Міністерство освіти і науки України
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики

Кафедра дискретного аналізу
та інтелектуальних систем

Курсова робота

на тему:

"Індуктивні методи самоорганізації агента в задачах навігації"

Виконав
ст. групи ПМі-43
Накрийко А.В.

Науковий керівник
ас. Годич О.В.

Львів - 2009

Зміст

1 Вступ

Проблема побудови автоматичих контролерів для роботів або інших складних механізмів була серйозним викликом для вчених та інженерів з перших днів комп'ютерної ери і залишається не менш складною і сьогодні. Дуже часто постановка задачі контролю звучить наступним чином: *"Для досягнення мети агент повинен взаємодіяти з зовнішнім середовищем, виконуючи дії з множини допустимих дій. Яка буде оптимальна послідовність дій для досягнення поставленої мети?"* Простота постановки задачі оманлива. Навчити штучні машини повторювати людську поведінку, діяти логічно та ефективно у будь-якій ситуації, як виявилося, надзвичайно складно. Було розроблено безліч технік та методик, які намагаються вирішити дану проблему з більшим або меншим успіхом.

Ця робота є спробою застосувати одну з таких методик, а саме — навчання з підсиленням з використанням штучних нейронних мереж, для розв'язання задачі оптимального керування роботом в умовах сереодвища, заповненого перешкодами.

У парадигмі навчання з підсиленням, агенту дається можливість експериментувати з множиною допустимих дій для вироблення оптимальної стратегії шляхом проб та помилок. Якість вибраних дій підкріплюється через систему винагород та покарань, яка базується на результатах вибраних дій. Даючи більшу винагороду за дії, що наближають агента до мети та караючи за ті, що призводять до неефективного розв'язку, агент отримує можливість оцінити власну поведінку. В результаті численних повторень агентом буде вироблена оптимальна стратегія, яка найефективніше приводить до виконання завдання.

2 Навчання з підсиленням (reinforcement learning)

2.1 Короткий вступ

Навчання з підсиленням (Reinforcement learning) - навчання про те, які дії потрібно приймати залежно від ситуації таким чином, щоб максимізувати *числовий сигнал "винагороди"(reward signal)*. "Учневі" не вказується, які дії потрібно здійснювати, щоб досягти мети, а натомість дається можливість самому вибирати з певного набору допустимих дій. В результаті багаторазової взаємодії з середовищем шляхом методу спроб та помилок досягається оптимальна "стратегія" (послідовність дій, які ведуть до мети).

У загальному випадку, вибрані дії впливають не тільки на винагороду, що слідує безпосередньо після вчинення дії, але й на майбутні ситуації, що, в свою чергу, дає вплив на мабутні винагороди.

В навчанні з підсиленням присутні дві сутності — агент та середовище. Агент — це все те, на що ми можемо чинити безпосередній вплив. Накриклад, якщо ми маємо робота, то агентом може бути як весь робот, якщо ми здатні безпосередньо керувати його поворотом навколо своєї осі. Або агентом можна визначити лише електромотори, які зумовлюють поворот коліс та їх рух. В останньому випадку ми безпосередньо можемо впливати тільки на силу струму, що подається на електромотори. На самі ж колеса ми впливаємо опосередковано, тому вони вважаються середовищем, по відношенню до нашого агента.

Стан системи — набір параметрів, які задають наші знання (можливо неповні) про оточуючий світ в конкретний момент часу. Це, фактично, наше сприйняття поточної ситуації. Наприклад, у випадку автономного керування транспортним засобом, станом буде інформація про відстань до найближчої перешкоди, відстань до заданої цілі, можливо, поточна швидкість самого транспортного засобу тощо.

Складовими частинами системи навчання з підсиленнями є:

- *Стратегія (policy)*, яка визначає те, які дії будуть прийматися залежно від ситуації на даний момент часу. Стратегія — це, фактично, інтелект агента.
- *Функція винагороди (reward function)*, яка задає відображення зі стану системи та здійсненої дії у множину дійсних чисел — множину винагород. Тобто після кожної здійсненої дії, залежно від стану системи, агент отримує певне миттєве значення винагороди. Глобальною метою агента є максимізація загальної винагороди, отриманої протягом всього часу його дії.
- *Вартісна функція (value function)*, яка, на відміну від функції винагороди, задає не те, що є "корисним" безпосередньо зараз, а те, що є корисним в більш глобальному значенні: з урахуванням майбутніх станів.

Розглянемо ці поняття більш формально, базуючись на інформації з [2].

2.2 Модель навчання з підсиленням

Агент (agent) та *середовище (environment)* взаємодіють в кожен дискретний момент часу $t = 0, 1, 2, \ldots$. На кожному кроці t агент отримує певне представлення *стану системи (state)* $s_t \in \mathcal{S}$, де $\mathcal{S}$ — множина усіх станів системи, і на базі цього вибирає *дію (action)* $a_t \in \mathcal{A}(s_t)$, де $\mathcal{A}(s_t)$ — множина допустимих дій у стані s_t. На наступному кроці $t+1$, частково як наслідок виконаної дії, агент отримує числову *винагороду (reward)* $r_{t+1} \in \mathbb{R}$ і опиняється в стані s_{t+1}.

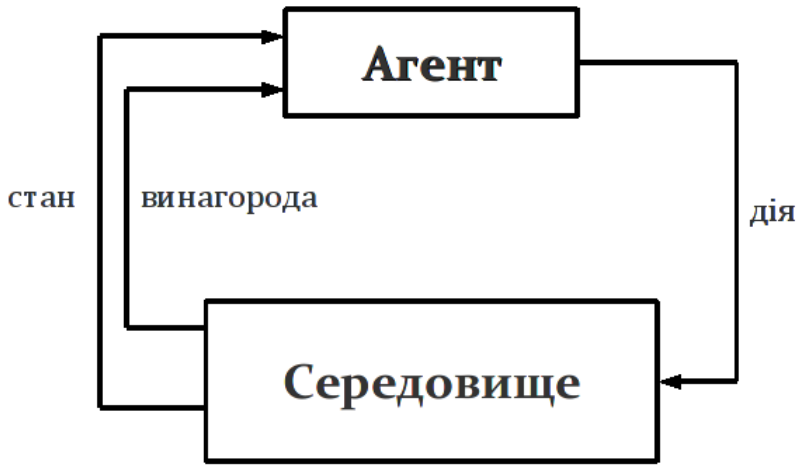


Рис. 1: Взаємодія агента та середовища в навчанні з підсиленням.

В загальному випадку, на кожному кроці агент реалізовує відображення станів системи на ймовірності здійснення можливої дії. Таке відображення називається *стратегією (policy)* агента і позначається π_t, де $\pi_t(s, a)$ — це ймовірність, що $a_t = a$, якщо $s_t = s$.

Методи навчання з підсиленням визначають, як саме агент змінює свою стратегію в результаті набутого досвіду. Мета агента, грубо кажучи,— максимізувати сукупну винагороду, яку він отримує протягом багатьох ходів.

2.3 Прибуток (return)

Визначимо більш точно мету агента. Нехай, починаючи з часу t, агент отримує послідовність винагород $r_{t+1}, r_{t+2}, r_{t+3}, \ldots$. Тоді основною метою агента буде максимізація *очікуваного прибутку (expected return)* R_t, де R_t — це певна функція від послідовності винагород. В найпростішому випадку прибуток є сумою усіх винагород:

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T, \tag{1}$$

де T — кінцевий момент часу. Такий підхід до визначення прибутку підходить там, де можна природно визначити фінальний момент часу, тобто в таких системах, в яких взаємодія агента і середовища природньо розбивається на окремі послідовності (*епізоди*), наприклад, партії в грі, проходи по лабіринту або будь-який інший вид повторюваних взаємодій.

Кожен епізод закінчується у *кінцевому (термінальному) стані (terminal state)*, після якого відбувається встановлення стану системи в певний початковий стан з множини можливих початкових станів і розпочинається наступний епізод. Задачі, в яких взаємодія агента і середовища відбувається у вигляді послідовності епізодів, називаються *епізодичними задачами (episodic tasks)*. В епізодичних задачах інколи розмежовують множину усіх нетермінальних станів системи $\mathcal{S}$ та множину усіх станів, включно з термінальними $\mathcal{S}^+$.

З іншого боку, не завжди можна здійснити природній поділ на епізоди, натомість взаємодія відбуваєть постійно без закінчення. Прикладом такої задачі може бути діяльність робота з великою тривалістю життєдіяльності. Такі задачі назвемо *неперервними задачами (continuing tasks)*. В такому випадку кінцевий момент часу $T = \infty$, а сам прибуток, який ми намагаємося максимізувати може легко прямувати до нескінченості (наприклад, якщо на кожному кроці $r_t = 1$). Щоб подолати ці труднощі, введемо дещо інше визначення прибутку.

Розглянемо додаткову концепцію — *дисконтування (discounting)*. Згідно з нею, агент намагаєтсья вибирати такі дії, щоб максимізувати суму дисконтованих винагород, які він отримає в майбутньому. Зокрема, агент вибирає таку дію a_t, яка максимізує очікуваний *дисконтований прибуток (discounted reward)*:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2}$$

де γ — це параметр, який називається *дисконтною ставкою (discount rate)*, $0 \leq \gamma \leq 1$.

Дисконтна ставка визначає поточну вагу майбутньої винагороди: винагорода, отримана через k кроків у майбутньому, має вагу лише γ^k, напротивагу одиничній вазі, яку б вона мала, якби була отримана на поточному кроці. Якщо $\gamma < 1$, то ця нескінченна сума має скінченне значення, якщо послідовність $\{r_k\}$ — обмежена. Якщо $\gamma = 0$, то агент є "недалекоглядним" і буде дбати лише про максимізацію миттєвої винагороди. Його задачею в такому випадку буде вибір такої дії a_t, щоб максимізувати лише винагороду r_{t+1}. Якщо дія агента не здійснює вплив на майбутні винагороди, то такий підхід дійсно дасть змогу вивчити оптимальну стратегію, однак в більшості випадків це не так, тому максимізація лише миттєвої нагороди може призвести до того, що буде втрачено нагоду "заробити" більше в майбутньому, здійснивши на даному кроці дію, яка принесе не максимальну миттєву винагороду. З наближенням γ до 1, агент буде надавати майбутнім винагородам все більшого значення: агент стає більш "далекоглядним".

2.4 Функція корисності (value function)

Практично всі алгоритми навчання з підсиленням базуються на оцінці *функції корисності (value function)* — функції станів системи (або пари стан-дія), що оцінює *наскільки добре* для агента перебувати в заданому стані (або наскільки добре здійснити задану дію, перебуваючи в заданому стані). Фраза "наскільки добре" визначається в термінах майбутніх винагород або, що є точнішим визначенням, в термінах очікуваного майбутнього прибутку. Очевидно, що майбутні винагороди залежать від того, які дії здійснювати, тобто від стратегії, тому ціннісна функція визначається з врахуванням певної стратегії.

Нагадаємо, що стратегія π — це відображення кожного стану $s \in \mathcal{S}$ та кожної допустимої дії $a \in \mathcal{A}(s)$ на ймовірність $\pi(s,a)$ здійснення дії a, будучи в стані s. *Корисністю* стану s згідно стратегії π, $V^\pi(s)$,— це очікуваний прибуток, який можна отримати, якщо почати зі стану s і діяти згідно стратегії π:

$$V^\pi(s) = E_\pi\Big\{R_t\Big|s_t = s\Big\} = E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle|s_t = s\right\}, \tag{3}$$

де E_π позначає математичне очікування в випадку, якщо агент діє згідно стратегії π. Потрібно відмітити, що цінність термінального стану завжди рівна нулю. Назвемо функцію V^π *функцією корисності станів (state-value function) для стратегії* π.

Аналогічно, визначимо корисність здійснення дії a в стані s згідно стратегії π, $Q^\pi(s,a)$, як очікуваний прибуток, отриманий в результаті перебування в стані s, здійснення дії a та подальшому слідуванні стратегії π:

$$Q^\pi(s,a) = E_\pi\Big\{R_t\Big|s_t = s, a_t = a\Big\} = E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle|s_t = s, a_t = a\right\}. \tag{4}$$

Назвемо функцію Q^π *функцією корисності дій (action-value function) для стратегії* π.

Функції корисності V^π та Q^π можуть бути оцінені з досвіду. Наприклад, якщо агент притримується стратегії π і підтримує середнє значення всіх прибутків, які були реально отримані, для кожного стану, в якому перебував агент, то середнє значення збіжиться до значення цінності для даного стану $V^\pi(s)$ за умови, що кількість перебувань в кожному зі станів прямує до нескінченності. Якщо зберігати середні значення окремо для кожної пари стан-дія, то таке ж твердження вірне і для функції Q^π. Такі методи оцінки цінових функцій називаються *методами Монте-Карло*, тому що вони використовують усереднення по випадкових прикладах реальних прибутків. Проте, якщо кількість станів та можливих дій велика, це унеможливлює збереження середніх значень для кожного стану (пари стан-дія). В такому випадку можна застосувати інші методи, зокрема такі, що використовують апроксиматори функцій. Ми розглянемо їх дещо пізніше.

Фундаментальна властивість функцій корисності, яка використовується в навчанні з підсиленням і динамічному програмуванні,— це певне рекурсивне співвідношення, якому задовільняє функція корисності. Для будь-якої фіксованої стратегії π та будь-якого стану s, наступне співвідношення завжди виконується:

$$\begin{aligned}
V^\pi(s) &= E_\pi\Big\{R_t\Big|s_t = s\Big\} \\
&= E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle|s_t = s\right\} \\
&= \sum_a \pi(s,a)\sum_{s'}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\middle|s_{t+1} = s'\right\}\right] \\
&= \sum_a \pi(s,a)\sum_{s'}\mathcal{P}^a_{ss'}\left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\right],
\end{aligned} \tag{5}$$

де $\mathcal{P}^a_{ss'}$ — це ймовірність переходу (transition probability) зі стану s в стан s' при виконанні дії a; $\mathcal{R}^a_{ss'}$ — очікувана винагорода, яка дається агенту при переході зі стану

s в стан s' при виконанні дії a. В даній формулі неявно розуміється, що $s \in \mathcal{S}$, дії a беруться з множини $\mathcal{A}(s)$, а $s' \in \mathcal{S}$ у випадку неперервної задачі або $s' \in \mathcal{S}^+$ — у випадку епізодичної задачі. Вказані ймовірності та очікувані винагороди визначають повну динаміку системи і на практиці зазвичай не відомі. Тому доводиться шукати методи, які б не потребували повних знань про систему.

Рівняння (5) називається *рівнянням оптимальності Белмана для V^π (Bellman optimality equation for V^π)*. Воно відображає зв'язок між корисністю стану s та корисністю станів s', в які можна потрапити з s. Можна довести, що функція V^π — єдиний розв'язок рівняння Белмана. Відмітимо також, що це рівняння — основа для багатьох методів обчислення, наближення та вивчення V^π.

2.5 Оптимальні функції корисності

Розв'язування задачі навчання з підсиленням означає, згрубша, знаходження такої стратегії, яка б давала великий прибуток в довгостроковій перспективі. Спробуємо визначити оптимальність стратегії наступним чином. Функції корисності визначають частковий порядок на множині усіх стратегій. Стратегію π називатимемо *кращою*, ніж стратегія π', якщо її очікуваний прибуток більший або рівний від такого ж для стратегії π' для усіх станів $s \in \mathcal{S}$:

$$\pi \geq \pi' \Leftrightarrow \Big(\forall s \in \mathcal{S}\Big) \Big\{V^\pi(s) \geq V^{\pi'}(s)\Big\} \tag{6}$$

Завжди існує хоча б одна така стратегія, яка краща або рівна, ніж усі інші. Така стратегія називається *оптимальною*. Хоча оптимальних стратегій може бути декілька, всіх їх будемо позначати як π^*. Усі оптимальні стратегії мають одну і ту ж функцію корисності станів, яка називається *оптимальною функцією корисності станів* V^*:

$$V^* = \max_\pi V^\pi(s), \qquad \forall s \in \mathcal{S}. \tag{7}$$

Оптимальні стратегії також мають одну і ту ж *функцію корисності дій* Q^*:

$$Q^* = \max_\pi Q^\pi(s,a), \qquad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s). \tag{8}$$

Ми можемо також записати функцію Q^* в термінах V^*:

$$Q^*(s,a) = E\Big\{r_{t+1} + \gamma V^*(s_{t+1}) \Big| s_t = s, a_t = a\Big\}. \tag{9}$$

3 Методи розв'язування задач навчання з підсиленням

Існує три широких класи методів, призначених для розв'язування задач навчання з підсиленням:

- методи динамічного програмування (dynamic programming methods);
- методи Монте-Карло (Monte Carlo methods);

- метод часової різниці (temporal-difference learning).

Кожен з цих класів має свої сильні та слабкі сторони. Так, методи динамічного програмування є добре розробленими з математичної точки зору, але потребують повної і достатньо точної моделі середовища, що часто просто неможливо задовільнити. Методи Монте-Карло не потребують моделі середовища і є концептуально простими, проте вони не є пристосованими до покрокових послідовних обчислень. Методи ж навчання з часовою різницею не потребують моделі середовища і добре пристосовані до послідовних покрокових обчислень, але є складнішими для аналізу. Ці методи також відрізняються в плані ефективності та швидкості збіжності.

В даній роботі був використаний саме останній метод, на якому зупинимося більш детально.

3.1 Навчання з часовою різницею (TD-методи)

Ідея методів *навчання з часовою різницею (temporal-difference, TD-навчання)* є, мабуть, центральною та найбільш новаторською для всього навчання з підсиленням. TD-навчання — це поєднання ідей методів Монте-Карло та динамічного програмування. TD-методи можуть, як і методи Монте-Карло, навчатися лише з досвіду, не потребуючи знання моделі динаміки середовища. З іншого боку, так само як і в методах динамічного програмування, TD-методи оновлюють свої оцінки частково на основі інших оцінок, не чекаючи закінчення епізоду. Така оцінка на базі інших оцінок отримала назву *"стартування" (bootstrap).*

3.1.1 TD-прогнозування

Як TD, так і Монте-Карло методи використовують отриманий досвід для розв'язування задачі прогнозування. Маючи певний досвід, отриманий в результаті притримування стратегії π, обидва методи використовують його для оновлення наближення V до V^π. Якщо в момент часу t був відвіданий нетермінальний стан s_t, то обидва методи оновлять свої наближення $V(s_t)$, базуючись на тому, що відбудеться в наступні моменти часу. Так, метод Монте-Карло чекає моменту, коли стане відомим прибуток, отриманий після відвідання поточного стану, а тоді використовує цей прибуток як наближення для $V(s_t)$. Простий алгоритм Монте-Карло для всіх візитів (every-visit Monte Carlo) виглядає наступним чином:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[R_t - V(s_t) \right], \tag{10}$$

де R_t — реальний прибуток, отриманий протягом поточного епізоду, починаючи з моменту часу t, α — сталий параметр величини кроку.

В той час, як MC-метод чекає до закінчення епізоду, TD-метод повинен чекати лише наступного моменту часу. В момент $t+1$ алгоритм формує наближення на основі отриманої винагороди r_{t+1} та наближення $V(s_{t+1})$. Найпростіший TD-алгоритм, знаний як TD(0), виглядає наступним чином:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]. \tag{11}$$

Ініціалізувати $V(s)$ довільним значенням
Ініціалізувати π стратегією, яка відповідає згенерованій $V(s)$
Для кожного епізоду повторювати:
    Ініціалізувати s
    Повторювати для кожного кроку епізоду:
        $a \leftarrow$ дія, пропонована поточною стратегією π для стану s
        Виконати дію a; спостерегти винагороду r та наступний стан s'
        $V(s) \leftarrow V(s) + \alpha\left[r + \gamma V(s') - V(s)\right]$
        $s \leftarrow s'$
    поки s — нетермінальний стан

Табл. 1: Алгоритм TD(0) для наближення V^π.

Раніше вже згадувалося, що

$$\begin{aligned} V^\pi(s) &= E_\pi\left\{R_t \middle| s_t = s\right\} \qquad (12)\\ &= E_\pi\left\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle| s_t = s\right\}\\ &= E_\pi\left\{r_{t+1} + \gamma\sum_{k=0}^{\infty}\gamma^k r_{t+k+2}\middle| s_t = s\right\}\\ &= E_\pi\left\{r_{t+1} + \gamma V^\pi(s_{t+1})\middle| s_t = s\right\}. \qquad (13) \end{aligned}$$

Таким чином, з попереднього запису видно, що MC-метод використовує наближення з (12) як бажане значення ціннісної функції, а DP-метод — з (13). Бажане значення для MC є наближенням, оскільки невідоме математичне сподівання. В випадку з DP воно є наближенням не через матсподівання, яке відоме з моделі динаміки середовища, а через те, що саме значення $V^\pi(s_{t+1})$ невідоме. Для TD-методу ми ж отримуємо наближення і через невідомість матсподівання, і через невідомість реального значення $V^\pi(s_{t+1})$. В табл. 1 поданий алгоритм TD(0).

3.1.2 Алгоритм Sarsa

Розглянемо тепер застосування TD-алгоритмів до задачі контролю. Натомість функції корисності станів ми будемо намагатися вивчити функцію корисності стану-дії $Q^\pi(s,a)$. Нагадаємо, що епізод складається з почергової послідовності станів та пар стан-дія:

Раніше ми розглядали переходи зі стану в стан і вивчення цінності станів. Тепер ми будемо розглядати переходи між парами стан-дія і вивчення цінності стану-дії. Відповідне правило навчання виглядає наступним чином:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]. \qquad (14)$$

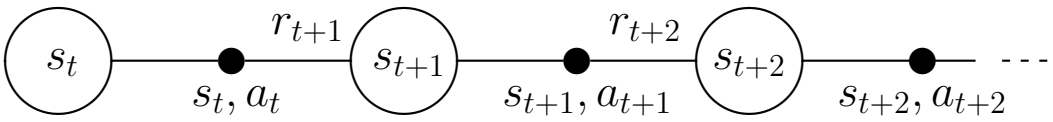


Рис. 2: Послідовність подій в Sarsa

Ініціалізувати $Q(s, a)$ довільним чином
Для кожного епізоду повторювати:
    Ініціалізувати s
    Обрати a зі стану s згідно стратегії, що базується на Q (наприклад, ε-жадібну)
    Повторювати для кожного кроку епізоду:
        Здійснити дію a, спостерегти r, s'
        Обрати a' зі стану s' згідно стратегії
        $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a)\right]$
        $s \leftarrow s'$; $a \leftarrow a'$;
    поки s — нетермінальний стан

Табл. 2: TD-алгоритм контролю Sarsa.

Це оновлення робиться після кожного переходу з нетермінального стану s_t. Якщо s_{t+1} — термінальний стан, то $Q(s_{t+1}, a_{t+1})$ визначається рівним нулю. Це правило використовує кожен елемент з п'ятірки подій, $(s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1})$ (рис. 2), що становлять перехід з однієї пари стан-дія в іншу. Ця п'ятірка і дала назву алгоритму — *Sarsa*.

3.1.3 Q-навчання

Ще одним алгоритмом, який широко застосовується в задачах контролю, є Q-навчання. Теоретично він значно краще розроблений, ніж Sarsa, проте на практиці зазвичай дає гірші результати. Правило оновлення для однокрокового алгоритму Q-навчання:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right]. \tag{15}$$

Незважаючи на незначні відмінності від Sarsa, Q-навчання є цілком іншим алгоритмом, реалізація якого дещо ускладнена, проте можливості для теоретичного аналізу значно кращі. І хоча Q-навчання теоретично більш обґрунтоване, на практиці саме Sarsa показує зазвичай кращі результати.

Ініціалізувати $Q(s,a)$ довільним чином
Для кожного епізоду повторювати:
  Ініціалізувати s
  Повторювати для кожного кроку епізоду:
    Обрати a зі стану s згідно стратегії, що базується на Q (наприклад, ε-жадібну)
    Здійснити дію a, спостерегти r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  поки s — нетермінальний стан

Табл. 3: Алгоритм Q-навчання для задачі контролю.

3.2 Стратегії вибору дій

В усіх розглянутих алгоритмах потрібно на кожному кроці робити вибір дії, яку повинен здійснити агент. Це можна робити різними способами, проте, для збіжності стратегії до оптимальної, необхідною умовою є те, щоб кожна можлива дія була випробувана безліч раз при безмежній кількості ітерацій. Таку умову задовільняють так звані soft-стратегії, в яких ймовірність $\pi(s,a) > 0,\ \forall \pi, \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$.

Одним з найпростіших випадків soft-стратегії є ε-жадібна стратегія. В такій стратегії зазвичай вибирається дія, яка дає поточний максимум ціннісної функції, проте інколи, з ймовірністю ε, приймається рішення про здійснення довільної випадкової допустимої дії. Таким чином будь-яка не жадібна дія отримує ймовірність виконання, рівну $\frac{\varepsilon}{|\mathcal{A}(s)|}$, а жадібна дія, відповідно, $1 - \varepsilon + \frac{\varepsilon}{|\mathcal{A}(s)|}$.

Саме таку стратегію вибору дій було використано в цій роботі.

3.3 Труднощі у випадку неперервного простору станів

Якщо множина станів системи $\mathcal{S}$ дуже велика, як у випадку багатьох класичних ігор, або ж неперерва, то постає дуже важливе питання представлення функцій корисності. Існує багато варіантів вирішення цієї проблеми. І хоч ефективність кожного з методів сильно залежить від типу розв'язуваної задачі, одним з найбільш універсальних методів є лінійні та нелінійні апроксиматори фукнцій. При цьому лінійні апроксиматори, хоч і мають менші можливості, проте в більшості задач чудово справляються зі своїми завданнями і при цьому для випадку сумісного використання певних алгоритмів навчання з підсиленнями з лінійними апроксиматорами функцій були дані теоретичні гарантії збіжності до оптимальних стратегій при виконанні певних умов ([3]).

У випадку з нелінійними апроксиматорами, такими як штучні нейронні мережі, в силу їх більшої складності немає ніяких гарантій успішної збіжності до оптимальних стратегій. Більше того, для певних алгоритмів та задач було доведено, що використання нейромереж призводить до осциляції нейромережі (більш детальну інформацію можна знайти в [2]). Тим не менше, нейромережі все ж таки успішно використовуються сумісно з різноманітними алгоритмами навчання з підсиленнями. Більше то-

го, здатність нейромереж до узагальнення, дає можливість швидше навчати агента, оскільки визначивши вартість певного стану, вона певним чином наближає і вартості близьких станів. Таким чином потрібно менше ітерацій навчання для того, щоб функція корисності збіглася до оптимальної.

Як приклад одного з найуспішніших використань нейромереж у самонавчанні, можна навести програму TD-Gammon гри в нарди (backgammon), розроблену Gerald Tesauro ([5]), яка в своїх останніх версіях грала на рівні з найкращими гравцями світу. Також успішні застосування нейромереж та методів навчання з підсиленнями можна знайти, наприклад, у [1] та [3].

В даній роботі ми також використовували нейромережу як апроксиматор функцій і, як буде видно з результатів, досить успішно.

4 Постановка задачі

4.1 Проблема навігації робота

У цій роботі ми розглянемо проблему навігації агента (робота) в умовах середовища, заповненого перешкодами. Ми спробуємо створити адаптивну систему контролю агента (робота), завданням якого є самостійних рух у віртуальному середовищі, оминання перешкод, що зустрічаються на його шляху та якомога швидше добирання до заданого пункту призначення. Щоб якомога краще наблизити поведінку автопілота до людської поведінки, система керування роботом сприймає лише об'єкти, які потрапляють у поле зору, обмежене певним кутом та дальністю огляду. Базуючись на цій обмеженій інформації, навчена система контролю повинна бути здатна ефективно керувати роботом і приводити його без зіткнень з перешкодами до заданої мети.

За останні роки цю задачу пробували розв'язувати багатьма різними способами. Більшість цієї роботи було сконцентровано на плануванні шляху, коли повністю відоме середовище аналізується для того, щоб знайти оптимальний шлях до мети. Це, наприклад, метод обчислення потенціальних полів, які "відштовхують" робота від перешкоди і "притягують" до цілі або метод дискретизації простору для того, щоб здійсювати швидкий пошук вільних від перешкод траекторій. Обмеженням цих методів в тому, що вони потребують попередніх повних знань про розташування перешкод в середовищі, і, відповідно, кожен раз, як змінюється їх конфігурації, потребують повного переобчисленння. Також, знайдені оптимальні траекторії абсолютно не враховують динаміки самого робота, тому необхідним стає розроблення ще одного контролера, який би відповідав за проведення робота запланованою траекторією.

На противагу цим методам, реактивні контролери використовують лише поточний стан системи, який вони здатні сприймати, для того, щоб зробити рішення про наступну дію. В результаті здійснення кожної дії відбувається зміна стану, і, відповідно, здійснюється рішення про дію в новій ситуації. Для прикладу, системи на основі бази знань опираються на заздалегідь визначені правила поведінки для того, щоб зробити рішення про наступну дію. В такому підході розробник повинен в своїх правилах враховувати динаміку робота і, відповідно, підлаштовувати їх для кожного конкретного застосування.

Поставлена задача — одна з тих, в яких можна з успіхом застосувати методику навчання з підсиленням. Перевагою цього підходу є те, що система самотійно вчиться досягати мети, використовуючи будь-яку інформацію про стан системи та наявні дії. В умовах обмеженої та неповної інформації розробляється поведінка, яка максимально враховує та використовує усю надану інформацію. Більш детальний огляд методів розв'язування даної задачі можна знайти у [1].

4.2 Дедуктивні та індуктивні методи навчання

Серед широкого спектру методів та підходів, розроблених в рамках машинного навчання, можна виділити дві групи методів, які кардинально відрізняються у своєму підході до проблеми набуття агентом знань. Це — індуктивні та дедуктивні методи навчання.

З цих двох груп методів найбільш дослідженими та теоретично обґрунтованими є дедуктивні методи навчання, основна ідея яких полягає в тому, що для того, щоб навчити агента діяти "інтелектуально" в умовах зовнішнього середовища, потрібно заздалегідь надати йому певну експертну інформацію, тобто інформацію, зібрану, опрацьовану та структуровану людьми, які є спеціалістами в даній галузі. Такі методи навчання ще називають навчанням із вчителем (supervised learning), оскільки необхідною умовою успіху є наявність певного "вищого інтелекту", який володіє знаннями і передає їх агенту. Для дедуктивних методів надзвичайно важливим є питання повноти експертних знань, їх коректності та точності. При цьому основним питанням, поряд з проблемою ефективних методів навчання на основі зібраної інформації, виступає проблема підготовки та організації готових знань, її структуризація таким чином, щоб вона могла бути якомога ефективніше використана агентом під час попереднього навчання.

На відміну від дедуктивних методів, індуктивні методи намагаються уникнути необхідності у попередніх знаннях. При цьому, якщо попередні знання присутні, вони можуть бути певним чином "інтегровані" в агента, тим самим додаткову інформацію також можливо ефективно використати. Але при цьому, навіть при відсутності готових знань, правил, індуктивні методи дають можливість агенту самому виробити ефективну та "розумну" поведінку шляхом багаторазової взаємодії з зовнішнім середовищем. В результаті численних спроб та помилок, агент здатний на основі набутого "досвіду" виробити власну стратегію оптимальної поведінки. Тобто агент, завдяки індуктивному навчанню, вибудовує внутрішню модель поведінки, базуючись лише на отриманому в результаті взаємодії з середовищем досвіді.

4.3 Порівняння з попереднім підходом

Ми вже намагалися розв'язати схожу задачу дедуктивним методом навчання, а саме використовуючи нейромережу та навчальну вибірку як експертну систему для розроблення системи контролю автомобіля. Щоправда, поставлена задача дещо відрізнялася в тому, що від автомобіля вимагалося лише оминання перешкоди, без чітко зазначеного кінцевого пункту призначення. Задача ускладнювалася більш реалістичною фізикою руху автомобіля, аніж у задачі контролю роботом, що розглядається у цій роботі. Оскільки керування роботом не вимагає великих швидкостей, ми припускаємо, що здатні повертати робот на заданий кут, залишаючись при цьому на місці, а також здатні миттєво зупинятися або ж, навпаки, рушати. У випадку з автомобілем ці припущення були б дещо нереалістичними, тому у фізиці автомобіля присутні інерція та швидкість руху, а також реалістичне здійснення повороту.

Отримані результати в задачі контролю автомобілем важко охарактеризувати однозначно. З однієї сторони, система контролю на базі нейромережі досить адекватно керувала автомобілем з доволі складною фізикою, успішно оминаючи перешкоди та забезпечуючи безперервний рух автомобіля в середовищі, про яке вона не мала жодної попередньої інформації. Базуючись лише на даних про віддаль до найближчих об'єктів в полі зору, система контролю забезпечувала завчасне оминання перешкоди з плавною зміною швидкості та напрямку руху автомобіля. Очевидно, що велику роль в досягненні вказаної природності та адекватності керування відіграла вдало підібрана навчальна множина.

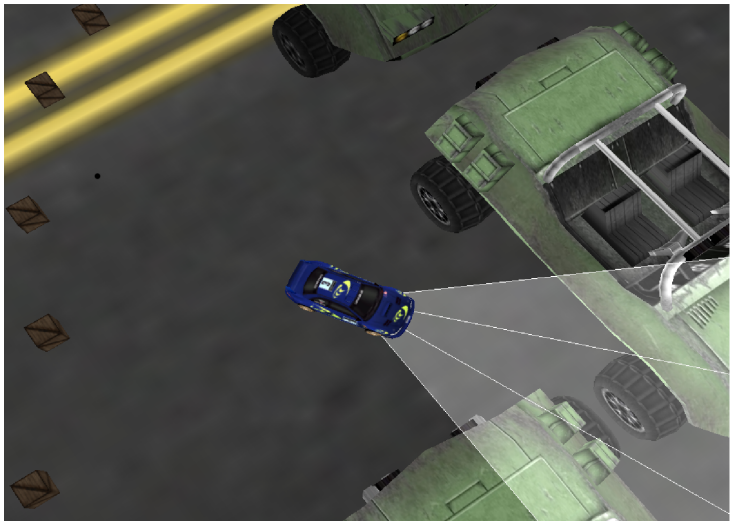


Рис. 3: Автомобіль застряг, заїхавши у кут

З іншого боку, найбільшою проблемою, з якою довелося зіткнутися, була наявність ситуацій, в яких автомобіль зупинявся і система контролю не могла вивести його з нерухомого стану. Це, зокрема, ситуація, подана на рис. 3, в якій автомобіль заїхав у глухий кут і застряг, не зумівши здійснити розворот назад. Цю проблему, теоретично, можна було б розв'язати, ввівши у визначення стану автомобіля значення поточної швидкості та задавши додаткові набори ключових експертних правил, які б дозволили автомобілю здійснити розворот у разі застрягання в глухому куті. Проте на практиці це достатньо проблематично, оскільки введення додаткової змінної стану призводить до ускладення поведінки всієї системи, підвищення рівня вимог до точності та репрезентативності експертних правил. Також для того, щоб зробити розворот автомобіля безпечнішим та природнішим, довелося б вводити також змінні, які б відповідали за задній огляд автомобіля, що призводить до ще більш строгих вимог до експертних правил.

Слід зазначити, що ще однією причиною (окрім вказаної обмеженості визначеного стану динамічної системи) такої поведінки є особливість фізичної моделі автомобіля. Мається на увазі природа автомобіля — для того щоб здійснити поворот, чи, тим більше, розворот, автомобіль повинен здійснювати поступальний рух, що в умовах, коли вже відбулося "застрягання", дуже проблематично. Можливим розв'язанням даної проблеми є зміна типу транспортного засобу. Якщо взяти транспортний засіб, здатний здійснювати поворот без поступального руху (як у випадку з роботом), то можна значно зменшити ймовірність його застрягання. Таким транспортним засобом, для прикладу, може бути танк, в якому шляхом незалежного обертання гусениць в різні сторони можна домогтися розвороту на будь-який кут, залишаючись при цьому на місці.

Важливою проблемою є вибір внутрішньої структури нейромережі. Якщо використати недостатню кількість нейронів та внутрішніх прошарків, то нейромережа не зможе в достатній мірі вивчити набір правил і, таким чином, не зможе повністю використати експертні знання. З іншого боку, використовуючи надто велику кількість нейронів, існує загроза надто точного запам'ятовування (overfitting) правил без належного узагальнення їх на схожі ситуації. В такому випадку нейромережа буде добре

поводитися у ситуаціях, заданих в експертних правилах, але навіть незначна зміна ситуації приведе до абсолютно неадекватної поведінки, яка значно відрізняється від заданої для близької еталонної ситуації.

Таким чином, при використанні наперед заданих експертних правил, з'являється велика кількість практичних питань, на які немає чітких теоретичних відповідей, а все доводиться вирішувати в результаті численних експериментів. Саме тому було вирішено відійти від підходу, який базується на заздалегідь відібраних експертних знаннях, а піти шляхом самоорганізації — дати можливість агенту розробити власну систему правил на основі отриманого внаслідок взаємодії з середовищем досвіду.

4.4 Індуктивний підхід — навчання з підсиленням

Оскільки використання попереднього підходу сильно залежить від якості навчальної вибірки (експертних правил), що при найменшому ускладненні сприйняття агентом середовища призводить до значних ускладнень експертних правил, було вирішено відійти від моделі навчання з учителем. Натомість був використаний самоорганізаційний підхід. Основна ідея такого підходу полягає в тому, щоб дати можливість агенту розробити власну систему правил щодо оптимальної поведінки в умовах середовища, завдяки безпосередній взаємодії з середовищем. Взаємодіючи з середовищем, агент отримує певний досвід і, якщо задати певний механізм оцінки агентом власних дій, то в результаті достатньої кількості "досвіду", можна надіятися, що агент розробить ефективну стратегію поведінки. Такий підхід отримав назву *навчання з підсиленням (reinforcement learning)*. При використанні навчання з підсиленням відпадає необхідність в складному і трудомісткому процесі розробки системи якісних і репрезентативних експертних правил, хоча, натомість, з'являєть необхідність в виборі механізму оцінки дій агента. Проте, для достатньо складних систем зазвичай значно легше визначити механізм безпосередньої оцінки дій, аніж розробити достатньо повну та якісну систему правил.

Розглянемо застосування такого індуктивного процесу навчання (на відміну від дедуктивного на базі системи правил) до вищезазначеної задачі навігації агента.

4.5 Середовище

Для випробування вищезазначеної ідеї було створено віртуальне середовище (див. рис. 4). Робот має п'ять секторів огляду, в межах яких він сприймає точну відстань до найближчої перешкоди. Центральний кут кожного сектору складає 18°. Таким чином робот сприймає об'єкти в межах поля зору в 90°. Роботу надається інформація про кут між напрямком його руху та напрямком на ціль, а також відстань до цілі (рис. 5). Таким чином, робот сприймає середовище як багатовимірний неперервний простір.

Середовище становить собою квадратну кімнату, в якій знаходять випадково згенеровані та розташовані перешкоди, що являють собою опуклі многокутники. Уся дія відбувається епізодами. Робот починає на початку кожного епізоду свій рух у випадковій позиції з випадковим початковим напрямом руху і повинен дістатися вказаної точки, яка також генерується випадково, не натикаючись на межі кімнати та на перешкоди.

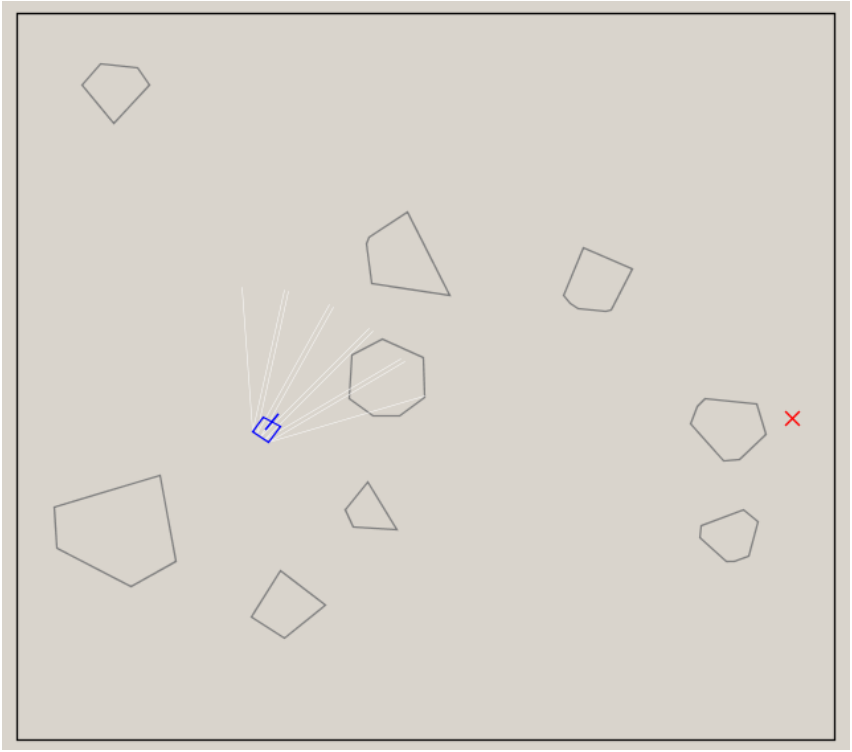


Рис. 4: Віртуальне середовище

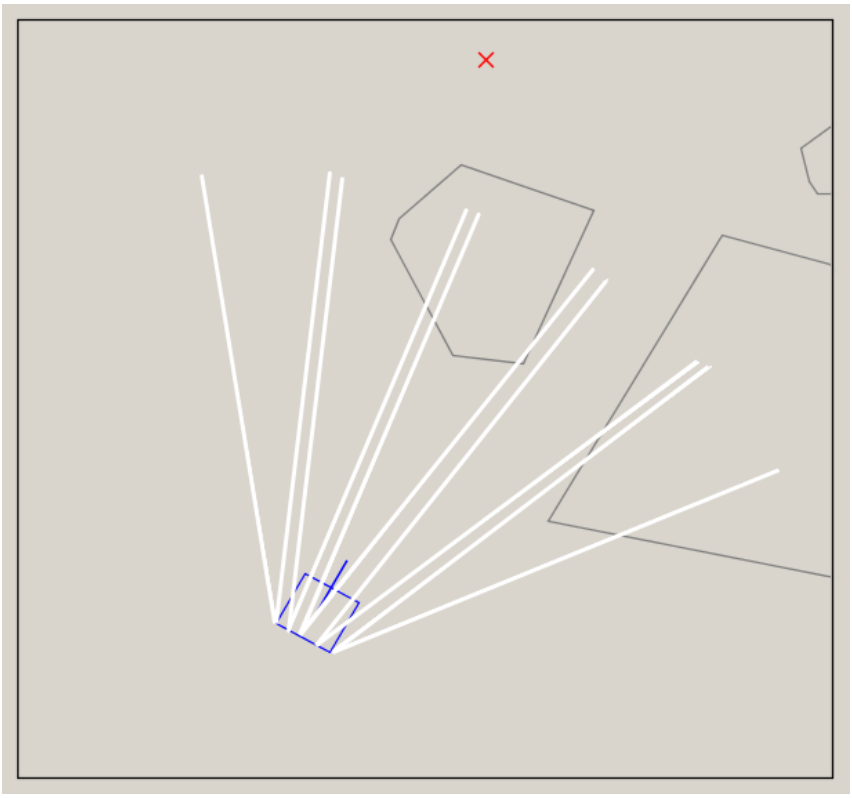


Рис. 5: Сприймання роботом середовища

Робот рухається, вибираючи на кожному кроці дію з дискретної множини допустимих дій, а саме:

- рухатись прямо на невелику відстань;
- рухатись прямо на невелику відстань, одночасно повернувши наліво на невеликий кут повороту;
- рухатись прямо на невелику відстань, одночасно повернувши направо на невеликий кут повороту;
- залишитися на місці, повернувши наліво на невеликий кут повороту;
- залишитися на місці, повернувши направо на невеликий кут повороту.

Робот робить рішення про наступний хід доти, доки не буде виконана одна з наступних умов:

- досягнуто цілі;
- зіткнення з перешкодою або межею кімнати;
- перевищено ліміт кількості кроків;

В усіх цих випадках епізод вважається закінченим. Після цього знову генерується випадковий епізод і все повторюється з початку.

4.6 Деталі реалізації

В нашій програмній реалізації було використано алгоритм Sarsa для навчання. Параметр дисконтування γ було прийнято рівним 0.99 для того, щоб враховувати якомога краще можливі наслідки прийнятих дій.

За кожен крок агент отримував винагороду, рівну -0.01, таким чином ми намагалися домогтися такої послідовності дій, яка б вела до цілі за найменшу кількість кроків. При досягненні мети, отримувалася нагорода, рівна 1.00. При зіткненні з перешкодою, покарання було найбільшим і становило:

$$-1 + \frac{1}{2}e^{-2\frac{d_{crash}}{d_{size}}},$$

де d_{crash} — це відстань від місця зіткнення до перешкоди, d_{size} — довжина сторони кімнати.

Таким чином, зіткнення було найнебажанішою подією, але при цьому для того, щоб заохотити робота рухатися в напрямку до цілі, покарання залежало від відстані до цілі. У випадку з перевищенням роботом допустимої кількості дій, покарання було таким же, як і у випадку з зіткненням, проте більшим на 0.3. Тобто безпечне пересування було пріоритетнішим.

Система винагород була вибрана досить довільно в результаті численних експериментів і з усіх випробуваних варіантів показала себе найефективнішою.

В якості апроксиматора функції була використана нейромережа прямого поширення сигналу, яка на кожній ітерації навчалася за допомогою одного проходу алгоритму зворотнього поширення помилки. Конфігурація нейромережі, використана в нашій програмі: $13 - 8 - 1$, при цьому для внутрішніх прошарків використовувалася логістична активаційна функція:

$$f(x) = \frac{1}{1 + e^{-x}},$$

а для вихідного прошарку одинична функція:

$$f(x) = x.$$

Для апроксимації функції $Q(s, a)$ використовувалися 5 окремих нейромереж, кожна з яких відповідала за свою дію. Цей підхід, згідно [1], є дещо ефективнішим, ніж використання однієї нейромережі з кількістю виходів, рівною кількості можливих дій. В такому випадку для ефективної роботи нейромережі достатньо меншої кількості ітерацій навчання, а також меншої кількості нейронів.

Крок навчання α спадав лінійно від 0.3 до 0.01, досягаючи мінімуму після одного мільйона навчальних кроків.

На вхід нейромережі подавалися спеціальним чином закодовані 7 відомих параметрів стану: 5 сенсорів відстані для кожного з секторів зору, а також відстань до цілі та кут. Кодування використовувалося те ж саме, що й у [1]. Кожен параметр стану кодувався дійсними числами з діапазону $[0, 1]$, які обраховувалися за формулою:

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

$$w_n = \frac{4N}{r}, \qquad b_n = \left(\frac{2n-1}{2}\right)\frac{r}{N},$$

де N — кількість входів нейромережі, які відповідають параметру, r — це діапазон $[0, r]$, в межах якого вхідні значення будуть найбільш чутливими.

Для сенсорів відстані до перешкоди було вибрано $N = 3$ та $r = 8$. Це дозволило зімітувати обмежену дальність видимості робота, що є значно реалістичніше в реальних застосуваннях, аніж абсолютно точні дані про відстань. Відстань до цілі кодувалася при $N = 5$ та $r = 25$, а відносний кут розбивався на дві частини: $\pi - \varphi$ та $\varphi - \pi$, вводячи поняття "ціль зліва" та "ціль справа", які кодувалися при $N = 3$ та $r = \pi$.

Ідея такого кодування полягає в тому, щоб при збільшенні важливості значення x, збільшувалася кількість "ввімкнутих" входів, тобто таких, значення яких близьке до одиниці. Таким чином, чим ближче робот знаходився до перешкоди, тим більше входів активізовувалося. Експеримент показав, що вказане кодування суттєво полегшує задачу навчання і робить його значно ефективнішим та якіснішим.

Усі вагові коефіцієнти нейромережі були початково ініціалізовані нулями, що завдяки одиничній функції у вихідному прошарку, початково давало значення нуль для будь-якого входу. Це є досить важливим моментом. В літературі ([2]) рекомендують при можливості використовувати таке початкове наближення функції корисності $Q(s, a)$, яке б було оптимістичним, по відношенню до реальної функції вартості. В

такому випадку агент, вибравши певну дію і отримавши винагороду, яка є нижчою, ніж початкове оптимістичне наближення, наступного разу в тій же ситуації буде схилятися до вибору іншої дії, таким чином пробуючи кожного разу іншу дію. Все це забезпечить швидше та більш точне наближення реальної ціннісної функції.

В нашому випадку, робот спочатку діє дуже неефективно. Оскільки ймовірність добратися до цілі, минаючи перешкоди, в умовах випадково згенерованого середовища дуже невелика, то початково постійно буде отримуватися негативне значення винагороди. Тому нульова ініціалізація нейромережі становить собою оптимістичні сподівання, і, відповідно, є корисною евристикою.

Щодо конкретних параметрів середовища та робота, то вони наступні. Робот становить собою квадрат розміром 1×1, середовище — також квадрат розмірністю 40×40. Сприймання перешкод відбувалося в межах п'яти секторів, кожен з кутом $18°$, тобто сумарно поле зору становило $90°$. Вважалося, що робот досягнув цілі, якщо він знаходився на відстані меншій 1 від цілі. Зіткнення зараховувалося при відстані меншій 0.01. На кожному кроці робот міг рухатися вперед на відстань 0.9.

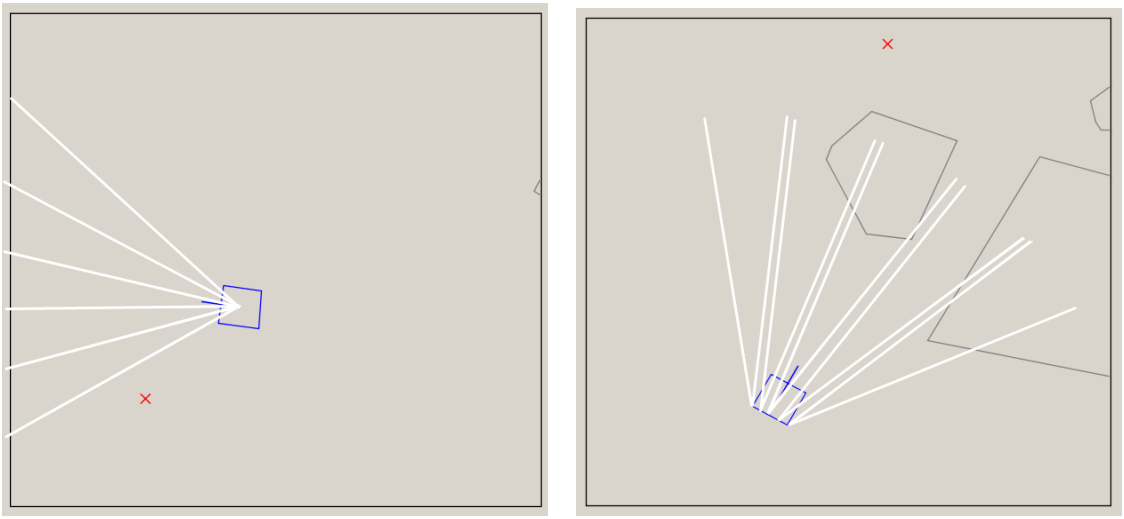
Розташування перешкод в кімнаті генерувалося автоматично, при цьому випадковим чином вибиралася кількість перешкод з діапазону $[4, 10]$, для кожної перешкоди випадково генерувалася позиція її центру та радіус з діапазону $[1.5, 4.0]$.

4.7 Результати

Практичні експерименти показали, що агенту достатньо 1.2 мільйонів кроків навчання для того, щоб вивчити власну стратегію, після чого вона практично не змінюється. Ефективність контролера оцінювалася у відсотках успішних добирань до цілі за останні 5000 епізодів, щоб знівелювати вплив фактично випадкових неадекватних дій на початку навчання.

Початково значення ε у ε-жадібній стратегії було фіксованим на позначці $0,1$, що становило собою розумний компроміс між випадковістю поведінки та можливістю дослідження "нерозвіданих" стратегій. При цьому вершини секторів, які відповідали за зір робота, починалися у центральній точці робота. Таке поєднання налаштувань дозволяло агенту досягати успішності в $60\%-65\%$ випадків. При цьому робот поводив себе цілком адекватно, стараючись оминути перешкоди, проте часто зачіпаючи боковою стороною краї перешкоди. Це зумовлено обмеженістю поля зору (див.рис. 6a), оскільки сектори захоплювали лише ту частину простору, яка знаходилася безпосередньо перед роботом, ніяк не даючи знати про перешкоди, що знаходяться в безпосередній близькості збоку.

Для того, щоб дати більше інформації про оточуючі перешкоди, поле зору робота було дещо видозмінене (див. рис. 6b). Вершини секторів були зміщені в задню частину і рівномірно розподілені по всій задній стороні. Таким чином, у робота тепер були два сектори, які надавали йому інформацію про дуже близькі перешкоди, що знаходилися ліворуч та праворуч від нього. Ці вдосконалення дозволили підвищити успішність агента до позначки $80\% - 82\%$ в більшості тестових запусків. Аналіз нової стратегії показав, що поведінка робота дійсно стала значно безпечнішою та ефективнішою. Але при цьому часто можна було спостерігати як робот оминає перешкоду, об'їжджаючи її "впритул", і час від часу здійснює випадкові повороти в сторону перешкоди. Це було зумовлено тим, що значення ймовірності вибору випадкової дії ε



(a) Початковий варіант (b) Більш ефективний варіант

Рис. 6: Різні форми області сприймання перешкод

було вибрано досить великим — 0.1. Тобто в середньому в одному випадку з десяти агент здійснював випадковий вибір дії, що призводило до зіткнень при близькості до перешкоди.

Зменшення ε до 0.01 призвело до значного погіршення навчального процесу, оскільки алгоритму Sarsa не надавалося достатньої можливості досліджувати нові стратегії, натомість поведінка агента збігалася до локального мінімуму, який був далекий від оптимального.

В результаті цих спостережень ми вирішили використати змінний показник ε, який лінійно спадав від 0.2 до 0.01 протягом 1 мільйону кроків навчання, після чого фіксувався на мінімальній позначці. Таке вдосконалення повинно було б допомогти агенту на початку дослідження більш повно досліджувати можливі нерозвідані дії, при цьому зменшуючи ймовірність вибору неоптимальних дій тоді, коли в результаті багатьох ітерацій, вивчена функція цінності давала хороше уявлення про оптимальні дії і необхідність в дослідженні випадкових дій зникала.

Експеримент показав, що таке вдосконалення дійсно дає краще результати. З усіма вищезазначеними вдосконаленнями в 8 випадках із 10 успішність агента становила $87\% - 92\%$ успішних епізодів.

На рис. 7 показано траєкторії робота без будь-якого навчання.

На рис. 8 показано приклади траєкторій робота на проміжних стадіях навчання.

На рис. 9 наведено приклади траєкторій в умовах однієї і тієї ж кімнати після різної кількості навчальних кроків.

На рис. 10 показано приклади траєкторій робота після навчання.

На всіх цих ілюстраціях символом + позначено рух робота, + в квадраті — фінальна позиція робота, + в крузі — початкова позиція робота, x — ціль.

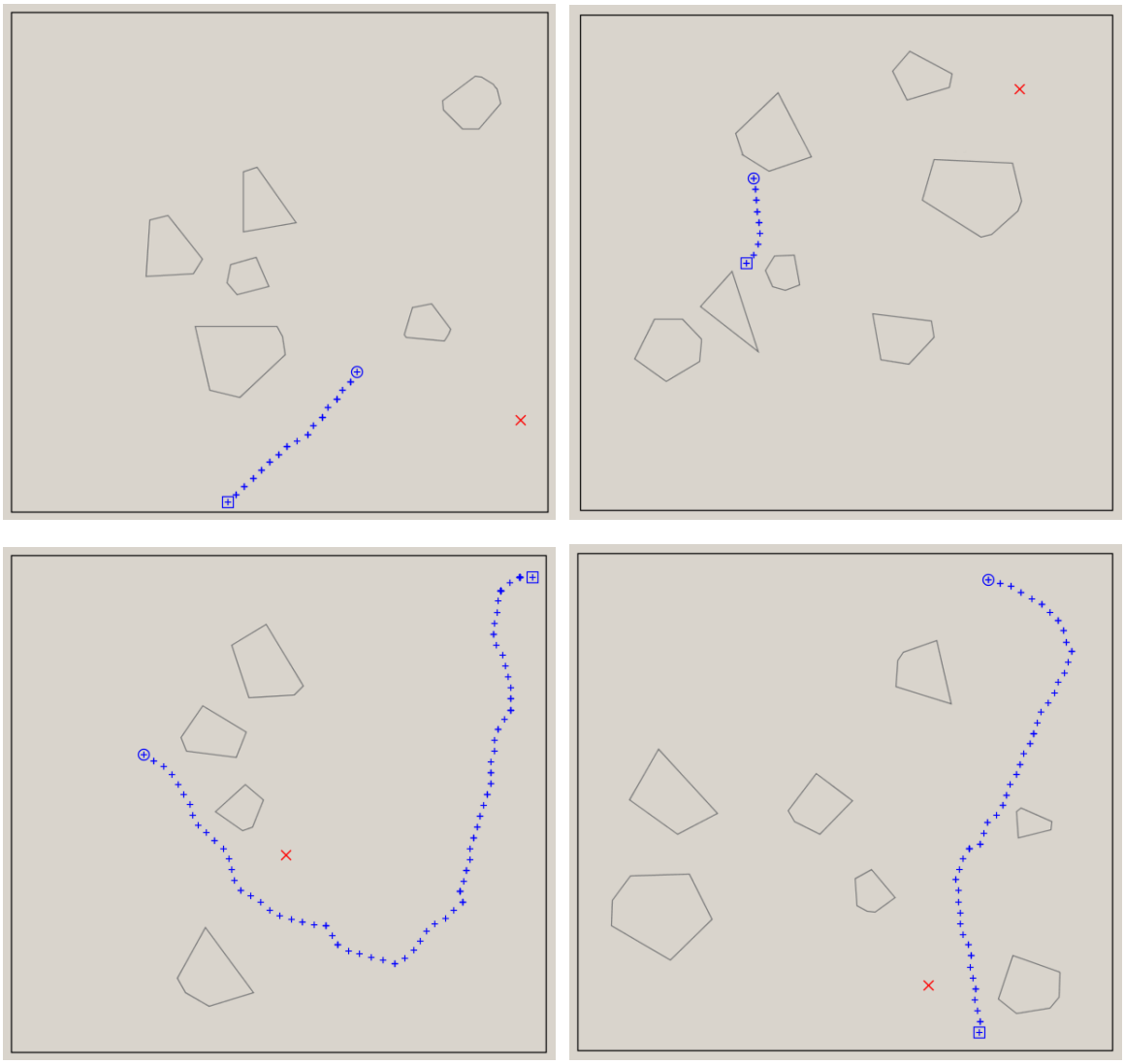


Рис. 7: Приклади початкових траекторій без навчання

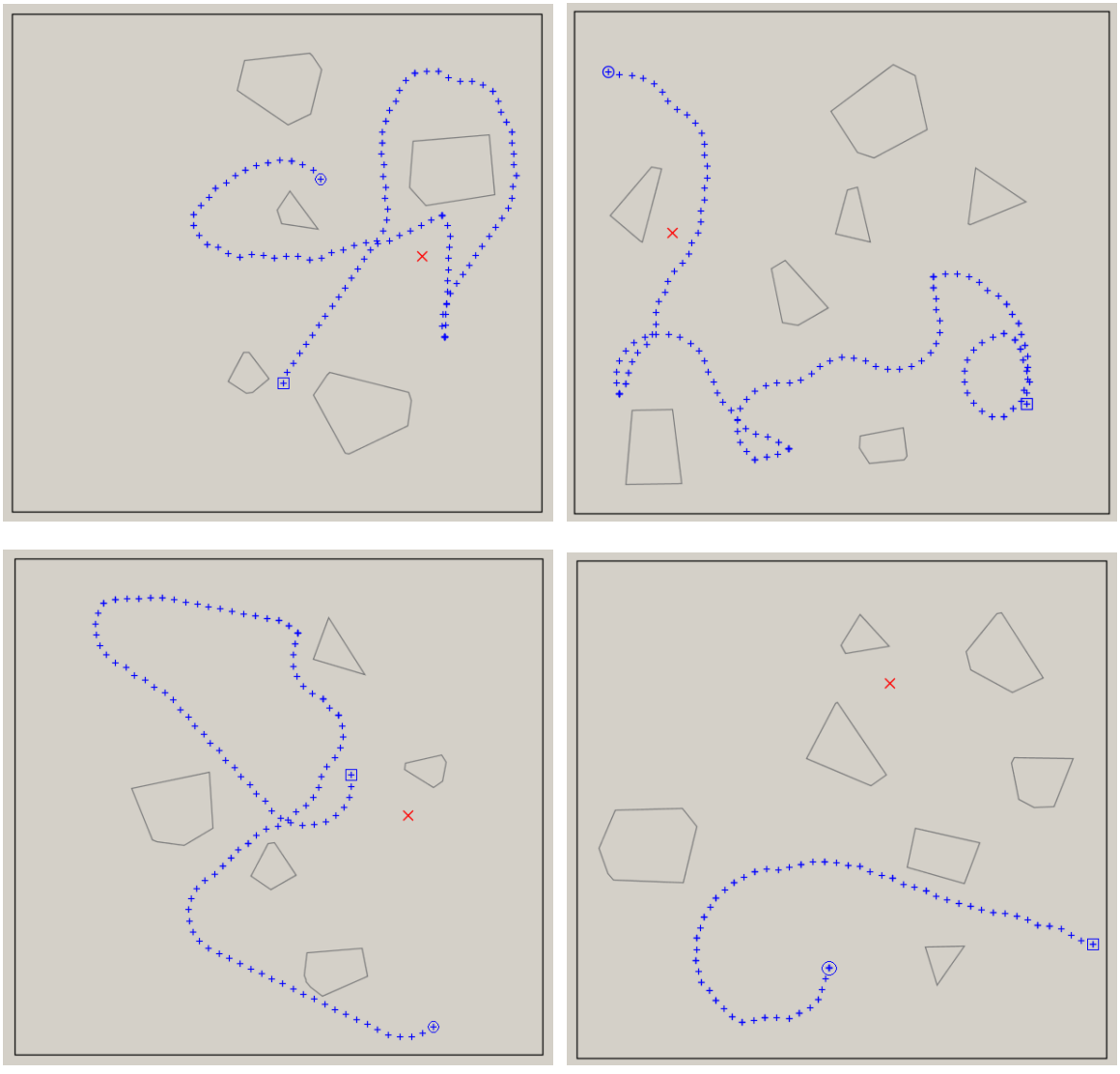
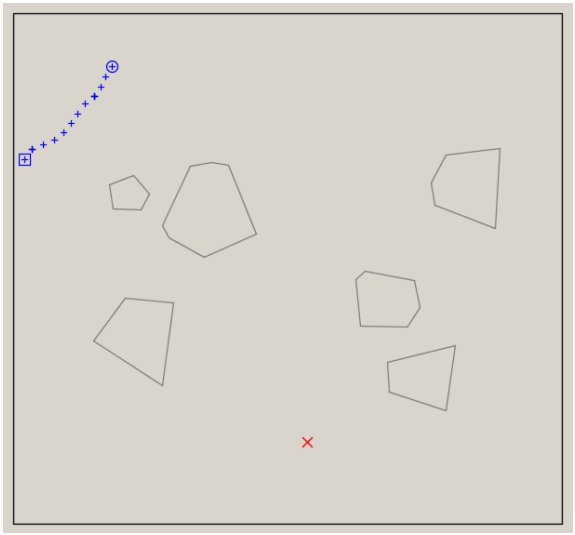
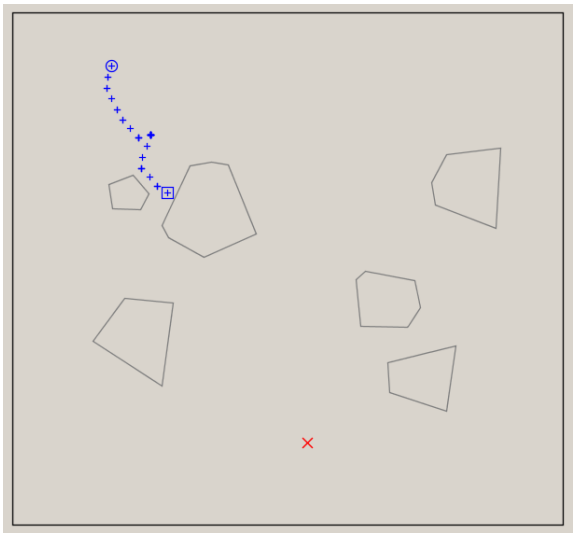


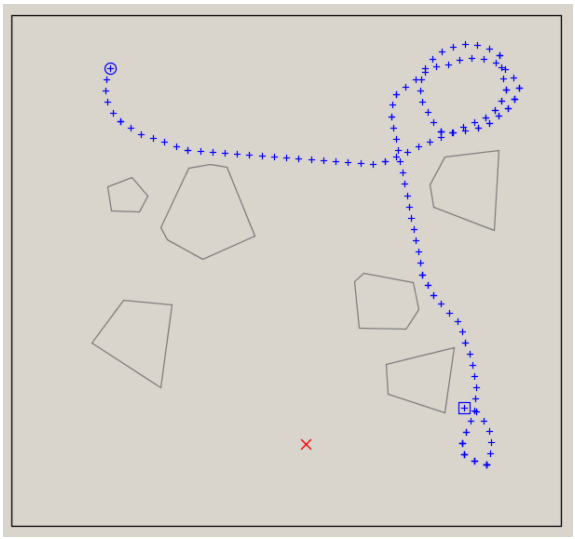
Рис. 8: Приклади траекторій на проміжних стадіях навчання



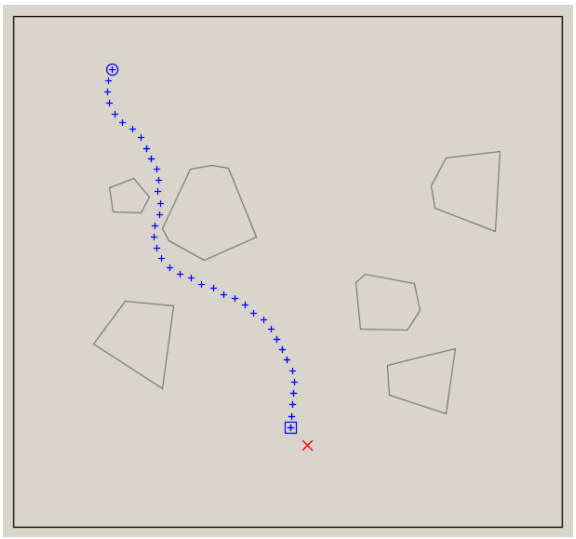
(a) Без навчання



(b) Після 100 тис. ітерацій



(c) Після 500 тис. ітерацій



(d) Після 1.2 млн. ітерацій

Рис. 9: Приклади траекторій в умовах одного і того ж середовища

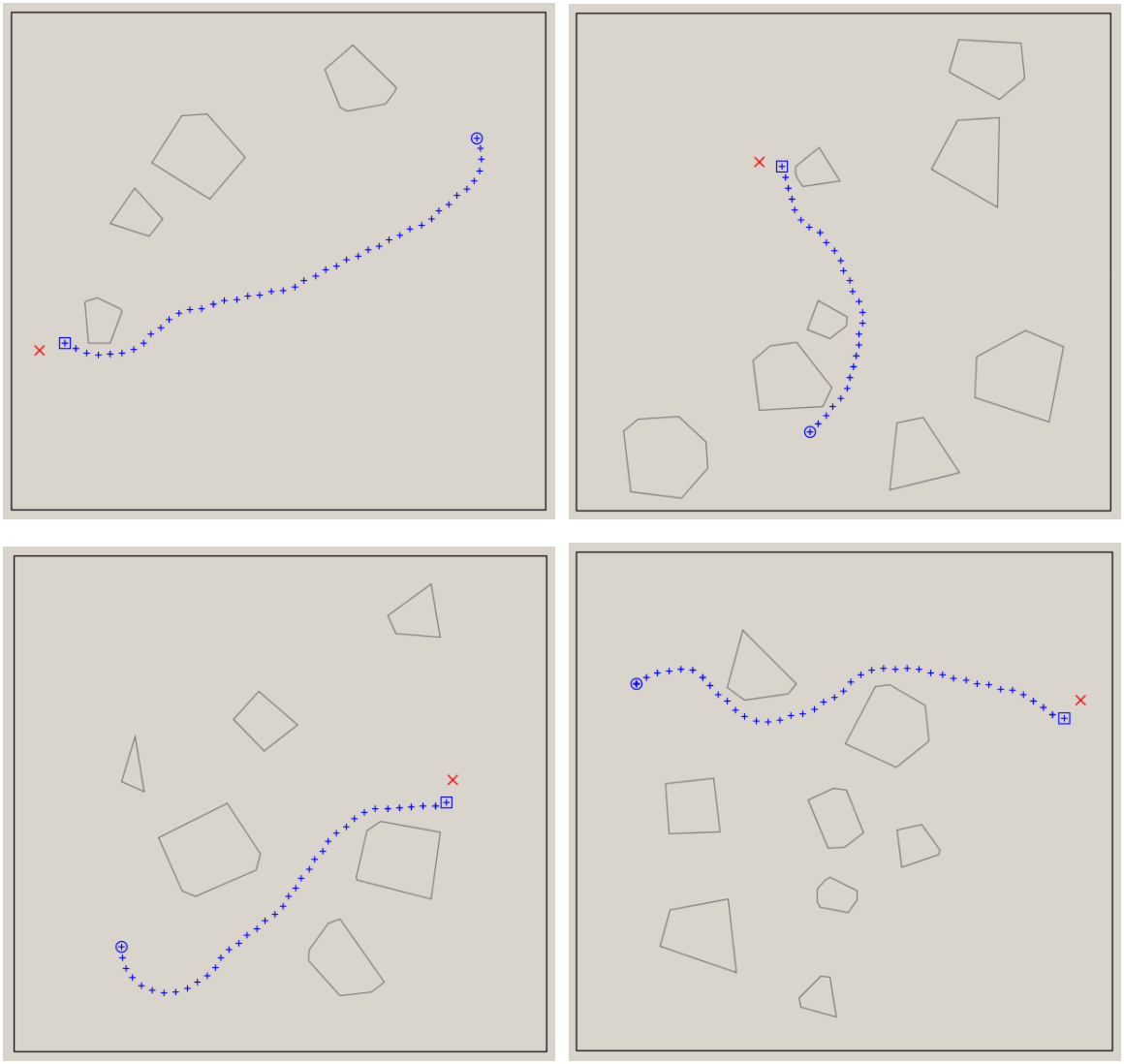


Рис. 10: Приклади траекторій навченого робота (після 1.2 мільйона навчальних кроків)

5 Висновки

Використовуючи парадигму навчання з підсиленням, було створено адаптивний контролер навігації робота, який базуючись на обмеженій інформації про навколишній світ, забезпечує добирання робота до цілі, оминаючи зіткнення з перешкодами. Як показали практичні експерименти, розроблений контролер є ефективним і забезпечує успішне добирання до цілі в абсолютній більшості випадків (в середньому $87\% - 92\%$).

Якщо порівнювати використаний підхід, з попереднім підходом на базі системи експертних правил, використаним нами раніше, можна відзначити, що навчання з підсиленням забезпечує значно більшу гнучкість, даючи можливість змінювати структуру стану системи, без жодних інших ускладнень. У випадку з системою експертних правил, така зміна одразу приводить до труднощів, оскільки потрібно розробляти абсолютно нову систему правил, що є надзвичайно трудомістким та складним процесом. На жаль, досить важко в кількісному відношенні порівняти два зазначених підходи, оскільки, по-перше, цілі в цих двох задачах були дещо різні: просто оминання перешкод в випадку з системою правил та добирання до цілі з оминанням перешкод у випадку з навчанням з підсиленням. По-друге, фізика моделей автомобіля та робота суттєво відрізняється, що унеможливлює об'єктивне порівняння. Проте, суб'єктивно, поведінка самостійно навченого робота виглядає значно раціональнішою, природнішою та більш адекватною, порівняно з поведінкою автомобіля у схожих ситуаціях (наприклад, при під'їзді до близько розташованої перешкоди).

Слід зазначити, що в обидвох підходах є як позитивні, так і негативні сторони. Для підходу на базі експертних правил характерним є швидкий процес навчання, проте сама розробка правил досить громіздка. Для навчання з підсиленням навпаки: немає затрат часу та сил на розробку системи правил, оскільки мобільний агент сам розробляє свої внутрішні правила, проте процес навчання триває досить довго. Для прикладу, здійснення 2 мільйонів навчальних кроків на середньостатистичному комп'ютері триває більше години часу. Неоднозначним для навчання з підсиленням є його залежність від евристичних параметрів: зміна параметру ε у ε-жадібній стратегії вибору дій з константного значення 0.1 на лінійно-спадну послідовність від 0.2 до 0.01 призвело до покращення результатів на $5\% - 10\%$.

Можливо, саме така залежність навчання з підсиленням від евристичних параметрів є поясненням того, що розроблена нами система контролю за своїми показниками уступає від аналогічної системи, розробленої G.A.Rummery([1]), для якої характерними були показники рівня успішності робота в межах $96\% - 99\%$. Очевидно, більш якісним підбором різноманітних евристичних параметрів навчання, можна було б досягти ефективнішої роботи контролера.

Тим не менше, зручність та легкість внесення будь-яких змін у структуру середовища, його стану чи фізики моделі без зміни навчального процесу, а також хороші показники ефективності роботи навченого контролера — безсумнівні переваги навчання з підсиленням. У випадку з проблемою навігації, незначна зміна сприймання роботом навколишнього середовища таким чином, щоб поле зору захоплювало незначну територію зліва та справа від нього, без будь-яких інших вдосконалень дає покращення результатів на $15\% - 17\%$.

Особливої уваги заслуговує використання штучної нейромережі у якості апро-

ксиматора функції. Хороші властивості апроксимації та узагальнення нейромережі — основна складова успішного застосування навчання з підсиленням для складних проблем з неперервним простором станів, як у випадку з навігацією робота. У нашому випадку нейромережа показала себе чудовим інструментом, який доповнює парадигму навчання з підсиленням. Цікавим є те, що, незважаючи на те, що в літературі досі не було дано будь-яких теоретичних доведень збіжності алгоритмів навчання з підсиленням при умові використання нелінійних апроксиматорів функцій, вкотре було показано практичну застосовність поєднання штучних нейромереж та алгоритму навчання з підсилення (зокрема, Sarsa) для складних задач контролю.

Проте, нейромережа вносить додатковий рівень складності у розроблені алгоритми, вимагаючи обережного та точного підбору параметрів навчання самої нейромережі, а також вибору оптимальної структури, кількості нейронів та їх активаційні функції.

Серед майбутніх напрямків роботи можна зазначити зміну завдання агента з добирання до вказаної цілі на завдання оптимального покриття обмеженого простору (кімнати) з оминаннями перешкод. Такий агент може бути корисним для використання у побутовій робототехніці, наприклад, для створення робота-порохотяга. В цьому ж руслі слід відзначити можливість використання різноманітних промислових сенсорів, наприклад, ехолокаторів, датчиків руху тощо. Успішне поєднання цих пристроїв з індуктивним навчанням приблизить нас до застосування розроблених методів у реальних пристроях, а не у віртуальних програмних середовищах.

Ще одним напрямком досліджень слід відзначити продовження дослідження в царині навчання без учителя. Одним з таких напрямків є ієрархічне навчання з підсиленням, яке зосереджується на тому, щоб залежно від стану агента та системи, перемикатися на виконання найбільш актуального та критичного завдання. Наприклад, у випадку з автономним роботом-порохотягом, якщо стан акумулятора буде свідчити про низький рівень заряду, то найбільш актуальним завданням для робота стане пошук найближчої док-станції, а не покриття кімнати.

Загалом, індуктивні методи навчання є дуже перспективним напрямком роботи, оскільки дають можливість створювати достатньо ефективні контролери для задач, з якими дедуктивні методи не справляються в силу надзвичайної складності поставленого завдання.

Література

[1] Rummery, G. A. 1995. Problem Solving with Reinforcement Learning. Ph.D. thesis. Cambridge University Engineering Department.

[2] Sutton, R. S., Barto, A. G. 2002. Reinforcement Learning: An Introduction. A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England.

[3] Coulom, R. 2002. Reinforcement Learning Using Neural Networks, with Applications to Motor Control. Ph.D. thesis. Institut National Polytechnique de Grenoble.

[4] Kaelbling L., Littman M., Moore A. 1996. Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4, 1996, 237-285

[5] Tesauro G. 1995. Temporal Difference Learning and TD-Gammon. Communications of the ACM, March 1995 / Vol. 38, No. 3