بهبود رفتار ربات در یک بازی بلادرنگ استراتژیک با استفاده از روش گرگ خاکستری

پرهام زیلوچیان مقدم[1]، فرید لطفعلی[2] ، دکتر جواد سلیمی سرتختی[3]

1- دانشجوی مهندسی کامپیوتر دانشگاه کاشان

Email: p.zilouchian@gmail.com

2- دانشجوی مهندسی کامپیوتر دانشگاه کاشان

Email: farid9lotfali@gmail.com

3- عضو هیئت علمی دانشکده برق و کامپیوتر دانشگاه کاشان

Email: salimisartakhti@gmail.com

چکیده

در این مقاله ما قصد داریم تا قدرت و نتایج استفاده از روش گرگ خاکستری را در طراحی و توسعه الگوریتم بازی ها بسنجیم . به طور خاص در این مقاله به بررسی بازی Planet Wars که برای رقابت هوش مصنوعی گوگل در سال 2010 انتخاب شد پرداخته می شود. این بازی به یک بازیکن مصنوعی نیاز دارد (در این بازی‌ها به آن‌ها ربات نیز گفته میشود) که قادر است تا با چندین هدف تعامل داشته باشد ، وقتی که قرار است به یک درجه‌ای خاص از سازگاری برسد برای این که حریفان متفاوتی را در سناریوها متفاوتی شکست دهند. موتور تصمیم این ربات براساس مجموعه‌ای از قوانین که براساس یادگیری تجربی به دست می آید تعریف میشود. الگوریتم‌های تکاملی (Evolutionary Algorithms) برای تنظیم مجموعه‌ای از مقادیر ثابت ، وزن‌ها واحتمالات که قوانین را تعریف میکنند استفاده میشود و بنابراین عملکرد عمومی ربات را تعریف میکنند. الگوریتم گرگ خاکستری یک روش متا هیوریستیک است که برگرفته از سلسله مراتب زندگی گرگ‌های خاکستری و نحوه شکار کردن آن‌ها در طبیعت است. به طور کلی گرگ‌های خاکستری را در چهار دسته به نام‌های آلفا ، بتا ، دلتا و امگا دسته‌بندی میکنند، از این دسته‌بندی برای بیان نحوه سلسله مراتب گرگ‌های خاکستری استفاده میشود. علاوه بر این موارد سه مرحله شکار، شامل جستجو برای شکار ، محاصره طعمه و حمله به طعمه نیز در این روش به کار گرفته میشود. الگوریتم گرگ خاکستری به وسیله 29 تابع تست معروف مورد آزمون و بنچ مارک قرار گرفته و نتایج آن به وسیله یادگیری مقایسه‌ای و الگوریتم بهبود ذره ذره‌ای (PSO)، الگوریتم جستجوی گرانشی (GSA)، تکامل دیفرانسیل (DE)، برنامه نویسی تکاملی (EP)و استراتژی تکاملی مورد تایید قرار گرفته‌اند. و تمامی این نتایج نشان‌داده اند که الگوریتم گرگ خاکستری در مقایسه با سایر روش‌های متا-هیوریستیک میتواند نتایج بسیار قابل توجهی را ارائه کند.

کلمات کلیدی مقاله: Planet Wars (جنگ سیاره‌ها) – Artificial Intelligence (هوش مصنوعی) – Gray Wolf Optimization (الگوریتم بهینه‌سازی گرگ خاکستری) - الگوریتم‌های تکاملی (Evolutionary Algorithms) – الگوریتم ژنتیک (GA) – برنامه نویسی تکاملی (EP) – Meta Heuristic – Google Artificial Intelligence Challenge (فرامکاشفه‌ای)

1. معرفی

معرفی کلی بازی‌های RTS

بازی های استراتژی زمان واقعی RTS (Real Time Strategy) یک زیرشاخه از بازی های مبتنی بر استراتژی هستند که در آن بازیکنان برای کنترل مجموعه ای از منابع، واحدها و سازه هایی که در عرصه بازی توزیع می شوند، تلاش می کنند. یک کنترل و استراتژی صحیح برای رسیدگی به این واحدها برای پیروزی در بازی ضروری است که این پیروزی ، به طور معمول از بین بردن تمام واحدهای دشمن است، یا در بعضی اوقات نیز زمانی که اهداف مشخصی از بازی به دست می آیند. ویژگی اصلی آنها طبیعت زمان واقعی (real time) آنهاست، یعنی بازیکن مجبور نیست منتظر نتایج بازی های دیگر هنگامی که نوبیت آنان است، باشد. Command & Conquer TM، Starcraft TM، Warcraft TM و Age of Empires TM نمونه هایی از بازی های RTS هستند.

دو سطح AI معمولا در بازی RTS در نظر گرفته می شود: اولین مورد، که توسط شخصیت غیر تماشایی (NPC) تفسیر می شود یک ربات است،که تصمیم گیری در مورد تمام مجموعه های واحد (کارگران، سربازان، ماشین آلات، وسایل نقلیه و یا حتی ساختمان ها) را بر عهده دارد؛ سطح دوم سطح اختصاصی است که مربوط به پیاده سازی رفتار هر یک از این واحدهای کوچک است. این دو سطح اقدامات، که می تواند به لحاظ استراتژیک و تاکتیکی در نظر گرفته شود، به طور ذاتی دشوار است که توسط یک انسان طراحی شود؛ اما این مشکل توسط ماهیت واقعی خود معمولا با محدود کردن زمان که هر ربات می تواند برای تصمیم گیری مورد استفاده قرار دهد، افزایش می یابد .به همین علت در این مقاله یک روش برنامه ریزی ژنتیکی (GP) به عنوان یک روش خودکار برای ایجاد موتور هوش مصنوعی (AI) در یک RTS پیشنهاد شده است. هدف GP ایجاد توابع یا برنامه هایی برای حل مشکلات تعیین شده است، که در آن طرز نمایش هر عنصر معمولا در قالب یک درخت تشکیل شده توسط اپراتورها (یا اولیه) و متغیرها (پایانه ها) است.

هدف از استفاده از GP در این زمینه، ایجاد موتورهای مبتنی بر قانون رفتاری است که به دنبال فرایند اکتشافی، الگوریتمیک و اتوماتیک است. بنابراین، به جای اجرای آنها از ابتدا توسط انسان (متخصص یا غیر متخصص)، این روش مجموعه ای از قوانینی را که می تواند پیچیده تر (و یا ساده تر) از آنچه که توسط انسان تعریف شده، تعریف کند. علاوه بر این، این الگوریتم قادر به ارزیابی هر مجموعه ممکن از قوانین است، و به این مجموعه با توجه به عملکرد ربات مربوطه (در طول جنگ) مقدار هایی را اختصاص می دهد. به این ترتیب، این مجموعه ها در طول الگوریتم اجرا می شود تا بتواند عملکرد ربات را افزایش دهند.

برنامه ریزی ژنتیکی (GP) یک نوع الگوریتم تکاملی (EA) است، یعنی یک الگوریتم جستجو و بهینه سازی احتمالاتی که از مدل تکامل داوطلبانه حاصل شده است، بر اساس این ایده که سازه های طبیعت سازگار هستند. بنابراین، تفاوت اصلی با GAs نمایش عنصر (individual) و اپراتورهای ژنتیکی است که اعمال می شود، که عمدتا بر مدیریت (و بهبود) این نوع ساختار تمرکز دارد.

جریان یک الگوریتم GP مشابه هر EA دیگر است: جمعیت به صورت تصادفی ایجاد می شود، هر فرد در جمعیت با استفاده از یک عملکرد تناسب ارزیابی می شود، افرادی که در فرآیند ارزیابی بهتر عمل می کنند، احتمال بیشتری دارند که انتخاب شوند و پدر و مادر برای جمعیت جدید نسبت به بقیه و جمعیت جدید ایجاد شده است و همچنین زمانی که افراد تحت عمل جراحی ژنتیک قرار میگیرند :(استفاده از روش متقاطع و جهش با یک احتمال خاص است. در آخر حلقه اجرا می شود تا یک معیار خاتمه از پیش تعریف شده برآورده شود.

معرفی بازی جنگ سیاره‌ها

مسابقات AI گوگل (GAIC) یک رقابت AI است که در آن افراد شرکت کننده برنامه هایی (ربات ها) برای رقابت با یکدیگر ایجاد می کنند. بازی انتخاب شده برای مسابقه، جنگ سیاره ها، در این مقاله مورد مطالعه قرار گرفته است به این سبب که موتور رفتاری یک ربات با کارایی بالا طراحی شود. جنگ سیاره ها یک نسخه ساده از Galcon است، از آنجایی که هدف از آن انجام مبارزه های ربات است. نسخه مسابقه‌ای این بازی برای دو بازیکن است.

مسابقه جنگ سیاره ها بر روی یک نقشه انجام می شود که شامل چندین سیاره است، روی هر یک از آنها یک عدد قرار دارد که نشان دهنده تعداد فضاپیما هایی است که میزبان آن است (شکل 1 را ببینید). در یک مرحله زمانی معین از بازی، هر سیاره تعداد خاصی از فضاپیما دارد و ممکن است متعلق به بازیکن، دشمن و یا خنثی باشد (یعنی به هیچ کس تعلق ندارد). مالکیت با یک رنگ نشان داده می شود، آبی برای بازیکن، قرمز برای دشمن، و خاکستری برای خنثی (شخصیت غیر بازی). علاوه بر این، هر سیاره دارای سرعت رشدی است که نشان می دهد که چه تعداد فضاپیما در طول هر مرحله از عملیات تولید می شود و به ناوگان فضاپیماهای بازیکن که مالک آن سیاره است اضافه می شود.

هدف بازی این است که تمام سیارات حریف تسخیر شود. اگر چه جنگ سیاره ها یک بازی RTS (استراتژی) است، پیاده سازی آن را به یک بازی نوبتی تبدیل کرده است، و هر بازیکن دارای حداکثر تعداد چرخش برای رسیدن به هدف است. بازیکن با ستاره های بیشتر در پایان مسابقه برنده می شود.

هر سیاره دارای خواص است: مختصات X و Y ، شناسه مالک سیاره ، تعداد فضاپیماها و نرخ رشد. بازیکنان ناوگان فضاپیماها را برای تسخیر سیارات دیگر (یا تقویت خود) ارسال می کنند، و هر ناوگان همچنین دارای مجموعه ای از خواص: شناسه مالک، تعداد فضاپیماها ، Source ، شناسه مقصد میباشد. زمان نوبت شبیه سازی شده یک ثانیه است و ربات فقط حداکثر در این زمان یک لیست اقدامات بعدی را انجام میدهد.

علاوه بر این، یک ویژگی از این مسئله این است که ربات قادر به ذخیره هر گونه دانش در مورد اقدامات خود در نوبت های قبلی، اقدامات حریف خود و یا نقشه بازی نیست. به طور خلاصه، در هر بار از شبیه سازی (منظور در هر نوبت)، ربات دوباره با یک نقشه ناشناخته مانند یک بازی جدید روبرو می شود. این ناتوانی در ذخیره سازی دانش در مورد گیم پلی ایجاد این ربات را به یک چالش جالب تبدیل میکند.

در حقیقت، هر ربات به عنوان یک تابع اجرا می شود که به عنوان ورودی لیست سیارات و ناوگان (وضعیت فعلی بازی)، که هر کدام از ورودی ها مقادیر خاص خود را دارند و ربات بر اساس آن‌ها یک سری اقدامات انجام می دهد. در هر شبیه سازی ، یک بازیکن باید جایی را برای ارسال ناوگان از فضاپیماها ، خروج از یکی از سیارات بازیکن و رفتن به سیاره دیگر بر روی نقشه انتخاب کید. این تنها نوع اقداماتی است که ربات مجاز به انجام آن است. ناوگان ها می توانند گام هایی برای رسیدن به مقصد خود بردارند. هنگامی که ناوگان به یک سیاره می رسد، با نیروهای موجود دشمن مبارزه می کند و در صورتی که تعداد واحدهای دشمن از تعداد واحد های فضاپیمای بازیکن کمتر باشد، بازیکن مالک آن سیاره می شود. اگر سیاره در حال حاضر به بازیکن تعلق دارد، ناوگان ورودی به عنوان تقویت به آن افزوده می شود. در هر زمان گام، نیروها در هر سیاره متعلق به بازیکن یا دشمن (نه سیاره های "خنثی") با توجه به سرعت رشد این سیاره افزایش می یابد. بنابراین، هدف طراحی یک تابع است که وضعیت نقشه را در هر شبیه سازی در نظر بگیرد و اقدامات لازم را برای به دست آوردن مزیت بیشتر نسبت به دشمن، و در نهایت، برنده شدن در بازی را تعیین کند .

محدودیت های مورد نظر در چالش AI گوگل (ربات نمی تواند هیچ اطلاعاتی را از یک نوبت به یک دیگر منتقل کند و محدودیت زمانی یک ثانیه) برای اجرای یک رویکرد (متاهیوریستیک) فراشناختی دشوار است. بنابراین برای این طراحی، الگوریتم تکاملی (EA)، الگوریتم ژنتیک (GA) پیشنهاد میشود.

معرفی روش‌ها متا–هیوریستیک (فرا مکاشفه‌ای)

روش‌های متا–هیوریستیک مانند الگوریتم ژنتیک (GA) ، الگوریتم بهینه سازی کلونی مورچگان (ACO) و الگوریتم بهینه‌سازی ذرات در طول دو دهه گذشته نه فقط در بین دانشمندان علوم کامپیوتر بلکه در میان سایز فیلدها نیز محبوب شده‌اند. به خاطر کارهای علمی زیادی که در این زمینه انجام شده است، این روش‌های بهینه سازی در فیلدهای گوناگون تحصیلی اضافه شده‌اند. یک سوالی که در اینجا پیش می‌آید این است که چرا روش‌های متا–هیورستیک تا این حد معمول شده‌اند. و پاسخ به این سوال میتواند در چهار دلیل اصلی خلاصه شود که عبارت‌اند از : راحتی ، قابلیت انعطاف ، مکانیزم غیرقابل مشتق سازی آن‌ها و همین طور جلوگیری از گیر افتادن در بهینه‌های محلی.

به طور کلی متا–هیوریستیک‌ها به دو دسته کلی تقسیم میشوند : براساس یک راه حل و مبتنی بر جمعیت.

در روش مبتنی بر راه‌حل ، فرآیند جستجو با یک راه‌حل کاندید شروع میشود. و آن تک راه‌حل در هر مرحله بهبود میابد. اما روش‌های مبتنی بر جمعیت بهبود را در براساس یک مجموعه‌ای از جواب‌ها (جمعیت) اعمال میکنند. در این حالت فرآیند جستجو با یک جمعیت اولیه داده شده شروع میشود (چندین راه‌حل) و این جمعیت در هر تکرار بهبود میابد. متا–هیوریستیک‌های مبتنی بر جمعیت یک سری مزیت‌ها در مقایسه با الگوریتم‌های تک راه حلی دارند:

- روش چند راه‌حلی اطلاعات را درباره فضای جستجو به اشتراک میگذارد که باعث پرش‌های ناگهانی به سمت قسمت‌های امیدارکننده فضای جستجو بشود.
- روش راه‌حلی به یکدیگر کمک میکنند تا از گیر افتادن در مینیمم‌های محلی راه حل‌ها جلوگیری شود.
- متا–هیوریستیک‌های مبتنی بر جمعیت به صورت عمومی حالت اکتشافی بیشتری در مقایسه با روش‌های تک راه‌حلی دارد.

حال به معرفی موارد به کار گرفته شده در مقاله میپردازیم:

2. الگوریتم بهینه سازی گرگ خاکستری(GWO)

در ابتدای توضیح الگوریتم بهینه‌سازی گرگ خاکستری خوب است به این موضوع هم اشاره کنیم که این الگوریتم را دو برادر ایرانی به نام سید علی و سید محمد میرجلالی با همکاری استاد خود با نام اندرو لوییس از دانشگاهی در استرالیا ابداع کرده‌اند. روش بسیار جالبی است که بر مبنای زندگی اجتماعی گرگ‌های خاکستری و نحوه شکار آن‌ها عمل میکند.

علت این که ما تصمیم گرفتیم از این الگوریتم استفاده کنیم شباهت بسیار زیاد نحوه عملکرد بازی Planet Wars با نحوه عمل این الگوریتم است به گونه‌ای که به راحتی میتوان این الگوریتم را با Planet Wars مطابقت داد.

در این بخش در ابتدا الهام بخشی روش گرگ خاکستری مطرح میشود. و سپس مدل ریاضی آن بیان میشود.

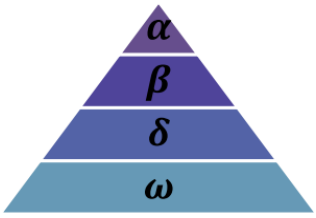
الهام بخشی

گرگ خاکستری به خانواده سگ سانان تعلق دارد. گرگ‌های خاکستری از بهترین شکارچیان به حساب می‌آیند. به این معنی است که آن‌ها در راس هرم غذایی هستند. گرگ‌های خاکستری بیشتر ترجیح میدهند تا به صورت گروهی زندگی کنند. و سایز گروه آن‌ها به طور متوسط بین 5 تا 12 است. به عنوان یک مثال گرگ‌های خاکستری یک سلسله مراتب بسیار سخت‌گیرانه مبتنی بر برتری دارند که در شکل1 نشان داده شده است. رهبران گروه نرها و یا ماده‌هایی هستند که آلفا نامیده میشوند. آلفاها عموما مسئول هستند تا در مورد شکار، مکان خواب، زمان بیدار شدن و ... تصمیم گیری کنند. و تصمیمات آلفاها به اعضای گروه دیکته میشود. هرند که بعضی رفتارهای دموکراتیک نیز مشاهده میشود که در آنها آلفاها سایر گرگ‌های دسته را دنبال میکنند. در گردهمایی‌های آن‌ها سایر گرگ‌ها، آلفاها را با بالابردن دست خود تصدیق میکنند. همچنین گرگ آلفا گرگ مافوق نیز نامیده میشود چون دستوراتش باید به وسیله دسته دنبال شود[46]. گرگ‌های آلفا همچنین فقط اجازه دارند تا در دسته خودشان جفت‌گیری کنند. نکته‌ای که جالب است این است که گرگ آلفا لزوما قوی‌ترین عضو گروه نیست اما بهترین عضو از لحاظ مدیریت گروه است. این موضوع نشان میدهد که تشکیلات، نظم و انظباط گروه بسیار از قدرت و زور مهم‌تر است.

سطح بعدی سلسله مراتب گرگ‌های خاکستری، بتا هست. بتا گرگ‌های زیردست هستند آلفا را در تصمیم‌گیری و سایر فعالیت‌های گروهی یاری میدهند. گرگ‌های بتا میتوانند نر یا ماده باشند، و همچنین بتا میتواند بهترین کاندید برای آلفا شدن باشد در حالتی که یکی از گرگ‌های آلفا بمیرد یا پیر شود. گرگ بتا باید به آلفا احترام بگذارد، اما به سایر گرگ‌های زیر دست درون دسته دستور میدهد. بتا نقش یک نصیحت‌کننده را برای آلفا و نقش نظم‌دهنده را برای دسته بازی میکند. بتا همچنین دستورات آلفا را در گروه پخش میکند و به اطلاع همه میرساند و فیدبکی که از گروه دریافت میکند را به اطلاع آلفا میرساند.

پایین‌ترین رتبه در دسته گرگ‌های هاکستری امگا نام دارد. امگا نقش قربانی را بازی میکند. امگا باید همواره گرگ‌های مافوق خود را تصدیق کند. آنها آخرین گرگ‌هایی هستند که اجازه دارند غذا بخورند. شاید اینطور به نظر برسد که امگا به تنهایی اهمیت چندانی در گروه نداشته باشد، اما مشاهده شده است که کل گروه با درگیری‌های داخلی و مشکلات زیادی موقع از دست دادن امگا مواجه میشود. این موضوع به خاطر تخلیه خشونت، ناامیدی و ناکامی تمامی گرگ‌ها به وسیله امگا(ها) است. این موضوع به رضایت‌مندی تمام دسته و پایداری ساختار تسلط در گروه کمک بسیاری میکند. در بعضی از مواقع نیز گرگ‌های امگا نقش نگهداری از بچه‌های گروه را بر عهده دارند.

اگر یک گرگ آلفا، بتا یا امگا نباید در نتیجه اون گرگ، گرگ تابع نامیده میشود (و در بعضی از منابع به آن‌ها دلتا نیز گفته میشود.). گرگ‌های دلتا باید آلفاها و بتاها را تایید کنند اما آنها بر امگاها غالب هستند و برآنها حکم فرما هستند. دیدبانان،ارشدها، مراقبان، کشیک‌ها، محافظان و شکارچیان به این دسته تعلق دارند. دیدبانان وظیفه دارند تا مرزهای قلمرو را دیدبانی بدهند و به گروه را در صورت خطر هشدار بدهند. نگهبانان از گروه محافظت میکنند و سلامت گروه را تضمین میکنند. ارشدها گرگ‌های باتجربه‌ای هستند قرار است آلفا یا بتا شوند. شکارچیان موقع شکار طعمه گرگ‌های آلفا و بتا را کمک میکنند و برای گروه غذا فراهم میکنند. و مراقبان نیز وظیفه دارند تا به ضعیف‌ها، زخمی‌ها و بیماران گروه کمک کنند و یا آن‌ها را حمل کنند.



شکل 1: سلسه مراتب گرگ‌های خاکستری (رتبه آن‌ها از بالا به پایین کاهش میابد).

علاوه بر سلسله مراتب اجتماعی گرگ‌ها، شکار گروهی یک رفتار اجتماعی جالب گرگ‌های خاکستری است.

فازها و قسمت‌های اصلی شکار گرگ‌های خاکستری عبارت‌اند از :

- پیگیری، تعقیب و نزدیک شدن به طعمه
- دنبال‌کردن، محاصره، خسته‌کردن و آزار و اذیت طعمه تا از حرکت باز بایستند.
- حمله کردن به سمت طعمه

این مراحل در شکل2 نشان داده شده اند.

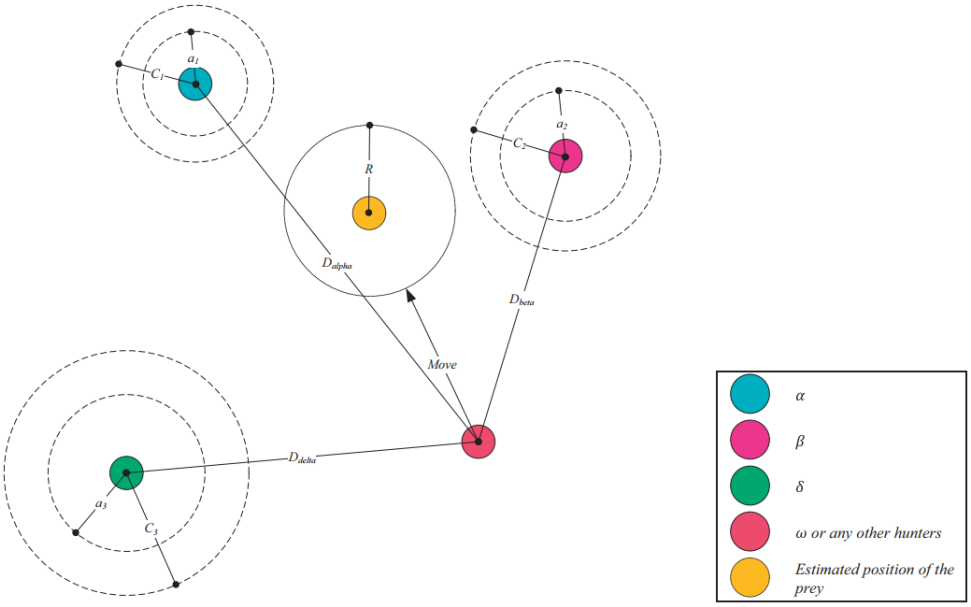
سلسله مراتب اجتماعی گرگ‌های خاکستری

برای این که به صورت ریاضی سلسله مراتب اجتماعی گرگ‌ها را موقعی که الگوریتم گرگ خاکستری را طراحی میکنیم، مدل کنیم، ما مناسب‌ترین راه حل را به عنوان آلفا (α). در نتیجه دومین و سومین بهترین راه‌حل به‌دست آمده بتا (β) و دلتا (δ) معرفی میشوند. و مابقی راه‌حل‌های کاندیدا به عنوان امگا (ω) معرفی میشوند. در الگوریتم گرگ خاکستری شکار (بهینه‌سازی) به وسیله α، β و δ انجام میشود. و گرگ‌های ω این سه دسته گرگ را دنبال میکنند.

حمله به طعمه

همان‌طور که در بالا اشاره شد، گرگ‌های خاکستری شکار را با حمله‌کردن به طعمه هنگامی که از حرکت باز می‌ایستد تمام میکنند. برای این که به صورت ریاضی نزدیک شدن به طعمه را مدل کنیم، ما مقدار $\vec{a}$ را کاهش دادیم. توجه کنید که بازه نوسان $\vec{A}$ نیز به وسیله $\vec{a}$ کاهش میابد. به بیان دیگر $\vec{A}$ یک مقدار تصادفی در بازه $[-2\,a.\,2\,a]$ است که a مقدارش از 2 تا 0 در هر تکرار کاهش میابد. وقتی که مقادیر تصادفی در بازه $[-1.1]$ هستند، موقعیت بعدی یک عامل جستجو میتواند در هر موقعیتی بین موقعیت فعلی و موقعیت قرارگیری طعمه باشد. [2]

شکل 2 : رفتار شکاری گرگ‌های خاکستری: (A) دنبال کردن، نزدیک شدن و ردیابی طعمه (B-D) تعقیب کردن، آزار و اذیت و محاصره کردن (E) متوقف کردن طعمه و حمله کردن



شکل 3 : به‌روزرسانی موقعیت در GWO

با متغیرهایی که تاکنون مطرح شده‌اند، الگوریتم GWO به عامل‌های جستجویش اجازه میدهد تا موقعیتشان را براساس محل قرارگیری آلفا، بتا و دلتا به‌روزرسانی کنند؛ و به سمت طعمه حمله کنند. هرچند، الگوریتم GWO به ایستایی در راه‌حل‌های محلی

با این متغیرها تمایل دارد. این موضوع درست است که روش محاصره‌کردن بیان شده اکتشاف را تا یم حدی نشان میدهد، اما GWO به متغیرهای بیشتری نیاز دارد تا به اکتشاف اهمیت بدهد.

جستجو برای طعمه (اکتشاف)

گرگ‌های خاکستری بیشتر بر طبق موقعیت آلفا، بتا و دلتا جستجو میکنند. آن‌ها از یک دیگر دور میشوند تا به جستجوی طعمه بپردازند و به یکدیگر نزدیک میشوند تا به طعمه حمله کنند. برای این که دورشدن را به صورت ریاضی مدل کنیم، ما $\vec{A}$ را با مقادیر تصادفی بیشتر از 1 یا کمتر از -1 به کار گرفتیم تا عامل جستجو را وادار کنیم که از طعمه دور شود. این موضوع به اکتشاف تاکید میکند و به الگوریتم GWO اجازه میدهد تا به صورت عمومی‌تر جستجو کند. شکل5(b) همچنین نشان میدهد که $|A| > 1$ گرگ‌های خاکستری را مجبور میکند تا از طعمه دور شوند به این امید که یک طعمه مناسب‌تری را پیدا کنند.

یکی دیگر از جزهای GWO که به اکتشاف توجه میکند، $\vec{C}$ هست. همین‌طور که امکان دارد در معادله (3. 4) ببینید، بردار $\vec{C}$ شامل مقادیر تصادفی در بازه $[0.2]$ میباشد. این جز وزن‌های تصادفی‌ای را برای طعمه در نظر میگیرد برای این‌که به صورت تصادفی به $(\vec{C} > 1)$ اهمیت بدهیم یا این‌که از اهمیت تاثیر طعمه $(\vec{C} < 1)$ در تعریف کردن فاصله در معادله (3.1) بکاهیم. این موضوع به GWO کمک میکند تا یک رفتار تصادفی‌تری در سراسر بهینه‌سازی از خود نشان بدهد، و اکتشاف و جلوگیری از مینیمم‌های محلی را دوست دارد. ارزشش را دارد که در اینجا اشاره‌ای کنیم که C به صورت خطی در تضاد با A کاهش نمیابد. ما عمدا به C نیاز داریم تا مقادیر تصادفی را در همه زمان‌ها برای ما فراهم کنند تا به اکتشاف اهمیت بدهیم نه فقط هنگام تکرارهای ابتدایی بلکه حتی در تکرارهای پایانی نیز باید به آن اهمیت بدهیم.

بردار C همچنین میتواند به عنوان نتیجه موانعی که در طبیعت به طعمه نزدیک میشوند، در نظر گرفت. به صورت کلی، موانع در طبیعت در مسیر شکار گرگ‌ها ظاهر میشوند و در حقیقت از نزدیک شدن راحت و سریع آن‌ها به طعمه جلوگیری میکنند. این دقیقا کاری است که بردار C انجام میدهد. بسته به موقعیت گرگ، او میتواند به صورت تصادفی به طعمه یک وزن بدهد و آن را برای رسیدن گرگ‌ها به او سخت‌تر و دورتر بکند. برای خلاصه در الگوریتم GWO فرآیند جستجو با ایجاد یک جمعیت اولیه‌ای از گرگ‌ها شروع میشود (راه‌حل کاندید). در هر مرحله تکرار، گرگ‌های آلفا، بتا و دلتا موقعیت احتمالی طعمه را پیش‌بینی میکنند. هر راه‌حل کاندید فاصله‌اش را تا طعمه به‌روزرسانی میکند. مقدار پارامتر a از 2 به 0 کاهش میابد برای این که به اکتشاف و بهره‌برداری به‌ترتیب اهمیت بدهد. راه‌حل‌های کاندید سعی میکنند تا از طعمه دور شوند وقتی که $|\vec{A}| > 1$ و به سمت طعمه بروند وقتی که $|\vec{A}| < 1$. در نهایت، الگوریتم GWO با رضایت از یک معیار پایانی، خاتمه میابد.

شبه کد الگوریتم GWO در شکل6 معرفی شده است.

برای این که بفهمید که به صورت تئوری چگونه GWO میتواند مسائل بهینه‌سازی را حل کند، یک سری از نکات اشاره شده است:

- سلسله مراتب اجتماعی پیشنهاد شده به GWO کمک میکند تا بهترین راه‌حل به دست آمده تا اینجای در تکرار را ذخیره کند.
- مکانیزم محاصره پیشنهاد شده یک همسایگی دایره‌ای شکلی را اطراف راه‌حل‌هایی که میتواند به ابعاد بالاتری تعمیم داده شود، بهعنوان یک دایره هایپر تعریف میکند.
- پارامترهای تصادفی A و C به راه‌حل‌های کاندید کمک میکند تا یک دایره هایپر با شعاع‌های تصادفی داشته باشیم.
- روش‌های شکار پیشنهاد شده به راه‌حل‌های کاندید اجازه میدهد تا موقعیت احتمالی طعمه را شناسایی کنند.

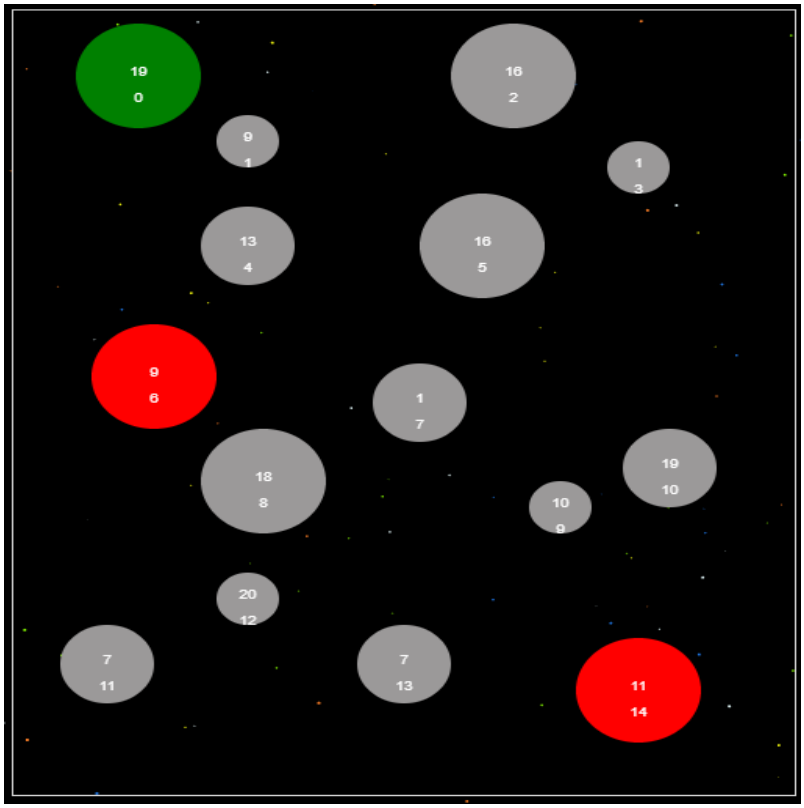
- اکتشاف و بهره‌برداری به‌وسیله مقادیر تطبیقی a و A تضمین میشود.
- مقادیر تطبیقی پارامترهای a و A به GWO اجازه میدهد تا به‌آرامی بین اکتشاف و بهره‌برداری جابه‌جا شود.
- با کاهش A، نصف تکرارها به اکتشاف اختصاص میابد ($|A| \geq 1$) و نصف دیگر به بهره‌برداری اختصاص میابد ($|A| < 1$).
- GWO تنها دو پارامتر اصلی دارد که باید تنظیم شوند ($a. c$).

```
Initialize the grey wolf population Xi (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
Xα=the best search agent
Xβ=the second best search agent
Xδ=the third best search agent
while (t < Max number of iterations)
    for each search agent
            Update the position of the current search agent by equation (3.7)
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update Xα, Xβ, and Xδ
    t=t+1
end while
return Xα
```

شکل 4 : شبه کد الگوریتم GWO

3. عملیات ربات گرگ خاکستری

همان‌طور که در الگوریتم گرگ‌خاکستری ما توضیح دادیم در اینجا نیز میخواهیم از شیوه شکار گرگ خاکستری برای طعمه، برای گرفتن سیاره‌ها استفاده کنیم. همان طور که در شکل6 مشاهده میکنید ما در نسخه طراحی شده خود سه نوع سیاره داریم که در اینجا سیاره قرمز که دشمن محسوب میشود درونش از الگوریتم گرگ خاکستری استفاده شده است. سیاره‌های خاکستری رنگ، سیاره‌هایی هستند که هنوز تسخیر نشده‌اند و همچنین سیاره‌های سبز رنگ ، سیاره‌هایی هستند که بازیکن آن‌ها را در اختیار دارد.



شکل 5 : در این شکل سیاره‌های قرمز رنگ که در آن از گرگ خاکستری استفاده شده است و همچنین خاکستری که هنوز مالکیت ندارند و سیاره سبز که در اختیار بازیکن است را میتوانید مشاهده کنید.

خب در ادامه خوب است ساز و کار و عملکرد گرگ خاکستری را در این بازی توضیح دهیم.

نحوه عملکرد

خب همان‌طور که پیش‌تر نیز به آن اشاره شد در بازی Planet Wars تیمی (از بین تیم‌های قرمز و سبز) برنده میشود که بتواند تمامی سیاره‌های دشمن را بگیرد و سیاره‌ای را در اختیار دشمن باقی نگذارد. خب ما در این‌جا پس از چند بار انجام بازی به این نتیجه رسیدیم که برای پیاده سازی الگوریتم گرگ خاکستری در این بازی راهی که وجود دارد این است که بیاییم و الگوریتم را به این شکل پیاده‌سازی کنیم که بسته به شرایط هوش‌مصنوعی (سیاره قرمز رنگ) بیاید و درصورتی که ما هنوز سیاره‌ای را نگرفته‌ایم سیاره‌های خاکستری (سیاره‌هایی که هنوز مالکیتی ندارند) با بزرگترین اندازه و کمترین تعداد نیروی پیش‌فرض درون آن‌ها را تصرف کند زیرا این سیاره‌ها تولید نیروی بیشتری برای ما در مقایسه با سیاره‌های کوچکتر دارند. و درصورتی که تیم مقابل هوش مصنوعی (سیاره‌های سبز رنگ) اقدام به تصرف سیاره‌ای کرده بودند بیاید و با بررسی آن سیاره و نیز سیاره‌های بدون مالکیت (خاکستری رنگ) باقی مانده ببیند کدام یک بهره‌وری بیشتری دارد و بیاید و آن یکی را تصرف کند. و کلا هدف ما با استفاده از این الگوریتم این است که در کمترین زمان ممکن بتوانیم سیاره‌ها را در اختیار بگیریم و بر بازیکن انسانی پیروز شویم.

در این استراتژی به این صورت عمل میکنیم که از آن سیاره ای که در ابتدای بازی در آن قرار داریم نیروها را به سمت سیاره دشمن هدایت می نماییم و سعی در تصرف این سیاره اولیه میکنیم . به دلیل آنکه دشمن سیاره اولیه خود را برای تصرف دیگر سیاره ها خالی کرده است تصرف آن راحت تر به نظر میرسد.

سپس به کمک سیاره های جدید و نیرو های آن شروع به محاصره مابقی سیاره هایی میکنیم که دشمن در آن قرار دارد برای این منظور از سیاره های خود به سمت سیاره دشمن که نزدیک تر است و اندازه بزرگ تری دارد (به این علت که تولید مثل در سیاره هایی که اندازه بزرگ تری دارد بیشتر است .) نیرو ارسال میکنیم. پس از تصرف یکی از سیارات به این شکل ادامه میدهیم تا دشمن در یک محدوده محاصره شود و چنانچه دشمن سعی در تصرف سیاره ای که خارج از محدوده ما است نمود نیروها را به آن سیاره ارسال کرده و آن سیاره را از او پس میگیریم و به همین صورت حلقه محاصره را بر او تنگ میکنیم و از تمام سیارات نیرو ها را به سیارات داخل محاصره خود ارسال میکنیم.

این استراتژی منطبق بر بر استراتژی گرگ خاکستری است. گرگ‌های خاکستری نیز در شکار طعمه خود به این صورت عمل میکنند که گرگ‌های آلفا حمله را رهبری میکنند و سایر گرگ‌ها از او تبعیت میکنند و حلقه محاصره را تنگ تر کرده و اجازه تحرک طعمه را از او میگیرند تا طعمه در مسیری که آن‌ها میخواهند حرکت کند و پس از خسته کردن طعمه (به دلیل عدم توانایی مانور طعمه) حمله نهایی را آغاز میکنند.

توضیح الگوریتم به کار برده شده

در ادامه ما انواع توابعی را که در این مسئله به کار برده‌ایم را توضیح میدهیم:

در ابتدای آغاز شروع به کار بازی، شروع میکند به ساختن یک جمعیت اولیه به صورت کاملا تصادفی در فضای مشخص شده از سیاره‌ها و این سیاره‌ها در سه اندازه شامل : کوچک، بزرگ و متوسط هستند.

پس از این کار شروع میکند به محاسبه شایستگی هر یک از عضوهای جمعیت براساس اساس معیاری که برای آن تعریف کرده‌ایم.

تمامی مقادیری که در ادامه اشاره میکنم تا زمانی که یکی از طرفین قرمز یا سبز در بازی پیروز نشود، ادامه پیدا میکند.

اول از همه که در یک فاصله زمانی مشخصی، مقادیر نیروهای موجود در داخل هر سیاره را بسته به اندازه‌ای که دارد محاسبه میکند و سپس یک تابعی اجرا میشود که برنده شدن یکی از طرفین را چک میکند و در صورتی که یکی از طرفین پیروز شده بود بازی خاتمه میابد.

سپس به ازای هر تغییری که در چینش نیروها صورت میگیرد تابع fitness برای محاسبه شایستگی تمام اعضا موجود فراخوانی میشود تا سیاره بعدی‌ای را برای حمله به آن مشخص کند و تابع شایستگی براساس تعداد نیروهایی که در آن است و اندازه سیاره و میانگین فاصله‌ای که از نیروهای ما دارد محاسبه میشود.

پس از محاسبه تابع شایستگی حمله انجام میشود و تمامی سیارات نیروهای خود را به سیاره‌ای که بیشترین شایستگی را دارد ارسال میکنند.

سپس مجددا موقعیت‌ها را به روزرسانی کرده و شایستگی‌های جدید را محاسبه میکند و به مرحله بعدی اجرای الگوریتم میرود.

Initialize the planets populations $X_i (i = 1, 2, 3, \dots, n)$

Calculate the fitness of each planet for attack at the start of the game $F_i (i = 1, 2, 3, \dots, n)$

Initiate the assault, based on the fitness of the planets that we have calculated earlier

While (none of the players have won)

Update the plenty of the forces in each planet

Calculate and update the new fitness based on the new situation

Choose the best planet for attack

Initiate the attack

Check for winner

End while

Return the winner

شکل 6 : شبه کد پیاده سازی انجام شده از بازی Planet Wars

پیاده‌سازی بازی جنگ‌های سیاره‌ای (Planet Wars)

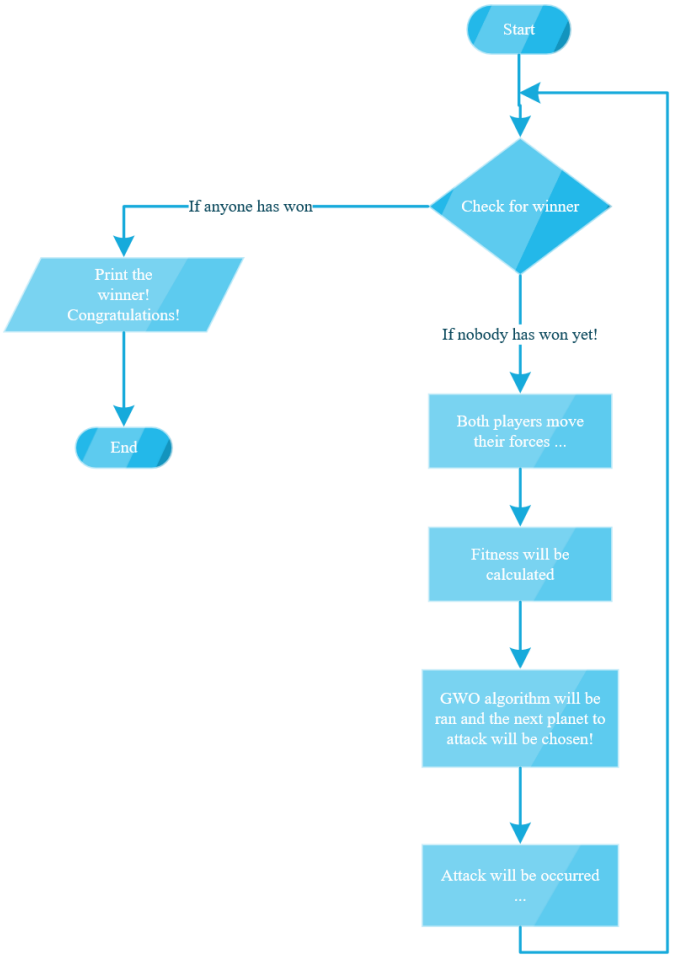
ما این بازی را برای این که بتوانیم به آن ظاهر گرافیکی خوبی بدهیم و پیاده‌سازی آن نیز زمان زیادی نگیرد، تصمیم گرفتیم که با استفاده از HTML, CSS, JavaScript پیاده سازی کنیم. خوبی این زبان‌ها این است که پیاده‌سازی سریعی دارند و به ما ظاهر خوبی را در کمترین زمان میدهند و قابلیت انعطاف بالایی دارند و نیز میتوان آن را به سارگی در مرورگر وب هر سیستمی بدون نیاز به نصب نرم‌افزار خاصی اجرا کرد.

ما نیز کدهای نوشته شده را در گیت‌هاب که یک سیستم مدیریت محتوا متن باز است بازگذاری کردیم به آدرس زیر که به راحتی میتوانید به آن دسترسی داشته باشید:

https://github.com/parhamzm/Planet-Wars_GWO

پس از دریافت فایل‌ها از گیت کافی است که فایل main.html را اجرا کنید تا بازی در صفحه مرورگر شما اجرا شود و در مرورگرهایی که ما تست کردیم مرورگرهای گوگل کروم(Google Chrome) و فایرفاکس(FireFox) بود و پیشنهاد میکنیم که شما نیز از همین مرورگرها استفاده کنید.

پس از شروع بازی مشاهده میکنیم که بازیکن قرمز رنگی که با الگوریتم گرگ خاکستری کار میکند به سرعت شروع به اجرا شدن و عمل کردن میکند و سیاره‌های بدون مالکیت با تعداد نیروهای کمتر را تصرف میکند و به وقتی که ما نیز اقدامی را انجام میدهیم و سیاره‌ای را درصدد تصرفش برمی‌آییم سریع وارد عمل میشود و آن را از ما میگیرد و به ما اجازه نمیدهد که قلمرومان را گسترش دهیم و حلقه محاصره را بر ما تنگ‌تر میکند و در لحظه آخر نیز حمله نهایی را به ما آغاز میکند و سیاره اصلی طرف مقابل (سبز رنگ‌ها) را میگیرد و بازی را میبرد.



شکل 7: فلوچارت نحوه عملکرد بازی Planet Wars با استفاده از الگوریتم GWO

ما در حدود 30 باری که بازی کردیم فقط یک بار تونستیم از هوش مصنوعی تقویت شده با گرگ خاکستری برنده شویم و تمام سیاره‌های قرمز رنگ را بگیریم و این نکته بهبود بسیار خوبی را در مقایسه با روش‌های پیشین استفاده شده مانند ربات Google AI و نیز AresBot و نیز GeneBot نشان میدهد. که برای این که بتوانید بازی را در نسخه موبایل که با این روش‌ها پیاده سازی شده است ببینید پیشنهاد میکنم که از طریف لینک زیر آن را دانلود کنید (البته این لینک برای سیستم عامل اندروید است):

https://play.google.com/store/apps/details?id=com.chris.pwars

****

4. منابع

[1]. Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: from natural to artificial systems: OUP USA; 1999.

[2]. Dorigo M, Birattari M, Stutzle T. Ant colony optimization. Comput Intell Magaz, IEEE 2006;1:28–39.

[3]. Kennedy J, Eberhart R. Particle swarm optimization, in Neural Networks, 1995. In: Proceedings, IEEE international conference on; 1995. p. 1942–1948.

[4]. Wolpert DH, Macready WG. No free lunch theorems for optimization. Evolut Comput, IEEE Trans 1997;1:67–82

[5]. Kirkpatrick S, Jr. DG, Vecchi MP. Optimization by simulated annealing. Science, vol. 220; 1983. p. 671–80.

[6]. Beni G, Wang J. Swarm intelligence in cellular robotic systems. In: Robots and biological systems: towards a new bionics?, ed. Springer; 1993. p. 703–12.

[7]. Basturk B, Karaboga D. An artificial bee colony (ABC) algorithm for numeric function optimization. In: IEEE swarm intelligence symposium; 2006. p. 12–4.

[8]. Olorunda O, Engelbrecht AP. Measuring exploration/exploitation in particle swarms using swarm diversity. In: Evolutionary computation, 2008. CEC 2008 (IEEE World Congress on Computational Intelligence). IEEE Congress on; 2008. p. 1128–34

[9]. Liang J, Suganthan P, Deb K. Novel composition test functions for numerical global optimization. In: Swarm intelligence symposium, 2005. SIS 2005. Proceedings 2005 IEEE; 2005. p. 68–75.

[10]. Mirjalili S, Lewis A. S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. Swarm Evolut Comput 2013;9:1–14

[11]. van den Bergh F, Engelbrecht A. A study of particle swarm optimization particle trajectories. Inf Sci 2006;176:937–71

[12]. Coello Coello CA, Mezura Montes E. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 2002;16:193–203

[13]. He Q, Wang L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 2007;20:89–99

[14]. Mezura-Montes E, Coello CAC. An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 2008;37:443–73