دوفصلنامه بازی جنگ، سال پنجم، شماره ۱۰، بهار و تابستان ۱۴۰۱

تاریخ دریافت: ۱۴۰۱/۰۴/۲۰ تاریخ پذیرش: ۱۴۰۱/۰۵/۱۰

از صفحه ۲۵ تا صفحه ۴۶

مسیریابی مبتنی بر هوش مصنوعی در بازی جنگ زمینی [۱]

حمید بیگدلی*[۲]
صابر جبارزاده[۳]
جلیل مظلوم[۴]

نوع مقاله: پژوهشی

چکیده

بازی جنگ[۵] در دنیا به عنوان یکی از روش‌های تمرین تصمیم‌سازی و تصمیم‌گیری فرماندهان نظامی مورد استفاده قرار می‌گیرد و درصورتی‌که این شبیه‌سازی با ابزارهای هوش مصنوعی[۶] تولید شود، قدرت تصمیم‌گیری را افزایش داده و تأثیر بسزایی در کاهش هزینه‌ها و جلوگیری از هدر رفت منابع خواهد داشت. یکی از موارد مورد استفاده در سامانه‌های بازی جنگ که موضوع این مقاله است، بخش مسیریابی می‌باشد. در این راستا ابتدا مسیریابی A^* مورد بررسی قرار گرفته و سپس معایب آن با ترکیب روش سلسله‌مراتبی، بهبود داده شده است. سپس مسیریابی با الگوریتم یادگیری تقویتی با پیش‌پردازش داده‌ها به صورت روش‌های ابتکاری به منظور همسان‌سازی نقشه‌ها، پیش از انجام فاز مسیریابی مورد مطالعه قرار گرفته است. در ادامه مسیریابی با یادگیری تقویتی به صورت سلسله‌مراتبی، بررسی شده است. همچنین زمان و سرعت یگان‌های مختلف در مسیریابی اعمال شده و نتایج آن مورد بررسی قرار گرفته است.

واژه‌های کلیدی:

هوش مصنوعی، الگوریتم مسیریابی، مسیریابی سلسله‌مراتبی، یادگیری تقویتی. [۷]

---

[1] Ai-Based Path Finding In Land War Game

[۲] استادیار، گروه مطالعات علم و فناوری، دانشگاه فرماندهی و ستاد آجا، تهران، ایران.

[۳] پژوهشگر، گروه مطالعات علم و فناوری، دانشگاه فرماندهی و ستاد آجا، تهران، ایران.

[۴] دانشیار، دانشکده مهندسی برق دانشگاه علوم و فنون هوایی شهید ستاری، تهران، ایران.

* نویسنده مسئول: Email: H.Bigdeli@casu.ac.ir

[5] Strategy Game

[6] Artificial Intelligence

[7] Reinforcement Learning

مقدمه

بازی جنگ عبارت است از نمایش و شبیه‌سازی عملیات نظامی که درآن دو نیروی متعارض که با استفاده از قوانین، جداول و روش‌هایی که برای شرح وضعیت واقعی یا فرضی در نظر گرفته شده‌اند باهم درگیر می‌شوند. بازی جنگ، فنی برای بررسی تصمیمات انسان‌ها با اهداف مختلف می‌باشد و به تحلیلی، آموزشی یا تجربی بودن تمرکز بازی جنگ بستگی دارد.

در قرن پیش رو، با افزایش هزینه جنگ‌های مدرن و رزمایش‌ها و تمرینات تاکتیکی، ارتش‌های کشورهای توسعه یافته پیش از ورود به میدان جنگ واقعی، با وارد نمودن اطلاعات مرتبط با وضعیت نیروهای خودی و دشمن احتمالی در سامانه بازی جنگ و در نظر گرفتن سناریوهای مختلف، نتایج حاصل از جنگ‌های آینده را موردبررسی قرارداد و ضمن کسب برآورد نسبی از نقاط قوت و ضعف خود، در راستای توسعه و تقویت توانمندی‌ها و از بین بردن نقاط ضعف موجود اقدام لازم را انجام می‌دهند. درواقع، بازی جنگ به‌عنوان روشی برای سنجش توان نظامی و دفاعی در کشورهای پیشرفته مورد استفاده قرارگرفته است.

هوش مصنوعی فناوری قدرتمندی است که حوزه‌های مختلف علوم و عرصه‌های گوناگون صنعت‌ها ازجمله صنایع دفاعی و نظامی جهان امروز را به سرعت دستخوش تحولات قابل توجه می‌کند. هوش مصنوعی که گاهی اوقات هوش ماشینی نیز نامیده می‌شود، به هوشمندی به نمایش در آمده به وسیله ماشین تا در شرایط مختلف، اطلاق می‌شود که در مقابل هوش طبیعی در انسان‌ها قرار دارد. به عبارت دیگر، هوش مصنوعی به سامانه‌هایی گفته می‌شود که می‌توانند واکنش‌هایی مشابه رفتارهای هوشمند انسانی ازجمله، درک شرایط پیچیده، شبیه‌سازی فرایندهای تفکری و شیوه‌های استدلالی انسانی و پاسخ موفق به آن‌ها، یادگیری و توانایی کسب دانش و استدلال برای حل مسایل را داشته باشند.

مسیریابی از مهم‌ترین مسائل هوش مصنوعی که نمود آن در صنعت بازی و علی‌الخصوص بازی‌های جنگ پیدا می‌باشد. مهم‌ترین چالش تا در این حوزه عبارت است از عبور از موانع موجود، پیدا کردن کوتاه‌ترین و سریع‌ترین مسیر از مبدأ به مقصد است. در برخی موارد حل همه چالش‌های بیان شده امکان‌پذیر نمی‌باشد، به‌عنوان مثال امکان دارد کوتاه‌ترین مسیر ارائه‌شده، سریع‌ترین مسیر ممکن نباشد فلذا استراتژی‌های مسیریابی در بازی‌ها به‌عنوان هسته اصلی هر سیستم استفاده می‌شوند که بایستی مورد توجه قرار گیرد.

راحله فرخی و همکاران (۱۳۹۹) در پژوهشی مربوط به بحث مسیریابی سلسله‌مراتبی از شبکه‌های عصبی و الگوریتم ژنتیک استفاده کرده و موردبررسی و تحلیل قرار داده‌اند و نیز ترکیب الگوریتم ژنتیک با روشی ابتکاری را بررسی نموده و با روش‌های دیگر مقایسه کرده و

نتیجه گرفته‌اند که ترکیب مناسب الگوریتم ژنتیک کارایی و اثربخشی مناسبی بر روی الگوریتم‌های مسیریابی سلسه مراتبی دارد.

ین‌لی و همکاران[8] (۲۰۱۲) در پژوهشی به مسیریابی سلسله‌مراتبی بر اساس درخت تصمیم پرداخته‌اند. ابتدا الگوریتم *A را مورد بررسی قرار داده و از آن جاییکه عملکرد مناسبی در محیط‌های بزرگ ندارد به بررسی الگوریتم *HPA پرداخته‌اند که بهبود بسیار مناسبی در تولید گراف نقشه محیط داشته است و سپس به تلفیق الگوریتم *HPA با درخت تصمیم پرداخته که الگوریتم *HPA توسط توسط درخت تصمیم تقسیم می‌شود که نتیجه بهتری به همراه داشته‌است و مسیرهای بهینه بیشتری پیدا می‌شد.

سولینگ‌یانگ (2007) [9]، به بررسی الگوریتم کوتاه‌ترین مسیر سلسله‌مراتبی و مدل تلفیقی آن با الگوریتم *A پرداخته‌اند و نتیجه حاصل شده نشان می‌دهد که پیچیدگی زمانی اجرای الگوریتم *A از نمایی به صورت لگاریتمی بدیل شده است که در مسائل دنیای واقعی نیز می‌تواند کاربرد پذیر باشد.

کرینگ و همکاران[10] (۲۰۱۰) ابتدا به بررسی *HPA پرداخته‌اند و سپس با بهبود و تلفیق آن با الگوریتم‌های جستجوی اول عمق، الگوریتم *DHPA را بررسی نموده‌اند که عملکرد زمانی آن را بهبود داده است و در مصرف منابع سخت‌افزاری نیز بسیار موثر واقع شده است و الگوریتم *SHPA نیز که حدود پنج برابر سریع‌تر از *HPA عمل کرده است ولی گاها مسیر بهینه را به دست نمی‌آورد.

۱. مرور کوتاه الگوریتم‌های مسیریابی[11]

مسیریابی بررسی چگونگی رسیدن از مبدأ به مقصد است. مسئله کوتاه‌ترین مسیر از موارد موردبحث امروزه در علوم کامپیوتر و هوش مصنوعی است. فرض کنید فضای مسئله گراف[12] وزن داری است که هدف جستجوی حداقل مسیر وزنی کل در گراف، بین جفت گره‌ها هست. (یک گراف شامل تعدادی گره به‌عنوان رأس و یال‌هایی است که آن‌ها را به هم متصل می‌کند و یک برچسب گراف دارای یک یا چند توصیف است که به هر رأس متصل شده و آن رأس را از هر رأس دیگر متمایز می‌کند. گراف جستجو به دو دسته جستجوی کورکورانه و جستجوی اکتشافی تقسیم می‌شود. جستجوی کورکورانه (یا جستجوی ناآگاهانه فقط قادر به تمایز حالت

[8] Yan Li
[9] Suling Yang
[10] Kring, A
[11] Pathfinding Algorithms
[12] Graph

غیر هدف از حالت هدف می‌باشد و در جستجوی آگاهانه عامل می‌داند که یک حالت غیر هدف نسبت به گره دیگر، امیدبخش‌تر است.

در ادامه به‌اختصار به بررسی مهم‌ترین الگوریتم‌های مسیریابی پرداخته‌ایم.

۱. الگوریتم جستجوی اول- عمق[۱۳]

اجرای این الگوریتم، نیاز به یک پشته دارد. به این صورت که با هر بار برخورد با یک رأس ملاقات نشده، آن رأس را در پشته گذاشته و هنگام عقب‌گرد رأس را از پشته حذف می‌کنیم؛ بنابراین در تمام طول الگوریتم اولین عنصر پشته[۱۴] در حال بررسی است. وقتی در گراف‌های بزرگ، عملیات جستجو را انجام می‌دهیم که امکان ذخیره کامل آن‌ها به علت محدودیت حافظه وجود ندارد، درصورتی‌که طول مسیر پیمایش شده توسط الگوریتم که از ریشه شروع رئوسی را که تابه‌حال دیده‌ایم، خیلی بزرگ شود، الگوریتم با مشکل مواجه خواهد شد. درواقع این راه‌حل ساده که همیشه کار نمی‌کند. چراکه ممکن است حافظه کافی برای این کار نداشته باشیم. البته این مشکل با محدود ذخیره کنیم کردن عمق جستجو در هر بار اجرای الگوریتم حل می‌شود که درنهایت به الگوریتم جستجوی عمیق کننده تکراری خواهد انجامید.

۲. الگوریتم جستجوی عمقی محدود[۱۵]

شبیه به الگوریتم اول-عمق، می‌باشد و این جستجو مشابه جستجوی اول-عمق عمل می‌کند، اما با تحمیل کردن یک محدودیت حداکثری روی عمق جستجو، از مشکلاتی که مربوط به تمامیت اول-عمق است، جلوگیری می‌کند. حتی اگر جستجو هنوز هم بتواند یک رأس فراتر از آن عمق را گسترش دهد، نمی‌تواند چنین کاری را انجام دهد و درنتیجه، مسیرهای با عمق نامحدود را دنبال نمی‌کند یا به عبارتی در حلقه‌ها گیر نمی‌افتد؛ بنابراین، جستجوی با عمق محدود در صورتی جواب را پیدا خواهد نمود که این جواب در فاصله اولین سطح تا عمق محدود قرار داشته باشد، که این محدودیت، حداقل تمامیت را روی تمامی گراف‌ها تضمین می‌کند.

۳. الگوریتم جستجوی اول- سطح[۱۶]

برای اجرای این الگوریتم از صف استفاده می‌شود. در این روش ابتدا گره ریشه گسترش می‌یابد. اگر ریشه جواب مسئله بود الگوریتم پایان می‌پذیرد در غیر این صورت فرزندان ریشه گسترش می‌یابند. حال به‌صورت سطحی در بین فرزندان ریشه

---

[13] Depth-First Search
[14] Stack
[15] Depth-Limited Search
[16] Breadth-First Search

بررسی می‌کنیم که آیا به جواب رسیده‌ایم یا خیر؟ اگر در این سطح به جواب نرسیده باشیم، تمام گره‌هایی که توسط ریشه تولیدشده‌اند خودشان گسترش می‌یابند و سطح بعدی را تشکیل می‌دهند. در هر سطح جستجو برای یافتن جواب انجام می‌گیرد تا نهایتاً به جواب برسیم. اگر راه‌حلی وجود داشته باشد، جستجوی اول- سطح ضمانت می‌کند که حتماً آن را بیابد (شرط کامل بودن[۱۷]) و اگر چندین راه‌حل وجود داشته باشد، جستجوی اول ضمانت می‌کند که کم‌عمق‌ترین جواب را مشخص کند، این الگوریتم بهینه است و در مقابل زمان و فضایی زیادی اشغال می‌کند.

۴. الگوریتم جستجوی اولین- بهترین

الگوریتم جستجوی اولین- بهترین مانند الگوریتم جستجوی اول- سطح از دو لیست (لیست باز و لیست بسته) برای حالت‌های محدود استفاده می‌کند. در هر مرحله، جستجوی اولین- بهترین، صف را با توجه به عملکرد تابع اکتشافی مرتب می‌کند. این الگوریتم به‌سادگی گره ملاقات نشده را با بهترین مقدار اکتشافی[۱۸] برای ملاقات بعدی انتخاب می‌نماید.

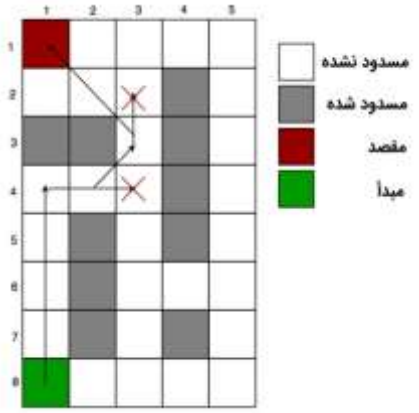
۵. الگوریتم A^*

الگوریتم A^* برخلاف دیگر روش‌های پیمایش گراف، دارای هوشمندی است و همین ویژگی، موجب تمایز آن از دیگر الگوریتم‌های مرسوم می‌شود. این موضوع، همراه با جزئیات بیشتر در ادامه تشریح می‌شود. شایان‌ذکر است که بسیاری از بازی‌ها و نقشه‌های مبتنی بر وب[۱۹]، از الگوریتم A^* برای پیدا کردن کوتاه‌ترین مسیر به‌طور کاملاً مؤثر، استفاده می‌کنند.

یک شبکه مربعی که موانع زیادی در آن وجود دارد مفروض است. همچنین، یک‌خانه به‌عنوان خانه شروع و یک‌خانه به‌عنوان خانه مقصد (هدف) در نظر گرفته می‌شود. کاربر می‌خواهد (در صورت امکان) از خانه شروع آغاز به کار کند و با سریع‌ترین حالت ممکن به خانه مقصد برسد. در اینجا، الگوریتم جستجوی A^* نقش‌آفرین می‌شود. فرض کنید g هزینه حرکت از نقطه آغاز به یک مربع خاص در شبکه، با دنبال کردن مسیری که برای رسیدن به آن تولیدشده است و h هزینه تخمین زده‌شده برای حرکت از یک‌خانه داده‌شده در شبکه به مقصد نهایی است. از h معمولاً با عنوان

[17] Completeness
[18] Heuristic
[19] Web-Based Maps

هیوریستیک یاد می‌شود. اکتشاف چیزی به‌جز نوعی حدس هوشمندانه نیست. کاربر واقع فاصله واقعی را تا هنگام یافتن مسیر نمی‌داند، زیرا هر مانعی (دیوار، رود و سایر موانع) ممکن است در مسیر باشد. عملکرد الگوریتم A^* به این صورت است که در هر گام، گره را متناسب با مقدار f که پارامتری مساوی با مجموع دو پارامتر دیگر g و h است انتخاب می‌کند. در هر گام، گره/خانه‌ای که دارای کمترین مقدار f است را انتخاب و آن گره/خانه را پردازش می‌کند.



تصویر (۱) اجرای الگوریتم A^*

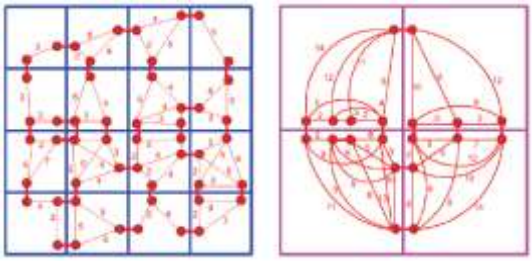
با توجه به تصویر ۱ الگوریتم A^* در هر گام، هوشمندانه‌ترین انتخاب را انجام می‌دهد. بنابراین می‌توان مشاهده کرد که الگوریتم از خانه (۴,۲) به (۳,۳) و می‌رود، نه به (۴,۳) (با علامت ضربدر نشان داده‌شده است). به‌طور مشابه، الگوریتم از (۳,۳) به (۲,۲) می‌رود، نه (۲,۳) (با علامت ضربدر نشان داده‌شده است).

۶. الگوریتم A^* سلسله‌مراتبی[۲۰]

الگوریتم A^* راه‌حل مناسب و کارآمدی جهت حل مسائل مسیریابی می‌باشد ولی بایستی این نکته را موردتوجه قرارداد که A^* در مسائلی که فضای نسبتا بزرگ دارند، فضا و زمان محاسبه عظیمی را خواهد داشت بنابراین الگوریتم HPA^* محدودیت‌های آن را برطرف کرده است و ایده‌ی اصلی آن بر اساس روش تقسیم و حل بر روش A^* می‌باشد. فضای بزرگ محاسبی به فضاهای کوچک‌تر تقسیم‌شده و درون هر یک الگوریتم A^* اجرا می‌شود، همچنین وجود سطح‌بندی در سلسله موردبحث، می‌تواند

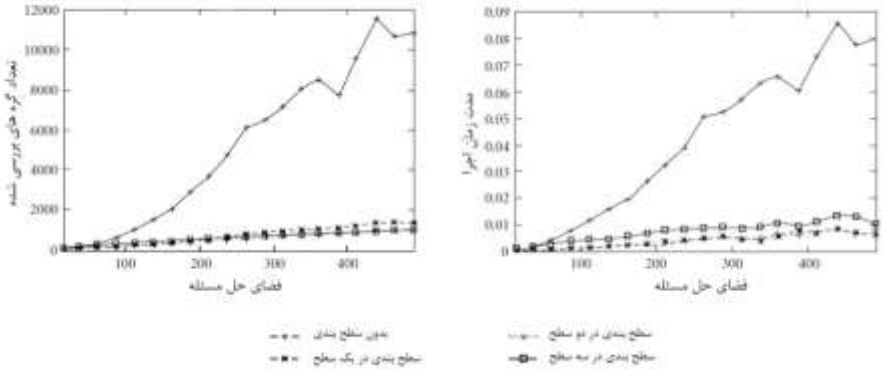
[20] HPA^*

جستجوی ما برای انتخاب بهترین مسیر را مخصوصا در محیط با اندازه بزرگ‌تر بهبود ببخشد. در یک گراف چند سطحی، گره‌ها و یال‌ها برچسب‌هایی مبنی سطح دارند. روال به این صورت است که ما هر سطح جدید را در سطح بالاتر سطح فعلی قرار می‌دهیم پس بدین‌صورت برای ارائه مسیر از S به G ما فقط سطوحی را موردبررسی قرار می‌دهیم که در بالاترین سطح قرار دارند. با فرض وجود مسیری از S به G، نتیجه جستجوی انجام‌شده توالی از گره‌ها خواهد بود که بهبود مسیر به‌صورت تکرارشونده، باعث ارائه مسیر مناسب و بهینه خواهد شد.



تصویر (۲) گراف سطح‌بندی شده

اگر نتایج استفاده از سلسله‌مراتب را با وضعیت بدون آن (یک سلسله‌مراتب) را بخواهیم مقایسه کنیم، نمودارهای زیر مفروض است:



نمودار (۱) تأثیر سطح‌بندی (سلسله‌مراتب) در حل مسئله

با توجه به نتایج به‌دست‌آمده، سطح‌بندی اضافه شده بر روش A^* (وجود سلسله‌مراتب)، در مواردی که فضای مسئله برای ما نامشخص و یا ابعاد بزرگی دارد، سبب بهبود فضای حل و مدت زمان اجرا و در کل سبب بهبود آن گشته است.

۲. بررسی مسیریابی مبتنی بر هوش مصنوعی

یادگیری تقویتی یکی از انواع مدل‌های یادگیری در هوش مصنوعی و حوزه یادگیری ماشین[۲۱] است که در آن‌یک عامل[۲۲] می‌آموزد که در یک محیط غیرقطعی[۲۳] و پیچیده به هدف برسد. با ذکر مثالی به توضیح آن می‌پردازیم:

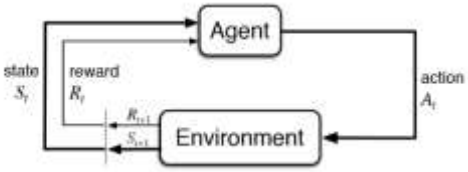
تصور کنیم که ما یک شرکت فروش داریم و یک کارمند استخدام می‌کنیم. در حال حاضر کارمند می‌تواند اقدامات مختلفی را انجام دهد، ازجمله تماس با مشتریان بالقوه و اطمینان از فروش که درازای آن کمیسیون دریافت می‌کند.

- این کارمند را عامل در نظر می‌گیریم.
- این عامل در شرکت کار می‌کند، پس شرکت را به‌عنوان محیط در نظر می‌گیریم.
- عامل در یک وضعیت[۲۴] قرار دارد و هر بار که عملیاتی را در محیط انجام می‌دهد، وضعیت عامل عوض می‌شود و به وضعیت جدیدی وارد می‌شود. نتیجه‌ی هر اقدام[۲۵] پاداش مثبت یا منفی به عامل می‌دهد.

برای مثال، اگر کارمند فروش را به اتمام برساند، کارمزد دریافت می‌کند و اگر اقدامات نادرستی انجام دهد و فروش را با موفقیت به اتمام نرساند، این کارمزد به او تعلق نمی‌گیرد. در این مثال، عامل به‌طور مداوم در حال یادگیری است. در طول این فرایند عامل اقداماتی را یاد می‌گیرد که به پاداش منتهی می‌شوند و رفته‌رفته عملکرد خود را بهتر می‌کند تا درنهایت به یک هدف نهایی برسد. پس اگر بخواهیم به‌طور خلاصه بگوییم، در یادگیری تقویتی، یک عامل در یک محیط وجود دارد که می‌تواند اقداماتی را انجام دهد، درست مانند ما انسان‌ها. هر اقدامی نتیجه‌ای در پی دارد. عامل در تلاش است تا پاداش‌های دریافتی خود را به حداکثر برساند. نتیجه‌ی هر اقدام یا یک پاداش مثبت یا منفی است.

---

[21] Machine Learning
[22] Agent
[23] Uncertain
[24] State
[25] Action



تصویر (۳) نمایش یادگیری تقویتی و تعاملات عامل با محیط

باگذشت زمان عامل از این نتایج یاد می‌گیرد تا اقدامات خود را بهبود دهد؛ ازاین‌رو، می‌توان گفت یادگیری تقویتی یادگیری مبتنی بر بازخورد است.

با توجه به موارد عنوان‌شده، حال به استفاده از الگوریتم Q-Learning بصورت سلسله‌مراتبی برای حل مسئله مسیریابی می‌پردازیم.

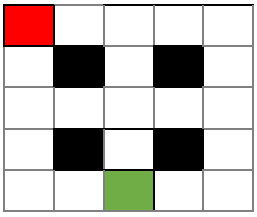
- الگوریتم Q-Learning

Q-Learning یک الگوریتم یادگیری تقویتی ارزش محور[۲۶] و بدون مدل[۲۷] است که برای پیدا کردن سیاست انتخاب عمل بهینه[۲۸]، از تابع Q استفاده می‌کند. اگر تابع Q را به‌صورت زیر تعریف کنیم:

$$NewQ(s,a) = Q(s,a) + \propto [R(s,a) + \gamma\, maxQ'(s',a') - Q(s,a)\,]$$

که در آن $Q(s,a)$ مقدار ارزش برای حالت s و اکشن a می‌باشد و مقدار α برابر با نرخ یادگیری[۲۹] و γ برابر با نرخ تخفیف می‌باشد.

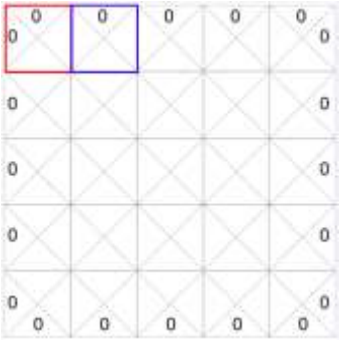
فرض کنید نقشه‌ای ساده (تصویر ۴) داریم به شکل زیر که عامل می‌خواهد از خانه قرمز به خانه سبز برسد و خانه‌های سیاه بیانگر موانع است:



تصویر(۴) نقشه مثالی

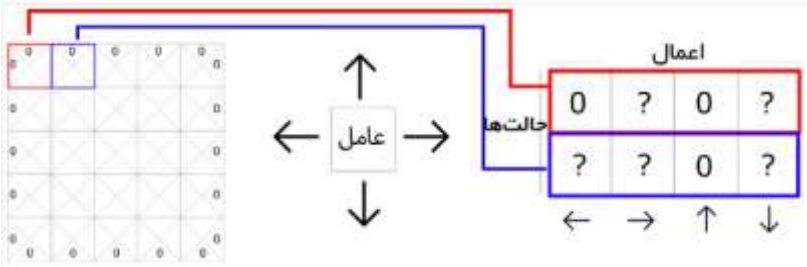
[26] Value-Based
[27] Model-Free
[28] Optimal Policy
[29] Learning Rate

اگر جدولی بسازیم به‌نحوی‌که که در آن بیشینه پاداش آینده مورد انتظار برای هر عمل در هر حالت محاسبه شود. با بهره‌گیری از این جدول، می‌توان مشخص نمود که بهترین عمل ممکن برای هر حالت چیست. در هر سلول (کاشی) چهار عمل ممکن است، این چهار عمل، پایین، بالا، چپ و راست رفتن هستند.



تصویر (۵) جدول حرکات ممکن

به‌منظور انجام محاسبات، این شبکه به یک جدول تبدیل می‌شود. جدول ساخته‌شده، Q-table[۳۰] نامیده می‌شود. ستون‌ها چهار عمل (چپ، راست، بالا و پایین) هستند. سطرها حالت‌ها هستند. مقدار هر سلول بیشینه پاداش آینده مورد انتظار برای حالت داده‌شده و عمل است.



تصویر(۶) – اعمال برای هر سلول

هر امتیاز Q-table، پاداش آینده مورد انتظار بیشینه‌ای است که عامل در صورت عمل کردن با بهترین سیاست داده‌شده، به آن دست می‌یابد. بنابراین عامل با انتخاب بالاترین امتیاز در هر سطر برای هر حالت، می‌داند که بهترین عمل قابل انجام چیست.

---

[۳۰] Q سرنام Quality است و مربوط به عبارت Quality Of The Action

حال بایستی بیان کنیم، ارزش برای هر جز Q-Table از تابع ارزش عمل به دست می‌آید. تابع ارزش عمل[۳۱] دو ورودی حالت و عمل را دریافت می‌کند. سپس، پاداش آینده مورد انتظار برای آن حالت و عمل را بازمی‌گرداند.

پس مراحل انجام به شرح زیر می‌باشد:

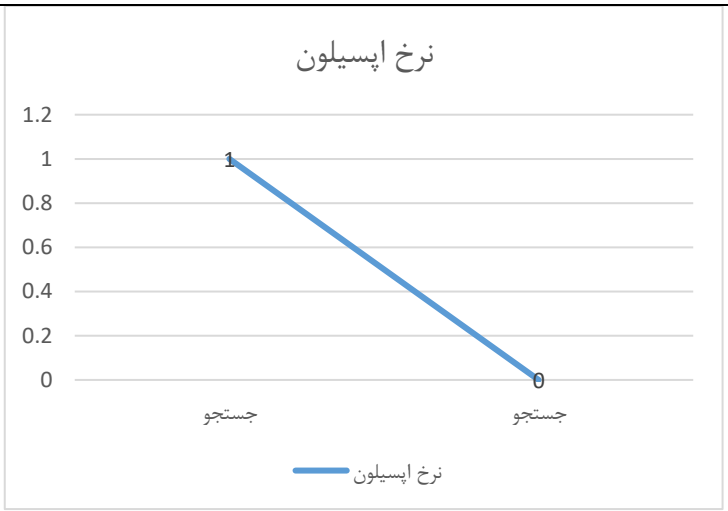
۱- مقداردهی اولیه Q-value

Q-table با m ستون (m = تعداد اعمال) و n سطر (n = تعداد حالت‌ها) ساخته شد. اکنون مقداردهی اولیه با صفر انجام می‌شود.

۲- انتخاب یک عمل

انتخاب یک عمل a در وضعیت کنونی s برمبنای ارزش کنونی Q-value تخمین زده‌شده. اما، اگر همه مقدارهای Q برابر با صفر باشند، می‌توان از استراتژی حریصانه اپسیلون[۳۲] استفاده کرد که نرخ جست‌وجوی اپسیلون تعیین‌شده و در آغاز برابر با یک قرارداده می‌شود. اپسیلون، نرخ گام‌هایی است که عامل به‌طور تصادفی انجام می‌دهد. در آغاز، این مقدار باید بیشترین ارزش ممکن باشد، زیرا عامل هیچ‌چیز درباره ارزش‌های موجود در Q-table نمی‌داند. این یعنی نیاز به جست‌وجوهای زیاد و انتخاب تصادفی اعمال توسط او است. یک عدد تصادفی تولید می‌شود. اگر این عدد بزرگ‌تر از اپسیلون بود، استخراج انجام می‌شود (این یعنی ازآنچه در حال حاضر مشخص است برای انتخاب بهترین عمل در هر گام استفاده می‌شود). در غیر این صورت، جست‌وجو صورت می‌پذیرد. ایده آن است که باید یک اپسیلون بزرگ در ابتدای راه آموزش تابع Q وجود داشته باشد و به‌تدریج و پس‌ازآنکه اطمینان عامل در تخمین Q-value تا افزایش یافت، مقدار آن کاهش پیدا کند.

[31] Q-Function
[32] Epsilon Greedy Strategy



نمودار (۲) - رابطه نرخ اپسیلون و جستجو

۳- انجام عمل a و مشاهده حالت خروجی s و پاداش

اکنون باید تابع Q(s,a) به‌روزرسانی شود. عمل a که در گام ۳ انتخاب‌شده بود انجام می‌شود و اجرای این عمل حالت s و پاداش r را در خروجی می‌دهد (همان‌طور که در فرآیند یادگیری تقویتی در قسمت اول این مطلب بیان شد). سپس، برای به‌روزرسانی Q(s,a)، از معادله بلمن[۳۳] استفاده می‌شود.

۴- تکرار مراحل ۲ و ۳مراحل ۲ و ۳ تا زمانی که به مقدار حداکثر تکرار برسد، تکرار خواهد شد.

آنچه در طی مراحل فوق بیان کردیم، برای یادگیری تقویتی و الگوریتم آموزش Q بود. اگر کمی مراحل مسیریابی را مرور کنیم، آنچه سبب بهبود اجرا مسیریابی می‌شود، تبدیل کل فضای مسئله به فضاهای کوچک‌تر بود. حال اجرای هوش مصنوعی بررسی‌شده به این صورت است که تقسیم‌بندی فضا انجام‌شده و الگوریتم Q یک‌بار در داخل فضاهای کوچک (خوشه‌ها) و یک‌بار هم به‌صورت کلی و بین خوشه‌ها انجام می‌پذیرد. می‌توان ضرایب مناسب و همچنین مقادیر اولیه Q برای هرکدام را به‌صورت تجربی و با آزمودن‌های مکرر به دست آورد.

۳. مسیریابی سلسله‌مراتبی مبتنی بر هوش مصنوعی

الگوریتم یادگیری Q به دلیل انعطاف‌پذیری و بهینه بودن و همچنین انعطاف محاسبه در نقشه‌های تصادفی و بازی‌های بلادرنگ[۳۴] مورد بسیار مناسبی جهت استفاده می‌باشد و

[33] Bellman Equation
[34] Real-Time Games

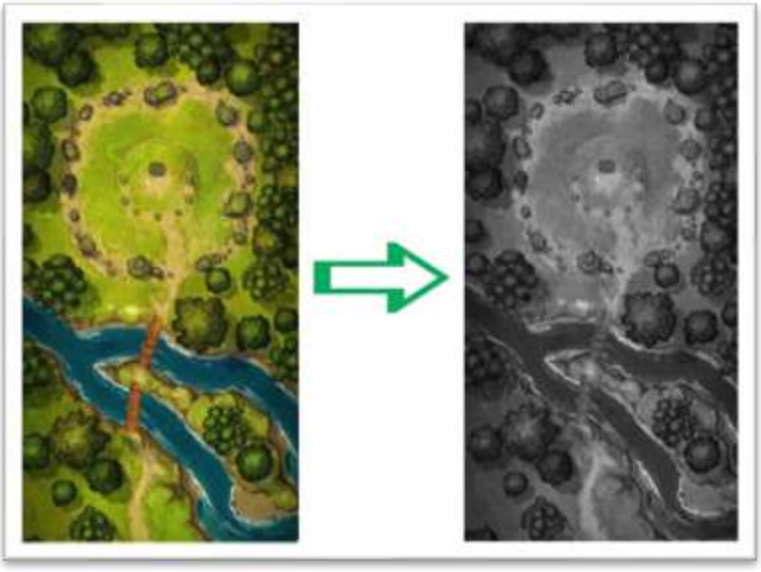
همچنین در محیط‌هایی که به‌صورت پویا[۳۵] تغییر می‌کنند و همچنین اطلاعات خاصی از آن محیط نیز در دسترس نیست، عملکرد مناسبی از خود نشان داده است، همچنین همانگونه که پیش‌تر پرداختیم، وجود سلسله‌مراتب در مسیریابی، سبب بهبود فضای جستجو و محاسبات می‌گردد فلذا در ادامه به بررسی الگوریتم یادگیری Q به صورت سلسله‌مراتبی می‌پردازیم.

ورودی‌های سیستم مسیریاب طراحی‌شده عبارت است از نقشه‌زمینی و مختصات نقاط شروع و هدف و نیز خروجی، مسیر بهینه است که در نقشه ارائه‌شده، معین است.

۱،۵. فاز پیش‌پردازش[۳۶]

پردازش نقشه ورودی به سیاه و سفید

با توجه به امکان ورودی‌های مختلف (نقشه‌های مختلف)، بایستی نقشه به صورتی تبدیل شود تا برای هوش مصنوعی، تفاوتی در روند اجرا ایجاد نشود و یا عملکرد آن دچار اخلال نگردد پس اولین پردازشی که بر روی ورودی به این منظور انجام می‌گردد، تبدیل نقشه‌ی ورودی به نقشه‌ای بصورت سیاه و سفید است. برای تبدیل تصویر، ابتدا ماتریس تصویر خاکستری را با توجه به میانگین سه رنگ rgb[37] از ماتریس تصویر اصلی استخراج می‌شود.

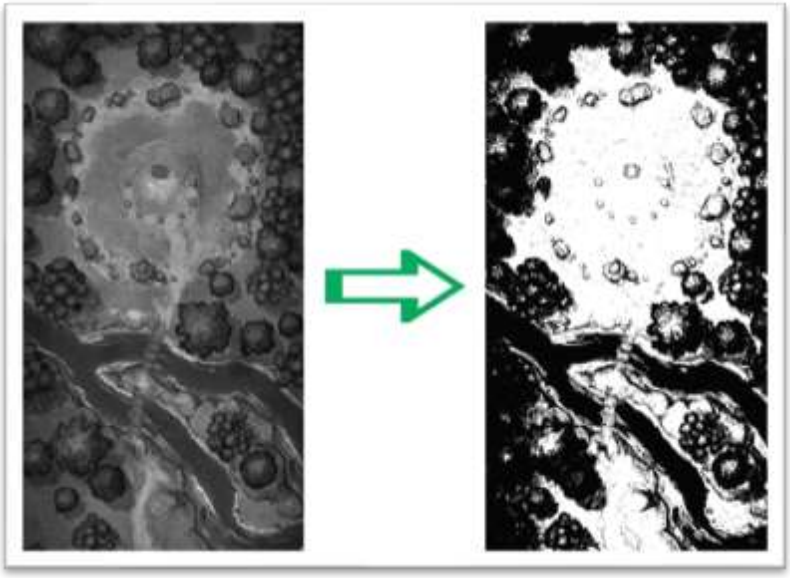


تصویر(۷) – تبدیل نقشه ورودی بصورت سیاه و سفید

[35] Dynamic
[36] Preprocess
[37] Red – Green - Blue

ایجاد نقشه شارپ

خروجی پردازش قبلی، به عنوان ورودی این پردازش استفاده شده. همچنین تولید این نقشه جهت تولید ماتریس نقشه که پردازش‌های بعدی بر روی آن انجام خواهد گرفت، بسیار مورد استفاده است و نیز به نوعی فضای پردازش هوش مصنوعی را به‌صورت ویژه‌ای کاهش خواهد داد و نقشه بصورت شارپ جهت امکان برچسب‌گذاری نقاط، تولید می‌گردد. برای تولید این نقشه، ماتریس خاکستری تولید شده در مرحله قبل و تعریف یک حد آستانه[۳۸]، بالاتر از آستانه را رنگ سیاه و پایین‌تر از آن را سفید در نظر می‌گیریم. در این مرحله می‌توان نویزها را با روش‌ها و یا ابزارهای ابتکاری حذف نمود.



تصویر (۸) – تولید تصویر شارپ

کاهش ابعاد

مشکل افزایش ناخواسته ابعاد، یک مشکل فراگیر است. این مشکل، ناشی از تثبیت اندازه‌گیری یا ذخیره‌سازی داده هاست که از گذشته وجود داشته است. این مسئله، مشکل جدیدی نیست، ولی اخیراً به دلیل افزایش داده‌ها، اهمیت بیشتری پیدا کرده است. اساساً، کاهش ابعاد به فرآیند تبدیل مجموعه‌ای از داده‌ها اطلاق می‌شود. این داده بایستی اطلاعات مشابه را به طور خلاصه منتقل می‌کند. حال در نقشه‌ی شارپ تولید شده، به جهت کاهش حجم فضای حل

[38] Threshold

مسئله، بهتر است ابعاد نقشه کاهش داده شود تا محاسبات کمتر و سریع‌تر صورت پذیرد، پس با در نظر گرفتن مقدار عرض و محاسبه طول آن از ابعاد نقشه، به نحوی که نسبت آن بهم نریزد، ابعاد آن کاهش داده می‌شود.

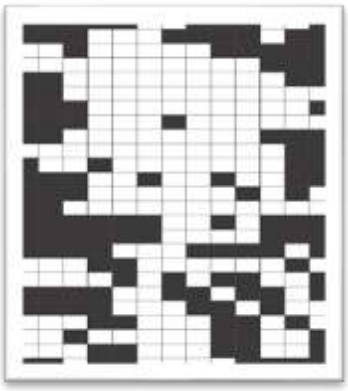
تبدیل سطح به جدول و خوشه‌بندی

کل سطح به جدول بزرگ با اندازه سلول‌های مساوی تقسیم خواهد شد (تصویر ۹) و تمام سلول‌ها از تمام سلول‌های قابل‌دسترس اسکن شده و مقادیر ویژه هر سلول برای تعیین هزینه آن برای استفاده در الگوریتم آموزش Q موردبررسی قرار می‌گیرد. هر سلول را می‌توان به‌عنوان یک گره در نظر گرفت و تمام اطلاعات سلول برای پردازش بیشتر در یک آرایه قرار خواهد گرفت. و همچنین بایستی هر سلول را بایستی اسکن نموده تا همسایگان موردبررسی قرار گیرد. (تصویر ۱۰) امکان دارد برای دسترسی به تمام همسایگان سلول، امکان‌پذیر نباشد که امری طبیعی است. (به دلیل وجود مراتع، پستی‌وبلندی‌ها، عوارض طبیعی یا...) و همچنین هزینه دسترسی نیز بایستی مدنظر باشد.

تصویر (۹) تبدیل نقشه به جدول

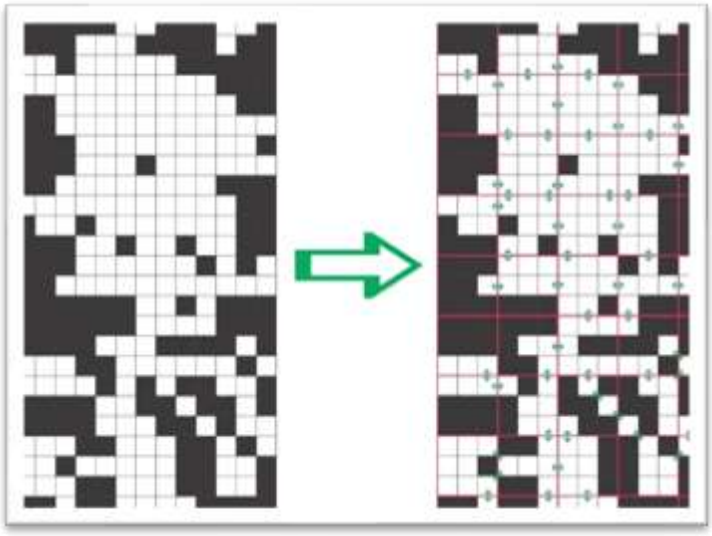
تصویر (۱۰) اسکن همسایه‌های هر سلول

سپس فرض کنید کل محیط به‌صورت جدول (تصویر ۱۱) درآمدهاست که نقاط غیرقابل‌دسترس (به عنوان مثال رودخانه، درختان، سنگ‌ها و...) به‌صورت رنگ سیاه درآمده‌اند.



تصویر (۱۱) تبدیل کل محیط به جدول

جدول فوق را به جدول‌های کوچک‌تر (خوشه کوچک‌تر) تقسیم می‌کنیم و امتداد مرز مشترک دو خوشه که مانعی بینشان وجود ندارد، به‌عنوان ورودی با امتیاز حداکثر در نظر گرفته می‌شود و ورودی‌ها به هر خوشه را بررسی می‌کنیم. در تصویر ۱۲ خوشه‌بندی انجام‌شده و ارتباط بین گره‌های درون خوشه و بیرون نشان داده‌شده است.



تصویر (۱۲)خوشه‌بندی و دسترسی بین خوشه‌ها

ایجاد گراف اتصال[۳۹]

از گراف برای مشخص نمودن مبدأ و مقصد (گره‌ها مبدأ و مقصد و یال بیانگر مسیر) استفاده می‌کنیم. که امکان دارد دنبال مسیر داخل خوشه‌ای (یال داخلی) و یا بین خوشه‌ای (یال خارجی) باشیم.

۲,۵. فاز مسیریابی

اضافه نمودن نقطه شروع و پایان و اجرای آموزش Q

گره شروع (S) و گره پایان (G) را اضافه می‌کنیم و خوشه‌هایی که S و G در آن قرار دارند را بررسی نموده و نقاط اتصال از هرکدام از خوشه‌ها به خوشه‌های دیگر را به‌واسطه‌ی گره‌های مرزی بررسی می‌کنیم.

الگوریتم یادگیری Q با توجه به آنچه در قسمت‌های پیشین شرح داده شد، برای پیدا کردن مسیر از S به G به کار گرفته می‌شود که آن مسیر انتزاعی با حرکت از S به مرز خوشه S، حرکت از G به خوشه G و پارامترهای اولیه را به شرح زیر تعیین نموده‌ایم:

جدول (۱) پارامترهای اولیه مسیریابی Q

نام پارامتر	شرح پارامتر	مقدار اولیه
پاداش حداقل	پاداش حداقل که برابر پاداش نقاطی است که بیشترین ارتفاع را در نقشه دارند.	-۵
پاداش حرکت	پاداش حرکت کردن از هر نقطه در داخل نقشه	-۱
پاداش اتصال	پاداش اتصال خوشه‌ها به یکدیگر	۵
حداکثر اتصال خوشه	حداکثر تعداد اتصال‌های ممکن بین دو خوشه در نقشه	۲۰
نرخ یادگیری	نرخ یادگیری	۰.۹۹
حداکثر اپیزود یادگیری	حداکثر اپیزود یادگیری	۲۰۰۰
خوشه در سطر	تعداد تقسیم‌بندی نقشه به خوشه‌ها در سطر	۴۰
خوشه در ستون	تعداد تقسیم‌بندی نقشه به خوشه‌ها در ستون	۲۰
مقدار کاهش	تعداد پیکسل تصویر کاهش داده شده در ستون	۱۰۰

[39] Connection Graph

لازم به ذکر است پارامترهای ذکر شده، صرفا با بررسی و تست و انجام آزمایشات متعدد تنظیم شده‌اند و قابل تغییر باتوجه می‌باشند.

آموزش Q یک بار در داخل خوشه‌ها و بار دیگر بین خوشه‌ها انجام می‌شود و مسیر گره‌هایی که بایستی از مبدأ به مقصد طی شوند، با رنگ سبز نمایش داده‌شده‌اند و طبق قانون نامساوی مثلثی، بهترین و نزدیک‌ترین مسیر، مسیری است که به‌صورت خطی باشد و هزینه مناسب‌تری در برداشته باشد.

برای این منظور از جدول Q-Value بهره می‌بریم که برای هر وضعیت[40]، عمل‌های ممکن (حرکت به خانه‌های اطراف که عبارت از حرکت به بالا، پایین، چپ و راست) مشخص شده است و مقدار ارزش تمامی آن‌ها برابر صفر است، پس در نتیجه جدولی به شکل زیر در ابتدا داریم:

جدول (۲) پارامترهای اولیه مسیریابی Q

وضعیت ها	عمل‌های ممکن			
	بالا	پایین	چپ	راست
(1,1)	0	0	0	0
(1,2)	0	0	0	0
…	…	…	…	…
(m,n)	0	0	0	0

فرض کنیم در نقشه تصویر ۱۱ در اولین خانه (خانه عرض ۵ و ارتفاع ۷) حرکت به سمت راست را مورد بررسی قرار میدهیم و در صورتی که به سمت راست حرکت کنیم، صرفا حرکت کرده‌ایم و به هدف نرسیده‌ایم، پس مقدار Q جدید برای آن را از طریق رابطه‌ی Q محاسبه می‌کنیم، اگر فرض کنیم رابطه به صورت زیر است:

$$NewQ(s,a) = Q(s,a) + \propto [R(s,a) + \gamma\, maxQ'(s',a') - Q(s,a)\,]$$

که در آن، $Q(s,a)$ برابر مقدار Q کنونی، $\propto$ برابر با نرخ یادگیری، $R(s,a)$ پاداش برای انجام آن عمل در آن حالت، γ برابر با نرخ تخفیف (که معمولا یک در نظر گرفته می‌شود)، $maxQ'(s',a')$ برابر بیشینه پاداش آتی پیش‌بینی شده با توجه به s' جدید است. (به این

[40] State

معنی که در پس از حرکت به راست، چه حالت‌های دیگری پیش رو داریم و بیشترین مقدار Q برای کدام حرکت است.

پس برای حالت شروع و به سمت راست داریم:

$$NewQ(start,right) = Q(start,right) + \propto [\Delta Q(start,right)]$$

و مقدار ΔQ برابر است با:

$$\Delta Q(start,right) = R(start,right) + \gamma \max Q'(s',a') - Q(start,right)$$

و مقدار $\max Q'(s',a')$ به این مفهوم است در صورتی که به سمت راست حرکت کنیم، در حرکت‌های بعدی، چه مقدار Qهایی داریم و بیشترین آن‌ها را انتخاب میکنیم. در وضعیت موجود، مقدار تمامی حالت‌های بعدی برابر با صفر است و همچنین پاداش حرکت در نقشه را ۱- تعریف کرده‌ایم و نرخ یادگیری را ۰. ۹۹، داریم:

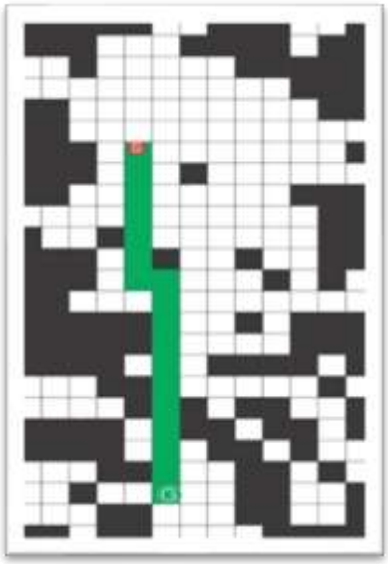
$$\Delta Q(start,right) = -1 + 0.99 * 0 - 0 = -1$$

لازم به توضیح است مقدار یک برای نرخ یادگیری، باعث حداکثر یادگیری خواهد شد و هر چقدر به صفر نزدیک‌تر باشد، یادگیری، کمتر خواهد بود. در ادامه با بروزرسانی جدول Q داریم:

جدول (۳) بروزرسانی مقدار برای خانه (5,7)

وضعیت ها	عمل‌های ممکن			
	بالا	پایین	چپ	راست
(1,1)	0	0	0	0
(1,2)	0	0	0	0
…	…	…	…	…
(5,7)	0	0	0	-1
…	…	…	…	…
(m,n)	0	0	0	0

و این محاسبه را تا جایی ادامه می‌دهیم که مقادیر جدول Q تغییر پیدا نکند و به این معنی است که یادگیری متوقف شده است و به پایان رسیده است. در تصویر ۱۳ نتیجه نهایی مسیریابی قابل مشاهده می‌باشد.



تصویر (۱۳) مسیریابی انجام‌شده

۳،۵. محاسبه سرعت حرکت و مدت زمان

پس از انجام روند مسیریابی و معین نمودن مسیر، حال می‌توان سرعت و مدت زمان طی مسیر توسط یگان‌های مختلف را محاسبه نمود. با توجه به اینکه هر یگان رزمی دارای متوسط سرعت حرکت معینی می‌باشد[41] و همچنین ضریب اصطکاک سطح برای هر آب و هوایی معین است، می‌توان سرعت حرکت و همچنین زمان رسیدن هر یگان به مقصد را محاسبه نمود. به عنوان مثال سرعت حرکت یگان پیاده طی تحقیقات صورت گرفته، به صورت متوسط عدد ۴ کیلومتر بر ساعت معین شده است، همچنین مسافتی که بایستی طی شود در مسیریابی محاسبه گردیده است و لذا زمان رسیدن به مقصد قابل محاسبه می‌باشد.

در صورتی که شرایط آب و هوایی، بصورت بارانی یا برفی باشد، قاعدتا امکان طی مسیر با همان شرایط امکان پذیر نیست فلذا یک ضریب اصطکاک نیز در محاسبه سرعت دخیل می‌باشد که

[41] https://hoi4.paradoxwikis.com/land_units

برای آب و هوای مناسب (مثلا آفتابی)، یک در نظر گرفته می‌شود و برای سایر شرایط آب و هوایی، کسری از آن خواهد بود.

نتیجه‌گیری

صنعت مدرن بازی‌های رایانه‌ای و علی‌الخصوص بازی‌های استراتژیک، هرساله با توجه به محیط‌ها و تعداد واحدهای موجود در صنعت، دچار تغییر شده و بزرگ‌تر و پیچیده‌تر می‌شوند. امروزه مطالعه مسیریابی با زمینه‌های دیگر ازجمله منطق، برنامه‌نویسی بازی، مدیریت عملیات، طراحی و تجزیه‌وتحلیل سیستم، مدیریت پروژه، شبکه و خط تولید و یادگیری و تفکر به‌مانند یک انسان در هوش مصنوعی همراه است.

چالش مسیریابی مولفه‌ای اساسی در محیط‌های بازی است. در پژوهش‌های پیشین انجام‌شده در این خصوص، به بهینه‌سازی مسیریابی A^* پرداخته‌اند و سعی نموده‌اند پیچیدگی فضای حل مسئله را با استفاده از روش‌های ابتکاری بهبود دهند و در پژوهشی به بهینه‌سازی الگوریتم با افزودن سلسه‌مراتب پرداخته‌شده است که با بررسی نتایج متوجه شدیم وجود سلسله‌مراتب هم به لحاظ اجرایی و هم نتیجه، تأثیر مناسبی در این الگوریتم داشته است و در سیستمی که موردبررسی قرار داده‌ایم و پیاده‌سازی کرده‌ایم، یکی از الگوریتم‌های مطرح هوش مصنوعی (آموزش Q) با تلفیق آن با ایده‌ی مسیریابی سلسله‌مراتبی A^* است و رویکرد سلسله‌مراتب در الگوریتم Q-learning موردبحث و اجرا قرارگرفته است. درواقع الگوریتم پیشنهادشده با کاهش فضای جستجو و تعداد عملیات مسیریابی، سرعت اجرای بالاتری دارد و به لحاظ دقت عملکرد نیز عملکرد قابل قبولی ارائه کرده است. جهت بهبود عملکرد در کارهای آینده، می‌توان Q-Learning را با در نظر گرفتن نحوه توزیع اشیاء و موانع بررسی نمود، بدین‌صورت که بهینه‌سازی برای کل نقشه انجام شود و سپس الگوریتم Q-Learning بر روی اجرا شود. همچنین هموارسازی مسیر نیز جزو مواردی ست که می‌توان جهت بهبود مسیر و کیفیت آن انجام داد، در این صورت شکل ظاهری مسیر کیفیت مناسبی خواهد داشت و هنگام انتخاب اهداف فرعی و تشکیل مسیرهای جزئی، این ضعف برطرف خواهد شد. همچنین موارد دیگری که می‌تواند سبب بهبود مسیریابی گردد، استفاده از یادگیری عمیق[۴۲] جهت شناسایی و تقسیم اشیاء محیط جهت بهبود روند مسیریابی می‌باشد. طراحی سیستم‌های تشخیص اشیا مبتنی بر الگوریتم‌های بینایی ماشین[۴۳]، کمک می‌کند سیستم در مقابل تغییرات گوناگونی در تصویر نظیر چرخش، انتقال و مقیاس، مقاوم و کارا باشد. الگوریتم‌های بینایی ماشین به شاخه‌های

[42] Deep Learning
[43] Machine Vision

متعددی تقسیم‌شده و راه‌های گوناگونی برای آن ارائه‌شده است که هرکدام دارای مزایا و معایبی می‌باشد و بایستی روش مناسب با توجه به کاربرد مسئله مسیریابی انتخاب شود. همچنین لازم به ذکر است کاربرد این الگوریتم‌ها در مسائل نظامی به پیشرفت آن کمک شایانی کرده است.

منابع

[1]فرخی، راحله. (۱۳۹۹). حل مدل مسیریابی وسایل حمل مواد خطرناک مبتنی بر کاهش ریسک با استفاده از رویکرد سلسله‌مراتبی، دانشگاه علم و صنعت ایران.

[2] K. D. Forbus, J. V. Mahoney, and K. Dill, (2002). How qualitative spatial reasoning can improve strategy game AIs, IEEE Intelligent Systems, vol. 17, no. 4, pp. 25–30.

[3] E. Bethke, (2003). Game Development and Production, Wordware, Plano, Tex, USA.

[4] K. Forbus, (1996). Qualitative reasoning, in CRC Handbook of Computer Science and Engineering, pp. 715–733, CRC Press, Boca Raton, Fla, USA.

[5] A. G. Cohn, (1997). Qualitative spatial representation and reasoning techniques, in Proceedings of the 21st Annual German Conference on Artificial Intelligence: Advances in Artificial Intelligence (KI ’97). vol. 1303 of Lecture Notes in Computer Science, pp. 1–30, Springer, Freiburg, Germany.

[6] A. Botea, M. Muller, and J. Schaeffer, (2004). Near optimal hierarchi- cal path-finding, Journal of Game Development, vol. 1, no. 1, pp. 7–28.

[7] S. Rabin. (2000). A∗ speed optimizations, in Game Programming Gems, M. DeLoura, Ed. , pp. 272–287, Charles River Media, Rockland, Mass, USA.

[8] R. S. Sutton and A. G. Barto, (1998). Reinforcement Learning: An Introduction, MIT Press, Cambridge, Mass, USA.

[9] I. Millington. (2006). Artificial Intelligence for Games, Morgan Kauf- mann, San Mateo, Calif, USA.

[10] D, Michie, D. J. Spiegelhalter, C. C. Taylor. (1994). Machine Learning, Neural and Statistical Classification, Prentice Hall, Upper Saddle River, NJ, USA.