УДК 004.8, 005.94 https://doi.org/10.15407/pp2025.01.082

І.Ю. Гришанова, Ю.В. Рогушина

МЕТОД ПОБУДОВИ ТА ОПТИМІЗАЦІЇ МАРШРУТУ ДЛЯ КОМПОЗИТНОГО ВЕБСЕРВІСУ НА ОСНОВІ Q-LEARNING

Запропоновано метод автоматизованої генерації маршруту для композиції вебсервісів відповідно до поставленої цілі з використанням методів машинного навчання з підкріпленням (reinforcement learning). Агент, використовуючи Q-learning, поступово накопичує знання про середовище та оновлює оцінку корисності своїх дій, яким відповідають наявні сервіси.

Задача поділяється на дві підзадачі: побудову можливих маршрутів – таких послідовностей сервісів, що результати виконання попереднього сервісу змінюють середовище, уможливлюючи виконання наступного; та вибір оптимального маршруту відповідно до критеріїв QoS, що адаптується до змін самого середовища.

Визначено основні складові навчання з підкріпленням, розглянуто їхню специфіку для аналізу сервісів. Розглянуто додаткові підходи, які дозволяють уникнути зациклення та використання непотрібних сервісів. Запропоновано модифікацію методу Q-learning для автоматичної генерації маршрутів на основі вхідних і вихідних даних вебсервісів і для вибору оптимального маршруту на основі аналізу їхніх якісних характеристик з використанням підходу з пам'яттю, де агент із кожним кроком розширює свої знання про середовище. Запропоновано програмну реалізацію розробленого методу, яка дозволяє оцінити його властивості.

Розглянуто можливості запропонованого методу на прикладі генерації оптимальних послідовностей вивчення матеріалів для індивідуальних освітніх траєкторій відповідно до особистих потреб студентів. Кожен навчальний об'єкт (інформаційний об'єкт, що використовується для освітніх потреб та описаний метаданими) розглядається як окремий сервіс, де входи та виходи представлені необхідними та результуючими компетенціями.

Ключові слова: вебсервіс, композиція сервісів, маршрут, машинне навчання.

I. Grishanova, J. Rogushina

FLOW CONSTRUCTING AND OPTIMIZING METHOD FOR COMPOSITE WEB SERVICE BASED ON Q-LEARNING

We propose a method of automated flow generation for the web services composition according to the defined target state based on reinforcement machine learning. An agent that uses Q-learning gradually accumulates knowledge about the environment to updates the evaluations of the usefulness of its actions (these actions correspond to the existing services).

The task is divided into two subtasks: - construction of possible flows represented as sequences of services where the results of the previous service execution change the current environment state and enable the execution of the next service; - choice of the optimal flow according to the history of interactions and to QoS criteria that is adapted to environment changes.

We determine the main components of reinforcement learning and analyze their specifics for service composition task. Additional approaches that allow avoiding looping and the use of unnecessary services are considered. We propose modification of the Q-learning method developed for automatic generation of flows based on input and output data of web services and for selecting the optimal flow based on the analysis of their qualitative characteristics. This modified method uses approach with memory where the agent expands its knowledge about the environment at each step. We consider characteristics of proposed method based on analysis of its software implementation.

Possibilities of proposed method are considered on example of generation an optimal study sequences used for individual educational trajectories in accordance with the personal needs of students. Every learning object (information object used for educational needs described by metadata) is considered as a specific service where inputs and outputs are represented by required and result competencies.

Keywords: web service, composition of services, flow, machine learning.

1. Вступ

У сучасних системах вебсервіси є ключовим компонентом для реалізації інтеграційних рішень. Вебсервіси забезпечують стандартизований інтерфейс для взаємодії між системами, а їх об'єднання дозволяє створювати складні композитні сервіси, що використовують кілька окремих вебсервісів для досягнення певної функціональності [1, 2].

Композиція вебсервісів – це процес вибору набору сервісів та послідовності їх виконання, що дозволяє перейти з наявного стану у бажаний (цільовий) стан. Водночас враховуються як функціональні (входи, виходи, як саме перетворюються дані), так і нефункціональні (QoS – швидкість роботи, вартість, доступність тощо) властивості сервісів [3]. Етапами створення композиції вебсервісів є генерація можливих маршрутів сервісів, що викликаються для досягнення цільового стану, та вибір серед них оптимального (ми використовуємо термін "*маршрут*" (flow) – за аналогією з теорією графів, де так позначається послідовність вершин та орієнтованих ребер – для опису послідовності виконання сервісів). Такий маршрут є основою для подальшої композиції сервісів.

Одним із сучасних підходів до вирішення задачі автоматизованої композиції вебсервісів є використання методів машинного навчання (Machine Learning, ML), які дозволяють ефективно адаптувати алгоритм до змін у середовищі, оптимізувати вибір сервісів на основі параметрів QoS та враховувати складні залежності між даними та процесами.

ML – це галузь штучного інтелекту, що досліджує алгоритми й статистичні моделі, які дозволяють комп'ютерам автоматично поліпшуватися під час виконання завдань, спираючись на накопичені дані та досвід, без явного опису алгоритму [4]. Тобто комп'ютерні програми можуть покращувати свою продуктивність, аналізуючи дані й досвід, а не виконуючи заздалегідь прописані інструкції. Сучасні алгоритми ML оптимізують продуктивність з точки зору певного критерію на основі накопиченого досвіду [5]. *Навчання з підкріпленням (reinforcement learning, RL)* – підклас ML, що використовує зворотний зв'язок у вигляді винагороди або штрафу залежно від корисності певної дії для досягнення обраної цілі [6].

В рамках базової RL-системи виділяють наступні складові:

Агент – це програма (наприклад, Q-learning), що шукає оптимальний шлях для досягнення цільового стану, тобто оцінює, які дії слід виконати у певному стані, щоб максимізувати винагороду і наблизитись до цільового стану. Агент навчається, удосконалюючи свої стратегії для досягнення максимальної винагороди в довгостроковій перспективі.

Середовище – оточення агента, з яким він взаємодіє. Воно змінює стан у відповідь на дії агента й генерує винагороду, яка стимулює або карає агента залежно від того, наскільки його дії сприяють досягненню мети.

Стратегія – правила, за якими агент обирає дії в кожному стані. Ці правила агент змінює в результаті навчання, щоб знайти дії для максимізації сумарної винагороди.

Функція винагороди – правило, яке визначає, яку винагороду отримує агент за дію в конкретному стані та забезпечує зворотний зв'язок, вказуючи агенту, чи його дії наближають до мети, і впливає на оновлення стратегії.

Функція цінності – оцінка впливу стану або пари "стан-дія" на винагороду, яка дозволяє агенту оцінити вигідність своїх дій не лише на короткостроковій, а й на довгостроковій основі.

Правила переходу між станами – визначення того, як середовище змінює свій стан після виконання агентом певної дії. Ці правила можуть бути як детермінованими, так і стохастичними.

Спостереження – інформація про стан середовища, доступна агенту залежно від типу середовища (повна або часткова).

Ці компоненти описують замкнений цикл взаємодії в RL-системі (агент

обирає дію → середовище реагує, змінюючи стан і надаючи винагороду → агент оновлює свою стратегію на основі отриманого досвіду), який повторюється, поки агент не навчиться оптимально виконувати завдання або досягати мети в середовищі.

У методі навчання з підкріпленням Q-learning агент будує таблицю Q-значень для різних станів і дій, поступово знаходячи оптимальну стратегію через спроби й помилки. Таблиця Q-значень (або Q-таблиця) — це матриця, яка використовується в алгоритмі Q-learning для зберігання оцінок корисності виконання певної дії в конкретному стані середовища: рядки відповідають станам, стовпці – можливим діям. Значення Q(s,a) в таблиці показує очікувану винагороду, яку агент отримає, почавши зі стану s, виконавши дію a, і дотримуючись оптимальної політики в подальшому. Завдяки тому, що агент постійно оновлює свої знання на основі отриманих винагород, він здатний адаптуватися до змін у середовищі, що робить його ефективним інструментом для задач, де середовище є динамічним або частково відомим.

2. Постановка задачі

У більшості відомих рішень з композиції вебсервісів побудова маршруту не автоматизована. Ми пропонуємо:

- алгоритм автоматичної побудови маршруту за описами характеристик вебсервісів та вимогами до композитного сервісу;

- метод вибору оптимального маршруту на основі порівняння властивостей за визначеним набором критеріїв QoS .

Для вирішення задачі композиції вебсервісів методом Q-learning ми пропонуємо змоделювати та описати цю задачу через поняття агента і навколишнього середовища, де агент вивчає, які сервіси обирати для досягнення цільового результату (отримати вихідні параметри композитного сервісу).

Побудова композитного сервісу з оптимальними параметрами QoS має враховувати додаткові вимоги:

- уникнення використання зайвих і непотрібних сервісів;

- уникнення зациклювання;

- спрощення введення початкових даних, щоб термінальні стани, які не є цільовими, обчислювалися автоматично. Досягнення таких термінальних станів означає помилку і призводить до закінчення навчання;

- адаптація до можливих змін характеристик сервісів та критеріїв QoS;

- масштабування методу для великої кількості сервісів;

- оптимізація знайденого маршруту.

3. Елементи композитного вебсервісу

Ми пропонуємо наступне визначення вебсервісу: *вебсервіс WS* – це п’ятірка

$$WS = <ID, IN, OUT, QoS, Descr>,$$

де:

ID – ідентифікатор вебсервісу, унікальний атрибут: він дозволяє однозначно ідентифікувати кожен вебсервіс у системі;

IN – множина вхідних параметрів $<inp_1; inp_2; \ldots; inp_n>$, $i = \overline{1,N}$ яка описує передумови для виконання веб-сервісу: чітке визначення вхідних даних дозволяє зрозуміти, які саме параметри має отримати вебсервіс для свого виконання;

OUT – множина вихідних параметрів $<out_1; out_2; \ldots; out_m>$, $m = \overline{1,M}$, яка описує результат роботи вебсервісу та використовується для створення композитного сервісу (виходи одного вебсервісу можуть використовуватися як входи для іншого);

QoS – кортеж нефункціональних характеристик <att_1; att_2; …; att_k>, $k = \overline{1,K}$ (наприклад, доступність, час виконання, пропускна здатність), що містить важливі елементи для задачі адаптивної композиції, які визначають, наскільки вебсервіс відповідає вимогам користувача та середовища, і впливають на вибір сервісів у динамічному середовищі;

Descr – текстовий опис вебсервісу: цей компонент корисний для розуміння призначення вебсервісу, а також при семантичній обробці вебсервісу для певних цілей. На даний час цей аспект ми не розглядаємо, однак вважаємо перспективним дослідження семантичної обробки таких описів.

Композитний вебсервіс (Composite Web Service) – це впорядкована сукупність вебсервісів, об'єднаних для досягнення спільної мети – цільового стану. На початку роботи композитний сервіс задається вхідними і вихідними параметрами (початковим і цільовим станами).

В результаті роботи нам необхідно отримати композитний вебсервіс як множину можливих маршрутів і оптимальний маршрут – послідовність сервісів з оптимальними параметрами *QoS*.

Визначимо специфіку композиції сервісів у термінах базових елементів ML. *Середовище* – це простір станів і дій, які агент може виконувати для досягнення цільового стану. *Станами* є можливі комбінації доступних вхідних і вихідних параметрів, що змінюються залежно від того, які сервіси виконано. Множина всіх можливих станів представляє *простір станів*, де окремо виділяють початковий і цільовий стани. *Поточний стан агента* – це те, які вхідні та вихідні дані відомі агенту після виклику певного вебсервісу. *Початковий стан* визначається вхідними параметрами композитного сервісу. *Цільовий стан* визначається набором вихідних даних (виходів композитного вебсервісу), які агент повинен досягти. Якщо агент досягає цільового стану, епізод завершується, і він отримує винагороду. Досягнення цільового стану вказує на успішну композицію сервісів.

Дія – це операції над станами в результаті виклику агентом доступного вебсервісу, що змінює поточний стан агента, та збільшує його знання про середовище.

Маршрут – це послідовність дій агента (викликів вебсервісів), які переводять його з початкового стану до цільового стану. Кожен крок маршруту відповідає виконанню певної дії (вебсервісу), яка змінює поточний стан агента на новий.

В результаті роботи алгоритму агент формує множину маршрутів – набір усіх припустимих послідовностей дій, які переводять агента з початкового до цільового стану. Кожному маршруту відповідає послідовність вебсервісів, що можуть бути використані для досягнення мети.

4. Q-learning для побудови оптимального композитного сервісу

В загальному вигляді програма навчання агента методом Q-learning має наступний вигляд [7]:

initialize Q(s, a) for all $s \in S$, $a \in A(s)$

for each episode do

s ← s0 // Початковий стан

while s ∉ terminal state do

Choose $a \in A(s)$

using ε – greedy policy

Execute action a, observe reward r and new state s'

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max a`Q(s`,a`) - Q)s,a)]$$

s ← s' // Перехід до нового стану

end while

end for

де:

s – поточний стан агента,

a – обрана дія,

r – винагорода за виконання дії a,

s′ – новий стан після виконання дії,

α – швидкість навчання (learning rate),

γ – фактор дисконтування (discount factor),

ε – параметр для ε-жадібної стратегії, що контролює баланс між дослідженням та експлуатацією.

Загальні характеристики методу Q-learning створюють основу для розробки алгоритму, що дозволяє агенту діяти в середовищі оптимально (саме цей алгоритм реалізовано у нашій програмі). Він забезпечує ефективне навчання, знаходження оптимальної стратегії і, зрештою, вирішення поставленої задачі – побудови оптимального композитного сервісу з урахуванням заданих параметрів QoS.

Розглянемо, як саме ми пропонуємо модифікувати Q-learning для побудови композитного сервісу.

У просторі станів S, що описує можливі комбінації вхідних даних, можна виділити наступні підмножини:

- $S_{ter\,min\,al} \subseteq S$ – множина термінальних станів, де $\forall s_{ter\,min\,al} \in S_{ter\,min\,al}$ – термінальний стан, у якому агент більше не може виконати жодну операцію або досяг завершення своєї мети.

- $\forall S \exists s_{target} \in S_{ter\,min\,al}$ – цільовий стан, в якому досягнуто бажаного результату (отримано необхідні вихідні дані).

- $S_{negative} \subseteq S$ – множина негативних станів, де $\forall s_{negativel} \in S_{negativel}$ – це стан, у якому агент не може знайти жодної дії, яка б наближала його до цільового стану, тобто не може рухатися до мети, і тому епізод має завершитися зі штрафом. .

Стан вважається негативним в наступних ситуаціях:

- для даного стану немає можливих дій (тобто жоден сервіс не може бути виконаний),

- немає доступних дій (сервісів), що переводять агента у новий корисний стан (тобто будь-які дії не наближають агента до цільового стану).

Якщо доступна дія не веде до цільового стану або не додає корисних виходів, можна розглядати її як помилкову і призначати штраф за її виконання.

Функція переходу — механізм, який визначає, як змінюється стан після виконання дії. У запропонованому методі зміна стану залежить від того, які виходи вебсервісу додаються до поточного стану (next_state), та від того, які обмеження враховуються для доступності сервісів. Ми пропонуємо функцію переходу визначати за допомогою таблиці переходів *transition_table*, яка зберігає нові стани для кожної пари (стан, дія). Такий підхід враховує лише доступні сервіси в поточному стані, що запобігає вибору некоректних дій. З кожним кроком агент, переходячи зі стану $s \in S$ в стан $s` \in S$, розширює свої знання про середовище.

Винагорода (Reward) в алгоритмі Q-learning – це кількісна оцінка того, наскільки корисною або шкідливою була ця дія для досягнення кінцевої мети, тобто отримання цільових вихідних даних у композитному сервісі. Агент отримує її після виконання певної дії (виклик/аналіз сервісу) у конкретному стані. Винагорода визначається за формулою:

$$R(s,a) = \sum_{i=1}^{n} w_i * r_i(s,a) - 1.$$

Стан $s \in S$, дія $a \in A$, w_i – вага i-го параметру QoS значення винагороди за стан s та дію a, $r_i(s,a)$ – нормалізована оцінка винагороди для i-го параметру QoS, що визначається за наступним правилом:

$$r_i(s,a) = \frac{att_i(s,a) - att_{min}}{att_{max} - att_{min}},$$ якщо параметр і потрібно максимізувати,

$$r_i(s,a) = 1 - \frac{att_i(s,a) - att_{min}}{att_{max} - att_{min}},$$ якщо параметр і потрібно мінімізувати.

$att_i(s,a)$ – значення i-го параметра QoS для дії a у стані s, а att_{min} та att_{max} – його мінімальне та максимальне значення серед усіх можливих сервісів.

Ми пропонуємо доповнювати визначення винагороди штрафами за непотрібні дії, що не наближають агента до цілі або не дають нових корисних виходів. Додається *функція втрат (Loss Function),* подібно до нейронних мереж, яка визначається через різницю між поточним Q-значенням і цільовим значенням, що використовується для оновлення Q-таблиці. Щокожного оновлення вона визначається як окрема метрика TD-помилки – Temporal Difference error .

Запропонований метод використовує популярну у Q-learning *ε-жадібну (epsilon-greedy) стратегію,* що забезпечує баланс між дослідженням нових дій і використанням вже відомих дій з високими значеннями Q-значень. Спочатку агент активно досліджує середовище, а після накопичення достатньої інформації поступово починає фокусуватися на оптимальних діях, зменшуючи випадкові вибори. Проведені нами експерименти на великих наборах даних виявили потребу в оптимізації процесу обчислення стратегії. Це зумовило необхідність розробки автоматизованого вибору методу розрахунку стратегії, що було реалізовано в нашій роботі.

Алгоритм навчання для побудови оптимального композитного вебсервіса має наступний вигляд:

1. *Ініціалізація середовища*: описуються входи, виходи та QoS-параметри вебсервісів; формується список цільових виходів та початковий стан композитного сервісу; задаються глобальні параметри важливості елементів QoS; генерується таблиця досяжних станів, які можна отримати, виконуючи сервіси послідовно; формується таблиця переходів, можливих за умови виконання наявних сервісів.

2. *Пошук можливих маршрутів* за допомогою модифікованого Q-learning:

- *Ініціалізація Q-таблиці* всіх можливих станів і дій (значення Q-функції ініціалізуються нулями);

- *Навчання.* Початковий стан задається відповідно до вхідних даних композитного сервісу. Агент вибирає дію (вебсервіс), яку можна виконати у цьому стані на основі обраної стратегії. Агент виконує дії та в результаті цього переходить з одного стану в інший. Винагорода нараховується за: *прогрес у досягненні мети* та *якість виконання сервісу* (значення QoS). Якщо досягнуто стан, коли жодна дія не наближає агента до цілі, то за виконання будь-якої можливої дії агент отримує негативну винагороду. Якщо через певну кількість перевірок ситуація не змінюється, епізод завершується невдачею.

3. *Вибір і контроль стратегії* (ε-жадібної стратегії дій агенту). На основі проведених експериментів на даний час ми використовуємо експоненційне та лінійне зниження ε. Найкращий метод зниження ε підбирається автоматично шляхом запуску тестових епізодів з оцінкою кількості знайдених маршрутів та часу виконання.

4. *Збереження досвіду.* Після завершення навчання Q-таблиця зберігається у файлі. Досвід агента (знайдені маршрути) також записується у форматі JSON.

5. *Оцінка ефективності*. Для контролю роботи алгоритму процес знаходження оптимального маршруту відображається наступним чином: виводяться всі маршрути, які досягають мети, із зазначенням їхніх сумарних QoS і наприкінці виводиться оптимальний. За результатами роботи будуються графіки сумарної винагороди, TD-помилки та зміни **ε** за епізодами.

6. *Використання збереженого досвіду*. Отримані результати роботи агента (Q-таблиця та список маршрутів) можливо зберігати і в подальшому **з**авантажувати для виконання цільового завдання без необхідності додаткового навчання.

5. Наукова новизна

Агент, який використовує Q-learning, може поступово вивчати та вдосконалювати стратегію вибору оптимальних вебсервісів, оновлюючи свою Q-таблицю – структуру, в якій зберігаються оцінки вигідності переходу з одного стану в інший через виконання конкретної дії (виклику вебсервісу). Під час цього процесу агент отримує винагороду за правильний вибір дії або штраф за неправильний і таким чином поступово адаптується до змін у середовищі.

Більшість досліджень [8-10], що стосуються композиції вебсервісів із використанням методів навчання з підкріпленням (RL), зосереджуються на оптимізації параметрів якості обслуговування (QoS), тоді, як проблему автоматичної побудови можливих маршрутів часто ігнорують або вирішують вручну чи напів-автоматично. В попередній роботі автори дослідили доцільність використання Q-learning для композиції сервісів за попередньо заданим

маршрутом та визначили потребу в автоматизації побудови таких маршрутів [11].

У роботі запропоновано модифікований алгоритм Q-learning для автоматичної генерації маршрутів на основі вхідних і вихідних даних вебсервісів і вибору оптимального згідно з параметрами QoS і глобальних ваг важливості цих параметрів.

Запропонований підхід побудови множини можливих маршрутів для заданого композитного сервісу і вибір оптимального відрізняється від традиційних тим, що адаптується до змін у середовищі в реальному часі. Агент навчається на основі зворотного зв'язку від середовища і самостійно коригує вибір вебсервісів для досягнення кращої якості. Цей метод не потребує попередньо повної інформації про весь простір станів вебсервісів. Невідомі стани обробляються окремо: якщо для певного стану немає можливих переходів, епізод завершується зі штрафом, що мінімізує ризик неправильних дій та переходів у невизначені стани.

Надана формалізація композиції вебсервісів через базові поняття методу Q-learning: стани, простір станів, дії, переходи таке інше, що дозволяє реалізувати автоматизацію процесу побудови маршрутів методом Q-learning, який раніше виконувався частково вручну.

Запропонований метод базується на даних, що збираються із описів вебсервісів, поданих стандартом OpenAPI [12], що додає гнучкості в адаптації для різних платформ і надає доступ до описів функціональності кожного сервісу.

Оптимізація процесу навчання здійснюється за рахунок виключення з розгляду негативних станів та повторних дій у поточному епізоді, що дозволяє скоротити обчислювальні витрати та зосередити пошук на більш релевантних станах та допомагає запобігти зациклюванню в епізоді. Реалізовано також механізми для запобігання зациклюванню шляхом обмеження на кількість кроків без прогресу, щоб уникати петлевих станів. Такі види оптимізації не входять до класичного Q-learning.

Запропонований метод для побудови композицій вебсервісів реалізує Q-learning алгоритм, що відрізняється від класичного адаптацією до специфічних вимог вибору і комбінації вебсервісів.

6. Основні відмінності запропонованого методу

Запропонований метод побудови множини можливих маршрутів має низку принципових відмінностей від традиційних підходів Q-learning.

Динамічний вибір і корекція дій

У запропонованому методі агент обирає послідовність вебсервісів з урахуванням специфічних параметрів QoS. Класичний Q-learning орієнтується переважно на статичне середовище, де набір дій визначений наперед, а в нашому випадку дії можуть залежати від поточного набору досягнутих станів і комбінації входів-виходів кожного сервісу.

Аналіз QoS.

Даний метод включає оцінку якості сервісів на основі QoS для формування винагороди, що дозволяє точніше оцінювати вигоду від кожного вибору сервісу. Кількість параметрів QoS не є фіксованою. Крім того, на відміну від поширених прикладів з оптимізацією QoS ми вводимо глобальні ваги параметрів QoS для врахування як важливості різних аспектів, так і їхній характер впливу: одні слід максимізувати, а інші — мінімізувати (наприклад, доступність, продуктивність треба максимізувати, а час виконання — мінімізувати) у процесі обчислення винагороди за використання певного вебсервісу. Таке узагальнення дає змогу адаптувати модель до конкретних умов задачі, підвищує її гнучкість і точність у виборі оптимального композитного сервісу. Класичний Q-learning зосереджений лише на кінцевій винагороді, не враховуючи QoS та їхньої модальності.

Використання OpenAPI.

Метод використовує дані, які збираються із специфікацій OpenAPI, що додає гнучкості в адаптації для різних платформ і надає доступ до описів функціональності кожного сервісу. Класичний Q-learning не містить аналогічного етапу інтерпретації структурованих вебданих, тоді як наданий метод можливо використовува-

ти з будь-якими стандартами подання вебсервісів, що містять узгоджені описи вхідних і вихідних даних.

Механізми для запобігання зациклюванню.

Враховуючи складність побудови багатокомпонентних вебсервісів, додатково запроваджені обмеження на кількість кроків без прогресу, щоб уникати петлевих станів. Це розширює можливості класичного Q-learning, де подібні зациклювання не є критичними.

Обробка безвихідних і неперспективних станів.

Запропоновано механізм, який запобігає виконанню дій, що не мають переходів для певного стану, і вводиться штраф за такі дії. Тобто агент перевіряє наявність можливих дій у кожному стані та отримує штраф за неперспективні дії. Це корисне розширення для обробки винятків або "неперспективних" станів, але класичний метод Q-learning цього не має.

Встановлення параметрів навчання залежно від розміру вхідних даних.

Автоматично визначаються:

- *кількість епізодів* – залежно від розміру таблиці станів та складності завдання;

- *обмеження кроків на епізод* – залежно від кількості станів і середньої довжини можливого маршруту;

- *порогова кількість кроків без прогресу* – для уникнення зациклювання залежно від складності середовища;

- гіперпараметри Q-learning: *стратегія навчання epsilon і метод зниження параметра epsilon epsilon_decay* – для забезпечення оптимального балансу між дослідженням та експлуатацією;

- *швидкість навчання alpha* – адаптується під максимальну довжину маршруту і стабільність навчання.

Така модифікація Q-learning дозволяє агенту не просто знаходити можливі маршрути через список доступних вебсервісів, а й адаптивно покращувати стратегію вибору на основі зворотного зв'язку від середовища. Це забезпечує системі можливість автоматично коригувати свої рішення, вибираючи такі вебсервіси, які оптимізують не тільки функціональність композитного сервісу, а й параметри QoS. Цей підхід є особливо корисним у динамічних середовищах, де характеристики вебсервісів, такі як час виконання, доступність або пропускна здатність, з часом можуть змінюватися. Використання Q-learning дозволяє автоматично підбирати найкращі варіанти композиції, враховуючи не лише доступні сервіси, а й їхню поточну продуктивність. Основні компоненти запропонованого методу залишаються в межах класичного Q-learning (формула оновлення, ε-жадібна стратегія тощо), але внесені вдосконалення роблять його реалізацію більш гнучкою і надійною, що робить його ефективнішим у середовищах із високою динамікою і складними вимогами до виконання.

7. Застосування алгоритму композиції сервісів для побудови послідовностей навчальних об’єктів

Розглянемо використання запропонованого методу на прикладі задачі вибору послідовності вивчення *навчальних об'єктів* (Learning Object, LO), що використовуються в індивідуальних освітніх траєкторіях [13], які дозволяють персоніфікувати процес здобуття освіти відповідно до потреб та можливостей конкретного студента.

LO – це сукупність інформації, яку об’єднує певна навчальна ціль [14]. Це автономний інформаційних об’єкт, що може багаторазово використовуватися у навчальному процесі. LO може містити лекційні матеріали, практичні завдання, засоби оцінювання знань, представлені як текст, відео, аудіо, програмний код тощо, а контент та роль LO у навчанні однозначно визначаються в його метаописі.

Використання метаописів є основою для автоматизованого аналізу LO, що забезпечує їхнє багаторазове використання для різних навчальних задач, оптимізує їхній вибір та зменшує час цієї процедури, дозволяє шукати, оцінювати та комбінувати LO без додаткового аналізу їхнього контенту.

Метадані LO дозволяють визначити вимоги до студента, який здатний оволодіти знаннями з LO, – вхідні компетенції, та результати такого вивчення – вихідні компетенції. Для забезпечення можливості співставлення компетенцій LO, викладачем курсу спочатку створюється тезаурус навчального курсу $Th_{course} = C_{in} \cup C_{out}$, що містить усі основні компетенції, що їх має отримати студент, який успішно вивчить цей курс $c_{oit_q} \in C_{out}, q = \overline{1,Q}$, та попередні вимоги до студентів $c_{in_p} \in C_{in}, p = \overline{1,P}$ [16]. Після цього викладач здійснює попередній відбір LO, $\forall l_k \in L \exists c_{in}(l_k) \in C_{in}, \exists c_{out}(l_k) \in C_{out}$ так, щоб у цих LO були надані можливості отримання всіх результуючих компетенцій курсу, тобто $\forall c_{out_p} \in C_{out} \exists l_k : c_{out_p} \in c_{out}(l_k)$.

Це дозволяє визначати семантичні відношення між окремими LO – наприклад, знаходити прийнятний порядок вивчення цих LO або перетин знань в їхньому контенті (Рис.1).

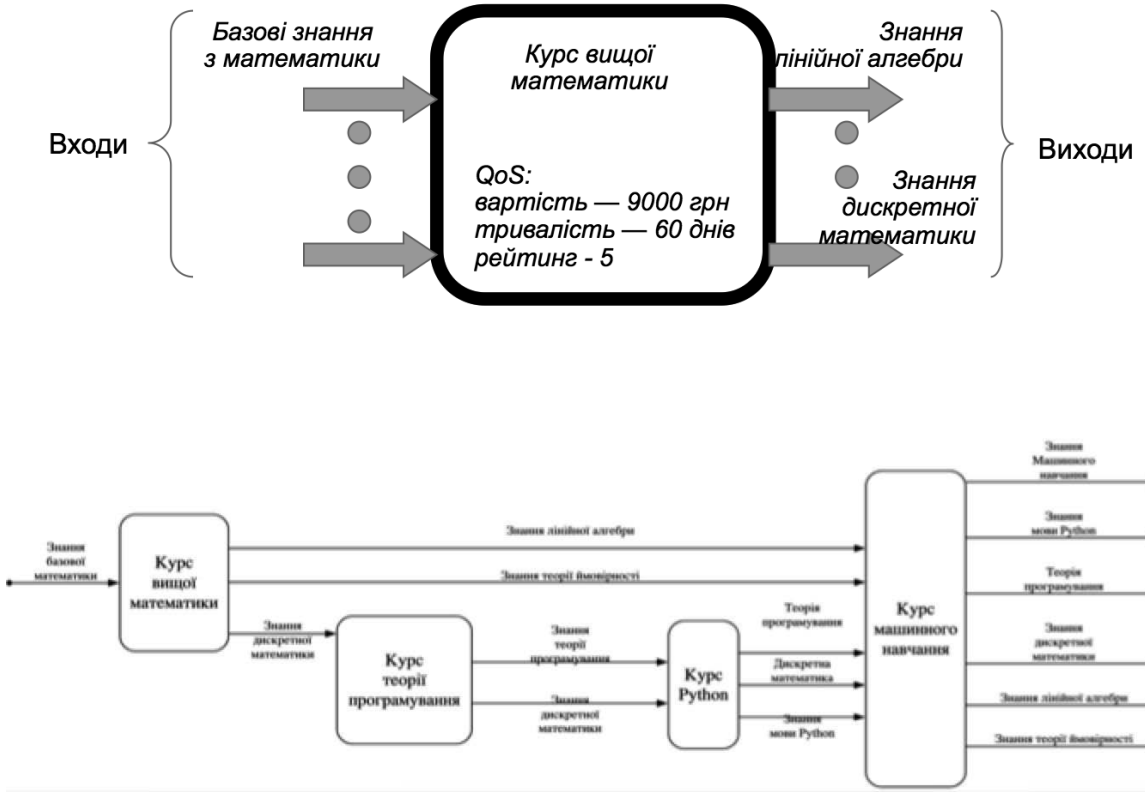

Рис.1. Процес побудови послідовності LO на основі аналізу компетенцій студента

Наступним кроком є оновлення метаданих LO з множини L, щоб пов'язати кожен з них із відповідними вхідними та вихідними наборами компетенцій курсу. Слід звернути увагу, що таке уточнення метаданих LO потрібно виконувати для кожного навчального курсу.

Аналізувати метадані LO можна вручну, але, якщо LO багато та вони досить часто оновлюються, то це забиратиме дуже багато часу, а результати такого аналізу можуть бути неактуальними.

Для автоматизації цього процесу пропонуємо вивчення кожного LO як вебервісу, що має функціональні вимоги і нефункціональні властивості.

Функціональні вимоги сервісу LO – це входи (компетенції студента, які необхідні для вивчення) та виходи (компетенції, які отримуються в результаті вивчен-

ня). Нефункціональні властивості сервісу LO – це характеристики, що можуть впливати на вибір одного із сервісів з однаковими функціональними вимогами – як-от, вартість і час вивчення, які треба мінімізувати, і рейтинг курсу, який треба максимізувати.

Задача полягає в тому, щоб за параметрами наявних LO (вхідні дані – компетенції, що потрібні для вивчення; вихідні дані – компетенції, які можна отримати в результаті навчання [14]) побудувати оптимальну послідовність вивчення LO, що дозволяє отримати певний набір компетенцій – цільовий стан. Цей цільовий стан формалізується як набір C_{out} з Th_{course}.

Сервіси LO функціонують у динамічному інформаційному середовищі, тобто може змінюватися набір доступних сервісів, умови їх використання та контент, а також поточний стан студента. Наприклад, у якийсь момент часу студент розуміє, що для отримання певної компетенції курсу він має доступ лише до таких LO, що подають інформацію природною мовою, якою він не володіє, тобто ситуація може розглядатися як негативний стан. Тоді студент може почекати появи LO тими мовами, котрі він знає, вивчити мову, яку використовують LO з відповідними компетенціями або очікувати на зменшення результуючих компетенцій курсу (у цьому дослідженні динамічні зміни цільового стану не розглядаються).

Крім того, щомиті можуть змінюватися критерії оцінювання та відносні пріоритети параметрів LO. Приміром, іноді час навчання стає важливішим за його вартість. Це викликає потребу у застосуванні методів машинного навчання, які забезпечують можливість автоматизованої побудови маршруту без фіксації його довжини та послідовності елементів.

Запропонований у роботі метод відповідає цим умовам, а тестування його на прикладі з 1000 LO (кількість станів 25913), показало прийнятний час виконання (час формування таблиць: 197.4923 секунд і час виконання: 2.5679 секунд) та високий рівень якості побудованої траєкторії навчання. Тестування виконувалось на Macbook Pro з чіпом Apple M1 Pro з 16Гб пам`яті.

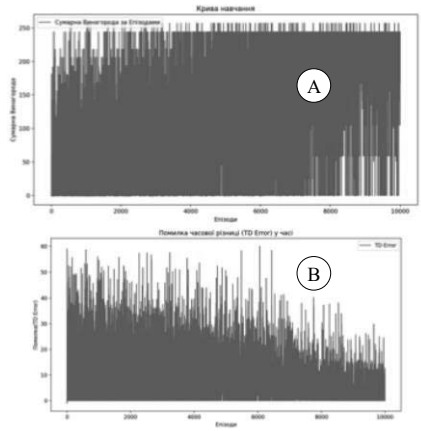

Рис.2. Результати тестування програми на прикладі з 1000 LO

Графік винагороди за епізодами (крива навчання) (Рис.2.A) показує, як змінювалася винагорода під час навчання агента. На початку роботи алгоритму фінальна винагорода за виконання дій була низькою, але поступово її значення збільшувалося, що свідчить про навчання агента і його здатність знаходити кращі маршрути навчання. Позитивна тенденція на графіку свідчить про успішність навчання та покращення алгоритму Q-learning.

Графік TD-помилки (Рис.2.B) відображає помилку часової різниці в процесі навчання. Спочатку помилка була високою, що очікувано, оскільки агент тільки починав вивчати навколишнє середовище. Із часом значення помилки зменшувалося, що свідчить про покращення прогнозів агента щодо майбутніх винагород і стабілізацію процесу навчання.

8. Висновки

Запропоновано новий підхід до побудови композиції вебсервісів, в якому система автоматично генерує можливі маршрути без необхідності попередньо задавати повну інформацію про простір станів і оптимізує їх на основі параметрів QoS. Оптимізація процесу навчання здійснюється шляхом видалення з розгляду неперспективних станів та обмеження повторних дій в межах поточного епізоду, що значно зменшує кількість обчислень та прискорює збіжність алгоритму. Якщо певний стан не

має можливих переходів, то епізод завершується зі штрафом, що запобігає неефективним обчисленням у нерелевантних напрямках. Це дозволяє швидше адаптуватися до змін і динамічно коригувати стратегію вибору сервісів, мінімізуючи час на дослідження та покращуючи ефективність у реальних умовах. Запропоновано засоби запобігання зациклюванню, яке може виникати на великих об'ємах даних.

Надалі для прискорення пошуку оптимальних сервісів рекомендується додати використання евристик для відбору сервісів, алгоритми прискореного навчання та ефективні методи балансу між дослідженням і експлуатацією, а для адаптації до динамічних змін QoS-параметрів – онлайн-навчання, відстеження змін у середовищі та регулярне оновлення стратегії на основі адаптивних алгоритмів. Крім того, для вирішення проблем, пов'язаних з довготривалим пошуком оптимальних сервісів та динамічними змінами параметрів QoS, доцільно застосувати кілька різних стратегій і методів, які дозволять покращувати ефективність навчання та адаптацію агента до змін середовища в реальному часі. Ці рішення допоможуть підвищити ефективність запропонованого методу.

Отож, запропоновано новий підхід до вирішення задачі автоматизації процесу композиції вебсервісів із динамічними характеристиками. Можливі подальші дослідження стосуються підвищення ефективності: покращення швидкості навчання агента, підвищення стабільності роботи, оптимізації таблиці станів і дослідження інших методів навчання з підкріпленням, таких як SARSA та глибокі нейронні мережі.

References

1. *Berners-Lee T.,* Web Services, Program Integration across Application and Organization boundaries, W3C, https://www.w3.org/DesignIssues/WebServices.html
2. Web Services Activity Statement, W3C, https://www.w3.org/2002/ws/Activity
3. *Kashyap V., Bussler C., Moran M.,* Web Service Composition. The Semantic Web: Semantics for Data and Services on the Web. In: The Semantic Web. Data-Centric Systems and Applications. Springer, Berlin, Heidelberg. 2008, pp.233-248 https://doi.org/10.1007/978-3-540-76452-6_10.
4. *Samuel A.,* Some studies in machine learning using the game of checkers. IBM Journal of research and development, 1959, 3(3), pp.210-229.
5. *Mitchell T.,* Machine learning. New York: McGraw-hill. 1997, Vol. 1, No. 9 http://www.pachecoj.com/courses/csc380_fall21/lectures/mlintro.pdf.
6. *Barto A., Sutton R.,* Reinforcement Learning: An Introduction, The MIT Press: Cam-bridge, Massauchsetts, 2018. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.
7. *Jang B., Kim M., Harerimana G., Kim J.W.*, Q-learning algorithms: A comprehen-sive classi-fication and applications. IEEE access, 2019.
8. *Lemos A.L., Daniel F., Benatallah B.,* Web service composition: a survey of techniques and tools. ACM Computing Surveys (CSUR), 2015, 48(3), 1-41.
9. *Uc-Cetina V., Moo-Mena F., Hernandez-Ucan R.,* Composition of web services using Mar-kov decision processes and dynamic programming. The Scientific World Journal, 2015 (1).
10. *Kalasapur S., Kumar M., Shirazi B.A.,* Dynamic service composition in pervasive com-puting. IEEE Transactions on parallel and dis-tributed systems, 2007, 18(7), pp.907-918.
11. *Gryshanova I., Rogushyna J.,* Technology of machine learning use for composite web service development. Problems in Programming, 2024, 4, pp.34-43. (in Ukrainian)
12. The OpenAPI Specification, https://swagger.io/specification/.
13. *Rogushina J., Gladun A., Anishchenko O., Pryima S.,* Semantic Support of Personal Learning Trajectory Development, in: UkrPROG 2024, CEUR Vol-3806, 2024, pp.487-505.
14. *Politis D., Tsalighopoulos M., Kyriafinis G.,* Designing Blended Learning Strategies for Rich Content, Handbook of Research on Building, Growing, and Sustaining Quality E-Learning Pro-grams, 2017. DOI: 10.4018/978-1-5225-0877-9.ch017.
15. *Rogushina J., Gladun A., Anishchenko O., Pryima S.,* Semantic analysis of learning objects: thesaurus approach for digital transformation of educational resources,

CEUR Workshop Proceed-ings (2024), CEUR, Vol-3771, 85–99. ceur-ws.org/Vol-3771/paper13.pdf

Одержано: 05.02.25
Внутрішня рецензія отримана: 03.03.25
Зовнішня рецензія отримана: 07.03.25

Про авторів:

Гришанова Ірина Юріївна,
науковий співробітник.
ORCID
http://orcid.org/0000-0003-4999-6294.

Рогушина Юлія Віталіївна,
кандидат фізико-математичних наук, доцент,
старший науковий співробітник.
ORCID
http://orcid.org/0000-0001-7958-2557.

Місце роботи авторів:

Інститут програмних систем
НАН України,
тел. (+38) 066 5501999,
e-mail: ladamandraka2010@gmail.com,