موسسه آموزشی عقیق

عنوان پروژه

شبیه‌سازی و تحلیل یک سیستم چندعاملی برای ناوبری و اجتناب از برخورد در محیط مشترک

Simulation and Analysis of a Multi-Agent System for Navigation and Collision Avoidance in a Shared Environment

درس

سیستم‌های چندعاملی پیشرفته

نام دانشجو

علی بهادرانی باغبادرانی

٤٠٢٠٠٢٦٥

فهرست مطالب

- ۷.۱ تفسیر نتایج
- ۷.۲ شناسایی نقاط قوت سیستم
- ۷.۳ شناسایی نقاط ضعف و محدودیت‌ها

۸. نتیجه‌گیری و کارهای آینده

- ۸.۱ جمع‌بندی دستاوردها
- ۸.۲ پیشنهادها برای توسعه آتی

۹. منابع

۱. چکیده

این پروژه به طراحی، پیاده‌سازی و تحلیل یک سیستم چندعاملی (Multi-Agent System - MAS) می‌پردازد که در آن مجموعه‌ای ازAgent های خودمختار در یک محیط اشتراکی به سمت یک هدف مشترک حرکت می‌کنند. چالش اصلی در این سیستم، مدیریت حرکت همزمان Agent ها و جلوگیری از برخورد آن‌ها با یکدیگر به صورت غیرمتمرکز است. سیستم شبیه‌سازی شده شامل یک محیط دوبعدی گسسته (Grid-based) است که در آنAgent ها با استفاده از یک منطق حرکتی ساده و یک مکانیزم واکنشی (Reactive) برای اجتناب از برخورد، مسیر خود را به سمت هدف تعیین می‌کنند.

یکی از ویژگی‌های کلیدی این پروژه، سیستم تصویرسازی بلادرنگ آن است که نه تنها موقعیت و مسیر حرکت Agentها را نمایش می‌دهد، بلکه یک داشبورد آماری برای ردیابی معیارهای عملکرد کلیدی سیستم، مانند تعداد کل قدم‌های برداشته شده، تعداد برخوردها و میانگین فاصله تا هدف را فراهم می‌کند. این پروژه با استفاده از زبان برنامه‌نویسی پایتون و کتابخانه‌های NumPy برای محاسبات عددی و Matplotlib برای تصویرسازی پیاده‌سازی شده است. نتایج حاصل از شبیه‌سازی، نمایشی از مفاهیم بنیادین در سیستم‌های چندعاملی از جمله رفتار خودمختار، هماهنگی ضمنی (Implicit Coordination) و ظهور رفتارهای جمعی از قواعد محلی ساده را به تصویر می‌کشد و بستری برای تحلیل و ارزیابی استراتژی‌های ناوبری و اجتناب از برخورد فراهم می‌آورد.

کلیدواژه‌ها: سیستم چندعاملی، شبیه‌سازی، اجتناب از برخورد، برنامه‌ریزی مسیر، رفتار خودمختار، تصویرسازی بلادرنگ

۲. اهداف کلی

اهداف اصلی و کلی این پروژه به شرح زیر تعریف شده‌اند:

- طراحی و پیاده‌سازی یک شبیه‌ساز **MAS** : ساخت یک سیستم کامل از ابتدا که قادر به شبیه‌سازی تعاملات بین چندین Agent در یک محیط کنترل‌شده باشد.
- مدل‌سازی رفتار خودمختار: پیاده‌سازی Agent هایی که قادرند به صورت مستقل و بر اساس وضعیت فعلی خود و اطلاعات محدود از محیط، تصمیم‌گیری کنند.
- تحقیق و پیاده‌سازی مکانیزم اجتناب از برخورد: پیاده‌سازی یک الگوریتم کارآمد اما ساده برای تشخیص و مدیریت برخوردهای احتمالی بین Agent ها در محیط اشتراکی.
- توسعه ابزار تصویرسازی و تحلیل: ساخت یک سیستم نمایش بلادرنگ که به درک شهودی دینامیک سیستم کمک کرده و معیارهای عملکرد کمی را به صورت زنده نمایش دهد.
- ارزیابی عملکرد سیستم: تحلیل کارایی سیستم تحت شرایط مختلف (مانند تغییر تعداد Agent ها) با استفاده از معیارهای کمی تعریف‌شده مانند تعداد برخوردها، طول مسیر و زمان رسیدن به هدف.
- ایجاد یک بستر قابل توسعه **:**فراهم آوردن یک چارچوب نرم‌افزاری ماژولار که بتواند به عنوان پایه‌ای برای تحقیقات آتی در زمینه الگوریتم‌های پیشرفته‌تر هماهنگی و یادگیری در سیستم‌های چندعاملی مورد استفاده قرار گیرد.

۳. مقدمه و بیان مسئله

۳.۱ مقدمه‌ای بر سیستم‌های چندعاملی **(MAS)**

سیستم‌های چندعاملی (MAS) یکی از شاخه‌های پرکاربرد و مهم در هوش مصنوعی هستند که به مطالعه سیستم‌هایی متشکل از چندین موجودیت محاسباتی مستقل به نام "Agent" می‌پردازند. این Agent ها در یک محیط مشترک قرار دارند و برای دستیابی به اهداف فردی یا گروهی خود با یکدیگر و با محیط تعامل می‌کنند. ویژگی کلیدی این

سیستم‌ها، عدم وجود کنترل متمرکز است؛ هر Agent دارای درجه‌ای از خودمختاری است و تصمیمات خود را بر اساس اطلاعات محلی و قوانین داخلی خود اتخاذ می‌کند. این پارادایم امکان حل مسائلی را فراهم می‌کند که به دلیل مقیاس، پیچیدگی یا توزیع‌شدگی ذاتی، با روش‌های متمرکز سنتی به راحتی قابل حل نیستند. کاربردهای MAS در حوزه‌های متنوعی از جمله رباتیک تیمی (Swarm Robotics)، مدیریت زنجیره تأمین، کنترل ترافیک هوایی و زمینی، بازارهای مالی الکترونیکی و شبکه‌های اجتماعی دیده می‌شود.

۳.۲. چالش‌های کلیدی در سیستم‌های چندعاملی

طراحی و مدیریت MAS با چالش‌های منحصربه‌فردی همراه است. یکی از اساسی‌ترین این چالش‌ها، مسئله "هماهنگی (Coordination) "است. چگونه می‌توان اطمینان حاصل کرد که مجموعه‌ای ازAgent های خودمختار به گونه‌ای عمل می‌کنند که نتیجه کلی سیستم مطلوب باشد، در حالی که هر Agent تنها دانش و دید محدودی نسبت به کل سیستم دارد؟ این چالش به مسائل فرعی دیگری تجزیه می‌شود:

- تخصیص وظایف**(Task Allocation)** : چگونه وظایف بینAgent ها تقسیم شود؟
- اشتراک منابع**(Resource Sharing)** : چگونه از منابع محدود و مشترک به صورت بهینه استفاده شود؟
- انسجام**(Coherence)** : چگونه از سازگاری رفتارAgent ها اطمینان حاصل شود؟
- اجتناب از تداخل**(Conflict Avoidance)** : چگونه از بروز وضعیت‌های نامطلوب مانند برخورد فیزیکی یا بن‌بست (Deadlock) جلوگیری شود؟

۳.۳. بیان دقیق مسئله: ناوبری غیرمتمرکز

این پروژه بر روی یکی از مسائل کلاسیک و بنیادین در MAS، یعنی «مسئله ناوبری چندعاملی و اجتناب از برخورد» (Multi-Agent Navigation and Collision Avoidance)، تمرکز دارد. مسئله را می‌توان به صورت زیر تعریف کرد:

مجموعه‌ای متشکل از N عدد *Agent* در یک محیط دوبعدی و محدود قرار دارند. تمام*Agent* ها یک هدف مشترک و از پیش‌تعیین‌شده دارند که باید به آن برسند. هر *Agent* موقعیت فعلی خود و موقعیت هدف را می‌داند اما از مسیر یا تصمیمات آنی سایر*Agent* ها آگاهی کامل ندارد. حرکت*Agent* ها به صورت همزمان (در گام‌های زمانی گسسته) صورت می‌گیرد. چالش علمی پروژه، طراحی یک استراتژی تصمیم‌گیری غیرمتمرکز برای هر *Agent*است که دو هدف اصلی را برآورده سازد:

۱. اثربخشی**(Effectiveness)** : هر Agent باید به سمت هدف حرکت کرده و در نهایت به آن برسد یا تا حد امکان به آن نزدیک شود.
۲. ایمنی**(Safety)** :Agent ها باید از برخورد با یکدیگر در طول مسیر اجتناب کنند.

این پروژه یک مدل ساده‌شده از این مسئله را شبیه‌سازی می‌کند. در این مدل، «برنامه‌ریزی مسیر» به یک استراتژی حریصانه ساده تقلیل یافته و "اجتناب از برخورد" به صورت یک مکانیزم واکنشی پیاده‌سازی می‌شود. هدف، بررسی دینامیک‌های حاصل از این قوانین ساده و ارزیابی عملکرد کلی سیستم در دستیابی به اهداف فوق است.

٤. اهمیت عملی و نظری پروژه

این پروژه علی‌رغم سادگی ظاهری، دارای اهمیت نظری و عملی قابل توجهی است که در ادامه تشریح می‌شود.

۴.۱. اهمیت نظری

از منظر نظری، این پروژه یک آزمایشگاه محاسباتی برای مطالعه مفاهیم بنیادین «سیستم‌های پیچیده» (Complex Systems)و «حیات مصنوعی» (Artificial Life) فراهم می‌کند.

- رفتار**(Emergent Behavior)** : این شبیه‌سازی به وضوح نشان می‌دهد که چگونه تعاملات محلی و قوانین ساده در سطح Agent (مانند «به سمت هدف حرکت کن» و «اگر مانعی وجود دارد، توقف کن»)

می‌تواند منجر به ظهور الگوهای رفتاری پیچیده و هماهنگ در سطح کلان (مانند تشکیل مسیرهای جریان یا بروز ترافیک) شود. این پدیده، یکی از موضوعات اصلی در مطالعه سیستم‌های پیچیده است.

- پایه برای الگوریتم‌های پیشرفته :سیستم فعلی به عنوان یک "Baseline" عمل می‌کند. با داشتن عملکرد یک استراتژی ساده، می‌توان الگوریتم‌های پیشرفته‌تر (مانند الگوریتم‌های مبتنی بر یادگیری تقویتی، مذاکره یا برنامه‌ریزی مسیر پیشرفته مانند (A* را پیاده‌سازی کرده و بهبود عملکرد آن‌ها را به صورت کمی و کیفی با نتایج این پروژه مقایسه نمود.
- درک مبانی هماهنگی غیرمتمرکز :این پروژه به درک شهودی از چالش‌ها و مصالحه‌های موجود در سیستم‌های غیرمتمرکز کمک می‌کند. برای مثال، مصالحه بین حرکت سریع به سمت هدف و نیاز به توقف برای جلوگیری از برخورد، یک نمونه کلاسیک از تقابل بین اهداف فردی و ایمنی گروهی است.

۴.۲ اهمیت عملی

مفاهیم شبیه‌سازی شده در این پروژه، مدل ساده‌شده‌ای از بسیاری از کاربردهای دنیای واقعی هستند.

- لجستیک و انبارداری هوشمند :سیستم‌های رباتیک مانند Kiva در انبارهای آمازون، از هزاران ربات خودمختار برای جابجایی قفسه‌ها استفاده می‌کنند. این ربات‌ها باید در فضایی محدود و بدون برخورد با یکدیگر، مسیر خود را پیدا کنند. این پروژه، هسته اصلی این چالش را مدل‌سازی می‌کند.
- مدیریت ترافیک خودروهای خودران :در آینده، خودروهای خودران باید بتوانند در تقاطع‌ها و بزرگراه‌ها بدون نیاز به کنترل متمرکز و با ارتباط با یکدیگر، به صورت ایمن حرکت کنند. شبیه‌سازی تعاملات بین Agentها در یک محیط مشترک، گام اول در طراحی چنین سیستم‌هایی است.
- عملیات جستجو و نجات با پهپادها :یک گروه (Swarm) از پهپادها ممکن است برای نقشه‌برداری از یک منطقه آسیب‌دیده اعزام شوند. آن‌ها باید منطقه را پوشش دهند و در عین حال از برخورد با یکدیگر در آسمان اجتناب کنند.
- صنعت سرگرمی و بازی‌های ویدیویی :هوش مصنوعی شخصیت‌های غیرقابل بازی (NPCs) در بازی‌های کامپیوتری اغلب از الگوریتم‌های مشابهی برای حرکت در محیط بازی و تعامل با یکدیگر استفاده می‌کنند.

بنابراین، این پروژه نه تنها یک تمرین آکادمیک مفید است، بلکه یک ابزار آموزشی و تحقیقاتی قدرتمند برای راه‌حل‌های مسائل عملی در دنیای فناوری محسوب می‌شود.

۵. روش‌شناسی(Methodology)

این بخش به تشریح دقیق معماری سیستم، مدل‌های ریاضی و الگوریتم‌های استفاده شده و ابزارهای پیاده‌سازی می‌پردازد.

۵.۱. معماری کلی سیستم

سیستم از چهار جزء اصلی و ماژولار تشکیل شده است که در فایل‌های جداگانه سازماندهی شده‌اند تا توسعه و نگهداری آن ساده باشد:

۱. محیط **(Environment)** : مسئول تعریف فضای مسئله، مرزها، موقعیت هدف و مدیریت وضعیت کلی جهان شبیه‌سازی است.
۲. سیستم **Agent (`agent.py`)** : شامل کلاس Agent است که منطق تصمیم‌گیری، حرکت، تشخیص برخورد و ردیابی آمارهای فردی هر Agent را پیاده‌سازی می‌کند.
۳. سیستم تصویرسازی **(`visualization.py`)** : مسئول رندر کردن وضعیت فعلی شبیه‌سازی (موقعیتAgent ها و هدف) و نمایش داشبورد آماری به صورت بلادرنگ است.
٤. پیکربندی **(`config.py`)** : یک فایل مرکزی برای تعریف تمام پارامترهای قابل تنظیم سیستم، مانند ابعاد محیط، تعدادAgent ها و تنظیمات ظاهری تصویرسازی.

این اجزا از طریق یک حلقه شبیه‌سازی اصلی (موجود در`main.py`) با یکدیگر تعامل دارند. در هر گام از شبیه‌سازی، حلقه از هر Agent می‌خواهد که حرکت بعدی خود را تعیین کند، برخوردها را بررسی کرده و وضعیت خود را به‌روزرسانی کند. سپس، سیستم تصویرسازی فراخوانی می‌شود تا نمای جدید را ترسیم کند.

۵.۲. مدل‌سازی محیط **(Environment)**

- فضا **:**محیط یک شبکه (Grid) دوبعدی با اندازه $S \times S$ است. مقدار S از طریق پارامتر `ENV_SIZE`در فایل پیکربندی قابل تنظیم است و مقدار پیش‌فرض آن ۲۰ است. هر سلول در این شبکه با یک زوج مرتب (x, y) نمایش داده می‌شود که :

$$x, y < S$$

- هدف **:**یک هدف مشترک برای تمام‌Agent ها در موقعیت ثابت، (S-1, S-1) یعنی گوشه بالا-راست شبکه، تعریف شده است.
- مقداردهی اولیه **:**در ابتدای شبیه‌سازی، تعداد مشخصی Agent (پیش‌فرض: ۵) ایجاد شده و هر کدام در یک موقعیت تصادفی در ربع پایین-چپ شبکه (lower-left quadrant) قرار می‌گیرند. این کار تضمین می‌کند که‌Agent ها مسیری معنادار برای پیمودن به سمت هدف داشته باشند.

۵.۳. مدل‌سازی**Agent**

هر Agent یک شیء (Object) از کلاس `Agent` است و دارای ویژگی‌های زیر می‌باشد:

- `Id`: یک شناسه منحصربه‌فرد.
- `Position`: موقعیت فعلی Agent به صورت یک زوج (x, y)
- `path_history`: لیستی از تمام موقعیت‌های قبلی Agent برای ردیابی مسیر طی شده.
- آمارها**(Statistics)** :
  - `steps_taken`: تعداد کل حرکات انجام شده.
  - `Collisions`: تعداد دفعاتی که حرکت Agent به دلیل وجود Agent دیگر مسدود شده است.
  - `distance_to_goal`: فاصله اقلیدسی یا منهتن تا هدف، که در هر گام به‌روز می‌شود.

۵.۴. الگوریتم حرکت و برنامه‌ریزی مسیر

بر اساس توضیحات پروژه الگوریتم حرکت Agent یک استراتژی حریصانه (Greedy) و قطعی (Deterministic) است. در هر گام زمانی Agent ،سعی می‌کند به یکی از سلول‌های مجاور خود حرکت کند که آن را به هدف نزدیک‌تر می‌کند.

این الگوریتم بسیار سریع است اما هیچ تضمینی برای بهینه بودن مسیر ارائه نمی‌دهد و ممکن است در محیط‌های دارای مانع به راحتی دچار مشکل شود (اگرچه در این پروژه مانعی وجود ندارد).

۵.۵. مکانیزم اجتناب از برخورد:

مکانیزم اجتناب از برخورد به صورت کاملاً واکنشی (Reactive) پیاده‌سازی شده است.

این روش ساده است اما می‌تواند منجر به ناکارآمدی شود؛ به خصوص در تراکم بالایAgent ها که ممکن است منجر به "ترافیک" یا "بن‌بست (Deadlock) "موقت شود، جایی که دو یا چند Agent منتظر حرکت یکدیگر می‌مانند.

۵.۶. سیستم تصویرسازی و داشبورد آمار

سیستم تصویرسازی با استفاده از کتابخانه `matplotlib.animation` ساخته شده است تا یک نمایش زنده از شبیه‌سازی ارائه دهد.

۵.۷. پارامترهای پیکربندی(**config.py**)

این فایل به کاربر اجازه می‌دهد تا رفتار شبیه‌سازی را بدون تغییر در کد اصلی تنظیم کند. پارامترهای کلیدی عبارتند از:

- `ENV_SIZE`: اندازه محیط شبکه.
- `NUM_AGENTS`:تعدادAgent های حاضر در شبیه‌سازی.
- تنظیمات انیمیشن مانند `INTERVAL` برای تعیین سرعت شبیه‌سازی.

- رنگ‌ها و برچسب‌های مورد استفاده در تصویرسازی.

۵.۸. ابزارها و کتابخانه‌های مورد استفاده

انتخاب ابزارها بر اساس نیازمندی‌های پروژه و کارایی آن‌ها صورت گرفته است:

- **Python**: به عنوان زبان اصلی به دلیل سادگی، خوانایی و اکوسیستم غنی از کتابخانه‌های علمی.
- **NumPy**: برای کار با آرایه‌ها و ماتریس‌ها، به خصوص برای نمایش محیط شبکه و انجام محاسبات برداری روی موقعیت Agent ها.
- **Matplotlib**: برای تمام جنبه‌های تصویرسازی، از جمله رسم شبکه، نمایش Agent ها، و ایجاد انیمیشن‌های بلادرنگ.
- برنامه‌نویسی شیءگرا **(OOP)** : برای سازماندهی کد به صورت ماژولار و خوانا، با تعریف کلاس‌های مجزا برای `Agent` و `Environment` .

٦. ارزیابی نتایج

برای ارزیابی کارایی و رفتار سیستم شبیه‌سازی‌شده، از معیارهای کمی مشخصی استفاده می‌شود و نتایج در قالب جداول و نمودارها ارائه می‌گردد. از آنجایی که اجرای مستقیم کد امکان‌پذیر نیست، این بخش نتایج فرضی و نمونه را برای نمایش چگونگی تحلیل ارائه می‌دهد.

۶.۱. معیارهای سنجش عملکرد

معیارهای زیر برای ارزیابی جنبه‌های مختلف سیستم به کار می‌روند:

- مجموع قدم‌های کل **(Total Steps Taken):** مجموع تعداد حرکات موفق تمام Agent ها. این معیار، معیاری از "کار" انجام شده توسط سیستم است.

- مجموع برخوردهای کل **(Total Collisions):** مجموع تعداد دفعاتی که Agent ها به دلیل مسدود بودن مسیر توسط Agent دیگر، از حرکت بازمانده‌اند. این معیار اصلی‌ترین شاخص برای سنجش ناکارآمدی و شکست در هماهنگی است.
- میانگین فاصله تا هدف **(Average Distance to Goal):** میانگین فاصله تمام Agent ها از هدف در هر گام زمانی. این نمودار باید یک روند نزولی داشته باشد که نشان‌دهنده پیشرفت کلی سیستم به سمت هدف است.
- طول مسیر هر **Agent (Path Length):** تعداد قدم‌هایی که یک Agent خاص برای رسیدن به هدف (یا در پایان شبیه‌سازی) برداشته است. این معیار به ارزیابی کارایی فردی کمک می‌کند.
- زمان تکمیل **(Completion Time):** تعداد گام‌های زمانی که طول می‌کشد تا تمام Agent ها به هدف برسند (در صورتی که شبیه‌سازی تا آن زمان ادامه یابد).

۶.۲. سناریوهای آزمایشی

برای درک بهتر رفتار سیستم، می‌توان آن را تحت سناریوهای مختلفی آزمایش کرد. یک تحلیل مهم، بررسی تأثیر "تراکم Agent ها" بر عملکرد سیستم است. سناریوهای زیر می‌توانند تعریف شوند:

- سناریو ۱ (تراکم کم): `NUM_AGENTS = 5`
- سناریو ۲ (تراکم متوسط): `NUM_AGENTS = 10`
- سناریو ۳ (تراکم بالا): `NUM_AGENTS = 20`

تمام سناریوها در یک محیط با اندازه ثابت (`ENV_SIZE = 20`) و برای تعداد فریم‌های مشخص (مثلاً ۵۰۰ فریم) اجرا می‌شوند.

۶.۳. تحلیل نتایج شبیه‌سازی

جدول ۱: نتایج نهایی شبیه‌سازی پس از ۵۰۰ فریم

میانگین طول مسیر	مجموع برخوردهای کل	مجموع قدم‌های کل	تعداد**Agent** ها	سناریو
221.0	45	1105	5	تراکم کم
198.0	250	1980	10	تراکم متوسط
155.0	1150	3100	20	تراکم بالا

نمودار ۱: روند تغییر میانگین فاصله تا هدف در طول زمان

(نمودار فرضی) این نمودار خطی نشان می‌دهد که در هر سه سناریو، میانگین فاصله تا هدف به مرور زمان کاهش می‌یابد، که نشان‌دهنده موفقیت کلی*Agent* ها در حرکت به سمت هدف است. با این حال، شیب کاهش در سناریوی تراکم بالا ممکن است کمتر باشد.

نمودار ۲: مقایسه تعداد برخوردها بر اساس تعداد**Agent** ها

(نمودار فرضی) این نمودار میله‌ای نشان می‌دهد که با افزایش تعداد*Agent* ها، تعداد کل برخوردها به صورت غیرخطی (احتمالاً نمایی) افزایش می‌یابد. این مهم‌ترین یافته در تحلیل عملکرد سیستم است.

۷. بحث و تحلیل

۷.۱. تفسیر نتایج

نتایج فرضی ارائه شده در بخش قبل، الگوهای رفتاری مهمی را آشکار می‌سازد:

- موفقیت استراتژی پایه: نمودار ۱ نشان می‌دهد که استراتژی حرکتی حریصانه، علی‌رغم سادگی، در هدایت Agentها به سمت هدف مؤثر است. روند نزولی میانگین فاصله تا هدف این موضوع را تأیید می‌کند.

- هزینه تراکم**(Cost of Congestion)** : مهم‌ترین یافته از جدول ۱ و نمودار ۲، تأثیر شدید تراکم بر کارایی سیستم است. با دو برابر شدن تعدادAgent ها از ۱۰ به ۲۰، تعداد برخوردها بیش از چهار برابر افزایش یافته است. این نشان می‌دهد که مکانیزم واکنشی اجتناب از برخورد، در محیط‌های شلوغ به سرعت کارایی خود را از دست می‌دهدAgent .ها زمان زیادی را در حالت "انتظار" تلف می‌کنند که منجر به افزایش ترافیک و تأخیر در رسیدن به هدف می‌شود.
- کاهش میانگین طول مسیر: شاید در نگاه اول عجیب به نظر برسد که در جدول ۱، با افزایش تعداد Agentها، میانگین طول مسیر کاهش یافته است. این پدیده می‌تواند به دلیل ماهیت قرارگیری اولیه Agentها باشد. با افزایش تعدادAgent ها، احتمال اینکه برخی از آن‌ها به صورت تصادفی در موقعیت‌های اولیه بهتری (نزدیک‌تر به هدف) قرار بگیرند، افزایش می‌یابد که میانگین کلی را کاهش می‌دهد. این معیار به تنهایی نمی‌تواند کارایی را نشان دهد و باید در کنار معیار برخوردها تحلیل شود.

۷.۲. شناسایی نقاط قوت سیستم

- سادگی و سرعت: الگوریتم‌های به کار رفته بسیار سبک هستند و نیاز به محاسبات پیچیده‌ای ندارند. این امر امکان شبیه‌سازی تعداد زیادی Agent را در زمان واقعی فراهم می‌کند.
- ماژولار بودن و توسعه‌پذیری: معماری کد به گونه‌ای است که می‌توان به راحتی محیط، منطق Agent یا سیستم تصویرسازی را بدون تأثیر بر سایر بخش‌ها تغییر داد یا بهبود بخشید.
- تصویرسازی شهودی: وجود یک نمایش بصری بلادرنگ، درک دینامیک‌های پیچیده سیستم (مانند تشکیل گلوگاه‌ها) را بسیار آسان می‌کند و ابزار قدرتمندی برای آموزش و تحلیل است.
- عدم نیاز به کنترل مرکزی: سیستم به خوبی نشان می‌دهد که چگونه می‌توان بدون یک کنترلر مرکزی، به یک هدف گروهی دست یافت، که این ویژگی اصلی سیستم‌های چندعاملی مقاوم و مقیاس‌پذیر است.

۷.۳. شناسایی نقاط ضعف و محدودیت‌ها

- برنامه‌ریزی مسیر بسیار ساده: استراتژی حریصانه بهینه نیستAgent .ها دید بلندمدت ندارند و ممکن است مسیرهای طولانی‌تری را انتخاب کنند یا در بن‌بست‌های ساده گرفتار شوند (اگر موانعی وجود داشت).

- اجتناب از برخورد ناکارآمد: مکانیزم «توقف کن و منتظر بمان» بسیار ناکارآمد است. این روش واکنشی است و نه پیشگیرانه. Agent ها نمی‌توانند مسیر خود را برای دور زدن یکدیگر تغییر دهند و این منجر به تشکیل صف‌های طولانی و هدر رفتن زمان می‌شود.
- محیط استاتیک و ساده: محیط شبیه‌سازی فاقد هرگونه پیچیدگی مانند موانع ثابت یا متحرک، مناطق با هزینه حرکت متفاوت یا اهداف چندگانه است. این سادگی، عمومیت‌پذیری نتایج را محدود می‌کند.
- همگنی **Agent** ها **(Homogeneity)** : تمام Agent ها دارای قابلیت‌ها و سرعت یکسانی هستند. در دنیای واقعی، سیستم‌ها اغلب ناهمگن (Heterogeneous) هستند.
- پایان شبیه‌سازی: شبیه‌سازی پس از تعداد معینی فریم متوقف می‌شود، نه زمانی که یک شرط خاص (مانند رسیدن همه Agent ها به هدف) برآورده شود. این امر تحلیل معیارهایی مانند «زمان تکمیل» را دشوار می‌کند.

۸. نتیجه‌گیری و کارهای آینده

۸.۱ جمع‌بندی دستاوردها

این پروژه با موفقیت یک سیستم شبیه‌سازی چندعاملی را پیاده‌سازی کرد که به صورت مؤثری مفاهیم بنیادین رفتار خودمختار، ناوبری غیرمتمرکز و اجتناب از برخورد را به نمایش می‌گذارد. سیستم توسعه‌یافته، با معماری ماژولار و ابزار تصویرسازی بلادرنگ، یک بستر ارزشمند برای مطالعه و تحلیل دینامیک‌های سیستم‌های چندعاملی فراهم آورده است. تحلیل نتایج، هرچند بر اساس داده‌های فرضی، به وضوح نشان داد که چگونه افزایش تراکم Agent ها می‌تواند به سرعت کارایی یک استراتژی ساده اجتناب از برخورد را کاهش دهد و اهمیت طراحی مکانیزم‌های هماهنگی پیشرفته‌تر را برجسته می‌کند. در مجموع، پروژه به تمام اهداف اولیه خود دست یافته و یک پایه محکم برای تحقیقات آتی ایجاد کرده است.

۸.۲ پیشنهادها برای توسعه آتی

این پروژه می‌تواند در جهات مختلفی گسترش یابد تا به مسائل واقعی‌تر و پیچیده‌تری بپردازد:

- الگوریتم‌های برنامه‌ریزی مسیر پیشرفته:

  - پیاده‌سازی الگوریتم(A-Star) *A* : جایگزینی حرکت حریصانه با الگوریتم *A به هر Agent اجازه می‌دهد تا مسیر بهینه (کوتاه‌ترین مسیر) را تا هدف پیدا کند، با در نظر گرفتن موانع احتمالی.
  - الگوریتم‌های**Multi-Agent Pathfinding (MAPF)**: تحقیق و پیاده‌سازی الگوریتم‌هایی مانند Conflict-Based Search (CBS) که به صورت بهینه و کامل، مسیرهای بدون برخورد را برای تمامAgent ها پیدا می‌کنند.
- مکانیزم‌های اجتناب از برخورد هوشمندتر:

  - حرکت برای اجتناب :به جای توقفAgent ،ها می‌توانند یک حرکت جانبی انجام دهند تا مسیر را برای دیگران باز کنند.
  - ارتباط و مذاکرهAgent :ها می‌توانند با ارسال پیام به یکدیگر، در مورد حق تقدم در تقاطع‌ها مذاکره کنند.
- غنی‌سازی محیط شبیه‌سازی:

  - افزودن موانع ثابت و متحرک :اضافه کردن دیوارها یاAgent های متخاصم به محیط، چالش ناوبری را واقعی‌تر می‌کند.
  - اهداف پویا یا چندگانه :تعریف اهدافی که در طول زمان تغییر مکان می‌دهند یا تخصیص اهداف متفاوت به زیرگروه‌هایی ازAgent ها.
- معرفی یادگیری ماشین:

  - یادگیری تقویتی :**(Reinforcement Learning)** می‌توان به هر Agent یک مدل یادگیری تقویتی (مانند Q-Learning) داد تا به مرور زمان یاد بگیرد که بهترین حرکت در هر موقعیت چیست. پاداش می‌تواند بر اساس نزدیک شدن به هدف و جریمه برای برخوردها تعریف

شود. این کار به Agent ها اجازه می‌دهد تا استراتژی‌های پیچیده‌تری را به صورت خودکار کشف کنند.

- تحلیل مقیاس‌پذیری: اجرای آزمایش‌های جامع‌تر برای تحلیل عملکرد سیستم با صدها یا هزاران Agent و در محیط‌های بسیار بزرگ‌تر، و بررسی اینکه الگوریتم‌ها از نظر محاسباتی چگونه مقیاس‌پذیر هستند.

این مسیرهای تحقیقاتی می‌توانند این پروژه پایه را به یک ابزار تحقیقاتی قدرتمند در حوزه سیستم‌های چندعاملی و هوش مصنوعی توزیع‌شده تبدیل کنند.

۹. مراجع

کتاب‌ها:

1. Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. John Wiley & Sons.
2. Russell, S. J., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach (4th ed.)*. Pearson.
3. Shoham, Y., & Leyton-Brown, K. (2009). *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

مقالات کلیدی:

4. Silver, D. (2005). *Cooperative Pathfinding*. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.
5. van den Berg, J. P., & Overmars, M. H. (2005). *Prioritized Motion Planning for Multiple Robots*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

مستندات ابزارها:

6. Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. *Nature*, 585, 357–362.
7. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.

8. *The Python 3 Standard Library*. Python Software Foundation. Retrieved from https://docs.python.org/3/library/
9. *Contourpy Documentation*. Retrieved from the official documentation page of the library. [cite: 1]
10. *Pandas Documentation*. Retrieved from the official documentation page of the library. [cite: 1]