دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی برق

پایان‌نامه کارشناسی
گرایش کنترل

مقایسه استفاده از روش‌های بر مبنای یادگیری تقویتی برای حل بازی‌های دیفرانسیلی

نگارش
سپهر کریمی آرپناهی

استاد راهنما
دکتر محمدباقر منهاج

استادمشاور
دکتر حیدرعلی طالبی

مهر ۱۴۰۰

صفحه فرم ارزیابی و تصویب پایان‌نامه- فرم تأیید اعضاء کمیته دفاع

به نام خدا

تاریخ:۱۴۰۰/۷/۱۸

تعهدنامه اصالت اثر

اینجانب سپهر کریمی آرپناهی متعهد می‌شوم که مطالب مندرج در این پایان نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است.
نقل مطالب با ذکر مآخذ بلامانع است.

سپهر کریمی آرپناهی

تقدیر و تشکر

اینجانب مراتب تشکر خود را نسبت به آقای دکتر محمدباقر منهاج، استاد راهنمای بنده که طی انجام این پایان‌نامه، بنده را بی‌دریغ راهنمایی و یاری وکردند ابراز می‌نمایم.

همچنین از خانم دکتر فرزانه عبدالهی که داوری این پایان‌نامه را بر عهده داشتند، کمال تشکر و قدردانی را دارم.

همچنین از مادر و پدرم برای صبر و بردباری فراوانشان نهایت سپس و ارادت را دارم.

چکیده

در این پایان‌نامه به بررسی بازی‌های دیفرانسیلی[۱] و روش‌های اجرا کردن آن‌ها می‌پردازیم. سپس یادگیری تقویتی[۲] و پیاده‌سازی آن را بررسی می‌کنیم.

در ادامه بازی تعقیب و گریز[۳] که یکی از معروف‌ترین بازی‌های دیفرانسیلی است را پیاده‌سازی می‌کنیم. در این بازی دو عامل مستقل در یک محیط بازی وجود دارند و محیط دارای موانع است. هدف عامل تعقیب‌کننده رسیدن به عامل گریزنده و هدف عامل گریزنده فرار از تعقیب‌کننده است. ما در این بازی می‌خواهیم با آموزش عامل‌ها به رویکرد بهینه[۴] دست یابیم که همان پاسخ معادله همیلتون-ژاکوبی-آیزاکس[۵] است.

در انتها به بررسی نتایج و مقایسه دو روش پیاده‌سازی این بازی به وسیله یادگیری تقویتی می‌پردازیم.

واژه‌های کلیدی:

یادگیری تقویتی، یادگیری-کیو[۶]، تئوری بازی‌ها[۷]، بازی‌های دیفرانسیلی

۱ Differential games

۲ Reinforcement learning

۳ Pursuit and evasion

۴ Optimal policy

۵ Hamilton–Jacobi–Isaacs

۶ Q-learning

۷ Game theory

فهرست مطالب

صفحه

فهرست اشکال

فهرست جداول

فهرست علائم

علائم لاتین

S	مجموعه حالت‌ها
A	مجموعه اعمال
R	مجموعه پاداش‌ها در یک حالت
P	احتمال آن که انجام عملی در حالت اولیه به حالت ثانویه برویم

علائم یونانی

γ	فاکتور تخفیف
Δ	تفاوت زمانی
α	نرخ یادگیری
π	رویکرد عامل

۱- فصل اول

مقدمه

۱-۱- مقدمه

در این پایان‌نامه، ما به بررسی و پیاده‌سازی بازی تعقیب و گریز به عنوان یکی از مهم‌ترین بازی‌های دیفرانسیلی با روش‌های یادگیری تقویتی یادگیری-کیو و سارسا می‌پردازیم.

بازی تعقیب و گریز از گذشته، در مقالات بسیاری مورد بررسی قرار گرفته است. در این بازی دو عامل (یا چند عامل) مستقل در یک محیط بازی وجود دارند و همچنین محیط دارای موانع است. هدف عامل تعقیب‌کننده رسیدن به عامل گریزنده و هدف عامل گریزنده فرار از تعقیب‌کننده است.

در پیاده‌سازی بازی در این پایان‌نامه تمامی عامل‌ها دارای فضای دید محدود هستند و حریف آن‌ها فقط در فضای دید مشخص شده قابل تشخیص است. یعنی نیم‌دایره ای به شعاع ۵ واحد از مبدا آن‌ها برایشان قابل دیدن است.

۱-۲- کارهای پیشین

بازی تعقیب و گریز از جمله مسائلی است که در گذشته بسیار مورد تحلیل قرار گرفته است. و تحلیل‌های بسیاری بر روی آن انجام شده است. از آن جایی که این بازی بسیار برای مسائل کنترل بهینه مناسب است، معمولا در مقالات از روش‌های کنترل بهینه و حل معادلات دیفرانسیلی برای آن استفاده می‌کنند. [1] [2]

مقاله [3] استفاده از دو روش حریصانه[۸] و ماکس-جهانی[۹]را در این بازی‌ها بررسی می‌کند. همینطور در این مقاله رویکردهای خاصی به گریزنده داده شد.

در مقاله [4] استفاده از زنجیره مارکوف و فیلترهای کالمن برای ساخت مسیرهای هموار برای عامل دنبال‌کننده بررسی شده است. در این مقاله تمرکز بر استفاده از سنسورهای سطح پایین است و آن‌که گریزنده از شعاع دید دنبال‌کننده خارج نشود.

[۸] greedy

[۹] Global-max

روش این پایان‌نامه نشان می‌دهد چگونه فرایندهای تصمیم‌گیری مارکوف با یادگیری تقویتی(سارسا[۱۰] و یادگیری-کیو) می‌توانند باعث شوند که گریزنده تصمیم‌های سریع و خوبی بگیرد.

۱-۳- کارهای انجام شده در پایان‌نامه

در این پایان‌نامه ابتدا توضیحاتی درباره تئوری بازی‌های دیفرانسیلی و روش‌های اجرای آن‌ها دادیم. سپس یادگیری تقویتی را تحلیل کرده و روش‌های یادگیری-کیو و سارسا را شرح دادیم.

همچنین یک محیط گرافیکی دارای مانع که شامل دو عامل است ساخته و با استفاده از دو روش یادگیری-کیو و سارسا آن‌ها را اموزش دادیم و در آخر نتایج را تحلیل کردیم.

۱-٤- ساختار پایان‌نامه

این پایان‌نامه ۵ بخش دارد که به شرح زیر هستند:

- در فصل دوم به توضیحات تئوری روش‌های یادگیری تقویتی و پیاده‌سازی آن‌ها می‌پردازیم.
- در فصل سوم به شرح بازی‌های دیفرانسیلی و شیوه‌های پیاده‌سازی آن می‌پردازیم.
- در فصل چهارم یک محیط گرافیکی ساخته و بازی را با دو الگوریتم یادگیری تقویتی پیاده‌سازی کرده و نتایج آن را تحلیل خواهیم کرد.
- در فصل پنجم نیز نتیجه گیری کلی می‌کنیم و کارهایی که در آینده در این پایان‌نامه قابل انجام است را عنوان می‌کنیم.
- همچنین در انتها نیز مراجع استفاده شده در این پایان‌نامه را معرفی می‌کنیم.

[۱۰] SARSA

۲- فصل دوم
یادگیری تقویتی

۲-۱- مقدمه

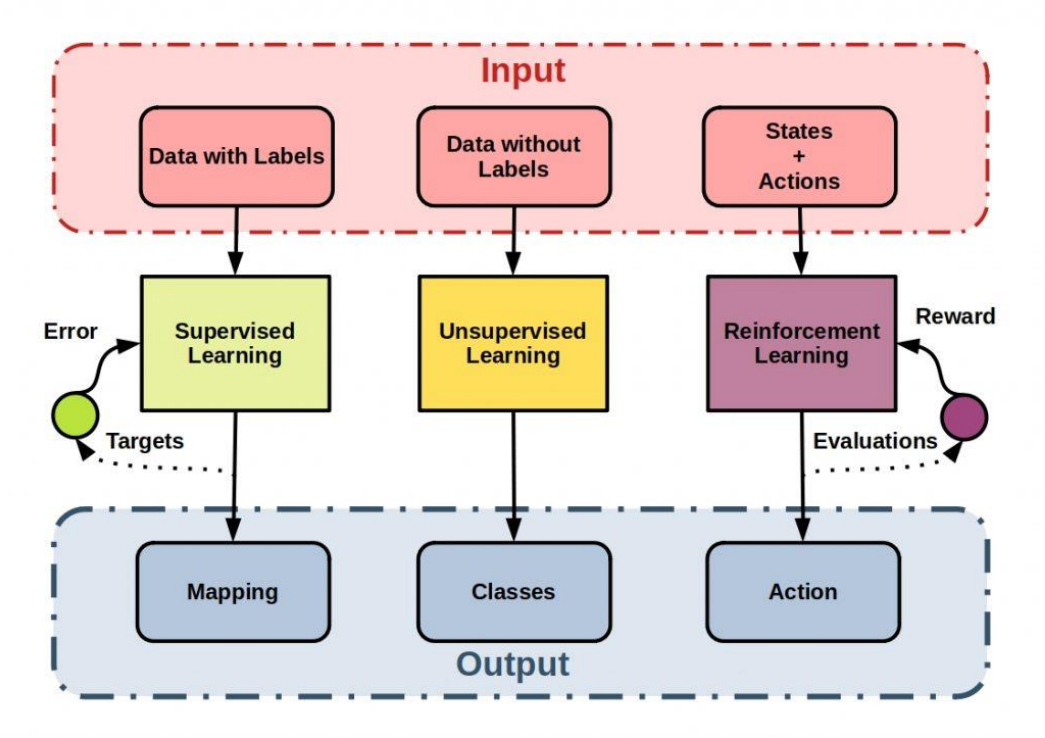
یادگیری ماشین توانایی یک برنامه برای توسعه بر مبنای مشاهده[۱۱] کنونی از محیط و رفتار منطقی[۱۲] است. [5]

در یادگیری ماشین سه نوع یادگیری وجود دارد: یادگیری نظارت‌شده[۱۳], یادگیری نظارت‌نشده[۱۴] و یادگیری تقویتی. یادگیری نظارت‌شده به دانش قبلی از محیط و فضای حالت بستگی دارد. علی‌رقم آنکه یادگیری نظارت شده کاربرد بسیاری دارد، در کاربردهایی که فضای حالت نامعلوم است، یادگیری ناقصی دارد. در این شرایط روش یادگیری نظارت‌نشده کاربرد دارد.

یادگیری نظارت‌نشده به دنبال الگوهای قبلاً کشف نشده در یک مجموعه داده بدون برچسب[۱۵] قبلی و با حداقل نظارت می‌باشد. برخلاف یادگیری نظارت‌شده که از داده‌های برچسب‌دار استفاده می‌کند.

یادگیری تقویت شده برخلاف نظر قدیمی محققان در حوزه یادگیری ماشین، در دسته یادگیری نظارت نشده قرار ندارد. زیرا در یادگیری تقویتی ما به دنبال ماکزیمم کردن پاداش هستیم. حال آنکه در یادگیری تقویت‌نشده به دنبال پیدا کردن الگوهای کشف نشده در مسئله هستیم. [6]

[۱۱] Observation
[۱۲] Rational
[۱۳] Supervised Learning
[۱۴] Unsupervised Learning
[۱۵] unlabeled



شکل ۱: مقایسه ۳ روش یادگیری ماشین *[7]*

یادگیری تقویت‌شده از آزمایش و خطا[۱۶]، فرایندهای تصمیم‌گیری مارکوف[۱۷]، یادگیری تفاوت زمانی[۱۸] تشکیل می‌شود. فرایندهای تصمیم‌پذیر مارکوف محیط را حالت‌بندی می‌کند و در بخش بعدی توضیح داده خواهد شد. یادگیری تفاوت زمانی، به ارزیابی حالت‌های فرآیندهای تصمیم‌پذیر مارکوف می‌پردازد. از ترکیب فرایندهای تصمیم‌پذیر مارکوف با یادگیری تفاوت زمانی می‌توان یک عامل یادگیرنده[۱۹] را ساخت. [8] یادگیری آزمایش و خطا به این معنی است که عامل از اشتباهات خود یاد می‌گیرد. از ترکیب ۳ المان بالا می‌توان تکنیک یادگیری تقویتی را ساخت که عاملی یادگیرنده همراه با کنترل بهینه می‌باشد. کنترل بهینه عامل را قادر می‌سازد تا وظیفه خود را به صورت بهینه انجام دهد.

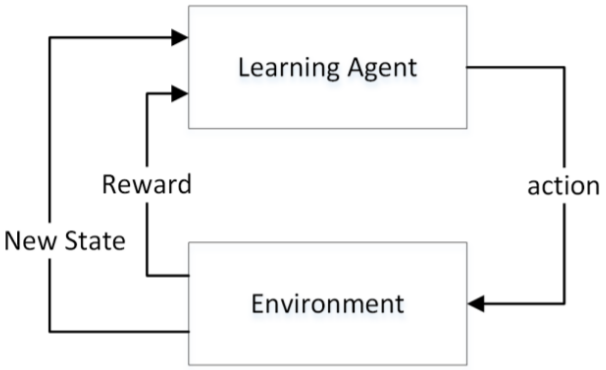
[۱۶] Trial and Error
[۱۷] Markov Decision Process
[۱۸] Temporal Difference
[۱۹] Learning agent

ربات‌هایی که از یادگیری تقویتی در آن‌ها استفاده شده، به وسیله سنسورهای خود محیط را مشاهده می‌کنند. مشاهده ربات به وسیله یادگیری تقویتی به فضای حالت تبدیل می‌شود.

یادگیری تقویتی بر پاداش به عامل تمرکز دارد. پاداش عامل نشان می‌دهد که عامل چه مقدار به هدف نزدیک است.

الگوریتم یادگیری تقویتی باید تعادلی بین اکتشاف[۲۰] و بهره‌برداری[۲۱] به وجود آورد. بهره‌برداری یعنی عامل باید میزان موفقیت عمل‌های قبلی را به یاد داشته باشد و بهترین آنها را تکرار کند. همچنین اکتشاف یعنی عامل در برخی مواقع باید تصمیم‌های تصادفی بگیرد. اکتشاف برای آن است که عامل در اکسترمم[۲۲]های محلی متوقف نشود.

در صورت وجود تنها یک عامل هرچیزی که با عامل تعامل داشته باشد، محیط را تشکیل می‌دهد. حالت کنونی عامل S_t می‌شود. هر عامل برای رسیدن به هدف نیازمند انجام اعمال است و اعمال باعث ایجاد تغییر در محیط شده و حالت را عوض می‌کنند. در شکل ۲ نحوه عملکرد یک عامل یادگیری تقویتی نشان داده شده است. عامل یک (یا چند) عمل را انجام داده و سپس حالت جدید S_{t+1} را مشاهده می کند. همچنین عامل پاداش r_{t+1} را دریافت می‌کند که به معنی میزان خوب یا بد بودن عمل قبلی است.



شکل ۲: یادگیری تقویتی *[12]*

[۲۰] Explore
[۲۱] exploit
[۲۲] extremum

رویکرد[۲۳] Π_t استراتژی عامل را در زمان t نشان می‌دهد. به این معنی که احتمال انجام عمل a_t را از مجموعه اعمال ممکن برای عامل، در حالت S_t نشان می‌دهد. همچنین این احتمال پس از انجام عمل و دریافت پاداش دوباره به‌روز می‌شود.

به عبارت دیگر عامل با رویکرد Π_t عمل $a=a_t$ را با احتمال $\pi_t(s, a)$ در حالت S_t انجام می‌دهد. همچنین رویکرد Π_t^* رویکرد بهینه عامل در آن حالت را نشان می‌دهد. یادگیری تقویتی می‌تواند بسیار به رویکرد بهینه نزدیک شود.

همانطور که گفته‌شد، یادگیری تقویتی در تلاش است تا پاداش عامل را افزایش دهد. نحوه محاسبه پاداش کل در یادگیری تقویتی به دو نوع است:

- اگر محاسبه در زمان محدود باشد، پاداش‌های آینده مورد انتظار به صورت معادله (۱) محاسبه می‌شود:

$$R_t = \sum_{k=1}^{T} r_{t+k} \qquad (1)$$

در معادله بالا R_t با تمام پاداش‌های آینده عامل، برابر است.

- اگر محاسبه در زمان نامحدود باشد باید پاداش- تخفیف-داده-شده آینده[۲۴] را محاسبه کنیم که به صورت معادله (۲) است:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \qquad (2)$$

که در آن γ مثبت و کوچکتر از ۱ است. همچنین عامل در هر قدم در تلاش است تا پاداش یعنی R_t را ماکزیمم کند.

---

[۲۳] Policy

[۲۴] Future discounted rewards

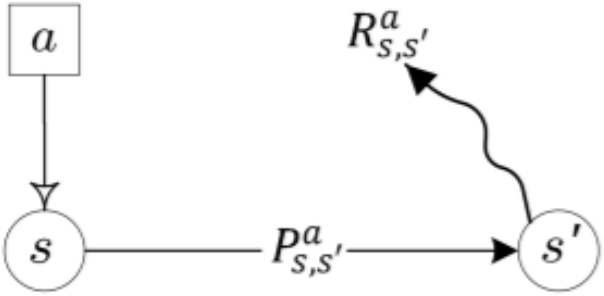
۲-۲- فرایندهای تصمیم‌گیری مارکوف

در فرایندهای تصمیم‌گیری مارکوف حالت کنونی پاداش آینده را تعیین می‌کند. یعنی به گذشته عامل وابستگی ندارد. هر فرایند تصمیم‌گیری مارکوف از ۵ توپل[۲۵] $G = (S, A, P, R, \gamma)$ تشکیل می‌شود.

- S: مجموعه حالت‌های عامل می‌باشد.
- A: مجموعه عمل‌های ممکن برای عامل می‌باشد.
- $R(s, a, s')$: پاداش آن عمل برای انجام a و گذر از حالت s به s' می‌باشد.
- $P(s, a, s')$: احتمال آن که با انجام عمل a در حالت s به حالت s' برویم.
- γ: فاکتور تخفیف می‌باشد، که برای محاسبه پاداش- تخفیف-داده-شده آینده استفاده می‌شود.

عامل در فرآیندهای تصمیم‌گیری مارکوف باید شرایط زیر را دارا باشد:

- عامل در دنباله‌ای از حالت‌ها حرکت کند.
- حالت کنونی عامل همیشه مشخص باشد.
- هر عمل به گذر از یک حالت به حالت دیگر منجر شود و پاداشی (یا تنبیه) دارد.



شکل ۳ : فرآیندهای تصمیم‌گیری مارکوف *[6]*

[۲۵] tuple

همانطور که در شکل ۳ : فرآیندهای تصمیم‌گیری مارکوفشکل ۳ مشاهده می‌شود، عامل با انجام عمل a از حالت s با احتمال P به حالت s' می‌رود و پاداش R را دریافت می‌کند.

۲-۳- یادگیری تفاوت زمانی

از مشکلات در برخی محیط‌ها این است که پاداش‌ها قابل مشاهده آنی نیستند. مثلا در بازی "دوز"، پاداش فقط بعد از اخرین حرکت بازی داده می‌شود و بقیه حرکت‌ها در طول بازی پاداش صفر می‌گیرند. یادگیری تفاوت زمانی یک روش یادگیری نظارت‌نشده است است که مقدار مورد انتظار یک متغیر را پیش بینی می‌کند. در یادگیری تفاوت زمانی به جای محاسبه کل پاداش‌های آینده[۲۶] (مانند معادله(۲))، پاداش کنونی را با مقدار حالت بعدی جمع می‌کند، تا مقدار حالت کنونی[۲۷] بدست آید.

$$V(s) = r_{t+1} + \gamma V(s') \tag{3}$$

- V(s): مقدار حالت کنونی
- r_{t+1}: پاداش کنونی
- V(s'): مقدار حالت بعدی

همچنین تفاوت زمانی Δ به صورت معادله (۴) تعریف می‌شود:

$$\Delta = r_{t+1} + \gamma V(s') - V(s) \tag{4}$$

هنگامی که تفاوت زمانی محاسبه شد، تابع مقدار حالت به صورت معادله (۵) به‌روز می‌شود:

$$V_t \leftarrow V_t + \alpha(r_{t+1} + \gamma V_{t+1} - V_t) \tag{5}$$

[۲۶] Future rewards

[۲۷] State value

- α : پارامتر یادگیری است. این پارامتر مقداری بین ۰ تا ۱ دارد و نشان می‌دهد اگر نرخ یادگیری پایین باشد، عامل به کندی یاد می‌گیرد زمان بیشتری طول می‌کشد تا تابع به مقدار بهینه میل کند. اما اگر نرخ یادگیری بالا باشد ممکن است تابع نتواند به مقدار بهینه میل کند [6]. همچنین نرخ یادگیری صفر به این معنی است که V_t به‌روزرسانی نمی‌شود.
- γ : نرخ تخفیف است. نشان دهنده میزان ارزش پاداش آینده است.

ارزش هر حالت بر طبق سیاست π به صورت معادله (۶) تعریف می‌گردد:

$$V_t^{\pi} \equiv r_{t+1} + \gamma V_{t+1}^{\pi} \quad (6)$$

که در آن V_{t+1} حالتی است که از حالت V_t طبق سیاست π به آن خواهیم رسید. همچنین γ عددی بین ۰ و ۱ است($0 < \gamma < 1$) .

در نتیجه برای سیاست بهینه π^* خواهیم داشت:

$$V^*_t \equiv V_t^{\pi^*} = \max_a \{R_{t+1}(a) + \gamma V_{t+1}^{\pi^*}\} \quad (7)$$

حال عبارتی که باید از آن نسبت به هر عمل، ماکزیمم گرفته شود را تابع Q تعریف می‌کنند.

$$Q\ (s_t, a_t) = r_t + \gamma V_{t+1}^{\pi} \quad (8)$$

۲-٤- یادگیری-کیو

یادگیری-کیو یک الگوریتم یادگیری تقویتی خارج-از-رویکرد[۲۸] و بدون مدل[۲۹] است. بدون مدل برای آن که برای انجام اعمال به رویکرد وابسته نیست که از برتری‌های آن محسوب می‌شود.

در این الگوریتم به هر زوج حالت-عمل[۳۰] یک مقدار $Q(\mathrm{s}, a)$ نسبت داده می‌شود . این مقدار عبارت است از مجموعه پاداش‌های دریافتی، وقتی عامل از حالت S شروع و عمل a را انجام دهد و در ادامه رویکرد موجود را پیروی کرده باشد، تا زمانی که به مقدار بهینه همگرا شود.

این الگوریتم با استفاده از معادله (۹) به‌روزرسانی می‌شود و همانطور که گفته شد نیازی به داشتن مدلی از محیط ندارد. [9]

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{\mathrm{a}} Q_k(s_{t+1}, a_{t+1}) \; - \; Q(s_t, A_t) \right] \qquad (9)$$

که r_t پاداش s_t و α نرخ یادگیری ($0 < \alpha < 1$) می‌باشد. مقدار فاکتور تخفیف γ بگونه‌ای است که $0 < \gamma < 1$ باشد.

همانطور که دیده می‌شود، یادگیری-کیو یک الگوریتم خارج از رویکرد است و به همین دلیل ماکزیمم حالت-عمل‌های بعدی را انتخاب می‌کند. این کار باعث می‌شود همیشه از رویکرد بهینه حرکت کند، که بدی‌ها و خوبی‌های این کار در مقایسه آن با روش سارسا و همینطور پیاده‌سازی در فصل بعد بررسی می‌شود.

یک اپیزود الگوریتم، هنگامی که s_{t+1} به وضعیت نهایی برسد، پایان می‌یابد. $Q(s_f, a)$ برای همه وضعیت‌های نهایی هیچگاه به‌روز نمی‌شود و مقدار اولیه خود را حفظ می‌کند.

---

[۲۸] Off policy

[۲۹] Model free

[۳۰] Action-state

۲-۵- الگوریتم سارسا

سارسا یک الگوریتم طبق رویکرد[۳۱] و بدون مدل از یادگیری تقویتی می‌باشد. سارسا مخفف حالت-عمل-پاداش-حالت-عمل[۳۲]می‌باشد. اسم این روش نشان‌دهنده آن است که معادله به‌روزرسانی آن مقدار Q ها در جدول-Q را با استفاده از $(s_t, a_{t,}, r_t, s_{t+1}, a_{t+1})$ به‌روزرسانی می‌کند. که به ترتیب برابر حالت کنونی، عمل کنونی، پاداش کنونی، حالت بعدی و عمل بعدی می‌باشد.

محاسبه تابع حالت-عمل و به‌روزرسانی آن به شکل معادله (۱۰) می‌باشد:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_t + \gamma\, Q(s_{t+1}, A_{t+1}) - Q(s_t, A_t)] \qquad (10)$$

که r_t پاداش s_t و α نرخ یادگیری ($0 < \alpha < 1$) می‌باشد. مقدار فاکتور تخفیف γ بگونه‌ای است که $0 < \gamma < 1$ باشد.

۲-۵-۱- مقایسه سارسا و یادگیری-کیو

هر دو الگوریتم یادگیری-کیو و سارسا از الگوریتم‌های پر استفاده در یادگیری تقویتی هستند.

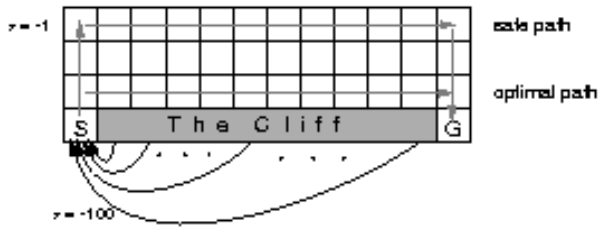
در هر دو الگوریتم عمل a_{t+1} را به یک شکل و بر اساس بهترین عمل ممکن انتخاب می‌کنند. اما از انجایی که الگوریتم یادگیری-کیو یک الگوریتم خارج-از-رویکرد است، به‌روزرسانی Q_t با ماکزیمم Q_{t+1} از میان تمام عمل‌ها انجام می‌شود. اما در سارسا به‌روزرسانی Q_t با $Q(s_{t+1}, a_{t+1})$ انجام می‌شود.

برای آن‌که تفاوت این دو روش مشهود شود، یک مثال از کتاب یادگیری تقویتی [6] می‌آوریم.

---

[۳۱] On-policy

[۳۲] State–action–reward–state–action



شکل ۴: مساله تپه

در این مثال یک عامل می‌خواهد از نقطه S به نقطه G برود. پاداش‌های مساله به این صورت است که اگر عامل از تپه بیوفتد، منفی ۱۰۰ امتیاز می‌گیرد و اگر از مسیر بالایی برود، منفی یک امتیاز می‌گیرد. در روش یادگیری-کیو همیشه از مسیر بهینه که از بالای تپه[۳۳]است، می‌رود. اما به دلیل انتخاب عمل‌ها به صورت اپسیلون-حریصانه، هرازگاهی از روی تپه می‌افتد (و پاداش منفی ۱۰۰ می‌گیرد). اما روش سارسا مسیر امن را یاد می‌گیرد و از مسیر بالایی می‌رود.

شکل ۵: مقایسه دو روش یادگیری-کیو و سارسا در مساله تپه

از مساله بالا نتیجه زیر حاصل می‌شود:

[۳۳] cliff

- یادگیری-کیو به صورت مستقیم رویکرد بهینه را یاد می‌گیرد اما سارسا در حین اکتشاف به رویکرد بهینه نزدیک می‌شود.

- یادگیری-کیو به دلیل داشتن واریانس بالا در عمل‌هایش ممکن است نتواند همگرا به هدف شود. به دلیل اینکه همیشه با بهترین Q به‌روزرسانی میشود. اما سارسا همراه با اکتشاف، به سوی هدف همگرا می‌شود. در واقعیت در محیط‌هایی که که اشتباه کردن هزینه زیادی دارد، مانند یک ربات در واقعیت، بهتر است از روش‌های محتاطانه مانند سارسا استفاده کنیم.

۳- فصل سوم
بازی‌های دیفرانسیلی

۳-۱- مقدمه

بازی‌های دیفرانسلی برای اولین بار توسط آیزاکس معرفی شد که در آن‌کتاب بازی "راننده قاتل" که آن هم یک نوع بازی دنبال‌کننده-گریزنده است حل شده‌است. در بازی راننده قاتل، راننده تلاش می‌کند یک عابر را زیر بگیرد. در این بازی ماشین سریع‌تر حرکت می‌کند، اما عابر توانایی مانور بیشتری دارد. [1]

بازی‌های تعقیب و گریز یکی از مسائل قدیمی بازی‌های دیفرانسیلی است، که در این فصل به آن می‌پردازیم. در این بازی یک عامل به عنوان تعقیب‌کننده و یک عامل به عنوان گریزنده داریم. در این بازی به تعقیب‌کننده یا گریزنده میتوان رویکردهای خاص داد. همچنین می‌توان مانع در محیط آن‌ها ایجاد کرد و یا محدودیت زمانی برای پیدا کردن گریزنده توسط دنبال‌کننده گذاشت. همچنین در این بازی‌ها می‌توان از چندین عامل استفاده کرد، که باعث پیچیده‌تر شدن بازی شده اما آن را به واقعیت نزدیک‌تر می‌کند.

۳-۲- روش‌های پیاده‌سازی بازی‌های دیفرانسیلی

در پیاده‌سازی بازی‌های دیفرانسیلی چندین روش وجود دارد، که روش اول که از کنترل بهینه و معادلات دیفرانسیلی عامل‌ها استفاده می‌کند، عام‌ترین روش آن است که به شرح آن می‌پردازیم.
از آنجایی که ما در این پایان‌نامه می‌خواهیم تعقیب‌کننده-گریزنده با اطلاعات ناقص را شبیه‌سازی کنیم، نمی‌توانیم از معادلات عامل‌ها بهره ببریم و به همین علت از روش دوم در این پایان‌نامه استفاده کردیم.

- **روش اول : حل بازی‌های دیفرانسیلی با معادلات دیفرانسیلی**

اکثر روش‌هایی که تا امروز در مقالات استفاده می‌شده است، حل به وسیله تئوری کنترل بهینه است. در این روش ابتدا معادلات مدل حرکتی هر دو ربات را به شکل معادله (۱۱) بدست آورده:

$$\frac{dx_t}{dt} = f\left(x_t, a_e, a_p\right) \qquad (11)$$

که در آن x_t حالت هر دو عامل (شامل سرعت و مکان هر دو عامل)، همچنین $t \in R$ متغیر زمان پیوسته است. a_e و a_p عمل‌هایی از مجموعه اعمال A_e و A_p هستند. در این روش از آنجایی که زمان و مکان پیوسته هستند، مدل مساله بسیار پیچیده می‌شود. یک راه نوآورانه، بدست آوردن مدل زمان گسسته معادله (۱۱) است[۳۴]:

$$x_{t+1} - x_t = f'(a_e, a_p) \qquad (12)$$

معادله (۱۲) مکان بعدی عامل‌ها را از جمع مکان فعلی و تابعی از عمل‌ها بدست می‌آورد. همچنین با استفاده از معادله آیزاکس، نقطه تعادل زینی[۳۵] را پیدا می‌کند. حل معادله آیزاکس همان حل یک معادله مینیمم-ماکزیمم است.

در این روش بنا بر این گذاشته شده که دو عامل از مدل حرکتی و مکان یکدیگر اطلاع دارند. اما در روش دوم که در این بازی پیاده‌سازی شده است، ربات‌ها با اطلاع ناقص از یکدیگر با استفاده از یادگیری تقویتی، همان رویکرد بهینه را که از معادله ایزاکس بدست می‌آمد، بدست می‌آورند. [10] [11]

- **روش دوم : حل بازی‌های دیفرانسیلی ناقص با یادگیری تقویتی**

در این پایان‌نامه ما از روش دوم حل بازی تعقیب‌کننده-گریزنده بهره می‌بریم. این روش همان مدل کردن به یک بازی مارکوف است که پیاده‌سازی بازی تعقیب‌کننده-گریزنده با اطلاعات ناقص می‌باشد.

در این روش، بازی را به زمان و مکان گسسته تقسیم میکنیم و هر عامل در هر قدم (یک واحد از زمان گسسته) یک عمل انجام داده و پاداش دریافت کرده، به حالت جدید رفته و جدول-Q خود را آپدیت می‌کند.

همینطور می‌توان نشان داد تعادل نش[۳۶] برای بازی‌های دیفرانسیلی با اطلاع ناقص وجود دارد. [11]

[۳۴] با فرض آن که x_t فقط شامل مکان باشد

[۳۵] saddle-point equilibrium

[۳۶] Nash equilibrium

همانطور که گفته شد در پیاده‌سازی این بازی دیگر دو عامل اطلاع دقیقی از موقعیت هم ندارند و فقط به وسیله سنسورها همدیگر را تشخیص می‌دهند. پیاده‌سازی این بازی‌ها با الگوریتم‌های یادگیری تقویتی در فصل بعد به صورت مفصل بحث خواهد شد.

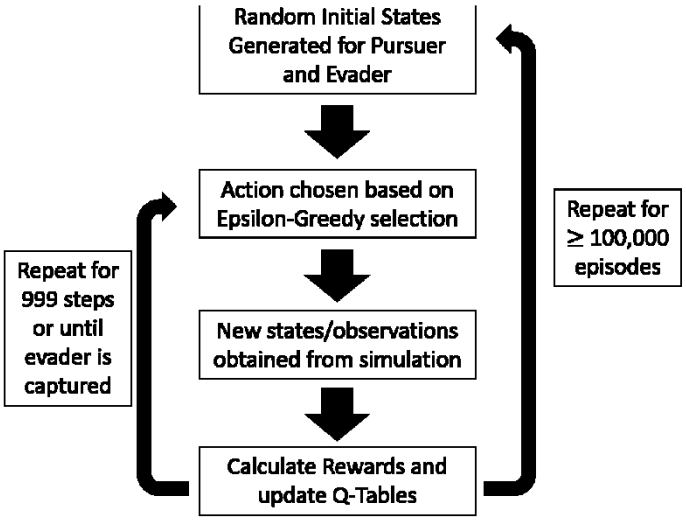
٤- فصل چهارم
پیاده‌سازی

٤-١- فرموله‌سازی مساله:

در پیاده‌سازی این بازی هدف ما این بود که هر دو عامل (عامل تعقیب‌کننده و عامل گزیزنده) را به صورت جداگانه آموزش[۳۷] دهیم. مطابق شکل ۶ هر دو عامل در انتهای هر قدم[۳۸]به صورت مستقل از هم آموزش داده می‌شوند.

در انتخاب قدم‌ها، هر عامل از روش اپسیلون-حریصانه[۳۹] استفاده می‌کند. این روش باعث می‌شود عامل علاوه بر انتخاب بهترین عمل ممکن در اکثر موارد (انتخاب حریصانه)، در برخی موقعیت‌ها عمل را به صورت تصادفی(با احتمال اپسیلون) انتخاب کند تا در اکسترمم‌های محلی گرفتار نشود.

مانند تمام الگوریتم‌های یادگیری تقویتی، ابتدا یک حالت اولیه به عامل‌ها می‌دهیم. سپس عامل‌ها در هر قدم، به صورت حریصانه یا تصادفی یک عمل را انتخاب می‌کند. سپس به حالت جدیدی رفته و پاداش یا تنبیه را دریافت می‌کند و جدول-Q خود را به‌روزرسانی می‌کنند. همانطور که در شکل ۶ آمده است، در هر اپیزود، حداکثر ۹۹۹ قدم اجرا می‌کنیم.



شکل ٦ : الگوریتم یادگیری-کیو استفاده شده در این بازی

[۳۷] train

[۳۸] step

[۳۹] epsilon-greedy

از آن‌جایی که نرخ یادگیری کوچک انتخاب کرده‌ایم، تعداد اپیزودهای تکرار مساله را بالا (بیشتر از ۱۰۰۰۰۰)انتخاب می‌کنیم تا عامل‌ها به خوبی آموزش‌دیده و به رویکرد بهینه نزدیک شوند.

۱-۱-۴- پاداش عامل‌ها در بازی:

نحوه دادن پاداش به هر عامل به ازای شرایط مختلف به شرح زیر است:

در این بازی هر عامل اطلاعات ناقصی از وضعیت حریف خود دارد که به وسیله سنسورها و با توجه به بازه دید تعیین‌شده، به آن عامل می‌رسد. به همین علت حالت دقیق حریف بر هرکدام از عامل‌ها پوشیده است.

در یادگیری تقویتی ما به وسیله پاداش مثبت دادن به عامل تعقیب‌کننده در صورت احساس[۴۰] کردن گریزنده و تنبیه (پاداش منفی) به گریزنده، در صورت احساس تعقیب‌کننده، آن‌ها را آموزش می‌دهیم.

پاداش تعقیب‌کننده و گریزنده به شرح زیر تعریف می‌شود:

$$R_{Pursuer} = \begin{cases} No\ Capture & R_{Pursuer} = -1 \\ Evader\ Observed & R_{Pursuer} = +2 \\ Evader\ Captured & R_{Pursuer} = +100 \end{cases}$$

$$R_{evader} = \begin{cases} No\ Capture & R_{Pursuer} = +1 \\ Evader\ Observed & R_{Pursuer} = -2 \\ Evader\ Captured & R_{Pursuer} = -100 \end{cases}$$

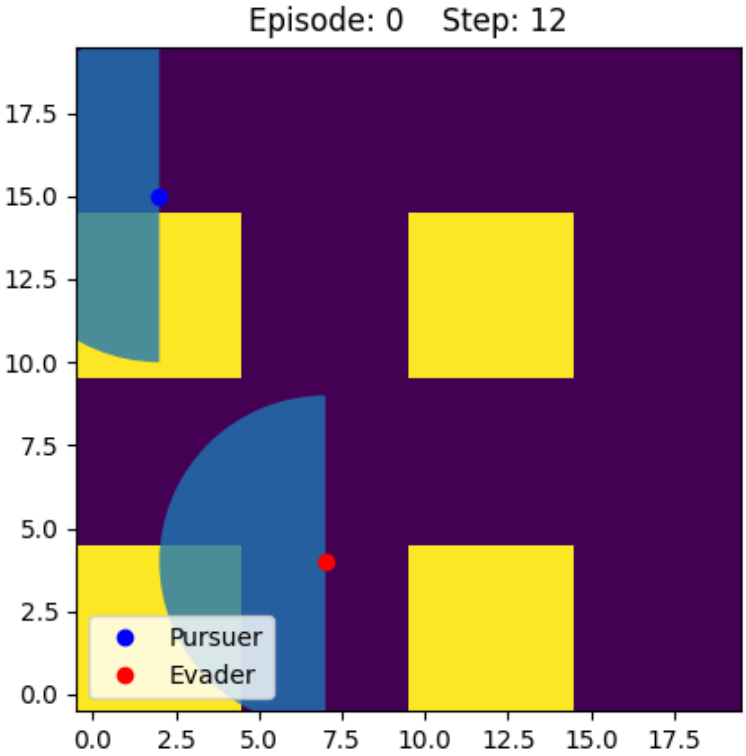
برای دنبال‌کننده در هر قدمی که گریزنده را احساس نکند و یا آن را نگیرد، ۱ واحد پاداش منفی می‌دهیم. و در صورت احساس گریزنده (گریزنده در بازه دید دنبال‌کننده باشد)، ۲ واحد پاداش مثبت و در صورت گرفتن گریزنده ۱۰۰ واحد پاداش مثبت دریافت می‌کند.

پاداش‌های عامل گریزنده عکس پاداش‌های عامل دنبال‌کننده انتخاب شد. زیرا در این بازی دارای دو عامل خصمانه هستیم، که جمعشان صفر است.

[۴۰] Sense

۲-۱-۴- محیط بازی

فضای محیط بازی را به صورت: $x = \{x, y, \theta, z\}$ می‌باشد. هر عامل x و y خود را دارد که نشان-دهنده موقعیت آن عامل در فضای گرافیکی بازی می‌باشد. همچنین جهت عامل نیز با θ مشخص می‌شود. z نیز سنسور عامل را نشان می‌دهد که اگر حریف خود را دید؛ یعنی حریف در بازه دید قرار گرفت، برابر با یک می‌شود، در غیر این صورت صفر است.



شکل ۷ : فضای گرافیکی بازی

محیط گرافیکی بازی یک مربع ۲۰ در ۲۰ است که دارای ۴ مانع است. هر عامل می‌تواند ۴ جهت بالا، پایین، چپ و راست را در بازی داشته باشد. همچنین این محیط به صورت زنده به‌روزرسانی می‌شود و بازی را به نمایش می‌گذارد. در این محیط عامل رنگ آبی دنبال‌کننده و عامل رنگ قرمز، گریزنده است .

هر عامل در هر قدم یکی از ۶ حرکتی که در جدول۱ آمده است را در صورتی که بتواند در آن جهت‌ها حرکت کند، می‌تواند انجام دهد. این حرکات فضای عمل[۴۱] عامل‌ها را تشکیل می‌دهد و شامل متوقف ماندن، حرکت رو به جلو، چرخیدن به چپ یا راست، چرخیدن به چپ یا راست و حرکت رو به جلو در جهت جدید می‌باشند. این حرکات برای هر عامل به صورت مستقل می‌باشند.

جدول 1: فضای دید هر ربات

عمل	حرکت خطی	حرکت دورانی
Stop	0	0
Forward	1	0
Turn Left	0	1
Turn Right	0	-1
Move and Turn Left	1	1
Move and Turn Right	1	-1

همچنین فضای دید هر ربات به صورت $x = \{z\}$ می‌باشد. اگر ربات دیگر در این بازه حس شود، $z=1$ می‌شود و ربات حریف حس شده است. در غیر این صورت $z=0$ است. دید عامل به صورت ۱۸۰ درجه تعریف شده است و همچنین شعاع دید آن نیز ۵ واحد تعریف شده است.

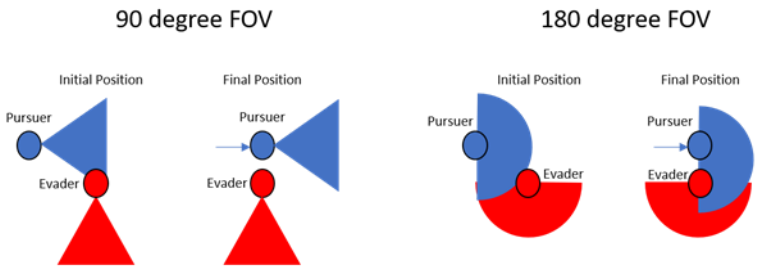
در ابتدا عامل با چند بازه دید بر اساس فاصله تعریف شده بود. یعنی اگر ربات مقابل در فاصله ۳ بود، پاداش با فاصله ۱ متفاوت بود. اما به دلیل ایجاد کندی در اموزش عامل‌ها و افزایش فضای حالت، حس کردن به حالت باینری (بودن یا نبودن در بازه دید)، تغییر داده شد.

[۴۱] Action space

جدول2 : فضای دید عامل گریزنده

Observation	*Returned*
حس شده	0
حس نشده	1

مطابق **شکل ۸** فضای دید ربات به صورت ۱۸۰ درجه انتخاب شده‌است تا عامل ها بتوانند حریف خود را در بیشتر از یک قدم ببینند. انجام این کار باعث شد دو عامل هنگام نزدیکی بیشتر در بازه دید یکدیگر باشند و در مقایسه با بازه دید۹۰ درجه، سرعت آموزش بالاتر رود.



شکل ۸: اثر بازه دید عامل‌ها در تشخیص حریف در حال گذر

٤-۲- انتخاب پارامترهای یادگیری

بازی‌های تعقیب و گریز بازی‌های نسبتا پیچیده‌ای محسوب می‌شوند. پیچیدگی این بازی‌ها هنگامی که نسبت به هم اطلاعات ناقص داشته باشند، بیشتر هم می‌شود. در این بازی یکی از چالش‌انگیزترین قسمت‌های آموزش عامل‌ها، تنظیم پارامترهای یادگیری بود.

برای آن که پارامترهای یادگیری به درستی تنظیم‌شوند، با تغییر اعداد و بررسی نمودارهای پاداش-اپیزود[۴۲]، اطمینان حاصل شد که پاداش هر دو عامل به یک مقدار میل کند.

- نرخ یادگیری:

[۴۲] Reward-episode

برای آن که از بیش-یادگیری[۴۳] جلوگیری کنیم از نرخ یادگیری کوچک استفاده می‌کنیم. زیرا ممکن است در اپیزودی دو عامل به طور تصادفی به هم برسند و ما نمی‌خواهیم این جدول-Q را تغییر زیادی دهد.

- تعداد تکرار:

از آن‌جایی که نرخ یادگیری کوچک استفاده کردیم باید تعداد تکرار زیادی داشته باشیم تا به رویکرد بهینه میل کنیم. در این مسئله از تعداد اپیزود بالاتر از ۱۰۰۰۰۰ استفاده می‌کنیم و مطابق نتایجی که در فصل ۵ گفته می‌شود، می‌توانیم به رویکرد بهینه میل کنیم.

- شعاع دید عامل:

شعاع دید عامل را به صورت صحیح و خطا بر روی ۵ تنظیم کردیم. زیرا در صورتی که شعاع بزرگ باشد، از واقعیت به دور است و همینطور در صورتی که شعاع بسیار کوچک باشد، آموزش عامل‌ها بسیار طولانی خواهد شد.

- فاکتور تخفیف:

نشان‌دهنده میزان اهمیت پاداش آینده است.

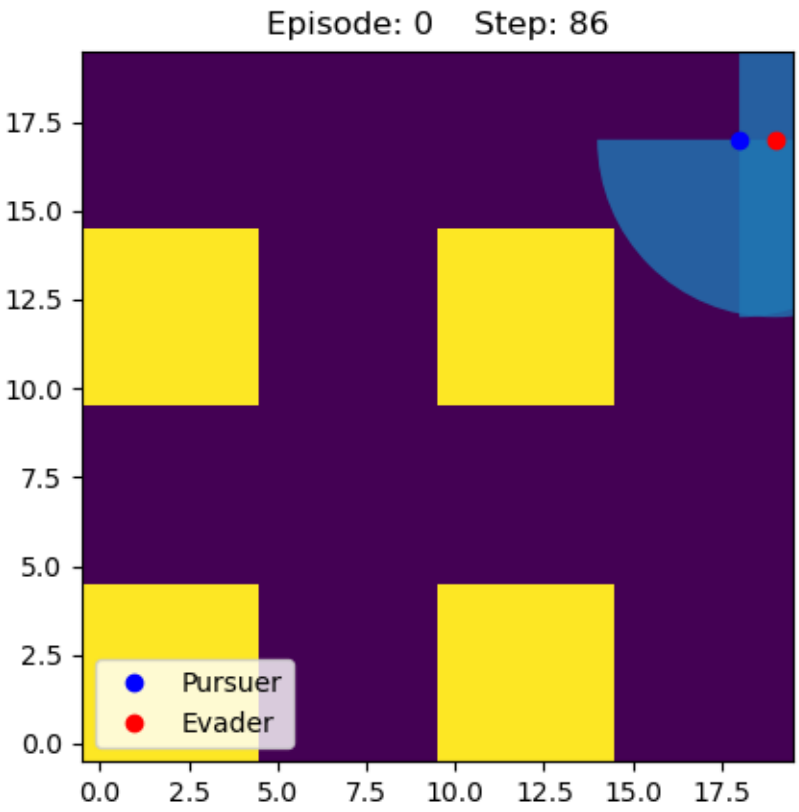
پس از امتحان چندین پارامتر و بررسی نمودارهای آن‌ها، همچنین بررسی چندین زاویه دید، پارامترهای زیر بدست آمد. که با هر دو الگوریتم یادگیری پیاده‌سازی شده در زمان نسبتا منطقی رویکرد بهینه را پیدا کرده به سمت آن میل میکنند.

پارامترهایی که با آن‌ها توانستیم عامل‌ها را آموزش دهیم به شرح زیر هستند:

- نرخ یادگیری: ۰.۰۱
- فاکتور تخفیف: ۰.۹
- تعداد تکرار: ۱۰۰۰۰۰
- تعداد قدم در هر اپیزود: ۹۹۹

[۴۳] Overtraining

- زاویه دید عامل: ۱۸۰ درجه
- شعاع دید عامل: ۵



شکل ۹:لحظه رسیدن دنبال کننده به گریزنده

٤-۳- پیاده‌سازی الگوریتم‌های یادگیری تقویتی

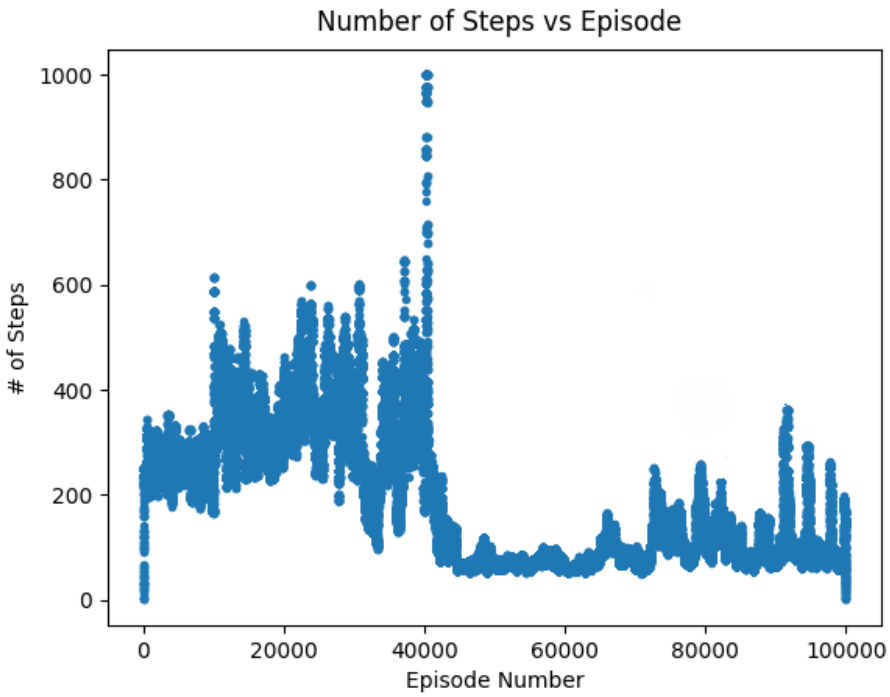
در این قسمت ابتدا هرکدام از روش‌های یادگیری-کیو و سارسا را پیاده‌سازی کرده و درباره آن‌ها بحث می‌کنیم. ابتدا نمودار تعداد قدم و مقدار پاداش بر اساس تعداد اپیزود اجرا شده را تحلیل می‌کنیم. سپس چند آزمایش بر روی مدل آموزش داده شده اجرا کرده و نتایج را نشان می‌دهیم.

٤-٣-١- پیاده‌سازی الگوریتم اول : یادگیری-کیو

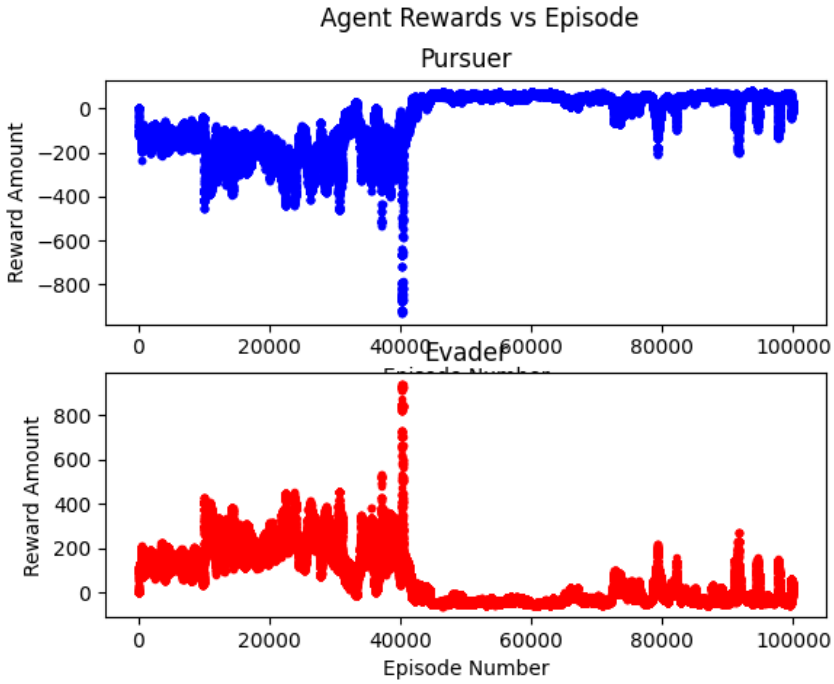
در پیاده‌سازی الگوریتم یادگیری-کیو، ابتدا پارامترهای آموزش را تعیین می‌کنیم. سپس در یک حلقه به تعداد اپیزودهای بازی، بازی را اجرا می‌کنیم. در هر قدم از بازی، ابتدا عامل یک عمل را بر اساس روش اپسیلون-حریصانه انتخاب کرده و سپس مطابق معادله ۹، جدول-Q را به‌روزرسانی می‌کند.

در آنالیز داده‌ها، با آنکه در ابتدا نتایج بسیار متفاوتی در هر اپیزود بدست امده بود، هنگامی که فیلتر بر روی نتایج زده شد، الگوها از آن استخراج شد.

همانطور که در **شکل ۱۰** و **شکل ۱۱** دیده می‌شود، در ۴۰۰۰۰ اپیزود اول، گریزنده پاداش نهایی مثبت و دنبال‌کننده پاداش نهایی منفی بیشتری دارد و بازی بیشتر طول می‌کشد (به طور میانگین ۳۰۰ قدم). اما پس از اپیزود ۴۰۰۰۰ تغییر رخ داده و پاداش‌ها به صفر نزدیک می‌شود که باعث می‌شود دنبال‌کننده بر گریزنده غلبه کرده و زودتر به آن برسد (به طور میانگین در ۱۰۰ قدم).



شکل ۱۰: تعداد قدم‌های دنبال کننده به گریزنده در روش یادگیری-کیو



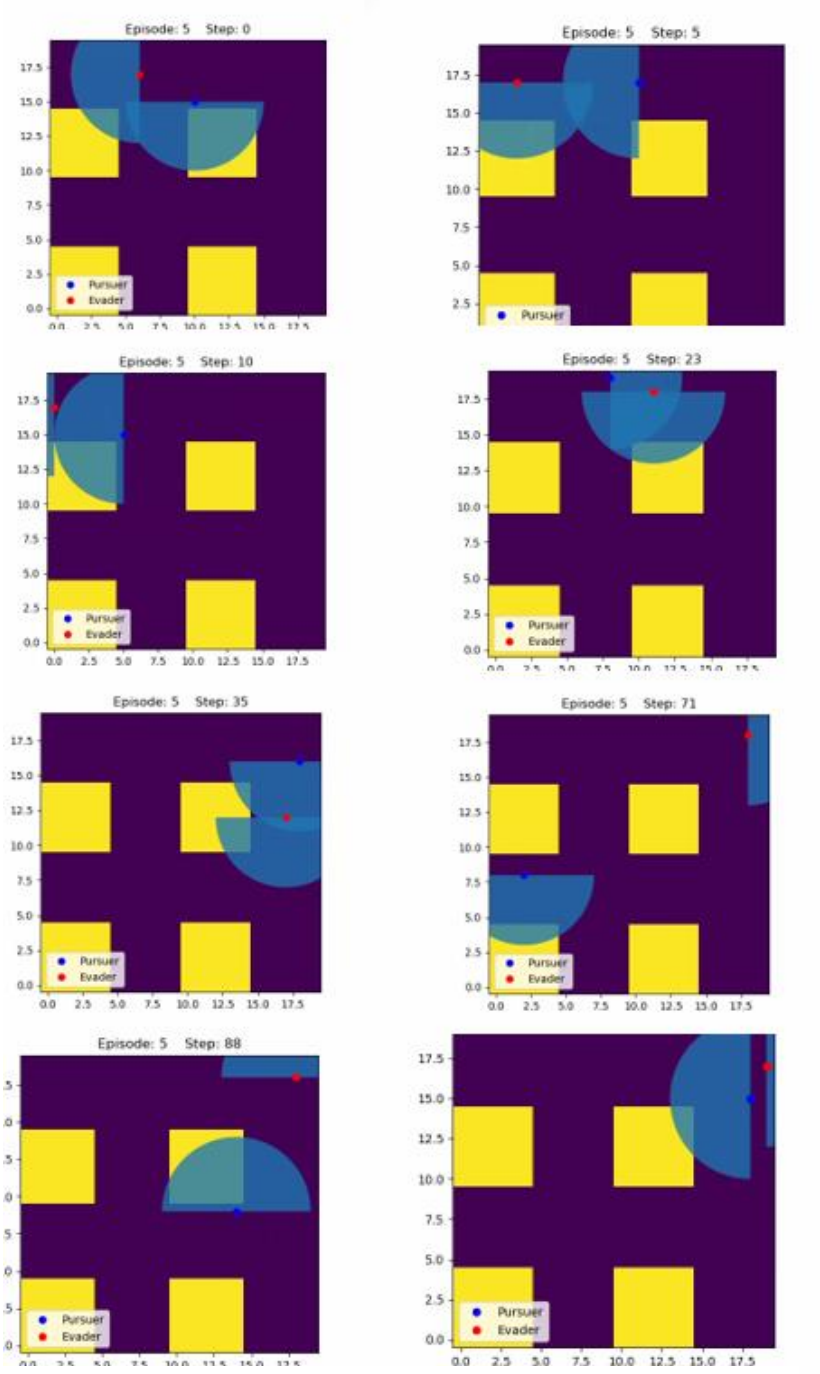
شکل ۱۱: پاداش نهایی هر یک از عالم‌ها بر حسب تعداد اپیزود در روش یادگیری-کیو

حال به بررسی و آزمون عامل‌های اموزش داده شده می‌پردازیم: (کلیه ویدیوهای آزمایش‌ها در گیت‌هاب این پایان‌نامه موجود است[۴۴].)

در هر تست ماکزیمم قدم‌ها ۲۵۰ قدم در نظر گرفته شده‌است. یعنی اگر دنبال‌کننده در ۲۵۰ قدم گریزنده را پیدا کند، دنبال‌کننده برده است. در غیر این صورت، گریزنده بازی را می‌برد.

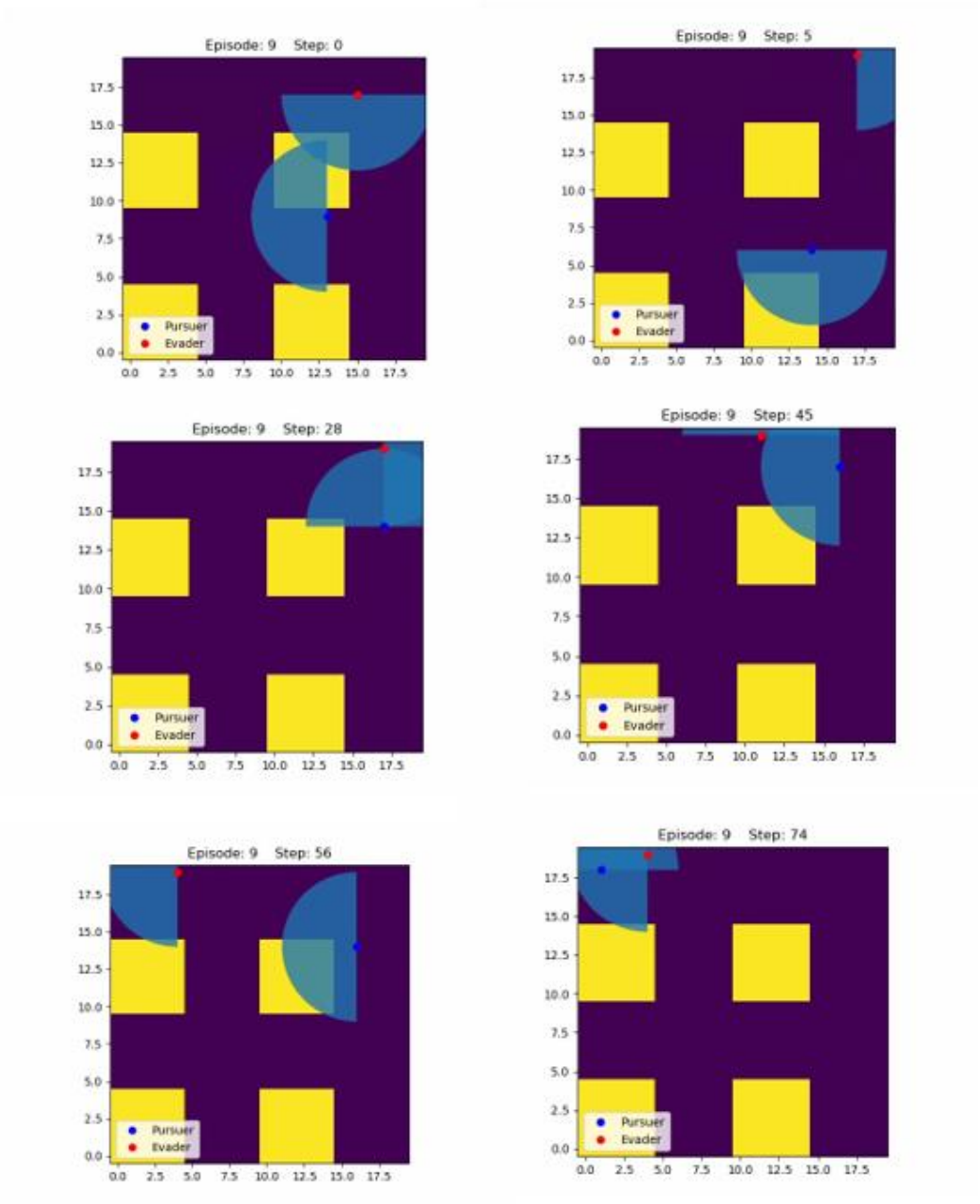
[۴۴] https://github.com/sepehr0007/BSc-Thesis-report

- آزمایش اول: در این آزمایش ابتدا عامل گریزنده به گوشه سمت چپ-بالای محیط می‌رود. اما دنبال‌کننده آن را پیدا می‌کند. سپس تغییر مکان داده به به گوشه سمت راست بالا می‌رود، و عامل دنبال‌کننده پس از ۹۴ قدم می‌تواند او را بگیرد.



شکل ۱۲: آزمایش اول الگوریتم یادگیری-کیو

- آزمایش دوم: این آزمایش بسیار جالب بود. در این آزمایش ابتدا دنبال‌کننده به سمت گریزنده می‌رود، سپس گریزنده به سمت شمال غربی زمین فرار می‌کند اما دنبال‌کننده می‌تواند در ۷۴ قدم او را شناسایی کند.

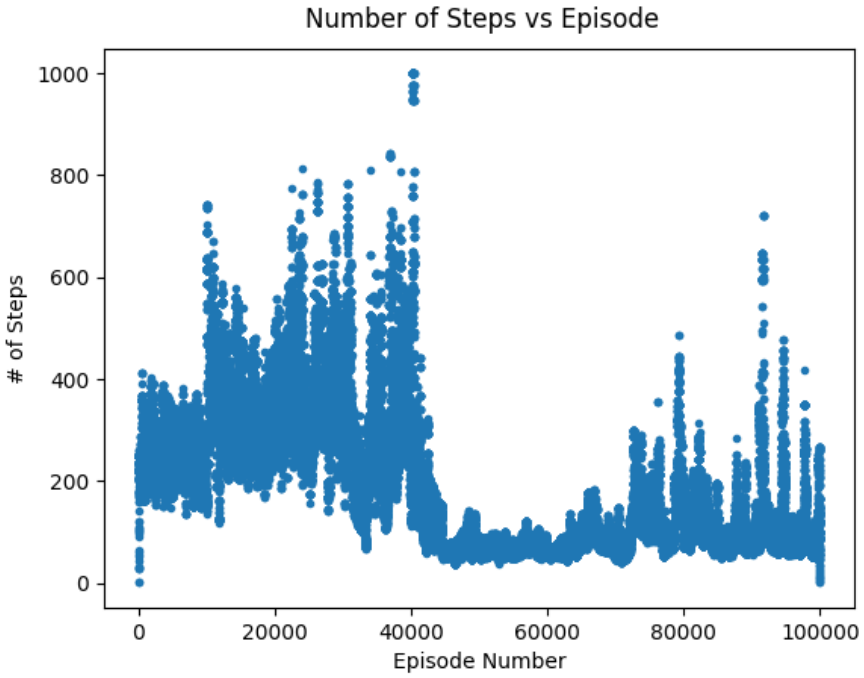


شکل ۱۳: آزمایش دوم یادگیری-کیو

٤-٣-٢- پیاده‌سازی الگوریتم دوم : سارسا

همانطور که اشاره شد، تفاوت اصلی الگوریتم سارسا و یادگیری-کیو در به‌روزرسانی جدول-Q می‌باشد. پس با پیاده‌سازی الگوریتم سارسا با استفاده از معادله ۱۰، و انتخاب پارامترهای یادگیری مانند الگوریتم یادگیری-کیو، بازی را اجرا می‌کنیم.

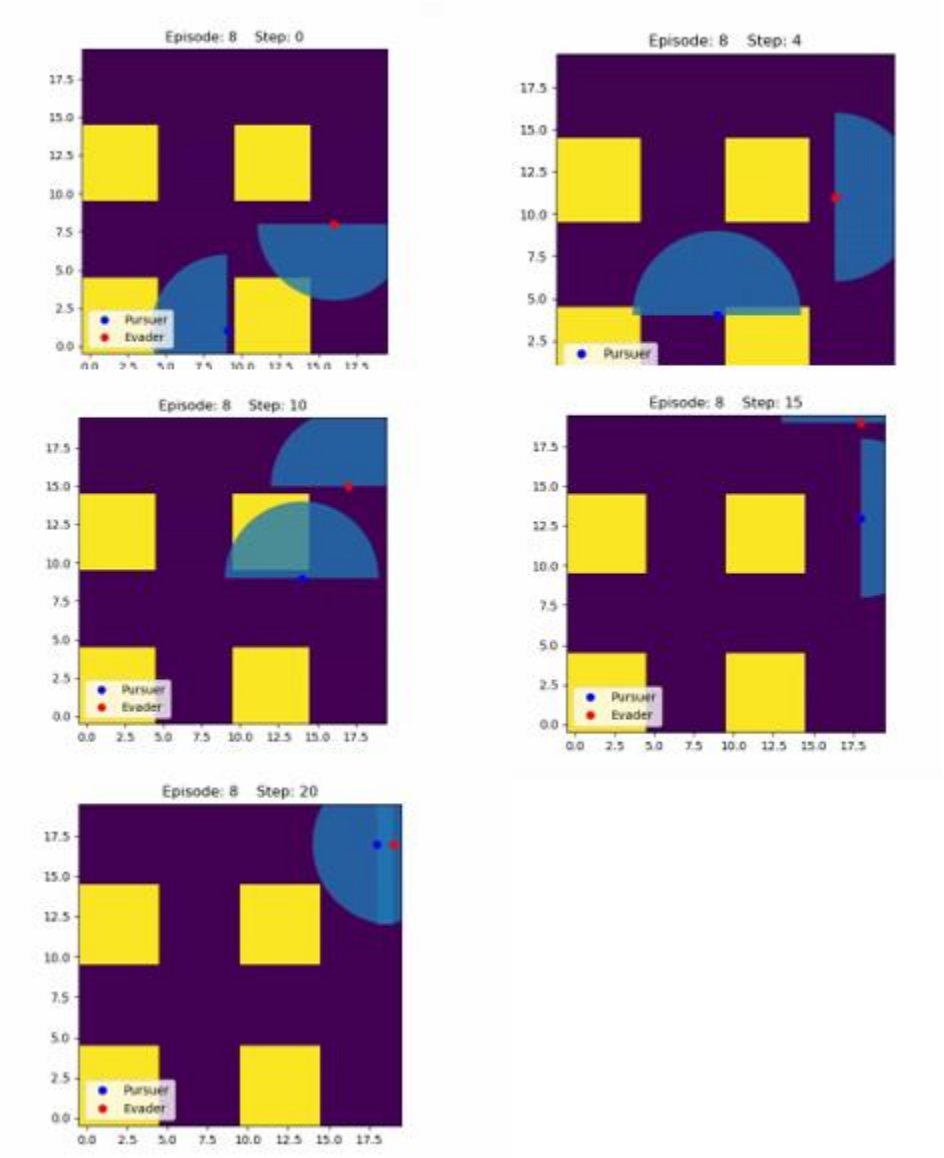
پس از یادگیری عامل‌ها و تحلیل نتایج با استفاده از استفاده از فیلتر بر روی نمودار، الگوریتم سارسا نیز تقریبا مانند الگوریتم یادگیری-کیو عمل کرد. یعنی پس از ۴۰۰۰۰ اپیزود، تقریبا به رویکرد بهینه میل کرد. البته از آن‌جایی که الگوریتم سارسا، در همیشه رویکرد بهینه را انتخاب نمی‌کند و به اکتشاف می‌پردازد، میانگین تعداد قدم‌های آن در هر اپیزود بالاتر از یادگیری-کیو است.



شکل ۱۴: تعداد قدم‌های دنبال کننده به گریزنده در روش سارسا

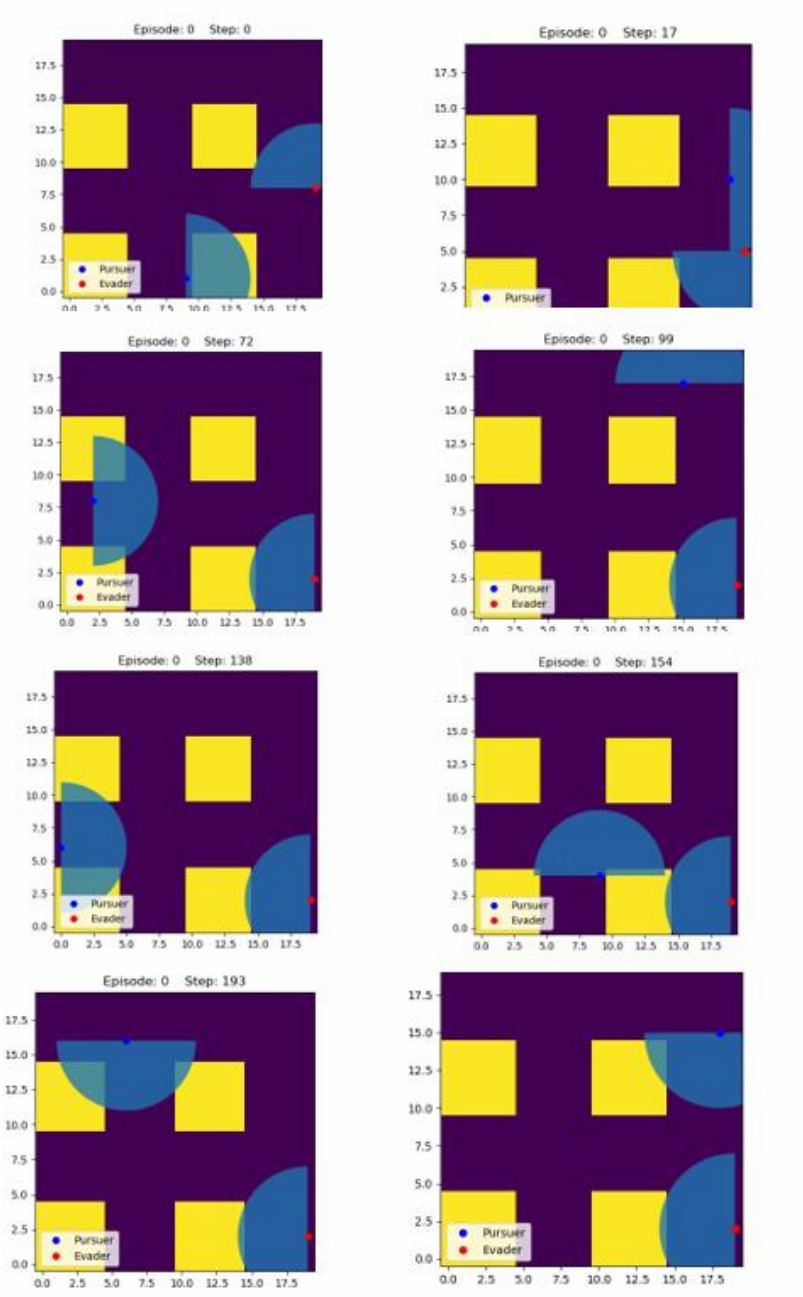
پس از آموزش عامل‌ها، حال به بررسی و تست آن‌ها می‌پردازیم:

- آزمایش اول: در آزمایش اول الگوریتم سارسا، دنبال‌کننده به خوبی و در ۲۰ قدم می‌تواند گریزنده را شناسایی کرده و خود را به آن برساند.



شکل ۱۵: آزمایش اول الگوریتم سارسا

- آزمایش دوم: در این آزمایش دنبال‌کننده در ۲۵۰ قدمی که برای تست الگوریتم در نظر گرفتیم، نمی تواند گریزنده را شناسایی کند و گریزنده یاد گرفته است که اگر در ضلع جنوب-شرقی زمین پنهان شود، دنبال‌کننده نمی‌تواند آن را پیدا کند.



شکل ۱۶: آزمایش دوم الگوریتم سارسا

٤-٤- نتیجه‌گیری :

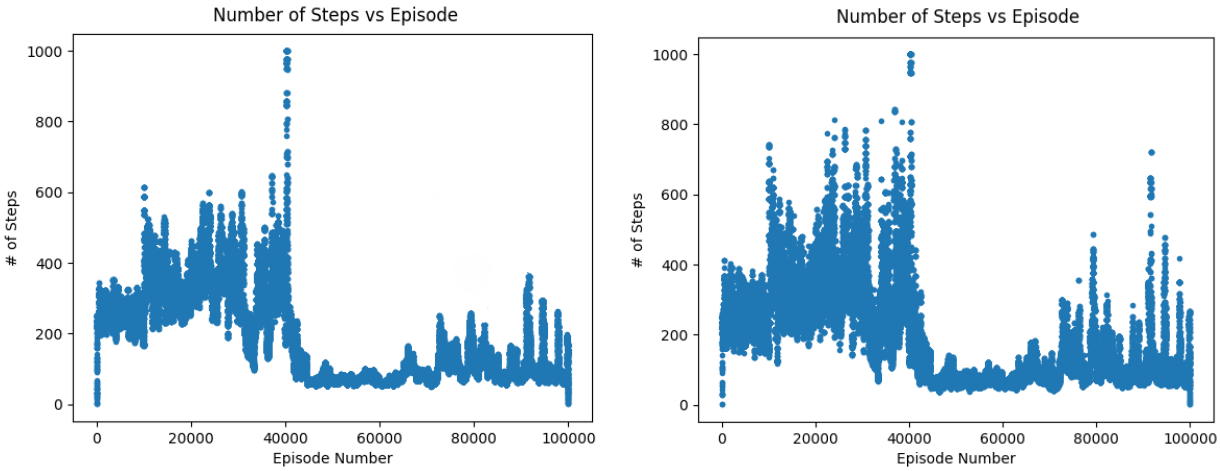
همانطور که در **شکل ۱۷** قابل مشاهده است، هر دو روش به خوبی و تقریبا مشابه هم عمل می‌کنند. اما روش یادگیری-کیو کمی بهتر عمل کرده و زودتر به رویکرد بهینه میل می‌کند.

تفاوت اولی که بیشتر به چشم می‌آید این است که از آن‌جایی که روش سارسا بیشتر به اکتشاف می‌پردازد، میانگین تعداد قدم‌های آن در هر اپیزود بیشتر از روش یادگیری-کیو می‌باشد.

تفاوت دوم این است که در یادگیری-کیو ما به رویکرد بهینه می‌رسیم اما در سارسا ما به رویکرد بهینه فقط نزدیک می‌شویم. به همین علت حتی پس از آموزش عامل‌ها، میانگین تعداد قدم‌های سارسا بیشتر از یادگیری-کیو می‌باشد.

البته ایرادی که در کتاب مقدمه‌ای بر یادگیری تقویتی [6] به آن اشاره شده نیز وجود دارد. این کتاب می‌گوید: از آنجایی که یادگیری-کیو در هر حالت بهترین انتخاب در آن زمان را انجام می‌دهد(اکسترمم‌های محلی)، در برخی مواقع به علت استفاده از روش اسپیلون-حریصانه و انتخاب عمل تصادفی، عملی بسیار متفاوت انجام دهد. اما الگوریتم سارسا محتاطانه به رویکرد بهینه میل می‌کند. و اجازه خطاکردن در هنگام انجام گردش را می‌دهد. در حالی که یادگیری-کیو آن‌ها را نادیده گرفته و همیشه بهترین رویکرد ممکن را انتخاب می‌کند.

در واقعیت اگر اشتباه کردن هزینه داشته باشد، نکته آخر بسیار مهم است و استفاده از سارسا که اعمال محتاطانه انجام می‌دهد، امن‌تر است. زیرا اگر عامل ما یک ربات باشد، آسیب به آن دارای هزینه است.



شکل ۱۷: مقایسه دو روش یادگیری-کیو و سارسا

۵- فصل پنجم
جمع‌بندی و نتیجه‌گیری و کارهای آتی

۵-۱- جمع‌بندی و نتیجه‌گیری

بازی تعقیب و گریز به وسیله الگوریتم‌ها و با بهینه‌سازی‌های متفاوتی بررسی و حل شده است. در این پایان‌نامه، ما یک محیط گرافیکی ایجاد کردیم. سپس دو روش یادگیری تقویتی را بر روی دو عامل که با یکدیگر رفتار خصمانه دارند پیاده‌سازی کرده، نتایج هر روش را توضیح داده و روش‌ها را با هم مقایسه کردیم. با یک دنبال‌کننده و یک گریزنده ما توانستیم یک محیط ساده که توانایی پردازش آن برای یک ربات وجود داشته باشد را شبیه‌سازی کنیم.

با به کارگیری الگوریتم‌های یادگیری-کیو و سارسا، همچنین استفاده از فریمورک OpenAI Gym، ما توانستیم مدل موفقی از بازی تعقیب و گریز پیاده‌سازی کنیم، که با استفاده از سنسورهای عامل و جدول-Q با پاداش‌های مناسب، عامل‌ها را اموزش دهیم.

فهمیدیم بیش‌آموزش یکی از مسائل اصلی در الگوریتم‌های یادگیری تقویتی است. همچنین تنظیم فراپارامترها و پیاده‌سازی سنسوری که هم توانایی پردازش اطلاعات آن وجود داشته باشد و هم کارساز باشد، می‌تواند چالش انگیز باشد.

در تحلیل نتایج با ابرپارامترهای استفاده شده در پایان‌نامه، هر دو روش به خوبی عمل کرده و کارکرد نزدیکی دارند. البته روش یادگیری-کیو کمی زودتر به رویکرد بهینه میل کرد و دارای میانگین پاداش کمتری نسبت به سارسا است. البته این روش خالی از ایراد نبوده و ایراد های آن نیز در فصل ۴ بررسی شد.

۵-۲- کارهای آتی

در آینده می‌توان این پروژه را به بازی‌های چندعاملی گسترش داد که امروزه بسیار موضوع پرطرفداری می‌باشد. یعنی یک یا چند عامل در تیم دنبال‌کنندگان و یک یا چند عامل در تیم گریزندگان باشد، که در این صورت علاوه بر رفتار خصمانه میان عامل‌ها، نیازمند همکاری عامل‌ها نیز می‌باشد.

همچنین می‌توان در هر عامل از چند سنسور استفاده کرد. در این صورت، هر عامل درک محیطی بهتری از محیط اطراف خود و عامل‌های احتمالی، پیدا می‌کند. البته این کارها پیچیدگی فضای عامل‌ها را افزایش

می‌دهد. که می‌توان با افزایش قدرت پردازش عامل‌ها، یا استفاده از پردازش ابری این مشکل را حل کرد. پیشنهاد دیگر می‌تواند استفاده از دوربین و پردازش تصویر باشد که در آن شبکه‌های عمیق را با روش های یادگیری تقویتی تلفیق می‌کنیم تا نتیجه مطلوب به دست آید.

منابع و مراجع

[١] R. Isaacs ،Differential Games Wiley ،New York, NY .١٩٦٥ ،

[٢] T. B. a. G. Olsder ،Dynamic non-cooperative game theory ،New york: Academic press .١٩٨٢ ،

[٣] R. V. D. H. S. O. S. a. S. S. H. J. Kim ،A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles ،In Proceedings of the 40th IEEE Conference on Decision and control .

[٤] D. T. a. R. Fierro. ،Adaptive sampling for tracking in pursuit-evasion .٢٠١١ ،

[٥] P. Kulkarni ،Reinforcement and Systemic Machine Learning for Decision Making. .٢٠١٢ ،

[٦] Barto, R. S. Sutton and A. G. ،Introduction to Reinforcement Learning. Cambridge ،MIT Press, 1st ed. .١٩٩٨ ،

[7] "Supervised vs Unsupervised vs Reinforcement Learning – Knowing the differences is a fundamental part of properly understanding machine learning," [Online]. Available: https://starship-knowledge.com/supervised-vs-unsupervised-vs-reinforcement. [Accessed 10 9 2021].

[٨] Raslan, Hashem and Schwartz, Howard and Givigi, Sidney ،A learning invader for the guarding a territory game ٢٠١٦ ،Annual IEEE Systems Conference (SysCon) ، .٢٠١٦

[۹] C. D. P. Watkins ،Q-learning .۱۹۹۲ ،

[۱۰] X. Huang ،Adversary agent reinforcement learning for pursuit-evasion .۲۰۲۱ ،

[۱۱] M. E. Khan ،Game Theory Models for Pursuit Evasion Games .۲۰۰٦ ،

[12] "Reinforcement Learning," [Online]. Available: https://en.wikipedia.org/wiki/Reinforcement_learning. [Accessed 10 9 2021].

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Electrical Engineering

BSc. Thesis

Comparison between applying reinforcement learning based methods to solve differential games

By
Sepehr Karimi Arpanahi

Supervisor
Dr. M. B. Menhaj

October 2021