

Після навчання метод градієнтного спуску у формулі (9) використовується для налаштування параметрів нейронної мережі. Процес навчання повторюється багато разів, поки значення помилки не стане стабільним на мінімумі. Після навчання нейронну мережу можна використовувати для прогнозування майбутнього QoS.

Адаптивний алгоритм композиції вебсервісу. Розроблено новий адаптивний алгоритм композиції вебсервісів, який поєднує навчання з підкріпленням і прогнозування QoS. Перш ніж представити алгоритм, нам потрібно вирішити дві проблеми оцінки QoS у композиції вебсервісу. По-перше, у композиції вебсервісу якість сервісу-кандидата пов'язана не лише з атрибутом QoS самого вебсервісу, а й з подальшим робочим процесом, що є результатом вибору цього вебсервісу. Наприклад,

хоча вебсервіс має кращі атрибути QoS, QoS у подальшому процесі з вибором цього вебсервісу може бути поганим, що призводить до незадовільної загальної якості композиції вебсервісу. Тому в оцінці QoS ми використовуємо алгоритм Q-навчання та значення Q для представлення QoS у композиції вебсервісу. Відповідно до формули значення Q (7), значення Q фактично є лінійною суперпозицією кожного значення винагороди за вебсервіс, тому його також можна оцінити методом прогнозування часових рядів. Крім того, якщо значення Q оцінюється за допомогою нейронної мережі LSTM, формулу (7) потрібно переписати так:

$$Q_{real}(s,a) = R(s'|s,a) + \gamma \max LSTM(s',a') \quad (21)$$

де LSTM(s′, a') представляє прогнозоване значення нейронної мережі LSTM, що відповідає стану s′ і вебсервісу a′, яка є оцінкою значення Q для пари стан–дія (s', a').

До того ж, із запуском алгоритму композиції вебсервісу нові дані продовжуватимуть додаватися до набору історичних даних. Взагалі нейронна мережа навчається на фіксованому навчальному наборі, а потім застосовується до нових даних. Але в процесі створення вебсервісу історичні дані поступово накопичуються, тому потрібне поступове навчання нейронної мережі. Навчання нейронної мережі необхідно проводити, якщо додається певна кількість нових даних. Крім того, з накопиченням історичних даних вони можуть займати багато місця і також впливатимуть на ефективність навчання нейронної мережі. Тому необхідно обмежити розмір набору історичних даних і відкинути попередні історичні дані, що може підвищити ефективність навчання нейронної мережі. Детальний алгоритм наведено в Алгоритмі 2.

Алгоритм 2. ***WSC на основі прогнозування QoS і навчання з підкріпленням***

Ініціалізація LSTM(*s,a*), випадкових параметрів дослідження є, довжину вхідних даних n, ємність історичних даних d, поріг оновлення LSTM u; вхідної моделі DWSC;
repeat
  Встановити s = s_0;
  repeat
    Вибрати вебсервіс a за є − жадібною стратегією (використання LSTM для оцінки якості обслуговування);
    Викликати a, отримати винагороду за формулою (17), перейти в стан s′;
    Обчислити значення Q за формулою (21), додати ці дані до набору історичних даних;
  if кількість даних у наборі історичних даних set = d, **then**
    Відмова від найдавніших історичних даних;
  end if
  if кількість нових даних у наборі історичних даних = u,
    then
      Навчіть LSTM за формулою (20);
  end if
  s ← s′;
  until досягти стану завершення;
until Збіжність (або до заданої кількості циклів);

Алгоритм закінчується. (Прогнозне значення QoS кожного сервісу може визначити найкращий сервіс-кандидат в кожному стані для завершення композиції вебсервісу)

Висновок і майбутня робота.

Композиція вебсервісу пропонує один із найважливіших способів повторного використання програмного забезпечення шляхом поєднання існуючих вебсервісів для створення нових систем, які відповідають складним вимогам користувачів. Зі збільшенням кількості функціонально схожих вебсервісів виникає необхідність вибору відповідних вебсервісів із великого пулу кандидатів для формування якісного композитного вебсервісу. Однак у динамічному мережевому середовищі QoS не є статичним, і якість може коливатися з часом. Таким чином, метод композиції вебсервісів потрібно динамічно коригувати, щоб адаптуватися до змін навколишнього середовища. Запропонований підхід враховує особливості динамічної зміни QoS. Для оцінки часових рядів QoS розроблено рекурентну модель на основі нейронної мережі. Модель DWSC використовується для формального представлення складу вебсервісу. Розроблено адаптивний алгоритм композиції вебсервісів, який інтегрує навчання з підкріпленням і динамічне прогнозування QoS.

Запропонований метод адаптивної композиції вебсервісів на основі навчання з підкріпленням використовує таблицю значень Q

і рекурентні нейронні мережі. Це вимагає встановлення відповідної рекурентної нейронної мережі для кожного можливого вебсервісу-кандидата, що призводить до високої обчислювальної вартості для великої кількості вебсервісів-кандидатів. Для цієї проблеми ми плануємо вивчити глибоке навчання з підкріпленням і використати відображення нейронної мережі, щоб замінити таблицю Q-значення в нашій майбутній роботі. Ми також плануємо вивчити багатоагентну систему, де кілька агентів спілкуються та координують один одного для підвищення ефективності.

References

1. Sangsanit, W. Kurutach, S. Phoomvuthisarn, REST web service composition: A survey of automation and techniques, in: 2018 International Conference on Information Networking, ICOIN, 2018, pp. 116–121.
2. Kritikos, K., Pernici, B., Plebani, P., Cappiello, C., Comuzzi, M., Benbernou, S.: I, B., Kertŕesz, A., Parkin, M., Carro, M.: A survey on service quality description. ACM Comput. Surv. 46(1), 1:1–1:58 (2013)
3. Mistry, S., Bouguettaya, A., Dong, H., Qin, A.K.: Predicting dynamic requests behavior in long-term iaas service composition. In: IEEE International Conference on Web Services (ICWS) 2015, pp. 49–56. IEEE (2015)
4. Wang, S., Zhu, X., Yang, F.: Efficient QoS management for qos-aware web service composition. Int. J. Web Grid Serv. 10(1), 1–23 (2014)
5. Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. 30(5), 311–327 (2004)
6. Y. Yan, P. Poizat, L. Zhao, Repair vs. recomposition for broken service compositions, in: International Conference on Service-Oriented Computing, Springer, 2010, pp. 152–166.
7. D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Trans. Softw. Eng. 33 (6) (2007) 369–384.
8. F. Li, L. Zhang, Y. Liu, Y. Laili, QoS-aware service composition in cloud manufacturing: a Gale-Shapley algorithm-based approach, IEEE Trans. Syst.Man Cybern. Syst. (2018) 1–12.
9. Y. Yan, P. Poizat, L. Zhao, Self-adaptive service composition through graphplan repair, in: Web Services (ICWS), 2010 IEEE International Conference on, IEEE, 2010, pp. 624–627.
10. Y. Yan, P. Poizat, L. Zhao, Repairing service compositions in a changing world, in: Software Engineering Research, Management and Applications 2010, Springer, 2010, pp. 17–36.
11. M. Alrifai, D. Skoutas, T. Risse, Selecting skyline services for QoS-based web service composition, in: Proceedings of the 19th International Conference on World Wide Web, ACM, 2010, pp. 11–20.
12. V.A. Permadi, B.J. Santoso, Efficient skyline-based web service composition with QoS - awareness and budget constraint, in: 2018 International Conference on Information and Communications Technology, ICOIACT, 2018, pp.855–860.
13. O. Hammas, S.B. Yahia, S.B. Ahmed, Adaptive web service composition insuring global QoS optimization, in: 2015 International Symposium on Networks, Computers and Communications, ISNCC, IEEE, 2015, pp. 1–6.
14. S.K. Gavvala, C. Jatoth, G. Gangadharan, R. Buyya, QoS-aware cloud service composition using eagle strategy, Future Gener. Comput. Syst. 90 (2019) 273–290.
15. H. Wang, X. Zhou, X. Zhou, W. Liu, W. Li, A. Bouguettaya, Adaptive service composition based on reinforcement learning, in: International Conference on Service-Oriented Computing, Springer, 2010, pp. 92–107.
16. H. Wang, G. Huang, Q. Yu, Automatic hierarchical reinforcement learning for efficient large-scale service composition, in: 2016 IEEE International Conference on Web Services, ICWS, 2016, pp. 57–64.
17. H. Wang, X. Chen, Q. Wu, Q. Yu, Z. Zheng, A. Bouguettaya, Integratingon-policy reinforcement learning with multi-agent techniques for adaptive service composition, in: Service-Oriented Computing, Springer, 2014, pp. 154–168.
18. Y. lei, P.S. Yu, Multi-agent reinforcement learning for service composition, in: 2016 IEEE International Conference on Services Computing, SCC, 2016, pp. 790–793.
19. P. Kendrick, T. Baker, Z. Maamar, A. Hussain, R. Buyya, D. Al-Jumeily, An efficient multi-cloud service composition using a distributed multiagentbased, memory-driven approach, IEEE Trans. Sustain. Comput. (2019) 1–13.
20. W. Li, J. Cao, K. Hu, J. Xu, R. Buyya, A trust-based agent learning model for service composition in mobile cloud computing environments, IEEE Access 7 (2019) 34207–34226.

21. D. Billsus, M.J. Pazzani, Learning collaborative information filters, in: Icml, vol. 98, 1998, pp. 46–54.
22. J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: SIGIR '99: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15–19, 1999, Berkeley, CA, USA, 1999, pp. 230–237.
23. H.N. Kim, A.T. Ji, I. Ha, G.S. Jo, Collaborative filtering based on collaborative tagging for enhancing the quality of recommendation, Electron. Commer. Res. Appl. 9 (1) (2010) 73–83.
24. K. Chen, H. Mao, X. Shi, Y. Xu, A. Liu, Trust-aware and location-based collaborative filtering for web service QoS prediction, in: 2017 IEEE 41st Annual Computer Software and Applications Conference, COMPSAC, vol. 2, 2017, pp. 143–148.
25. C. Wu, W. Qiu, X. Wang, Z. Zheng, X. Yang, Time-aware and sparsitytolerant QoS prediction based on collaborative filtering, in: 2016 IEEE International Conference on Web Services, ICWS, 2016, pp. 637–640.
26. G. Zou, M. Jiang, S. Niu, H. Wu, S. Pang, Y. Gan, QoS aware web service recommendation with reinforced collaborative filtering, in: International Conference on Service-Oriented Computing, Springer, 2018, pp. 430–445.
27. R.N. Calheiros, E. Masoumi, R. Ranjan, R. Buyya, Workload prediction using ARIMA model and its impact on cloud applications x2019; QoS IEEE Trans. Cloud Comput. 3 (4) (2015) 449–458.
28. I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.
29. A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Acoustics, Speech and Signal Processing (Icassp), 2013 Ieee International Conference on, IEEE, 2013, pp. 6645–6649.
30. R.-G. Cirstea, D.-V. Micu, G.-M. Muresan, C. Guo, B. Yang, Correlated time series forecasting using multi-task deep neural networks, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, ACM, 2018, pp. 1527–1530.
31. L. Yunpeng, H. Di, B. Junpeng, Q. Yong, Multi-step ahead time series forecasting for different data patterns based on LSTM recurrent neural network, in: 2017 14th Web Information Systems and Applications Conference, WISA, 2017, pp. 305–310.
32. J.D. Hamilton, Time Series Analysis, Princeton University Press, Princeton, NJ, 1994.
33. J. Contreras, R. Espinola, F.J. Nogales, A.J. Conejo, ARIMA Models to predict next-day electricity prices, IEEE Trans. Power Syst. 18 (3) (2003) 1014–1020
34. R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, Adaptive computation and machine learning, MIT Press, 1998
35. D.E. Goldberg, J.H. Holland, Genetic algorithms and machine learning, Mach. Learn. 3 (2) (1988) 95–99.
36. R.H. Crites, A.G. Barto, Elevator group control using multiple reinforcement learning agents, Mach. Learn. 33 (2–3) (1998) 235–262.
37. Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
38. A. Karpathy, J. Johnson, L. Fei-Fei, Visualizing and understanding recurrent networks, 2015, ArXiv preprint, arXiv:1506.02078.
39. C. Goller, A. Kuchler, Learning task-dependent distributed representations by backpropagation through structure, in: Neural Networks, 1996, IEEE International Conference on, vol. 1, IEEE, 1996, pp. 347–352.
40. S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
41. A. Plaat, Deep Reinforcement Learning, Springer Nature, 2022.

Одержано: 30.06.2025
Внутрішня рецензія оримана: 07.07.2025
Зовнішня рецензія оримана: 09.07.2025

Про автора:
Мороз Григорій Борисович,
кандидат технічних наук,
старший науковий співробітник,
провідний науковий співробітник.
https://orcid.org/0000-0001-8666-9503

Місце роботи автора:
Інститут програмних систем
НАН України
тел. +38-044-526-60-33
E-mail: moroz170@gmail.com