ISSN 2078–5097. Вісн. Львів. ун-ту. Сер. прикл. матем. та інф. 2013. Вип. 19. С. 106–125
Visnyk of the Lviv University. Series Appl. Math. and Informatics. Issue 19. P. 106–125

І Н Ф О Р М А Т И К А

УДК 004.4

АПРОКСИМАЦІЯ ПРОСТОРУ СТАНІВ І ДІЙ ШТУЧНИМИ НЕЙРОННИМИ МЕРЕЖАМИ В ЗАДАЧАХ НАВІГАЦІЇ

О. Годич[1], П. Кушнір[2], Ю. Щербина[2]

[1]Національний університет "Львівська політехніка",
вул. С. Бандери, 12, Львів, 79013
[2]Львівський національний університет імені Івана Франка,
вул. Університетська, 1, Львів, 79000, e-mail: dais@franko.lviv.ua

Розглянуто доцільність використання штучних нейронних мереж як апроксиматора для розв'язання задачі навігації. Створено програмне забезпечення для автоматизації проведення експериментів. Розглянуто різні підходи подання простору станів і дій. Проведено аналіз впливу параметрів і структури штучної нейронної мережі на швидкість навчання та якісні характеристики стратегії, отриманої після навчання. На підставі проведеного аналізу побудовано штучну нейронну мережу, здатну отримати близьку до оптимальної стратегію з використанням порівняно невеликої навчальної вибірки.

Ключові слова: задача навігації, інтелектуальні мобільні агенти, програмне забезпечення, моделювання, штучні нейронні мережі, навчання з підкріпленням.

1. ВСТУП

Ми розглядатимемо практичну доцільність використання штучних нейронних мереж як апроксиматора в алгоритмах навчання з підкріпленням для розв'язання задачі навігації.

Задача навігації (задача пошуку шляху) мобільного агента полягає в автоматичному визначенні шляху, який би пролягав між кількома наперед заданими точками. Цю задачу можна звести до задачі пошуку шляху з вихідної позиції агента до деякої наперед заданої точки на карті. Знайдений шлях повинен оминати перешкоди розміщені в навколишньому середовищі. На рис. 1 зображено приклад такого шляху.

Для розв'язання цієї задачі використовують парадигму *навчання з підкріпленням* [6]. Головна ідея цієї парадигми полягає в обчисленні агентом власної стратегії щодо поведінки в середовищі внаслідок безпосередньої взаємодії з ним. Взаємодіючи з середовищем, агент отримує досвід, який використовують для поліпшення стратегії поведінки. Це реалізовується за допомогою *функції корисності*, яка дає змогу агенту вибирати ту дію, яка, як свідчить досвід, принесе найбільшу користь.

При використанні навчання з підкріпленням виникає проблема вибору ефективної системи оцінювання дій агента. Зазвичай ця задача значно простіша, ніж відшукання оптимальної стратегії дій [10].

Алгоритми навчання з підкріпленням працюють з дискретним простором станів і дій. Водночас задача навігації потребує використання неперервного простору станів і, залежно від реалізації, неперервного простору дій.

Для подолання труднощів з неперервним простором станів застосовують апроксиматори функцій, які не тільки дають змогу подати функцію неперервних

станів, а й здатні прискорити збіжність завдяки своїм узагальнюючим властивостям. Як апроксиматор ми використали штучні нейронні мережі.

Розв'язання задачі навігації проводили за допомогою комп'ютерного моделювання. Моделлю навколишнього середовища є двовимірна площина з розташованими на ній перешкодами. Перешкоди мають вигляд опуклих многокутників. Середовище, в якому відбувається рух агента, може бути відкритим або закритим. Це досягається без уведення додаткових понять – адже границя середовища є також перешкодою.

Ціль, яку потрібно досягнути агенту після розв'язання задачі навігації, також є опуклим многокутником, що не перетинається з перешкодами. Епізод навчання вважається успішним, якщо агент завершив свій рух у будь-якій точці цілі за прийнятну кількість ітерацій. Проходження через ціль ще не є достатньою умовою для того, щоб вважати епізод успішним.

Приклад середовища з заданою на ньому ціллю зображено на рис. 1. Цифрами позначено: 1, 2 – перешкоди, 3 – початкова точка руху агента, 4 – ціль, яку потрібно досягнути, 5 – шлях агента.

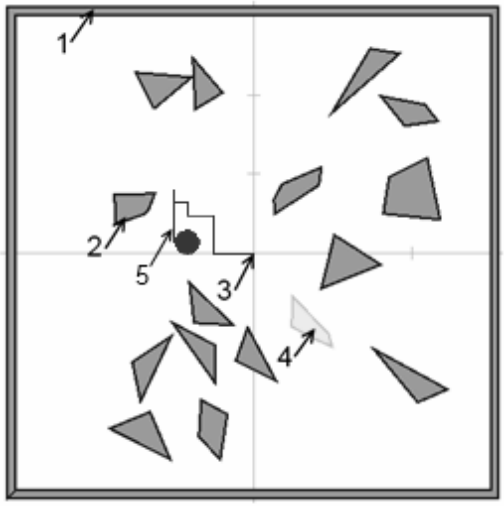


Рис. 1. Віртуальне середовище для моделювання поведінки агента

Час приймають дискретним (кожна ітерація алгоритму виконується за одиницю часу), проте від цього не втрачається загальність, оскільки при достатньо малому часі виконання ітерацій алгоритму дискретність часу є несуттєвою. З іншого боку, дискретність часу дає змогу уникнути багатьох проблем пов'язаних з розробленням алгоритму.

Кожна дія агента є рухом у заданому напрямі з заданою швидкістю на деяку відстань. Так моделюється рух агента, який виконує поворот на місці та далі рухається прямолінійно вперед (наприклад, такими є мобільні роботи на гусеничній основі). Кут огляду навколо агента розбито на рівні сектори, в яких визначається відстань до об'єктів. Відстань до перешкод у кожному секторі визначається як відстань від агента до перетину бісектриси сектора з найближчою перешкодою (або нескінченість, якщо такої перешкоди немає). Такий підхід дає змогу вирішити проблеми з недостатністю інформації про перешкоди, які описуються в [12].

2. НАВЧАННЯ З ПІДКРІПЛЕННЯМ. АЛГОРИТМ SARSA

У підході навчання з підкріпленням використовують поняття середовища та агента, який діє у ньому. Агент здатний сприймати середовище та має набір допустимих дій, якими він впливає на середовище (у випадку задачі навігації – рухаючись у ньому).

Уся сукупність інформації, доступна агенту в кожен дискретний момент часу *t*, називається станом і позначається s_t. Відповідно, можна ввести поняття множини усіх можливих станів середовища *S*. У випадку задачі навігації множина станів неперервна, а це значно ускладнює навчання. Подібно визначають множину дій агента, які є допустимими у стані s_t, яку позначають $A(s_t)$. Множина допустимих дій у загальному випадку задачі навігації також неперервна.

Схематично взаємодію агента та середовища зображено на рис. 2.

Рис. 2. Схема взаємодії агента і середовища

Середовище у певний момент часу *t* надає агенту інформацію у вигляді значень сенсорів s_t (у випадку задачі навігації це може бути відстань до навколишніх об'єктів, яку отримали за допомогою сонара; інформація про зіткнення тощо). На підставі цієї інформації агент обирає одну з можливих у поточному стані дій $a_t \in A(s_t)$. Виконавши цю дію за допомогою маніпуляторів, він отримує миттєву винагороду $r_t \in \Re$.

Мета агента – визначити таку стратегію, яка б кожній можливій ситуації s_t ставила у відповідність таку дію a_t, що дає в перспективі максимальну сумарну винагороду (очікуваний прибуток R_t). Цю стратегію прийнято позначати як відображення $\pi : S \to A$, яка ставить у відповідність кожному стану системи оптимальну дію в ньому. Очікуваний прибуток R_t визначають за формулою (2.1)

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{2.1}$$

де параметр $\gamma \in [0,1]$ визначає коефіцієнт зважування миттєвих винагород. Його зміною можна гнучко регулювати важливість обраних дій залежно від їхньої віддаленості у часі. Якщо $\gamma = 0$, то агент буде дбати лише про максимізацію миттєвої винагороди, отриманої лише на поточному кроці. У більшості задач часто буває вигідно на певному кроці виконати дію, яка не забезпечує максимальної винагороди на поточному кроці, зате дає змогу в майбутньому отримати більший сумарний прибуток. Із наближенням γ до 1 агент надаватиме майбутнім винагородам усе більшого значення, а отже, його поведінка ставатиме ближчою до оптимальної.

Існують різноманітні підходи до навчання агента. Одним з таких підходів є алгоритм, який ґрунтується на побудові функції корисності $V^{\pi}(s_t)$, що визначає максимальний очікуваний прибуток у стані s_t за умови строгого дотримання оптимальної стратегії π [10]. Саме такий підхід ми використали у цій роботі.

З огляду на формулу (2.1) отримуємо співвідношення (2.2) для *корисності стану* s_t стосовно деякої фіксованої стратегії π

$$V^{\pi}(s_t) = E_{\pi}\{R_t\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k}\right\}. \tag{2.2}$$

Тут E_{π} визначає математичне сподівання прибутку за умови дотримання агентом фіксованої стратегії π.

Часто використовують розширення функції $V^{\pi}(s_t)$, яке допомагає визначити не лише корисність стану s_t, а й виконаної в ньому дії a_t. На перший погляд вона збігається з функцією корисності стану, проте в цьому випадку враховується також дія, яка була виконана в певному стані (ця дія може не збігатися з тією, яка передбачена стратегією). Для цього використовують функцію *корисності дії* $Q^{\pi}(s_t, a_t)$, яка визначає очікуваний максимальний прибуток, у стані s_t по дії a_t

$$Q^{\pi}(s_t, a_t) = E_{\pi}\{R_t\} = E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k}\right\}. \tag{2.3}$$

Функції корисності можна використовувати для порівняння стратегій агента. Стратегію π називатимемо *не гіршою*, ніж стратегія π', якщо справджується твердження (2.4)

$$\forall s_t \in S\left(\pi \geq \pi' \Leftrightarrow V^{\pi}(s_t) \geq V^{\pi'}(s_t)\right). \tag{2.4}$$

Завжди існує стратегія, яка є не гіршою від усіх інших. Таку стратегію називатимемо *оптимальною*. Хоча оптимальних стратегій може бути декілька, усі вони мають ту саму функцію корисності станів, яку називатимемо оптимальною функцією корисності станів V^*

$$\forall s_t \in S\left(V^* = \max_{\pi} V^{\pi}(s_t)\right). \tag{2.5}$$

Аналогічно, усі оптимальні стратегії також мають ту саму функцію корисності дій, яку називатимемо оптимальною функцією корисності дій Q^*

$$\forall s_t \in S, \forall a_t \in A(s_t)\left(Q^* = \max_{\pi} Q^{\pi}(s_t, a_t)\right). \tag{2.6}$$

Завдання алгоритмів навчання з підсиленням – знайти таку стратегію дій π, яка б мала функції корисності максимально близькі до оптимальних [12].

Одним з методів навчання з підкріпленням є *метод часової різниці*. Детально цей метод описаний у працях [6] і [8]. Ми застосовуємо алгоритм Sarsa, що

використовує метод часової різниці для знаходження стратегії дій з близькою до оптимальної функцією $Q^{\pi}(s_t, a_t)$. Розглянемо цей алгоритм з пристосуванням до задачі навігації.

Епізодом називатимемо рух агента в деякому середовищі від початкової точки і до досягнення *термінального стану*. Термінальним станом може бути успішне досягнення цілі, перевищення часу пошуку, зіткнення з перешкодою тощо. Впродовж кожного епізоду алгоритм Sarsa намагається наблизити функцію корисності дій до оптимальної, не чекаючи його завершення. Це становить головну відмінність методів часової різниці – для поліпшення стратегії не треба чекати до закінчення епізоду.

Кожен епізод є дослідженням нового середовища. Причому про це середовище не відомо жодної інформації, окрім значень, які отримуються від сенсорів. На підставі цих даних будують стани s_t у процесі роботи алгоритму [8], [10], [14].

Суть алгоритму Sarsa така: починаючи з довільно ініціалізованої функції $Q^{\pi}(s_t, a_t)$ і початкового стану s_0 виконувати поліпшення функції корисності дії. Для цього агент обирає найліпшу для поточного стану s_t дію a_t, виконує її й отримує від середовища відповідну винагороду r_t. Агент переходить у новий стан s_{t+1}. Знову обирають найліпшу дію a_{t+1} для стану s_{t+1}. На підставі цих даних агент оновлює функцію корисності за правилом

$$Q^{\pi}(s_t, a_t) \leftarrow Q^{\pi}(s_t, a_t) + \alpha\left[r_t + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t)\right]. \qquad (2.7)$$

Схематично послідовність подій у Sarsa зображено на рис. 3.

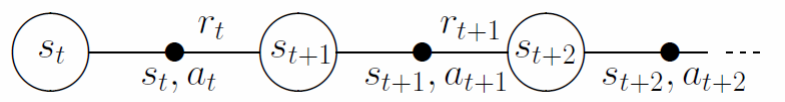


Рис. 3. Послідовність подій в алгоритмі Sarsa

Для реалізації функції $Q^{\pi}(s_t, a_t)$ використовують штучну нейронну мережу. Вона допомагає отримати значення функції в точках, які раніше не були визначені, але внаслідок неперервності функції корисності можуть бути досить точно оцінені на підставі даних про точки, які розташовані достатньо близько до заданої.

Розглянемо детальніше покрокову реалізацію алгоритму S.A.R.S.A. для розв'язання задачі навігації.

2.1. S – КОДУВАННЯ СТАНУ

Основний недолік нейронної мережі – невелика швидкість навчання. Для того, щоб отримати адекватне наближення шуканої функції корисності, потрібно навчати нейронну мережу протягом десятків, а то й сотень тисяч ітерацій. У випадку комп'ютерного моделювання така кількість ітерацій є прийнятною, але для реальних застосувань (наприклад, на справжньому роботі) це є критичним обмеженням [14]. Ці проблеми частково можна вирішити підбором параметрів навчання, структури нейронної мережі та правильним її використанням. Зокрема, це стосується способу подання вхідних значень. Отже, для успішного використання нейронної мережі як апроксиматора функції корисності дії $Q^{\pi}(s_t, a_t)$ треба вибрати кодування поточного стану s_t, в якому перебуває агент, для його подання на вхід нейронної мережі.

Один зі шляхів розв'язання цієї підзадачі розглянуто в [8]. Його й взяли за основу. Кожне значення *x*, яке кодується (наприклад, відстань до цілі, відстань до

перешкод тощо), передається на N входів нейронної мережі. Для цього значення входу $n = \overline{1..N}$) кодують за формулою (2.8)

$$i_n = \frac{1}{1+e^{w_n(b_n - x)}}, \tag{2.8}$$

де

$$w_n = \frac{4N}{r}, \tag{2.9}$$

$$b_n = \left(\frac{2n-1}{2}\right)\frac{r}{N}. \tag{2.10}$$

Значення N задає кількість входів нейронної мережі, а r задає проміжок $[0, r]$, в якому входи будуть найбільш чутливі (найбільше змінюватись зі зміною x). Це відповідає проекції вхідного значення на значення сигмоїд у відповідних точках, як зображено на рис. 4.

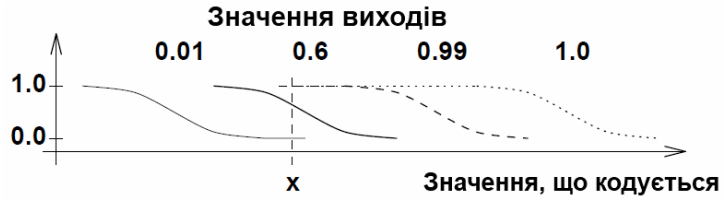


Рис. 4. Кодування вхідного значення, $N = 4$

Вибір такого методу кодування грунтується на ідеї збільшення кількості збуджених вхідних нейронів (їхнє вхідне значення близьке до 1) у момент, коли значення кодованої змінної найбільш значуще. Зокрема, в задачі навігації це відбувається, коли відстань до перешкоди стає дуже малою.

Для передачі на вхід нейронної мережі значення деякого кута α, який є частиною стану агента (в задачі навігації це може бути, зокрема, кут напряму до цілі), використовують пару значень $(\alpha,-\alpha)$, які потім кодують відповідно до описаного раніше методу з $r = \pi$. Використання такої пари значень зумовлене тим, що близькі значення кутів будуть мати близькі значення закодованих входів на проміжку $[0, r]$.

Розглянемо приклад кодування відстані до цілі x. Нехай $x = 3$; $N = 4$; $r = 5$. Для того, щоб отримати значення входів нейронної мережі, послідовно обчислюємо значення i_n, $n = \overline{1..4}$ згідно з формулами (2.8)–(2.10)

$$w_{1..4} = \frac{4\times 4}{5} = 3\frac{1}{5},$$

$$b_1 = \left(\frac{2-1}{2}\right)\frac{5}{4} = \frac{5}{8},\ \ b_2 = \left(\frac{4-1}{2}\right)\frac{5}{4} = 1\frac{7}{8},$$

$$b_3 = \left(\frac{6-1}{2}\right)\frac{5}{4} = 3\frac{1}{8},\ \ b_4 = \left(\frac{8-1}{2}\right)\frac{5}{4} = 4\frac{3}{8}.$$

Отримаємо відповідні значення входів, які кодують відстань до цілі

$$i_1 = \frac{1}{1+e^{\frac{16}{5}\left(\frac{5}{8}-3\right)}} \approx 0.999,\ \ i_2 = \frac{1}{1+e^{\frac{16}{5}\left(\frac{15}{8}-3\right)}} \approx 0.973,$$

$$i_3 = \frac{1}{1+e^{\frac{16}{5}\left(\frac{25}{8}-3\right)}} \approx 0.401,\ \ i_4 = \frac{1}{1+e^{\frac{16}{5}\left(\frac{35}{8}-3\right)}} \approx 0.012.$$

Як бачимо, перші два входи виявились близькими до одиниці, що призведе до збудження відповідних нейронів.

2.2. A – ВИБІР ОПТИМАЛЬНОЇ ДІЇ

Іншою підзадачею, яку треба розв'язати для успішного використання штучної нейронної мережі як апроксиматора, є проблема вибору оптимальної дії $a_t \in A(s_t)$ з неперервного простору дій $A(s_t)$.

Для розв'язання цієї підзадачі можливе використання наперед заданого невеликого фіксованого набору дій, або ж використання методів розбиття простору допустимих дій $A(s_t)$ для знаходження тієї, яка максимізує $Q^\pi(s_t, a_t)$. Градієнтні методи пошуку максимуму прийнятних результатів не дали, оскільки функція корисності, особливо на початку навчання, має складний вигляд, тому часто знаходили лише локальний максимум, а не глобальний.

2.2.1. ВИКОРИСТАННЯ НАПЕРЕД ЗАДАНОГО НАБОРУ ДІЙ

Можливим підходов до пошуку *оптимальної* дії a_t, на якому досягається $\max\limits_{a_t} Q^\pi(s_t, a_t)$, є пошук у наперед заданій множині дій *L*. Це найліпший варіант, коли простір дій дискретний і його розмір невеликий, оскільки можливо досягнути виконання умови $\forall s_t \in S\big(A(s_t) \subseteq L\big)$.

Проте й у випадку неперервного простору дій, за умови достатнього розміру набору *L*, цей підхід може дати адекватні результати [6]. У такому випадку апроксиматору передається закодоване значення стану, в якому перебуває агент, а на виході отримують значення функції корисності $Q^\pi(s_t, a_t)$ для кожного $a_t \in L$.

Дію, що відповідає максимальному значенню функції корисності, вибирають як оптимальну. Згідно з [6] доцільно для кожної дії $a_t \in L$ використовувати окремий апроксиматор для наближення функції $Q^\pi(s_t, a_t)$. Це ефективніше, ніж використовувати, наприклад, нейронну мережу з кількістю виходів, що дорівнює $|L|$. Хоча це забезпечує приріст у точності наближення $Q^\pi(s_t, a_t)$, проте в такому підході треба для кожної з можливих дій проводити окреме навчання апроксиматора, що пов'язано зі значними обчислювальними затратами.

Ми використовували окремий апроксиматор для кожної можливої дії. На вхід нейронній мережі передавали такі значення:

- відстань до центра цілі;
- кут напряму на ціль;
- відстань до найближчої перешкоди в кожному зі секторів огляду.

На виході отримували одне значення – прогнозоване значення функції корисності відповідної дії.

Експериментально виявили, що саме ці параметри є вагомими для вирішення задачі навігації. Додаткові параметри, наприклад, дія вибрана на попередньому кроці,

лише сповільнюють швидкість навчання нейронної мережі, незначно збільшуючи успішність роботи.

2.2.2. ВИКОРИСТАННЯ НЕПЕРЕРВНОГО ПРОСТОРУ ДІЙ

Іншим підходом до пошуку оптимальної дії є послідовне звуження області простору $A(s_t)$, в якому відбувається пошук. У такому випадку апроксиматору передається закодоване значення стану, в якому перебуває агент, і дії, які заплановано виконати. На виході отримують значення функції корисності $Q^{\pi}(s_t, a_t)$. Дію, що відповідає максимальному значенню функції корисності, вибирають як оптимальну. Алгоритм 1 використовують для пошуку оптимальної дії в неперервному просторі дій.

Ініціалізація. $X \leftarrow A(s_t)$

Виконувати.

1. Побудувати розбиття множини X на підмножини $X_j^{'}$.
2. Вибрати таку підмножину $X_j^{'}$, яка найімовірніше містить оптимальну дію.
3. $X \leftarrow X_j^{'}$.
4. Якщо X достатньо велика – перейти до кроку 1.

Як оптимальну дію вибрати довільне $a_t \in X$.

Алгоритм 1. Алгоритм пошуку оптимальної дії в неперервному просторі дій

Для повноти розгляду детально опишемо реалізацію кроків 1 і 2 алг. 1 при застосуванні його до моделі задачі навігації використаної в цій праці.

Як множини дій використаємо значення кута, на який треба повернути агенту на поточній ітерації. Швидкість агента і відстань, яку треба пройти, вибирають на підставі даних про відстань до цілі. У такому випадку початкове значення $X = [0, 2\pi]$. Для розбиття X використовуємо поділ поточного сектора пошуку на k менших секторів $X_j^{'} : j = \overline{1..k}$ однакового розміру. Щоб знайти сектор $X_j^{'}$, який найімовірніше містить оптимальне значення кута повороту, вибирають довільні значення $a_j \in X_j^{'}$. Обирають такий сектор $X_j^{'}$, що для відповідного кута a_j досягається максимальне значення функції корисності $\max\limits_{a_t \in X_j} Q^{\pi}(s_t, a_j)$. Такий поділ виконується m разів, де m – деяка наперед задана константа. Внаслідок виконання алгоритму отримаємо множину X таку, що $|X| = \frac{2\pi}{k^m}$ і з неї можна буде вибрати довільну дію як оптимальну.

У цьому підході на вхід нейронній мережі передавали такі значення:

- відстань до центра цілі;
- кут напряму на ціль;
- напрям, в якому агент буде рухатись;
- відстань до найближчої перешкоди в кожному з секторів огляду.

На виході отримували одне значення – прогнозоване значення функції корисності цієї дії. Схоже, як і в попередньому підході, "зайві" параметри лише зменшували швидкість навчання.

Розглянемо приклад пошуку оптимальної дії. Для наочності використаємо умовну одновимірну функцію корисності $Q(a) = \sin(a)$. Приймемо простір дій $A = [0, 2\pi]$, $k = 4$, $m = 2$.

1. Ініціалізація $X = [0, 2\pi]$.
2. Розбиття множини X

   $$X_1' = \left[0, \frac{\pi}{2}\right);\ X_2' = \left[\frac{\pi}{2}, \pi\right);\ X_3' = \left[\pi, \frac{3\pi}{2}\right);\ X_1' = \left[\frac{3\pi}{2}, 2\pi\right].$$

3. Вибір підмножини, що найімовірніше містить оптимальну дію

   $$\alpha_1 = \frac{\pi}{8};\ \alpha_2 = \frac{3\pi}{4};\ \alpha_3 = \frac{11\pi}{8};\ \alpha_4 = \frac{7\pi}{4}.$$

   $$Q(a_1) = 0.383;\ Q(a_2) = 0.707;\ Q(a_3) = -0.924;\ Q(a_4) = -0.707.$$

   Максимум досягається на 2 множині.
4. Отже,

   $$X \leftarrow X_2' = \left[\frac{\pi}{2}, \pi\right).$$

5. **Друга ітерація.** Розбиття множини X

   $$X_1' = \left[\frac{\pi}{2}, \frac{5\pi}{8}\right);\ X_2' = \left[\frac{5\pi}{8}, \frac{3\pi}{4}\right);\ X_3' = \left[\frac{3\pi}{4}, \frac{7\pi}{8}\right);\ X_4' = \left[\frac{7\pi}{8}, \pi\right].$$

6. Вибір підмножини, яка найімовірніше містить оптимальну дію

   $$\alpha_1 = \frac{17\pi}{32};\ \alpha_2 = \frac{11\pi}{16};\ \alpha_3 = \frac{3\pi}{4};\ \alpha_4 = \frac{63\pi}{64}.$$

   $$Q(a_1) = 0.995;\ Q(a_2) = 0.831;\ Q(a_3) = 0.707;\ Q(a_4) = 0.049.$$

7. Максимум досягається на 1 множині
8. Отже,

   $$X \leftarrow X_2' = \left[\frac{\pi}{2}, \frac{5\pi}{8}\right).$$

9. Оскільки ми прийняли, що $m = 2$, то кінець алгоритму. Виберемо довільне значення з множини X, яке і будемо вважати оптимальною дією

   $$\alpha = \frac{17\pi}{32} \in X.$$

Як бачимо, ми достатньо точно знайшли дію, що максимізує функцію корисності, виконавши мінімум обчислень.

2.3. R, S – ВИНАГОРОДА ВІД СЕРЕДОВИЩА. ПЕРЕХІД У НОВИЙ СТАН

Після виконання будь-якої дії агент отримує миттєву винагороду від середовища. У цій праці всі дії, які не приводять у термінальний стан, мають однакову винагороду. Всі термінальні стани мають своє значення винагороди.

Епізод може завершитись однією з таких подій (всі вони характеризують термінальні стани):

- *зіткнення* – агент зіткнувся з перешкодою;

- *перевищення часу виконання* – агент вичерпав кількість ітерацій, відведених йому для досягнення цілі;
- *успіх* – агент завершив рух у деякій точці цілі за прийнятну кількість ітерацій.

Вибір правильних значень винагород суттєво впливає на швидкість навчання агента і на отриману стратегію. Наприклад, використання невеликого від'ємного значення винагороди за дію, що не приводить у термінальний стан, спонукає агента відпрацьовувати стратегію спрямовану на зменшення кількості ітерацій в епізоді. У підсумку агент навчиться вибирати з можливих шляхів до цілі коротший.

Після вибору та виконання оптимальної дії агент переходить у новий стан. У задачі навігації дією є рух у просторі. У цьому дослідженні після кожної ітерації агент рухається по прямій, попередньо вибравши напрям руху.

2.4. A – ОНОВЛЕННЯ ЗНАЧЕННЯ ФУНКЦІЇ КОРИСНОСТІ

Після вибору наступної дії агент повинен оновити функцію корисності $Q^{\pi}(s_t, a_t)$ згідно з формулою (2.7). Це виконується з допомогою алгоритмів навчання нейронної мережі. Загальна схема алгоритмів навчання нейронної мережі така:

1) ініціалізувати ваги та функції активації малими ненульовими значеннями;
2) подати на вхід нейронної мережі один набір значень і обчислити вихід;
3) обчислити похибку;
4) змінити ваги та параметри функції активації так, щоб похибка зменшилась;
5) повторювати кроки 2-4, доки похибка не перестане зменшуватись, або досягне достатньо малого значення.

Ми вибрали алгоритм зворотного поширення похибки. При навчанні на кожній ітерації параметри змінюються відповідно до напряму антиградієнта функції похибки. Сам алгоритм фактично полягає в швидкому обчисленні антиградієнта [13].

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТИ

3.1. ІНФОРМАЦІЙНА СИСТЕМА МОДЕЛЮВАННЯ ПОВЕДІНКИ АГЕНТА

Для моделювання поведінки агентів у задачі навігації, а також тестування, відлагодження алгоритмів навчання й автоматизації проведення експериментів було розроблено інформаційну систему, яка уможливлює проведення експериментів. Мета цього середовища – підтримати процес відлагодження існуючих і розроблення нових алгоритмів для розв'язання задачі навігації.

Деталі реалізації та використання цієї системи описано в [11].

3.2. РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМУ З ВИКОРИСТАННЯМ НАПЕРЕД ЗАДАНОГО НАБОРУ ДІЙ

Під час аналізу досліджували вплив на роботу алгоритму таких чинників:

- крок навчання в алгоритмі Sarsa (параметр α);
- коефіцієнт γ алгоритму Sarsa, що відповідає за прогнозування винагород;
- ймовірність ε, з якою вибирається випадкова дія під час навчання;
- винагорода, яку дають за перевищення ліміту ітерацій при виконанні дії, яка не приводить у термінальний стан;
- кількість можливих напрямів руху;
- параметри N та r для кодування відстані до перешкод, відстані до цілі, кута напряму до цілі;

- структура нейронної мережі (кількість шарів і кількості нейронів у шарах);
- крок навчання для нейронної мережі.

Було проведено навчання агента на близько 20 000 різних наборів зазначених параметрів. Виявили і врахували такі закономірності:

- використовувати більше, ніж 2 шарів у нейронній мережі недоцільно. Мережі з більшою кількістю шарів значно довше вчаться;
- винагороди за успіх і за зіткнення можна прийняти такими, що дорівнюють 1 та – 1, відповідно, без втрати загальності;
- оптимальною функцією активації для нейронів є логістична;
- найліпше використовувати кількості напрямів руху кратні 4-м, оскільки при такому поділі агент має змогу рухатись вздовж осей координат;
- передача на вхід нейронної мережі додаткових параметрів (попередні дії, попередні нагороди тощо) лише сповільнює швидкість навчання та незначно поліпшує результати.

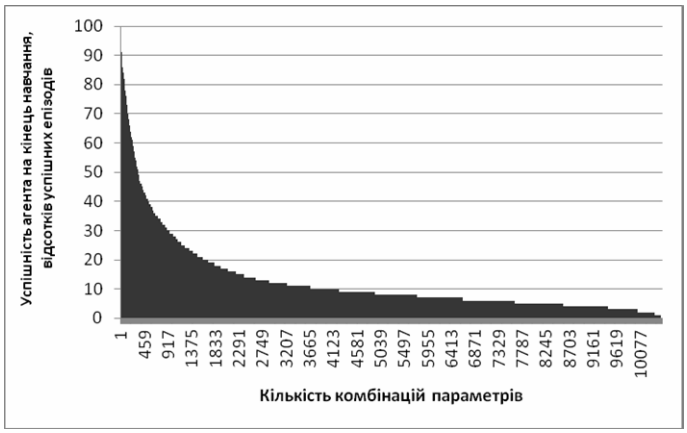
Зокрема, під час проведеного аналізу використовували всілякі можливі комбінації значень параметрів подані в табл. 1.

Таблиця 1

Комбінації параметрів, які аналізували

Параметр	**Значення**
Крок навчання в алгоритмі Sarsa	0.5; 0.8; 0.9; 1.0
Коефіцієнт γ алгоритму Sarsa	0.9; 0.95; 0.98
Ймовірність вибору випадкової дії	0.1; 0.2
Винагорода за перевищення ліміту ітерацій	-0.5; 0; 0.5
Винагорода за дію, яка не приводить у термінальний стан	-0.005; -0.0005
Кількість можливих напрямів руху	4; 8
Параметр N у кодуванні відстані до цілі	6
Параметр r у кодуванні відстані до цілі	4
Параметр N у кодуванні напряму до цілі	3; 6
Параметр N у кодуванні відстані до перешкоди	2; 3
Параметр r у кодуванні відстані до перешкоди	1
Кількість нейронів у внутрішньому шарі нейронної мережі	3; 4; 7
Залежність кроку навчання нейронної мережі від поточної ітерації	Стала, яка дорівнює 1 Лінійно спадна від 1 до 0.1 за 100000 ітерацій Лінійно спадна від 1 до 0.1 за 500000 ітерацій

Загалом це становило 10 368 дослідів. Загальна успішність під час аналізу цих комбінацій параметрів подана у діаграмі 1. Проаналізували відсоток успішних епізодів з останніх 100 під час навчання. Як бачимо, лише 11 комбінацій параметрів (~0.1 %) змогли завершити успішно більше, ніж 90 епізодів зі 100.



Діаграма 1. Успішність навчання

У навчанні агента використовували інкрементальний підхід. Спочатку навчання проводили у середовищах з меншою кількістю перешкод, згодом їхня кількість збільшилась.

У результаті аналізу вдалось досягнути стратегії близької до оптимальної вже після навчання протягом 1400 епізодів.

У цьому разі агента навчали у закритих середовищах розміром 6x6 метрів. За одну ітерацію агент мав змогу пройти максимум 0.2 метри і виконати не більше 50 ітерацій за епізод. Відносна похибка, яку давали сенсори, що визначали відстань до навколишніх перешкод, становила 10%. Максимальна ширина/висота перешкод і цілі становила 1 метр. Мінімальна відстань між перешкодами – 0.1 метра. Мінімальна площа кожної перешкоди і цілі становила 0.3 метра квадратного. Агент на початку руху розміщувався в початку координат, на відстані 0.5 метра навколо нього не було перешкод. Ціль знаходили мінімум за 0.7 метра від агента. Перші 500 епізодів у середовищі була одна перешкода, наступні 200 – 2 перешкоди, наступні 300 – 3 перешкоди, останні 400 епізодів у середовищі було 5 перешкод.

Серед усіх наборів параметрів найліпший подано у табл. 2.

Таблиця 2

Оптимальний набір параметрів

Параметр	**Значення**
Крок навчання алгоритму Sarsa	0.9
Коефіцієнт γ алгоритму Sarsa	0.9
Ймовірність вибору випадкової дії	0.2
Винагорода за перевищення ліміту ітерацій	0
Винагорода за дію, що не приводить у термінальний стан	-0.005
Кількість можливих напрямів руху	4
Кодування відстані до цілі	N=6, r=4
Кодування напряму до цілі	N=6
Кодування відстані до перешкоди	N=3, r=1
Кількість нейронів у внутрішньому шарі нейронної мережі	7
Залежність кроку навчання нейронної мережі від поточної ітерації	Стала, яка дорівнює 1

На рис. 5 зображено співвідношення епізодів, які завершились успіхом (штрихова крива), перевищенням допустимої кількості ітерацій (пунктирна крива) і зіткненням (суцільна крива) залежно від кількості епізодів, впродовж яких навчався алгоритм. На рис. 6 показано середню кількість ітерацій впродовж епізоду залежно від кількості епізодів, протягом яких навчався алгоритм.

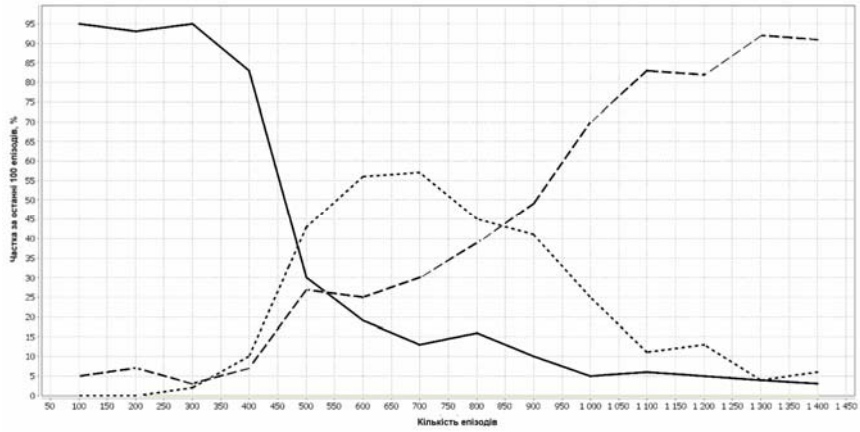


Рис. 5. Процес навчання алгоритму. Співвідношення результатів: штрихова крива – успіх; пунктирна крива – перевищення допустимої кількості ітерацій; суцільна крива – зіткнення

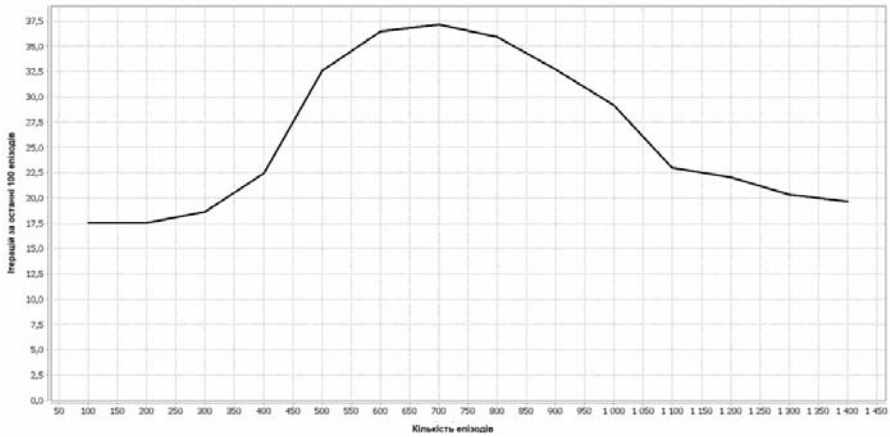


Рис. 6. Процес навчання алгоритму. Середня кількість ітерацій

Аналіз ефективності роботи алгоритму проводили у схожих середовищах, як і під час навчання, але в середовищі було розташовано 5 перешкод. Аналіз тривав протягом 5000 епізодів. Результати аналізу роботи алгоритму подано в табл. 3. На рисунку 7 зображено приклади руху агента в кількох випадково вибраних середовищах під час аналізу.

Таблиця 3

Результати аналізу алгоритму

Показник	**Значення**
Загальна кількість епізодів	5000
Загальна кількість ітерацій	74547
Успішних епізодів	4609 (92.18%)
Епізодів, які завершились зіткненням	44 (0.88%)
Епізодів, що закінчились перевищенням допустимої кількості ітерацій	347 (6.94%)

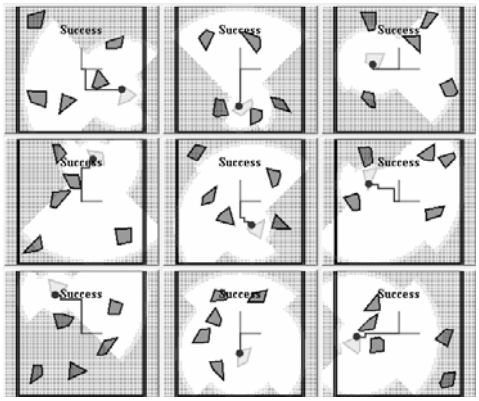


Рис. 7. Приклад роботи агента в середовищах під час аналізу

3.3. РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМУ З ВИКОРИСТАННЯМ НЕПЕРЕРВНОГО ПРОСТОРУ ДІЙ

Під час аналізу досліджували вплив на роботу алгоритму таких чинників:

- крок навчання в алгоритмі Sarsa (параметр α);
- коефіцієнт γ алгоритму Sarsa, який відповідає за "прогнозування" винагород;
- ймовірність ε, з якою вибирають випадкову дію під час навчання;
- винагорода, яку дають за перевищення ліміту ітерацій за виконання дії, що не приводить у термінальний стан;
- параметри N та r для кодування відстані до перешкод, відстані до цілі, кута напряму до цілі, напряму в якому агент буде рухатись;
- параметри k та m алгоритму поділу простору дій;
- структура нейронної мережі (кількість шарів і кількості нейронів у шарах);
- крок навчання для нейронної мережі.

Провели навчання агента на близько 500 різних наборів зазначених параметрів. Виявили та врахували такі закономірності:

– використовувати більше, ніж 2 шари у нейронній мережі недоцільно. Мережі з більшої кількістю шарів значно довше вчаться;

– винагороди за успіх і за зіткнення можна прийняти такими, що дорівнюють 1 та – 1, відповідно, без втрати загальності;
– оптимальною функцією активації для нейронів є логістична.

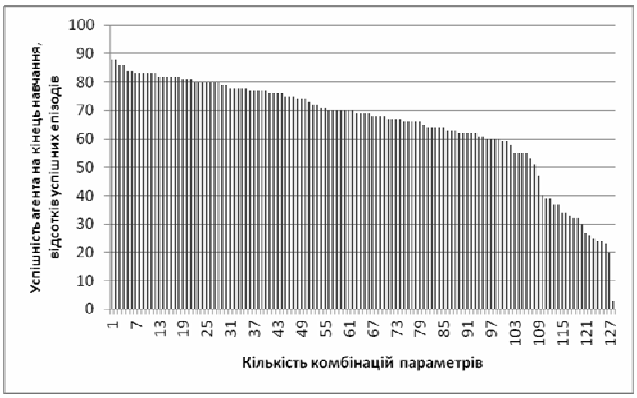
Під час проведеного аналізу використовували всілякі можливі комбінації значень параметрів поданих у табл. 4.

Таблиця 4

Комбінації параметрів, які аналізували

Параметр	**Значення**
Крок навчання алгоритму Sarsa	0.9
Коефіцієнт γ алгоритму Sarsa	0.98
Ймовірність вибору випадкової дії	0.2
Винагорода за перевищення ліміту ітерацій	0
Винагорода за дію, яка не приводить у термінальний стан	-0.005
Параметр *N* у кодуванні відстані до цілі	3; 5
Параметр *R* у кодуванні відстані до цілі	3; 5
Параметр *N* у кодуванні напряму до цілі	4; 6
Параметр *N* у кодуванні напряму руху	4; 6
Параметр *N* у кодуванні відстані до перешкоди	2; 3
Параметр *R* у кодуванні відстані до перешкоди	2; 3
Параметри *k* та *m* алгоритму	*k*=8; *m*=2
Кількість нейронів у внутрішньому шарі нейронної мережі	4; 8
Залежність кроку навчання нейронної мережі від поточної ітерації	Стала, яка дорівнює 1

Загалом це становило 128 дослідів. Загальна успішність під час аналізу цих комбінацій параметрів зображена на діаграмі 2. Проаналізували відсоток успішних епізодів з останніх 100 під час навчання.



Діаграма 2. Успішність навчання

У навчанні агента використовувались середовища аналогічні використаним у навчанні й аналізі попереднього підходу. Довелось продовжити навчання з 1400 епізодів до 5000. Перші 500 епізодів у середовищі була одна перешкода, наступні 2000 – 2 перешкоди, наступні 1500 – 3 перешкоди, останні 1000 епізодів у середовищі було 5 перешкод. Серед усіх наборів параметрів найліпше себе виявив поданий у табл. 5.

Таблиця 5

Оптимальний набір параметрів

Параметр	**Значення**
Крок навчання алгоритму Sarsa	0.9
Коефіцієнт γ алгоритму Sarsa	0.98
Ймовірність вибору випадкової дії	0.2
Винагорода за перевищення ліміту ітерацій	0
Винагорода за дію, яка не приводить у термінальний стан	–0.005
Кодування відстані до цілі	N=3; r=3
Кодування напряму до цілі	N=6
Кодування напряму руху	N=6
Кодування відстані до перешкоди	N=3; r=2
Параметри *k* та *m* алгоритму	*k*=8; *m*=2
Кількість нейронів у внутрішньому шарі нейронної мережі	8
Залежність кроку навчання нейронної мережі від поточної ітерації	Стала, яка дорівнює 1

На рис. 8 зображено співвідношення епізодів, які завершились успіхом (штрихова крива), перевищенням допустимої кількості ітерацій (пунктирна крива) і зіткненням (суцільна крива) залежно від кількості епізодів, впродовж яких навчався алгоритм. На рис. 9 подано середню кількість ітерацій впродовж епізоду залежно від кількості епізодів, впродовж яких навчався алгоритм.

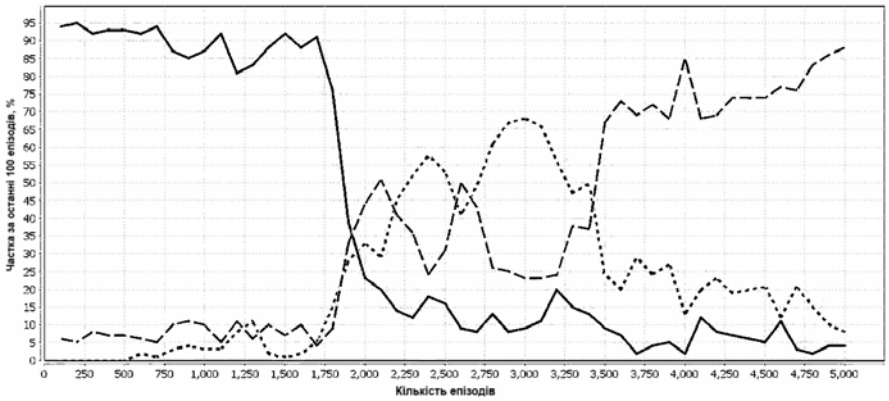


Рис. 8. Процес навчання алгоритму. Співвідношення результатів: штрихова крива – успіх; пунктирна крива – перевищення допустимої кількості ітерацій; суцільна крива – зіткнення

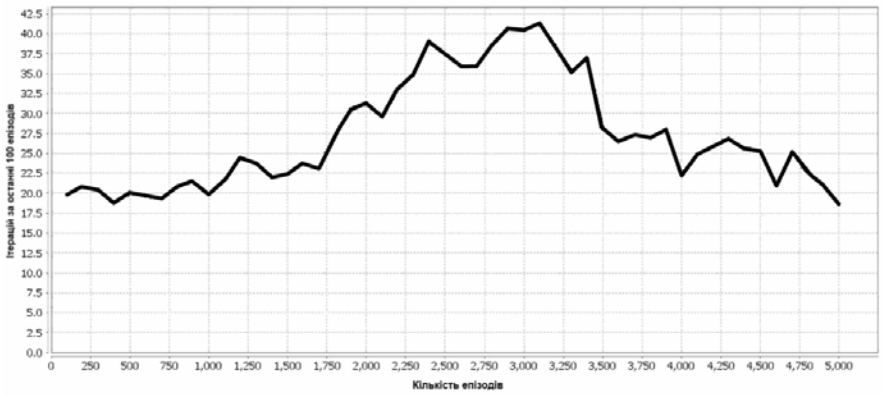


Рис. 9. Процес навчання алгоритму. Середня кількість ітерацій

Аналіз ефективності роботи алгоритму проводили в аналогічних середовищах, як і під час аналізу попереднього підходу. Результати аналізу роботи алгоритму подано в табл. 6. На рис. 10 зображено приклади руху агента в кількох випадково вибраних середовищах під час аналізу.

Таблиця 6

Результати аналізу алгоритму

Показник	**Значення**
Загальна кількість епізодів	5000
Загальна кількість ітерацій	70995
Успішних епізодів	4386 (87.72%)
Епізодів, які завершились зіткненням	384 (7.68%)
Епізодів, які завершились перевищенням допустимої кількості ітерацій	230 (4.6%)

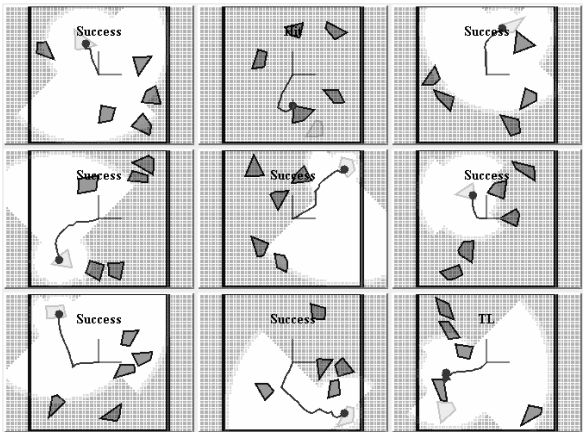


Рис. 10. Приклад роботи агента в середовищах під час аналізу

3.4. ПОРІВНЯННЯ РОБОТИ АЛГОРИТМІВ, ЯКІ ВИКОРИСТОВУЮТЬ РІЗНІ МЕТОДИ КОДУВАННЯ ДІЙ

Для порівняння ефективності в алгоритмі, який ґрунтується на наперед заданому наборі дій, використовували набір параметрів, описаний у табл. 2, а в алгоритмі, що використовує неперервний простір дій, – описаний у табл. 5.

Табл. 7 і 8 містять результати такого порівняння. У табл. 7 подано статистику, яка засвідчила загальну успішність алгоритмів під час роботи. У табл. 8 подано статистику, яка засвідчує ефективність роботи алгоритмів один стосовно одного. Клітинка (i, j) цієї таблиці містить інформацію про кількість епізодів, що алгоритм з заданим набором дій завершив їх з результатом, що відповідає i-му рядку, а алгоритм з неперервним простором дій завершив їх з результатом, що відповідає j-му стовпцю.

Таблиця 7

Порівняння роботи алгоритмів. Загальна статистика

Показник	**Алгоритм з заданим набором дій**	**Алгоритм з неперервним простором дій**
Загальна кількість епізодів	5000	
Загальна кількість ітерацій	73378	70779
Успішних епізодів	4640 (92.8%)	4433 (88.66%)
Епізодів, які завершились зіткненням	37 (0.74%)	352 (7.04%)
Епізодів, які завершились перевищенням допустимої кількості ітерацій	323 (6.46%)	215 (4.3%)

Таблиця 8

Порівняння роботи алгоритмів. Ефективність один стосовно одного

Алгоритм 1/ Алгоритм 2	**Успіх**	**Зіткнення**	**Тайм-аут**
Успіх	4169	304	167
Зіткнення	11	12	14
Тайм-аут	253	36	34

Як бачимо, ліпшої якості результату та швидкості навчання вдалось досягнути за допомогою алгоритму, який ґрунтується на наперед заданому наборі дій. Це зумовлено тим, що за такого підходу агенту завжди відома приблизна відстань до перешкоди за напрямом, у якому відбувається рух. Алгоритм, який ґрунтується на неперервному просторі дій, потребує подальшого дослідження і удосконалення.

Як видно з таблиці 8, алгоритми обчислили абсолютно різні стратегії, оскільки для деяких епізодів краще справляється один, а для деяких – інший алгоритм.

4. ВИСНОВКИ

Для зручного і якісного аналізу алгоритмів навігації інтелектуального агента створили інформаційну систему, яка допомагала проводити моделювання поведінки агента у віртуальному середовищі.

За допомогою цієї системи проаналізували роботу алгоритмів керування агента, які ґрунтуючись на обмеженій інформації про навколишнє середовище, забезпечують рух агента до цілі, уникаючи зіткнення з перешкодами.

Головним інструментом у роботі цих алгоритмів стали штучні нейронні мережі. Було підтверджено практичну доцільність використання штучних нейронних мереж як апроксиматора в алгоритмах навчання з підкріпленням для вирішення задачі навігації.

Проаналізовано вплив параметрів і структури нейронної мережі на швидкість навчання та якісні характеристики отриманої стратегії. Побудували нейронну мережу, яка виявила результати достатні для розв'язання задачі навігації реальним роботом у реальних умовах за допустимий час.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Azouaoui O*. Soft Computing Based Pattern Classifiers for the Obstacle Avoidance Behavior of Intelligent Autonomous Vehicles (IAV) / O. Azouaoui, A. Chohra. – Applied Intelligence. – 2002. – P. 249–272.
2. *Davis I.L*. A Modular Neural Network Approach to Autonomous Navigation / I.L. Davis. – 1996. – 154 p.
3. *Hachour O*. A Neural Network Based Navigation for Intelligent Autonomous Mobile Robots / O. Hachour // International Journal of Mathematical Models and Methods in Applied Sciences Processings. – 2010.
4. *Hachour O*. Cognitive tasks behavior of Intelligent Autonomous Mobile Robots / O. Hachour // International Journal of Mathematical Models and Methods in Applied Sciences Processings. – 2011.
5. *Janglová D*. Neural Networks in Mobile Robot Motion / D. Janglová // Inernational Journal of Advanced Robotic Systems. – 2004. – P. 15–22
6. *Lin L.-J*. Reinforcement Learning for Robots Using Neural Networks. Master's thesis / L.-J. Lin. – Carnegie-Mellon Univeristy of Pittsburgh Department of Computer Science, 1993.
7. *Ozkucur N.E*. Multi-Agent Visual-SLAM Algorithms on Autonomous Robots / N.E. Ozkucur. – LAP LAMBERT Academic Publishing, 2010. – 120 p.
8. *Rummery G.A*. Problem Solving with Reinforcement Learning / G.A. Rummery. – Ph.D. thesis. Cambridge University Engineering Department, 1995.
9. *Russel S*. Artificial Intelligence: A Modern Approach / S. Russel, P. Norvig. – Pearson, 2010. – 1152 p.
10. *Sutton R.S*. Reinforcement Learning: An Introduction / R.S. Sutton, A.G. Barto. – A Bradford Book, 1998. – 551 p.
11. *Годич О*. Інтелектуальна система моделювання поведінки агентів у задачі навігації / О. Годич, П. Кушнір, Ю. Щербина // Вісн. Львів. ун-ту. Сер. прикл. матем. та інформ. – 2012. – Вип. 18.– С. 208–221.
12. *Годич О*. Індуктивні методи моделювання навігації агента / О. Годич, А. Накрийко, Ю. Щербина / Вісн. Львів. ун-ту. Сер. прикл. матем. та інформ. – 2009. – Вип. 15.– С 291-312.
13. *Заенцев И.В*. Нейронные сети: основные модели / И.В. Заенцев. – 1999. – 76 с.
14. *Накрийко А*. Індуктивні методи самоорганізації агента в задачах навігації. Master's thesis / А. Накрийко. – ЛНУ ім. І. Франка, 2010.

Стаття: надійшла до редколегії 24.09.2012
доопрацьована 31.10.2012
прийнята до друку 14.11.2012

ПРИНЯТИЕ РЕШЕНИЙ АГЕНТОМ НАВИГАЦИИ С ПОМОЩЬЮ АППРОКСИМАЦИИ ПРОСТРАНСТВА СОСТОЯНИЙ И ДЕЙСТВИЙ ИСКУССТВЕННЫМИ НЕЙРОННЫМИ СЕТЯМИ

О. Годыч[1], П. Кушнир[2], Ю. Щербина[2]

[1]Национальный университет "Львовская политехника",
ул. С. Бандеры, 12, Львов, 79013
[2]Львовский национальный университет имени Ивана Франко,
ул. Университетская, 1, Львов, 79000, e-mail: dais@franko.lviv.ua

Рассмотрена целесообразность использования искусственных нейронных сетей в качестве аппроксиматора для решения задачи навигации. Разработано программное обеспечение для автоматизации проведения экспериментов. Исследованы различные подходы к кодированию пространства состояний и действий. Осуществлен анализ влияния параметров и структуры искусственной нейронной сети на скорость обучения и качественные характеристики стратегии, полученной в результате обучения. На основании проведенного анализа построена нейронная сеть, способная получить близкую к оптимальной стратегию с использованием небольшой обучающей выборки.

Ключевые слова: задача навигации, интеллектуальные агенты, программное обеспечение, моделирование, искусственные нейронные сети, обучение с подкреплением.

NAVIGATION AGENT DECISION BASED ON STATES AND ACTS SPACE APPROXIMATION WITH ARTIFICIAL NEURAL NETWORKS

O. Hodych[1], P. Kushnir[2], Y. Shcherbyna[2]

[1]Lviv Polytechnic "National University",
Bandery str., 12, Lviv, 79013
[2]Ivan Franko National University of Lviv,
Universytetska Str., 1, Lviv, 79000, e-mail: dais@franko.lviv.ua

This article discusses the possibility of using artificial neural networks for approximation in navigation problem solving. Developed software for experiment automation. Discussed different methods for encoding space of states and actions. Analyzed influence of artificial neural network parameters and structure on learning time and the quality of strategies. Using this analysis created artificial neural network able to obtain near optimal strategy using relatively small amount of training samples.

Key words: navigation problem, intelligent agent, software, modeling, artificial neural network, reinforcement learning.