روش محاصره هدف توسط یک سیستم چندعاملی مبتنی بر رویداد

بهروز مینایی عماد آقاجانی علیرضا صحاف نایینی

دانشگاه علم و صنعت ایران، دانشکده مهندسی کامپیوتر

[alireza@iust.ac.ir](mailto:alireza@iust.ac.ir) [emad@iust.ac.ir](mailto:emad@iust.ac.ir) [b\_minaei@iust.ac.ir](mailto:b_minaei@iust.ac.ir)

چکیده:

صنعت بازی‌های رایانه‌ای امروزه به یکی از پرطرفدارترین و سرگرم‌کننده‌ترین صنایع در جهان تبدیل شده است. یکی از اصلی‌ترین عوامل ایجاد هیجان و جذابیت در بازی‌های رایانه‌ای عنصر هوش مصنوعی است و 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... می‌باشد، که در این مقاله، مقوله پیچیده طراحی یک سیستم هوشمند برای تعقیب و محاصره و در نهایت دستگیری یک هدف (عامل خارجی) مورد بررسی قرار گرفته است.یکی از مهمترین چالش‌ها در طراحی این دسته از سیستم‌ها، که عموما چندعاملی می‌باشند، تعیین محل مناسب هر عامل برای هر چه بهتر رسیدن به هدف و محاصره آن است.

مدل پیشنهادی ما 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. در ادامه توضیحات این سیستم را بررسی می‌کنیم.

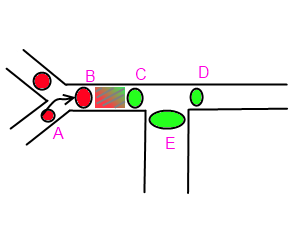
کلمات کلیدی: هوش مصنوعی، سیستم‌های چندعاملی، هوش مصنوعی مرکزی، هوش مصنوعی توزیع شده

1. مقدمه

دانش هوش مصنوعی نقش کلیدی در جذابیت و صنعتی شدن باز‌ی‌های رایانه‌ای دارد. در هوش مصنوعی به کارگیری عامل‌های چندگانه در بازی‌هایی که در آنها تعقیب و گریز مطرح است بسیار رایج است. در این مدل از هوش مصنوعی ما یک عامل مرکزی داریم که نقش فرماندهی و ارتباط یک عامل [1]با عامل‌های دیگر را ایفا می‌کند و یک هوش توزیع شده[2] 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. وظیفه‌ی کلی این سیستم دستگیری هدف بازیکنی است که در مکانی 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 منظور شده است در حال گشت‌زنی هستند و با دیدن هدف، به تعقیب او می‌پردازند. مسیریابی[3] عامل‌ها با استفاده از سیستم Navigation Mesh [3] و الگوریتم A\* [3] 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 فاصله‌ی واقعی هرکدام از عامل‌های مفید تا بازیکن که با استفاده از الگوریتم A\* بر روی خانه‌های ساخته شده با سیستم Navigation Mesh 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 و عامل آن را انجام می‌دهد. 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. **سیستم ساخت نقشه**

ما به منظور پیاده‌سازی ایده خود زمین و راه‌های آن را که معرّف نقشه بازی است، به یک گراف مدل کردیم. در این مدل راس‌ها به محل‌های ورود و خروج اطلاق می‌شود.



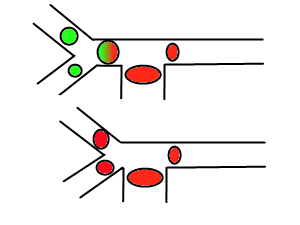
شکل 1: یافتن مسیر عامل

به عنوان مثال در شکل 1 نقاط مشخص شده نشان‌دهنده یک رأس در گراف هستند (چرا که محل ورود یا خروج یک مسیر جدید می‌باشند) و خطوط درون شکل نشان‌دهنده یال‌ها می‌باشند. در این گراف، یال AB نشان 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، ما قادر به تشخیص تغییر مسیرها خواهیم بود. به این منظور ما گراف را به کمک ماتریس مجاورت[[1]](#footnote-2) [4] پیاده‌سازی کردیم. 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 1 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، که مشخصا پردازش بسیاری کمی را نیاز دارد. از آن جا که در مدل پیشنهادی ما، رئوس گراف به منزله یک نقطه رهایی (trigger) می‌باشند، ما از نگرش پیشامد محور (Event Base) در انجام محاسبات فوق استفاده می‌کنیم. به این معنا که در هر بار عبور کردن فرد در حال تعقیب از یک گذرگاه، محاسبات فوق یک بار انجام می‌شود.

از آن جا که ما مسیر پیشروی فرد در دست تعقیب را به کمک ۲ رأس آخری که وی از آن‌ها عبور کرده است، محاسبه می‌کنیم، این نکته مهم است که این ۲ رأس باید متوالی باشند. از طرف دیگر، ما 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.

به عنوان مثال اگر فرد در دست تعقیب، بعد از عبور از گذرگاه‌های A، B و C، از گذرگاه E عبور کند ولی عاملی به او نرسد و مدتی بعد در حال عبور از گذرگاه D دیده شود، نشان می‌دهد که او توسط عاملی در حال تعقیب نبوده است و ما اطلاعاتی از وی نداشتیم. پس نمی‌توانیم تنها بر اساس رأس D نظر بدهیم. در اینجا ما بر اساس نقشه واقعی بازی (و نه مدل گراف آن) مسیر بعدی را به دست می‌آوریم. این کار می‌تواند به شکل‌های مختلف انجام گیرد که ما از روش شبیه‌سازی حرکت واقعی با سرعت بسیار بالا استفاده می‌کنیم.

همچنین ما در ادامه سعی می‌کنیم برای بهینه‌سازی گراف، تغییراتی در انتخاب رئوس داشته باشیم. بدین منظور ما سعی می‌کنیم بر اساس فاصله دو رأس متوالی و اهمیت ناحیه بین  ۲ رأس (که نماینده یک یال در گراف، و یک مسیر در نقشه است) رئوس را با یکدیگر ادغام کنیم. در شکل 2 دو نمونه از نحوه بهینه انتخاب رئوس شکل 1 آورده شده است.

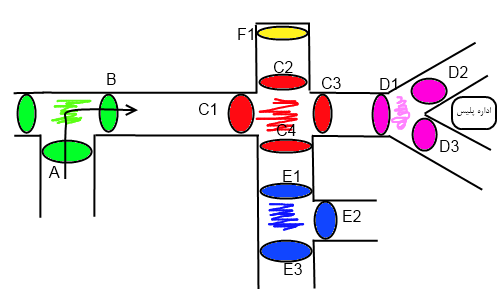


شکل 2: نحوه بهینه انتخاب رئوس

1. **درخت پیش‌بینی مسیر**

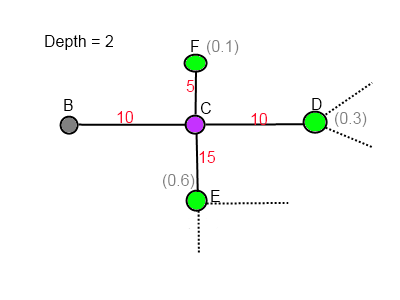
همان طور که در بخش 2 به درخت پیش‌بینی اشاره شد، در هر بار عبور از رئوس، ما یک پردازش برای بررسی محل بعدی فرد در حال گریز انجام می‌دهیم. ما همچنین همزمان با به‌روزرسانی اطلاعات فوق، یک درخت پیش‌بینی مسیر نیز تشکیل می‌دهیم. علت همزمانی این دو عملیات، عدم امکان تغییر مسیر در بین دو رأس است. یعنی پس از این 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 ما 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 ما ولی به شکل خاص استفاده می‌کند، به دست آوردن احتمال حرکت فرد در حال گریز به محل‌های در پیش روی اوست. 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 شکل3، رئوس C1 تا C4 از نظر مفهومی نمایانگر یک منطقه قابل پوشش توسط نیروهای تعقیب‌کننده است، پس ما در درخت خود، برای چنین رئوسی، یک رأس را به عنوان نماینده برای قرارگیری عامل استفاده می‌کنیم.



شکل 3: منطقه‌های قابل پوشش توسط عامل تعقیب‌کننده

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 4 خواهیم داشت:



شکل 4: مثال درخت جستجو در عمق 2

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:

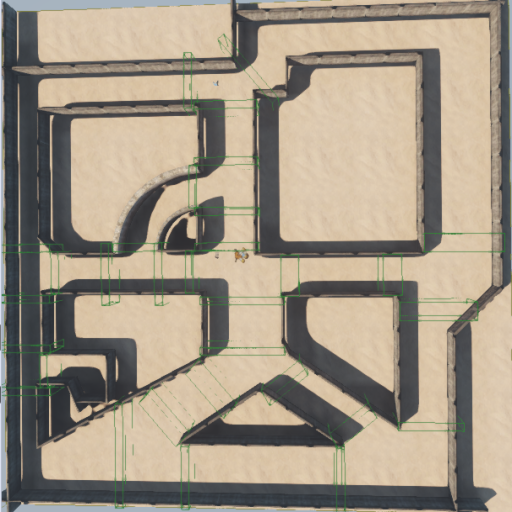
1**. میزان خطر گذرگاه‌ها:**

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.

2**. 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.

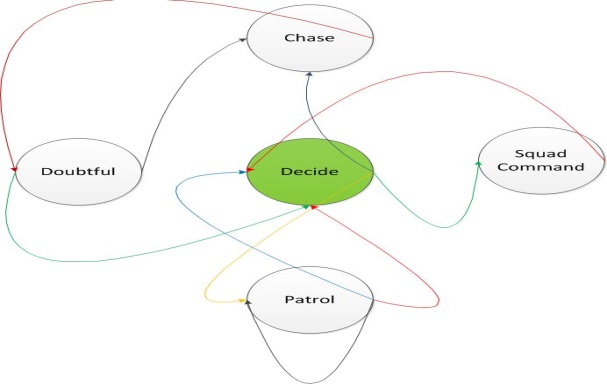
در شکل 5 نمای نقشه‌ای که تیم نگارندگان مقاله آزمایش‌های خود را در آن انجام داده، در کنار گذرگاه‌های آن نمایش داده شده است:



شکل 5: نقشه مسیرها و گذرگاه‌های آزمایش

1. آزمایش هوش مصنوعی فردی

سیستم هوش مصنوعی فردی با استفاده از ساختار Finite State Machine (FSM) [5] 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 6 حالت‌های مختلف در نظر گرفته شده را مشاهده می‌کنید.



شکل 6: ماشین گذر متناهی آزمایش

4-1 حالت تعقیب[[2]](#footnote-3)

این حالت 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 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 روز رسانی کرده و به فرمانده اطلاع می‌دهد.

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 فرمول زیر به دست آمده و عامل را به موقعیت Position می‌فرستیم تا در یک زمان به بازیکن برسد و او را دستگیر کند.

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.

4-2 حالت مشکوک[[3]](#footnote-4)

این حالت زمانی رخ می‌دهد که عامل در هنگام تعقیب دیگر موفق به دیدن بازیکن نشود. در این حالت ابتدا به آخرین جایی که بازیکن را دیده رفته و سپس در صورت مشاهده کردن بازیکن دوباره وارد حالت تعقیب می‌شود، در صورت عدم مشاهده بازیکن، مسیر روبه‌رو (در صورت وجود) یا یکی از مسیرهای ممکن را تا مدتی ادامه می‌دهد و در صورت عدم مشاهده بازیکن به حالت تصمیم‌گیری باز می‌گردد.

4-3 حالت گشت زدن[[4]](#footnote-5)

در این حالت وظیفه‌ی عامل رفتن به نقاط خاصی در نقشه و انجام کار خاصی در آن مکان به صورت تکرار شونده است. برای اجرای این حالت ما باید از قبل برای عامل‌ها مکان‌هایی را در نقشه مشخص کرده باشیم، برای مثال می‌توان گشت زنی در یک محله را با این سیستم به این صورت شبیه‌سازی کرد که عامل از اولین نقطه‌ی داده شده شروع کرده و به صورت ترتیبی به تمام نقاط مربوطه رفته و کار متناظر آن نقطه را انجام می‌دهد. (برای مثال انداختن نور چراغ قوه در مغازه‌ها و بررسی امنیت آن‌ها)

در صورت مشاهده مورد خاصی یا دریافت فرمانی از فرمانده وارد حالت تصمیم‌گیری می‌شود در غیر این صورت به گشت زدن خود می‌پردازد.

4-4 حالت دستور فرمانده

این حالت مربوط به انجام دستورات فرمانده است و در هنگامی که فرمانی از فرمانده می‌رسد، عامل وارد این حالت می‌شود. این حالت بر طبق قابلیت‌هایی که برای سیستم در نظر گرفته‌ایم، دارای پیاده‌سازی‌های متفاوتی است، که عمده‌ترین تفاوت آن‌ها فرمان‌های پشتیبانی شده توسط عامل است. در مدل ما به دلیل آنکه هدف نهایی محاصره و دستگیری بازیکن است، تنها فرمان ارسالی فرمانده به هر عامل دستور حرکت به نقطه‌ای خاص است.

برای اجرای این دستور فرمانده باید نقطه‌ای را که برای محاصره‌ی بازیکن بهتر است، به عامل معرفی کند. سپس عامل به سمت آن نقطه حرکت می‌کند و در صورتیکه بازیکن را در طول پیمودن مسیر مشاهده نکند خود را به آن نقطه می‌رساند. سپس از فرمانده یا دستور سکون یا حرکت به نقطه ای جدید دریافت می‌کند که به دستور جدید عمل می‌کند.

4-5 حالت تصمیم‌گیری[[5]](#footnote-6)

این حالت که اولین و اصلی‌ترین حالت عامل و به نوعی مغز عامل است، بر اساس اولویت‌های تعیین شده در داخل آن تصمیم به رفتن به حالت دیگری می‌گیرد.   
اولویت‌های قرار گرفته در مدل ما برای محاصره بازیکن به صورت زیر است:

1. در صورت داشتن دستوری از فرمانده رفتن به حالت دستور فرمانده
2. در صورت دیدن بازیکن و تحت تعقیب بودن آن رفتن به حالت تعقیب
3. 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 حلقه‌ی اصلی تفکر عامل تشکیل می‌شود.

1. پیاده‌سازی

تیم نگارندگان به منظور انجام آزمون از یکی از موتور‌های شبیه‌سازی گرافیکی رایگان به اسم UDK ورژن 2010 October استفاده کردند و نتایج نشان از موفقیت بالای عامل‌ها می‌داد. همچنین ابزار Analyser این محیط به ما در اندازه‌گیری میزان پردازش در حال انجام کمک کرد و نتایج این بخش همان طور که انتظار می‌رفت بسیار رضایت بخش بود.

1. نتیجه‌گیری

در این مقاله هدف اصلی ما 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 شدن 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 و استفاده همزمان از دو روش هوش مصنوعی توزیع شده و مرکزی و بدون استفاده از ترفندهای رایج بازی‌های امروزی، که مکان و مشخصات بازیکن را تمام عامل‌ها همیشه در اختیار دارند و باعث مصنوعی شدن هوش مصنوعی بازی می‌شود، باعث ایجاد محیطی پویا و در عین حال سبک از لحاظ پردازشی، برای کاربران می‌شود.

مراجع:

[1] Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2

[2] Gerhard Weiss , Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, The MIT Press 1999, ISBN 978-0262232036

[3] Xiao Cui; Hao Shi, “A\*-based Pathfinding in Modern Computer Games” IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, January 2011

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

[5] Thomas A. Sudkamp , Languages and Machines: An Introduction to the Theory of Computer Science (3nd ed.), Addison-Wesley PublishingCo.2007, ISBN 978-0321322210

1. Adjacent Matrix [↑](#footnote-ref-2)
2. Chase [↑](#footnote-ref-3)
3. Doubtful [↑](#footnote-ref-4)
4. Chase [↑](#footnote-ref-5)
5. Decide [↑](#footnote-ref-6)