روشی نوین برای محاصره هدف توسط یک سیستم چندعاملی مبتنی بر رویداد در بازی‌های رایانه‌ای

بهروز مینایی بیدگلی[۱] ، علیرضا صحاف نائینی[۲] ، عماد آقاجانی[۳] ، مهرداد آشتیانی [۴]

[۱] استاد دانشگاه علم و صنعت ایران ، مدیر عامل بنیاد ملی بازیهای رایانه‌ای ، B_minaei@iust.ac.ir

[۲] دانشجوی کارشناسی دانشگاه علم و صنعت ایران ، آزمایشگاه تحقیقاتی بازی‌سازی دانشگاه علم و صنعت ایران ، alisahaf70@comp.iust.ac.ir

[۳] دانشجوی کارشناسی دانشگاه علم و صنعت ایران ، آزمایشگاه تحقیقاتی بازی‌سازی دانشگاه علم و صنعت ایران ، Emad_Aghajani@comp.iust.ac.ir

[۴] دانشجوی دکتری دانشگاه علم و صنعت ایران ، آزمایشگاه تحقیقاتی بازی‌سازی دانشگاه علم و صنعت ایران ، m_ashtiani@comp.iust.ac.ir

چکیده

صنعت بازی‌های رایانه‌ای امروزه به یکی از پرطرفدارترین و سرگرم‌کننده‌ترین صنایع موجود در جهان تبدیل شده است. یکی از اصلی‌ترین عوامل ایجاد هیجان و جذابیت در بازی‌های رایانه‌ای عنصر هوش مصنوعی بازی است. سعی سازندگان بازی‌های رایانه‌ای استفاده هرچه بیشتر از این عنصر به منظور بهبود روند بازی است. هوش مصنوعی بازی‌ها خود شامل چندین بخش مانند شبیه‌سازی جمعیت، کنترل وسایل نقلیه و غیره است، که در این مقاله، یک سیستم هوشمند برای تعقیب و محاصره و در نهایت دستگیری یک هدف (عامل خارجی) طراحی و پیاده سازی گردیده است. یکی از مهمترین چالش‌ها در طراحی این دسته از سیستم‌ها، که اغلب چندعاملی هستند، تعیین محل مناسب هر عامل برای هر چه بهتر رسیدن به هدف و محاصره آن است. مدل پیشنهادی ارائه شده از یک سیستم رخداد محور و چند سیستم پیش‌بینی کلی تشکیل شده است و سعی دارد به کمک اطلاعات رسیده از هر عامل، بهترین چیدمان برای قرار دادن عامل‌ها را پیش‌بینی کند.

کلمات کلیدی: هوش مصنوعی، سیستم‌های چندعاملی، هوش مصنوعی مرکزی، هوش مصنوعی توزیع شده.

۱. مقدمه

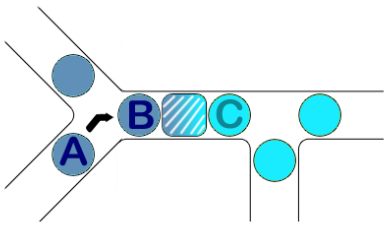
دانش هوش مصنوعی نقش کلیدی در جذابیت و صنعتی شدن بازی‌های رایانه‌ای دارد. در هوش مصنوعی به کارگیری عامل‌های چندگانه [۱] [۲] در بازی‌هایی که در آنها تعقیب و گریز مطرح است بسیار رایج است و بطور کلی مسایلی که به تعقیب و گریز در دنیای هوش مصنوعی می‌پردازند در قالب سیستم‌های چند عامله و با عنوان هایی گوناگون از قبیل "Cows Herders" [۳-۷] مطرح می‌شوند. در این مدل از هوش مصنوعی، یک عامل مرکزی وجود دارد که نقش فرماندهی و ارتباط یک عامل با عامل‌های دیگر را ایفا می‌کند و یک هوش توزیع شده [۲] که هرکدام از عامل‌ها را در انتخاب تصمیم‌ها یاری می‌کند. همچنین تعدادی عامل در اختیار داریم که هر کدام از آن‌ها در مکان‌های مختلفی از نقشه قرار گرفته‌اند. وظیفه کلی این سیستم، دستگیری بازیکن هدفی است که در مکانی نامشخص (از قبل تعیین نشده) از نقشه قرار دارد. بازیکن آزادانه می‌تواند در هر مسیری که بخواهد جابه‌جا شود. فرض می‌شود عامل‌ها در ابتدا بر اساس مسیری که برایشان از قبل تصویر شده است در حال گشت‌زنی هستند و با دیدن هدف، به تعقیب او می‌پردازند.

مسیریابی [۸] عامل‌ها با استفاده از سیستم تورجسم ناوبری[1][۹] و الگوریتم A* [۹] انجام می‌گیرد. هر عاملی که به تعقیب بازیکن بپردازد ابتدا به فرمانده (هوش مصنوعی مرکزی)، موقعیت و وضعیت خود را گزارش می‌دهد سپس فرمانده بر مبنای اطلاعات رسیده، عامل‌های مفید را از بقیه جدا می‌کند. (منظور از عامل مفید، عاملی است که توانایی کمک به تعقیب بازیکن را داشته باشد. برای مثال خیلی دور نباشد یا در ماموریت دیگری قرار نگرفته باشد). سپس با محاسبه‌ی فاصله واقعی هرکدام از عامل‌های مفید تا بازیکن که با استفاده از الگوریتم A* بر روی خانه‌های ساخته شده با سیستم تورجسم ناوبری به دست می‌آید، تصمیماتی را در خصوص این که هر کدام از عامل‌ها در کجا قرار بگیرند را اتخاذ می‌کند و به هر کدام از عامل‌ها تصمیم متناظر خودش را ارسال می‌کند. پس از دریافت دستور، عامل وظیفه اطاعت از آن را بر عهده خواهد داشت. هدف اصلی فرمانده محاصره کردن بازیکن و بستن راه‌های فرار وی است و این کار را با فرستادن عامل‌ها به انتهای مسیرهایی که احتمال حضور بازیکن در آن‌ها بیشتر است انجام می‌دهد.

[1] Navigation Mesh

۲. سیستم ساخت نقشه

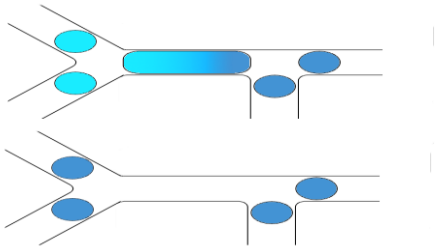
سیستم پیاده‌سازی شده به منظور دریافت و مدل‌سازی محیط اطراف خود، زمین و راه‌های قابل راه‌روی در آنرا، تحت قالب یک گراف راهروی مدل‌سازی می‌کند. در این مدل، راس‌ها به محل‌های ورود و خروج اطلاق می‌شود.



شکل ۱یافتن مسیر عامل

به عنوان مثال در شکل ۱، نقاط مشخص شده نشان‌دهنده یک رأس در گراف ایجاد شده هستند (به این دلیل که محل ورود یا خروج یک مسیر جدید هستند) و خطوط درون شکل نشان‌دهنده یال‌ها می‌باشند. در این گراف، یال ABنشان می‌دهد که یک مسیر از رأس A به B و بالعکس وجود دارد که نیاز به عبور از هیچ رأس دیگری ندارد. استفاده از این شیوه مدل‌سازی به ما کمک می‌کند که مستقل از مختصات مکانی هدف و تنها به کمک آگاهی از آخرین رئوسی که از آن‌ها عبور کرده، محل نسبی فرد را تشخیص دهیم. همچنین به کمک روشی که در زیر معرفی می‌شود، قادر به تشخیص تغییر مسیرها خواهیم بود. به این منظور گراف را به کمک ماتریس مجاورت[۲][۱۰] پیاده‌سازی کرده‌ایم. در نتیجه، اگر فرض کنیم که می‌دانیم فرد در حال تعقیب، به ترتیب از رأس‌های A و B عبور کرده باشد، مشخص می‌شود که این فرد در حال حرکت به سمت رأس C است و در ناحیه هاشور خورده مشخص شده در شکل ۱ است. این کار (تشخیص مکان بعدی و در نتیجه آن، محل نسبی فرد) به کمک تفریق مجموعه رئوس مجاور B از مجموعه رئوس مجاور رأس A به دست می‌آید، این کار سبب خواهد شد که بار پردازشی بسیار کمی بر روی پردازنده قرار گیرد. از آنجا که در مدل پیشنهادی ، رئوس گراف به منزله یک نقطه اطلاع‌رسانی[۳] هستند، در این روش از نگرش پیشامد محور [۴] در انجام محاسبات فوق استفاده می‌کنیم. به این معنا که در هر بار عبور فرد در حال تعقیب از یک گذرگاه، محاسبات فوق یک بار انجام می‌شود.

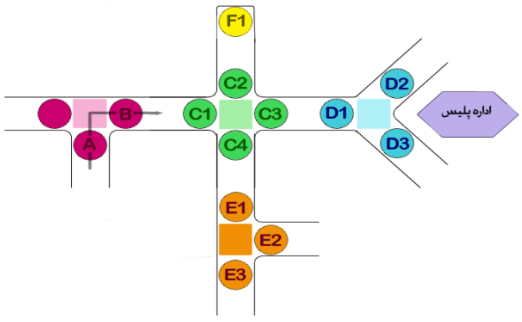
از آن جاکه مسیر پیشروی فرد در دست تعقیب را به کمک ۲ رأس آخری که وی از آن‌ها عبور کرده است، محاسبه می‌کنیم، الزام مهمی پدید می آید که این دو رأس باید متوالی باشند. از طرف دیگر، تنها در صورتی رخداد عبور از یک رأس را پردازش می‌کنیم که یکی از عامل‌های در حال تعقیب، شخص در حال فرار را در آن لحظه ببیند، در غیر این صورت مانند دنیای واقعی هیچ رخدادی صادر نمی‌شود. همچنین در ادامه سعی می‌کنیم برای بهینه‌سازی گراف، تغییراتی در انتخاب رئوس داشته باشیم. بدین منظور سعی می‌کنیم بر اساس فاصله دو رأس متوالی و اهمیت ناحیه بین دو رأس (که نماینده یک یال در گراف، و یک مسیر در نقشه است) رئوس را با یکدیگر ادغام کنیم. در شکل ۲ ، دو نمونه از نحوه بهینه انتخاب رئوس شکل ۱ آورده شده است.



شکل ۲نحوه بهینه انتخاب رئوس

۳. درخت پیش‌بینی مسیر

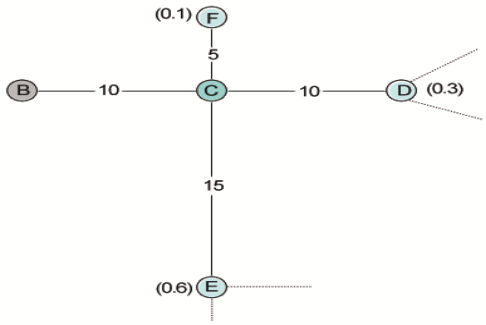
همان طور که در بخش ۲ به درخت پیش‌بینی اشاره شد، در هر بار عبور از رئوس، یک پردازش برای بررسی محل بعدی فرد در حال گریز انجام می‌دهیم. همچنین همزمان با به‌روزرسانی اطلاعات فوق، یک درخت پیش‌بینی مسیر نیز تشکیل می‌دهیم. علت همزمانی این دو عملیات، عدم امکان تغییر مسیر در بین دو رأس است. یعنی پس از این که فرد در حال تعقیب از یک گذرگاه عبور کرد، تا رسیدن فرد به رأس بعدی، تغییر مسیری نمی‌تواند صورت گیرد که این فرصت مناسبی را برای تشکیل درخت مهیا می‌کند. (شایان ذکر است که گذر از رئوس که منجر به ورود به یک ناحیه چند راهی می‌شود به شکل دیگری بررسی می‌شود که از بیان این حالت خاص صرف نظر می‌کنیم). هدف از تشکیل درخت فوق که به نوعی از گراف اصلی تولید شده ولی به شکل خاص استفاده می‌کند، به دست آوردن احتمال حرکت فرد در حال گریز به محل‌های در پیش روی او است. نتیجه‌ای که از این درخت برای ما اهمیت دارد، به دست آوردن مکان‌های مناسب برای قرار دادن عامل‌ها و بستن راه‌ها است. برای نمونه و بر اساس شکل ۳، رئوس C1 تا C4 از نظر مفهومی نمایانگر یک منطقه قابل پوشش توسط نیروهای تعقیب‌کننده است، پس در درخت خود، برای چنین رئوسی، یک رأس را به عنوان نماینده برای قرارگیری عامل استفاده می‌کنیم.



شکل ۳منطقه‌های قابل پوشش توسط عامل تعقیب کننده

[۲] Adjacent Matrix
[3] Reference
[4] Event Based

نحوه تشکیل این درخت را با یک مثال نشان می‌دهیم. اگر فرض کنیم فرد در دست تعقیب به ترتیب از گذرگاه‌های A و B گذشته باشد، برای عمق ۲، درختی مانند شکل ۴ خواهیم داشت:



شکل ٤مثال درخت جستجو در عمق ۲

در این درخت وزن یال‌ها نمایانگر طول مسیر و وزن برگ‌ها نشان‌دهنده احتمال آن مکان است (احتمال آنکه فرد در حال فرار به آن جا بگریزد). در محاسبه این احتمالات، ما ۲ معیار اصلی را مورد توجه قرار دادیم:

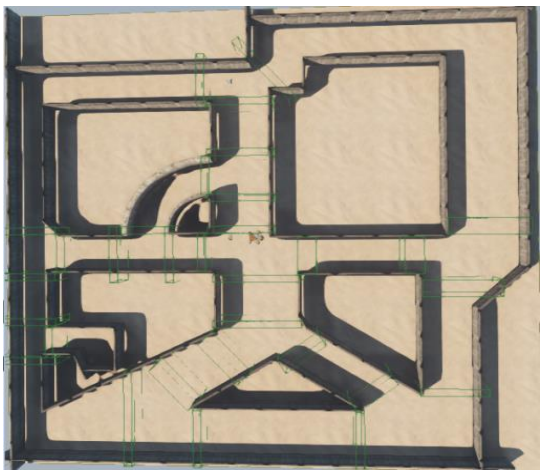
۱. **میزان خطر گذرگاه‌ها:** این معیار به صورت دلخواه ارزش‌دهی می‌شود و به شرایط فیزیکی یک رأس و عواملی از این دست بستگی دارد. به عنوان مثال، یک مکان که دارای آیتم‌هایی برای پنهان شدن بازیکن باشد، نسبت به یک مسیر که به بن‌بست ختم می‌شود، مناسب‌تر در نظر گرفته می‌شود و از ارزش بالاتری برخوردار خواهد بود.

۲. **تاریخچه مسیرهای انتخابی:** در این قسمت، یک سیستم ارزش‌دهی جداگانه برای گذرگاه‌ها در نظر گرفته می‌شود که بر اساس مشاهدات عامل‌های تعقیب‌کننده در حین تعقیب و بر اساس انتخاب‌های فرد در دست تعقیب، ارزش‌دهی می‌شود. به عنوان مثال، اگر بازیکن چند بار در یک دوراهی از مسیر سمت راست برود و این کار توسط عامل‌ها دیده شود، باعث می‌شود رأس انتهای مسیر سمت راست از ارزش بیشتری نسبت به رأس موجود در مسیر سمت چپ، برخوردار شود.

در شکل ۵ نمای نقشه‌ای که نگارندگان در موتور بازی‌سازی UDK [۱۱] برای انجام آزمایش‌های خود پیاده‌سازی کرده‌اند نشان داده شده است.

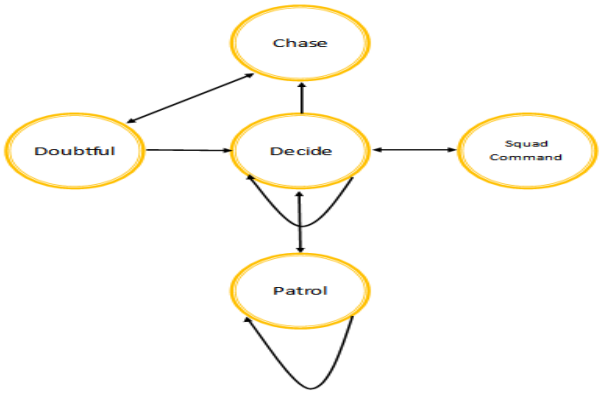
۴. آزمایش هوش مصنوعی فردی

سیستم هوش مصنوعی فردی با استفاده از ساختار ماشین حالت متناهی[5] [۱۲] ساخته شده است. در این سیستم، هر عامل دارای چندین حالت است که با توجه به شرایط و فرمان‌هایی که از سیستم هوش مرکزی می‌گیرد، وارد یکی از این حالت‌ها می‌شود. در شکل ۶ حالت‌های مختلف در نظر گرفته شده



شکل ٥نقشه مسیرها و گذرگاه‌های آزمایش

و در ادامه توضیح قسمت‌های مهم آن را مشاهده می‌کنید.



شکل ٦ماشین گذر متناهی آزمایش

۴-۱ حالت تعقیب

حالت تعقیب[6]، هنگامی رخ می‌دهد که عامل، کاربر را مشاهده کرده و از طرف فرمانده دستوری نداشته باشد. در این حالت بسته به موقعیت بازیکن و عامل دو حالت می‌تواند اتفاق بیفتد:

۱. اگر عامل از بازیکن عقب‌تر باشد و هر دو در یک جهت حرکت کنند (عامل در حال تعقیب از پشت سر بازیکن) عامل به مکانی که بازیکن را می‌بیند می‌رود و در هر لحظه مکان بازیکن را به روزرسانی کرده و به فرمانده اطلاع می‌دهد.

۲. در صورتی که عامل از پشت سر بازیکن در تعقیب او نباشد (به طور مثال از رو به رو یا از طرفین) مکان احتمالی برخورد با بازیکن توسط

[5] Finite State Machine

[6] Chasing State

فرمول زیر به دست آمده و عامل به موقعیت Position فرستاده شده تا در زمان صحیح به بازیکن رسیده و او را دستگیر کند.

$$\Delta S = s_{agent} - s_{Goal}$$

$$\Delta V = v_{agent} - v_{Goal}$$

$$T = \frac{\Delta S}{\Delta V}$$

$$\text{Position} = s_{agent} + v_{agent} \times T$$

در صورتی که در حین مراحل تعقیب، بازیکن از دید عامل ناپدید شد، عامل وارد حالت مشکوک شده و در صورت مشاهده بازیکن به حالت تعقیب باز می‌گردد.

۴-۲ حالت تصمیم‌گیری

حالت تصمیم‌گیری[۷]، که اولین و اصلی‌ترین حالت عامل و به نوعی واحد تصمیم‌گیری منطقی عامل است، بر اساس اولویت‌های تعیین شده در داخل آن تصمیم به رفتن به حالت دیگری می‌گیرد. اولویت‌های قرار گرفته در مدل ایجاد شده برای محاصره بازیکن به صورت زیر است:

۱. در صورت داشتن دستوری از فرمانده رفتن به حالت دستور فرمانده.
۲. در صورت دیدن بازیکن و تحت تعقیب بودن آن رفتن به حالت تعقیب.
۳. در صورت داشتن مسیر برای گشت زدن رفتن به حالت گشت‌زنی.

همچنین اکثر حالت‌ها پس از انجام کارهای تعیین شده به حالت تصمیم‌گیری وارد می‌شوند و به این صورت حلقه اصلی تفکر عامل تشکیل می‌شود.

۵. سیستم هوش مصنوعی مرکزی

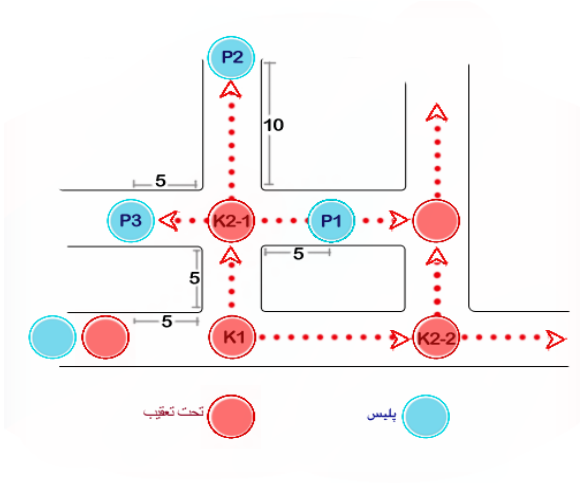
به منظور کنترل و هماهنگی بین نیروهای موجود از هوش جمعی استفاده می‌کنیم. این واحد مرکزی وظیفه دارد که اطلاعات و مشاهدات تمام نیروها را گرفته و پس انجام پردازش‌ها، دستورت لازم را به عامل‌ها ارسال کند [۱۳]. از آنجایی‌که این فرآیند به صورت متمرکز انجام می‌شود، به این واحد، واحد مرکزی اطلاق می‌کنیم. این واحد در دو مقطع زمانی به تصمیم‌گیری می‌پردازد:

۱. هنگامی که بازیکن در حال تعقیب از یک گذرگاه عبور کند و در عین حال توسط یک عامل قابل مشاهده باشد.
۲. دریافت اطلاعات از عامل‌ها.

بسته به مقطع زمانی تصمیم گیری دو حالت می‌تواند اتفاق بیفتد. این دو حالت در ادامه مقاله توضیح داده خواهند شد.

الف) حالت اول

حالت اول، همان طور که در بخش‌های قبل معرفی شد، با ایجاد یک درخت پیش‌بینی مسیر آغاز می‌گردد. عمق این درخت بر اساس معیارهای گوناگونی از قبیل تعداد عامل‌ها و موقعیت عامل‌ها. تعیین می‌گردد. در مرحله اول درختی با عمق یک ایجاد می‌کنیم. نتیجه این عمل، درختی با تنها یک برگ است. بر اساس اطلاعات قرار گرفته در برگ این درخت، امکان بسته شدن مسیر توسط یکی از نیروها بررسی می‌گردد. اگر بتوان عاملی بدین منظور پیدا کرد مشکل حل شده است، در غیر این صورت عمق درخت یک واحد افزایش می‌یابد. بر اساس عمق درخت و وزن یال‌ها و مکان فعلی عامل‌ها جدول شماره ۱ را از شکل ۷ به وجود می‌آوریم.



شکل ۷نقشه مسیرها و گذرگاه‌های آزمایش

فرض کنید تمامی افراد از جمله فرد در حال فرار، بتوانند با سرعت ثابت ۱ واحد بر ثانیه بدوند. در این صورت مدت زمانی که برای پیمودن M واحد لازم است، برابر M ثانیه خواهد بود.

جدول ۱جدول ساخته شده بر اساس شکل ۷

	K1	K2-1	K2-2
P1	-5	+5	+5
P2	-10	0	-10
P3	-5	+5	-5

با این فرض، خانه‌های جدول بالا، اختلاف زمان رسیدن فرد در حال فرار به یکی از مکان‌های حاصل از برگ درخت پیش‌بینی، نسبت به هر یک از عامل‌های P1 تا P3 را نشان می‌دهد. یعنی:

[7] Deciding State

(۱) $\Delta t = (Kx\text{-}T_{yP}) - (Kx\text{-}T_{yA})$

T_{yP}= y زمان رسیدن بازیکن به خانه(2)

T_{yA}= y زمان رسیدن عامل P به خانه (3)

اعداد منفی جدول، نشان‌دهنده میزان تاخیر عامل‌ها برای رسیدن به هدف در نقطه مشخص شده است. همان طور که از شکل ۷ و جدول ۱ مشخص است، در مرحله اول سعی می‌کنیم که در لایه‌ی اول به موفقیت برسیم. به این منظور باید به دنبال عاملی باشیم که بتواند قبل از رسیدن بازیکن در حال گریز، خود را به مکان K1 برساند. ولی از آن جا که برای این لایه تمام اختلاف زمان‌ها منفی است، مشخص است که امکان چنین چیزی وجود ندارد. پس باید به بررسی لایه بعدی بپردازیم.

موفقیت در هر لایه منوط به بسته شدن تمام برگ‌های آن لایه است که در مورد لایه دو، بسته شدن مکان‌های K_{1-2} و K_{2-2} مدنظر است. همان‌طور که در جدول مشخص است، این کار با قرار دادن نیروی P1 در مکان K_{2-2} و یکی از دو نیروی P2 یا P3 در مکان K_{1-2} قابل انجام است. اگر در شرایطی در این لایه نیز قادر به بستن تمام راه‌ها نمی‌شدیم، باید برای مسیرهای بسته نشده، گراف را تا یک لایه بعد گسترش دهیم. روند افزایش لایه‌ها زمانی خاتمه می‌یابد که یا در لایه فعلی به موفقیت برسیم و بتوانیم تمامی مکان‌های آن لایه را (برگ‌های آن لایه را) پوشش دهیم و یا مجموع اختلاف زمان‌های لایه فعلی از لایه قبلی کمتر شود یا تعداد برگ‌ها بیشتر از عامل‌های موجود گردد

هم‌چنین به منظور کاهش محاسبات و دریافت نتیجه بهتر، گسترش درخت به صورت ناحیه‌ای[8] صورت می‌پذیرد. به این معنا که اگر در یک لایه قسمتی از مکان‌ها قابل پوشش و قسمتی غیر قابل پوشش باشند تنها بررسی لایه بعد را از سمت برگ‌های غیر قابل پوشش ادامه می‌دهیم و گسترش به صورت ناحیه‌ای صورت می‌پذیرد که این کار باعث کاهش چشمگیری در هزینه پردازش‌ها می‌گردد. در رابطه با نحوه چیدمان عامل‌ها در مکان‌ها، چند حالت ممکن است اتفاق بیفتد، که هر یک را جداگانه شرح می‌دهیم:

۱. **تعداد عامل‌ها بیشتر یا برابر تعداد برگ‌ها باشد:** این حالت به معنی وجود عامل‌های اضافی است. در این حالت، از عامل‌ها با چند رویکرد متفاوت و بسته به مکان و فاصله آن‌ها از فرد در حال گریز و فاصله با سایر برگ‌ها استفاده می‌کنیم.

الف) بستن مسیر از پشت (این کار به منظور پشتیبانی عامل‌های اصلی بکار می‌رود)

ب) افزایش عامل‌ها در مکان‌های محتمل‌تر (با توجه به وزن برگ‌ها)

۲. **تعداد عامل‌ها کمتر از تعداد برگ‌ها باشد:** در این حالت، که معمول‌تر از حالت قبلی است، سعی بر آن است که یک عامل به دو یا چند ناحیه اختصاص داده شود. در این حالت، یک عامل بر اساس میزان احتمال برگ‌هایی که به وی اختصاص داده شده و فاصله وی از هر یک از آن‌ها، در حد فاصل برگ‌ها قرار می‌گیرد. مثالی از انجام این کار بدین شکل است که فرض کنید یک عامل داریم و دو نقطه باید بسته شود، مدت زمان رسیدن بازیکن به هر کدام از این دو نقطه را محاسبه و سپس بر مبنای کمترین زمان، عامل را در مسیر بین دو نقطه و در فاصله‌ای از نقطه نزدیک‌تر به بازیکن قرار می‌دهیم که اگر از طرف فرمانده متوجه شد که بازیکن وارد کدام مسیر شده است، زمان لازم برای رسیدن به آن نقطه را داشته باشد و خود را به آن مکان برساند.

ب) حالت دوم

برای کنترل بیشتر و طبیعی شدن رفتار عامل‌ها، از یک ارتباط دوطرفه بین واحد مرکزی و عامل‌ها استفاده می‌کنیم. بدین منظور از مشاهدات عامل‌ها در موقعیت‌های مختلف استفاده کرده و پس از بررسی شرایط عامل‌ها، دستور واحد مرکزی را اجرا خواهیم کرد.

۶. نتیجه‌گیری

در این مقاله، هدف اصلی، ارایه مدلی ساده و در عین حال کارا برای تعقیب و محاصره‌ی یک هدف خارجی توسط مجموعه‌ای از عامل‌ها است. نگارندگان در این مقاله، به منظور هرچه بیشتر طبیعی جلوه دادن مدل هوش مصنوعی و کاهش نیاز به تبادل اطلاعات مابین عامل‌ها، تاکید بر استفاده از روش رویداد محور، و نه مکان دقیق عامل‌ها دارند. مدل پیشنهادی فوق، علاوه بر سرعت پردازش بالا، امکان استفاده همزمان از دو روش هوش مصنوعی توزیع‌شده و مرکزی را به ارمغان می آورد. روش ارائه شده از ترفندهای رایج هوش مصنوعی در بازی‌های رایانه‌ای استفاده نمی‌کند. این حرف به این معنا است که در روش ارائه شده، مکان و مشخصات بازیکن در اختیار هیچ یک از عوامل موجود در دنیای بازی قرار ندارد. این موضوع می‌تواند منتهی به طبیعی شدن فراوان هوش مصنوعی به کار رفته در بازی شده و محیط پویای رقابتی با سربار پردازشی بسیار کمی را به مخاطبان خود ارائه کند.

۷. مراجع

[1] Simon Parsons and Michael Wooldridge. 2002. Game Theory and Decision Theory in Multi-Agent Systems. Autonomous Agents and Multi-Agent Systems 5, 3 (September 2002)

[2] Winikoff, M., Padgham, L.: Developing intelligent agent systems: a practical guide. In: WileySeries in Agent Technology. Wiley, New York, NY (2004)

[3] Adel T. Rahmani, Alireza Saberi, Mehdi Mohammadi, Amin Nikanjam, Ehsan Adeli Mosabbeb, and Monireh Abdoos. 2009. SHABaN: Multi-agent Team To Herd Cows. In Programming Multi-Agent Systems, Koen V. Hindriks, Alexander Pokahr, and Sebastian

[8] Regional

Sardina (Eds.). Lecture Notes In Artificial Intelligence, Vol. 5442. Springer-Verlag, Berlin, Heidelberg 248-252

[4] Steen Vester, Niklas Skamriis Boss, Andreas Schmidt Jensen, and Jørgen Villadsen. 2011. Improving multi-agent systems using Jason. Annals of Mathematics and Artificial Intelligence 61, 4 (April 2011), 297-307

[5] Vahid Rafe, Amin Nikanjam, and Mohammad Rezaei. 2011. Galoan: a multi-agent approach to herd cows. Annals of Mathematics and Artificial Intelligence 61, 4 (April 2011), 333-348

[6] Gregor Balthasar, Jan Sudeikat, and Wolfgang Renz. 2009. On Herding Artificial Cows: Using Jadex to Coordinate Cowboy Agents. In Programming Multi-Agent Systems, Koen V. Hindriks, Alexander Pokahr, and Sebastian Sardina (Eds.). Lecture Notes In Artificial Intelligence, Vol. 5442. Springer-Verlag, Berlin, Heidelberg 233-237

[7] Gregor Balthasar, Jan Sudeikat, and Wolfgang Renz. 2010. On the decentralized coordination of herding activities: a Jadex-based solution. Annals of Mathematics and Artificial Intelligence 59, 3-4 (August 2010), 411-426

[8] In-Cheol Kim. 2006. Real-Time search algorithms for exploration and mapping. In Proceedings of the 10th international conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part I (KES'06), Bogdan Gabrys, Robert J. Howlett, and Lakhmi C. Jain (Eds.)

[9] Xiao Cui; Hao Shi, "A*-based Path finding in Modern Computer Games" IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011

[10] Godsil, Chris; Royle, Gordon Algebraic Graph Theory, Springer (2001), ISBN 0-387-95241-1, p.164

[11] UDK Unreal Developer's Kit, http://www.unrealengine.com/udk , last visit: January 2013.

[12] Anna I. Esparcia-Alcazár, Anais Martínez-García, Antonio M. Mora, J. J. Merelo, and Pablo García-Sánchez. 2010. Genetic evolution of fuzzy finite state machines to control bots in a first-person shooter game. In Proceedings of the 12th annual conference on Genetic and evolutionary computation

[13] Karl Tuyls and Simon Parsons. 2007. What evolutionary game theory tells us about multiagent learning. Artif. Intell. 171, 7 (May 2007), 406-416