دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

پروژه اول هوش مصنوعی

رشته علوم کامپیوتر

بهبود الگوریتم *A

نگارش

محمدحسین رستمدار

استاد درس

جناب آقای دکتر مهدی قطعی

استاد کارگاه

جناب آقای بهنام یوسفی مهر

تاریخ

۱۴۰۳/۰۷/۱۷

چکیده

در این مقاله ما در بخش اول به تحلیل الگوریتم *A برای یک ماشین خودران میپردازیم و بعد از آن راه هایی را برای بهبود این الگوریتم برای جست و جوی ربات مسیریاب ارائه می دهیم. این الگوریتم های بهبود یافته ای را که در این گزارش مطرح کردیم عبارت اند از **الگوریتم *A مبتنی بر راهنما , نقاط کلیدی و گام های متغیر** هستند.

حال در بخش دوم برای طی یک مسیر خودران برای رسیدن به مقصد , الگوریتم های جستجوی سطح اول (BFS) , جستجوی هزینه یکنواخت (UCS) , حریصانه و *A را مطرح میکنیم و در نتایج آن را تحلیل میکنیم و آن ها را از نظر زمان اجرا , تعداد گره ها و در نهایت عمقی که الگوریتم مربوطه آن را طی میکند.

واژه‌های کلیدی:

الگوریتم بهبود یافته *A , وسایل نقلیه خودران , BFS , UCS , حریصانه

صفحه

فهرست مطالب

چکیده..... ۳

فصل اول مقدمه..... 3

فصل دوم الگوریتم بهبودیافته ی A-Star برای مسیریابی وسایل نقلیه خودران زمینی..... 5

۲-۱- مفاهیم اصلی و نوآوری مقاله..... 6

۲-۲- بررسی الگوریتم های معرفی شده به عنوان بهبود الگوریتم سنتی..... ۷

۲-۳- مقایسه الگوریتم سنتی *A با الگوریتم‌های معرفی شده و تحلیل نتایج..... ۱۰

فصل سوم الگوریتم های جست و جو..... ۱۴

۳-۱- تحلیل نتایج الگوریتم ها..... ۱۵

۳-۲- اثبات Consistent یا A d missib l e بودن راه حل..... ۲۰

فصل چهارم نتیجه‌گیری و جمع بندی..... ۲۶

منابع و مراجع..... ۲۸

فصل اول

مقدمه

مقدمه:

مقاله و تصویر ارائه شده به بررسی و بهبود الگوریتم‌های جستجو برای برنامه‌ریزی مسیر خودروهای خودران اختصاص دارد. این موضوع یکی از مسائل کلیدی در حوزه هوش مصنوعی و روباتیک است که هدف آن پیدا کردن سریع‌ترین و بهینه‌ترین مسیر بین دو نقطه با توجه به موانع موجود در محیط است. الگوریتم‌های جستجو از بازی‌های ویدیویی گرفته تا رباتیک و خودروهای خودران استفاده می‌شوند و به ماشین‌ها امکان می‌دهند مسیرهایی را پیدا کنند که از نظر زمان و فاصله بهینه باشند.

در این مقاله، الگوریتم کلاسیک A-Star با الگوریتم‌های دیگری همچون BFS، DFS، UCS و Greedy مقایسه می‌شود. در بخشی از تمرین عملی، کاربران با استفاده از این الگوریتم‌ها باید مسیریابی در محیطی شبیه به یک هزار تو را انجام دهند که موانع متعددی در آن وجود دارد. هدف از این تمرین، پیاده‌سازی و مقایسه کارایی الگوریتم‌های مختلف جستجو در شرایط مختلف است. تصویر موجود به خوبی پیچیدگی محیط و نحوه عملکرد این الگوریتم‌ها را به نمایش می‌گذارد.

فصل دوم

الگوریتم بهبودیافته ی A* برای مسیریابی وسایل نقلیه خودران زمینی

2-۱- مفاهیم اصلی و نوآوری مقاله:

مقاله ارائه‌شده به بهبود الگوریتم **A-Star** برای برنامه‌ریزی مسیر وسایل نقلیه خودران اختصاص دارد.

الگوریتم (A-star) ***A**یکی از مهم‌ترین و پرکاربردترین الگوریتم‌های جستجو است که در مسائل پیدا کردن مسیر (Pathfinding) و جستجوی گراف‌ها به کار می‌رود. این الگوریتم در مسائل مختلف از مسیریابی ربات‌ها گرفته تا جستجوی مسیر در بازی‌های ویدیویی، نقش مهمی ایفا می‌کند. ویژگی اصلی آن این است که با استفاده از تخمین فاصله (هزینه) به هدف، بهینه‌سازی در فرآیند جستجو را انجام می‌دهد و تلاش می‌کند تا مسیر بهینه از مبدا به مقصد را با حداقل هزینه پیدا کند.

نوآوری‌ها و مفاهیم اصلی این مقاله شامل موارد زیر است:

۱. **استاندارد ارزیابی** :الگوریتم‌های مسیریابی نیازمند ارزیابی بر اساس ایمنی، راحتی و بهینه‌سازی انرژی هستند، اما تاکنون معیار مناسبی برای ارزیابی کمی این الزامات وجود نداشته است. برای این منظور، یک استاندارد جدید ارزیابی معرفی شده است. از الگوریتم مسیریابی موجود در یک مقاله به عنوان نمونه استفاده شده و تاثیر پارامترهای مختلف جستجو بررسی می‌شود.

سه جهت حرکت (چپ، راست، مستقیم) در فرآیند جستجو استفاده شده و سرعت مجازی برای هر کدام از این تغییرات تعریف می‌شود. سرعت مجازی بین حرکت مستقیم و تغییر جهت‌ها در جدول مشخصی تنظیم شده است. با استفاده از ۹ مرحله جستجو مختلف، نتایج بررسی شده و مشخص شده که مرحله جستجو ۴ بهترین عملکرد را دارد.

Table 3. The results of estimating under different parameters.

Parameters	Epoints *N*	Ppoints	Length	Cost time *T*
Step = 1	89,879	238	249.8131	298.6263
Step = 2	51,922	119	248.8820	293.7640
Step = 3	48,318	88	246.0100	271.0201
Step = 4	16,117	66	245.0794	264.1587
Step = 5	14,168	54	245.8635	269.7270
Step = 6	12,207	46	247.9927	277.9854
Step = 7	9631	40	245.2979	285.5958
Step = 8	7226	36	248.2427	270.4855
Step = 9	—	—	—	—

همچنین از مزایای این روش در ارزیابی میتوان به نکات زیر اشاره کرد :

- استفاده از زمان سپری شده (T) برای سنجش کارایی الگوریتم‌ها که با شرایط واقعی همخوانی دارد.
- استفاده از فاصله و سرعت مجازی برای حفظ روند واقعی بدون نیاز به مدل فیزیکی خودرو.

۲. **الگوریتم A-Star مبتنی بر استفاده از راهنما** :به عنوان یک بهبود در نسخه کلاسیک A* شناخته می‌شود. این الگوریتم از یک **راهنما** بهره می‌گیرد که می‌تواند توسط انسان یا از طریق یک سیستم برنامه‌ریزی جهانی تولید شود. راهنما به عنوان یک **تابع هیوریستیک** عمل می‌کند و به الگوریتم در طول فرآیند جستجو کمک می‌کند تا مسیرهایی را انتخاب کند که به مسیر ایده‌آل نزدیک‌تر باشند.

۳. **الگوریتم A-Star مبتنی بر نقاط کلیدی** :الگوریتم مبتنی بر نقاط کلیدی پیرامون موانع، مسیر خودران را به‌گونه‌ای هدایت می‌کند که پیش از برخورد با مانع، مسیر را تغییر دهد و از آن دوری کند.

٤. **الگوریتم A-Star مبتنی بر گام‌های متغیر** :برای کاهش زمان محاسبات، گام‌های جستجو به‌صورت متغیر و متناسب با وضعیت موانع تنظیم می‌شوند. این روش در فضاهای باز از گام‌های بزرگتر و در اطراف موانع از گام‌های کوچکتر استفاده می‌کند تا کارایی محاسبات افزایش یابد.

این بهبودها باعث می‌شوند که الگوریتم پیشنهادی در مقایسه با روش‌های کلاسیک، بهینه‌تر و مناسب‌تر برای کاربردهای واقعی وسایل نقلیه خودران عمل کند.

۲-۲-بررسی الگوریتم های معرفی شده به عنوان بهبود الگوریتم سنتی:

۱-الگوریتم A-Star مبتنی بر استفاده از راهنما:

الگوریتم A-Star مبتنی بر راهنما با استفاده از یک **راهنما** (Guideline)که توسط انسان یا یک برنامه‌ریزی جهانی تولید شده، مشکلات نسخه سنتی را حل می‌کند. این راهنما، به عنوان یک مرجع خارجی به الگوریتم اضافه می‌شود که به آن کمک می‌کند تا مسیرهای طبیعی‌تر و امن‌تری را انتخاب کند. در واقع، این راهنما نقش یک نقشه ایده‌آل را بازی می‌کند که هدف آن هدایت الگوریتم به سمتی است که تصمیمات آن با اهداف راننده یا سیستم تطابق بیشتری داشته باشد.

تابع هیورستیک به شکل $F(i)=G(i)+H1(i)\cdot a1+H2(i)\cdot a2$تعریف می‌شود که در آن $H1(i)$ فاصله نقطه i از راهنما و $H2(i)$ فاصله از نقطه $g(i)$ در راهنما تا هدف است. این رویکرد به الگوریتم اجازه می‌دهد تا به مسیر راهنما نزدیک‌تر باشد و از مسیر بهتری در سناریوهایی با پیچ‌های متعدد پیروی کند، برخلاف A-Star سنتی که در این شرایط ممکن است مسیرهای نامناسبی را انتخاب کند.

چگونه الگوریتم مبتنی بر راهنما بهبود ایجاد می‌کند؟

۱. **هدایت بهتر در محیط‌های پیچیده** :در شرایطی که پیچ‌های تند یا موانع وجود دارند، راهنما به عنوان یک ابزار هدایتگر به الگوریتم کمک می‌کند تا بهترین مسیر ممکن را انتخاب کند، بدون اینکه فقط به تخمین فاصله به هدف متکی باشد.

۲. **کاهش خطاهای ناشی از تشخیص ناقص جاده‌ها** :الگوریتم سنتی A* به تشخیص دقیق جاده‌ها وابسته است و اگر کناره‌های جاده به درستی شناسایی نشوند، عملکرد آن مختل می‌شود. اما در الگوریتم مبتنی بر راهنما، حتی در صورت نقص در تشخیص کناره‌های جاده، مسیر بهینه‌تری به دلیل راهنما انتخاب می‌شود.
۳. **تطابق بهتر با شرایط واقعی** :به دلیل استفاده از اطلاعات اضافی (راهنما)، این الگوریتم توانسته است در شرایط واقعی مانند جاده‌های شهری و روستایی، عملکرد بهتری نسبت به نسخه سنتی داشته باشد. مسیرهای انتخاب شده با آنچه که در دنیای واقعی انتظار می‌رود، بیشتر تطابق دارند.

۲-الگوریتم A-Star مبتنی بر نقاط کلیدی:

الگوریتم A* سنتی و حتی نسخه بهبودیافته آن که از راهنما استفاده می‌کند، با وجود مزایای بهبود یافته، همچنان در زمانی که **راهنما از روی مانع عبور می‌کند** دچار مشکل می‌شود. در چنین مواردی، راهنما مسیر وسیله نقلیه را به سمت موانع هدایت می‌کند، که این می‌تواند خطرناک باشد و برخلاف اهداف راننده یا سیستم کنترل خودرو عمل کند. به همین دلیل، نسخه مبتنی بر نقاط کلیدی ارائه شد.

نقاط کلیدی (**Key Points**) نقاطی هستند که در دو طرف یک مانع قرار می‌گیرند و می‌توان از آن‌ها برای هدایت مسیر به شکلی ایمن و به دور از مانع استفاده کرد. به عبارت دیگر، این نقاط به الگوریتم اجازه می‌دهند که با شناسایی آن‌ها، مسیر را تغییر داده و از برخورد به موانع جلوگیری کند.

مراحل عملکرد الگوریتم A* مبتنی بر نقاط کلیدی به طور خلاصه به شرح زیر است:

۱. **تنظیم صحنه آزمایشی** :تعیین اندازه فضای جستجو، نقطه شروع، نقطه هدف، نقشه موانع و راهنما.

۲. **بررسی وجود موانع** :بررسی می‌شود که آیا در مسیر راهنما موانعی وجود دارد یا خیر.
۳. **شناسایی نقاط کلیدی** :شناسایی نقاط کلیدی در دو طرف موانع برای دور زدن آن‌ها.
٤. **تولید مسیرهای کاندیدا** :تولید مسیرهای جدید با اتصال نقاط کلیدی به نقطه شروع و هدف.
٥. **بررسی موانع و محاسبه هزینه‌ها** :بررسی اینکه آیا مسیرهای کاندیدا شامل موانع هستند و محاسبه هزینه زمانی هر مسیر.
٦. **انتخاب بهترین مسیر** :انتخاب مسیری که هیچ مانعی نداشته باشد و کمترین هزینه زمانی را داشته باشد.
۷. **جستجوی مسیر نهایی** :استفاده از الگوریتم A* برای جستجوی مسیر نهایی و برگشت نتیجه.

۳-الگوریتم A-Star مبتنی بر گام‌های متغیر:

الگوریتم مبتنی بر گام‌های متغیر از استراتژی جدیدی برای تنظیم اندازه گام جستجو استفاده می‌کند. به این معنا که:

- **گام‌های بزرگ‌تر در مناطق باز** :در فضاهای باز و بدون موانع، الگوریتم می‌تواند از گام‌های بزرگ‌تری استفاده کند، که به آن اجازه می‌دهد به سرعت به سمت هدف پیش برود و تعداد نقاط گسترش را کاهش دهد.
- **گام‌های کوچک‌تر نزدیک موانع** :وقتی الگوریتم به موانع نزدیک می‌شود، اندازه گام کاهش می‌یابد. این امر به الگوریتم کمک می‌کند تا با دقت بیشتری مسیر را جستجو کرده و از برخورد با موانع جلوگیری کند.

این الگوریتم از یک **قالب محاسباتی** برای محاسبه نقاط گسترش استفاده می‌کند که در مراحل اولیه الگوریتم ایجاد می‌شود. این روش به‌طور قابل توجهی عملیات محاسباتی مانند محاسبه توابع سینوس و کسینوس را کاهش می‌دهد و در نتیجه زمان محاسباتی را بهبود می‌بخشد.

۲-۳- مقایسه الگوریتم سنتی A* با الگوریتم‌های معرفی شده و تحلیل نتایج:

۱- الگوریتم سنتی A*

ویژگی‌ها:

- **تأسیس بر روی هزینه‌ها** :از دو تابع هزینه $G(i)$ (هزینه واقعی طی‌شده) و $H(i)$ (تخمین فاصله به هدف) برای تصمیم‌گیری در انتخاب مسیر استفاده می‌کند.
- **عملکرد ضعیف در شرایط خاص** :در مواجهه با پیچ‌ها و تقاطع‌ها ممکن است مسیرهای نامناسب یا خطرناک انتخاب کند.

محدودیت‌ها:

- در شرایط پیچیده مانند جاده‌های منحنی و تقاطع‌ها، عملکرد نامطلوبی دارد و احتمالاً اهداف راننده را به خوبی محقق نمی‌کند.
- پیچیدگی زمانی و فضایی بالایی دارد، به خصوص در محیط‌های بزرگ.

۲- الگوریتم *A مبتنی بر راهنما

ویژگی‌ها:

- **استفاده از راهنما** :راهنما به عنوان یک مرجع برای تعیین مسیر ایده‌آل بر اساس نیات راننده به کار می‌رود.
- **بهبود تابع هیوریستیک** :تابع هیوریستیک به دو جزء تقسیم می‌شود H1 : (فاصله از راهنما) و H2 (فاصله از نقطه‌ای روی راهنما به هدف).

مزایا:

- بهبود عملکرد در جاده‌های پیچیده و تقاطع‌ها با استفاده از راهنما.
- کاهش خطر انتخاب مسیرهای نامناسب.

نتایج:

- با استفاده از الگوریتم مبتنی بر راهنما، تعداد نقاط گسترش به ۹۳,۸۸۸ افزایش می‌یابد و زمان هزینه ۳۰۵.۸۴۳۸ ثانیه است که نشان‌دهنده کارایی کمتر نسبت به الگوریتم‌های دیگر است.

۳- الگوریتم *A مبتنی بر نقاط کلیدی

ویژگی‌ها:

- **مدیریت موانع** :شناسایی نقاط کلیدی در دو طرف موانع و دور زدن آن‌ها.
- **تولید مسیرهای کاندیدا** :از نقاط کلیدی برای ایجاد مسیرهای جدید استفاده می‌شود.

مزایا:

- افزایش امنیت و کارایی با جلوگیری از هدایت به سمت موانع.
- کاهش تعداد نقاط گسترش به ۱۱,۱۰۲ و زمان هزینه ۲۷۵.۲۸۹۰ ثانیه.

نتایج:

- این الگوریتم به وضوح بهترین عملکرد را از نظر تعداد نقاط گسترش و زمان هزینه دارد.

٤- الگوریتم A* مبتنی بر گام‌های متغیر

ویژگی‌ها:

- **استراتژی گام‌های متغیر** :اندازه گام بر اساس توزیع موانع متغیر است.
- **محاسبات بهینه** :استفاده از قالب محاسباتی برای کاهش بار محاسباتی.

مزایا:

- کاهش تعداد نقاط گسترش به ۱,۴۲۶ و بهبود قابل توجهی در کارایی محاسباتی.
- زمان هزینه ۲۸۰.۶۴۷۷ ثانیه که نزدیک به الگوریتم مبتنی بر نقاط کلیدی است.

نتایج:

- این الگوریتم عملکرد بسیار بهتری در کاهش بار محاسباتی و بهبود کارایی نسبت به الگوریتم‌های سنتی و حتی الگوریتم مبتنی بر نقاط کلیدی ارائه می‌دهد.

نتیجه‌گیری کلی

از مقایسه این چهار الگوریتم مشخص است که:

- الگوریتم A* مبتنی بر راهنما در بهبود عملکرد نسبت به الگوریتم سنتی A* موفق است، اما هنوز هم در شرایط خاص مانند عبور از موانع دچار مشکل می‌شود.
- الگوریتم A* مبتنی بر نقاط کلیدی بهترین عملکرد را از نظر تعداد نقاط گسترش و زمان هزینه دارد و امنیت بیشتری را ارائه می‌دهد.
- الگوریتم A* مبتنی بر گام‌های متغیر با کاهش بار محاسباتی و افزایش کارایی، به‌ویژه در محیط‌های دارای محدودیت منابع، به بهبود الگوریتم‌های پیشین کمک می‌کند.
- نتایج تجربی نشان می‌دهد که **الگوریتم مبتنی بر نقاط کلیدی** در زمینه اجتناب از موانع و **الگوریتم مبتنی بر گام‌های متغیر** از نظر کاهش زمان محاسبه عملکرد بهتری دارند، در حالی که **الگوریتم مبتنی بر راهنما** دقت مسیر را بهبود می‌بخشد.

در نتیجه، هر یک از این الگوریتم‌ها با بهبودهای خاص خود به رفع مشکلات الگوریتم A* سنتی کمک کرده و در شرایط خاص عملکرد بهتری را ارائه می‌دهند.

Table 5. The results of estimating for different algorithms.

Algorithm	Epoints N	Ppoints	Length	Cost time T
Autoware	33,654	68	246.6846	286.3693
Guideline	93,888	75	251.9219	305.8438
Key point	11,102	67	246.6445	275.2890
Variable step	1426	46	246.3238	280.6477

فصل سوم

الگوریتم های جست و جو

۳-۱- تحلیل نتایج الگوریتم ها

در وهله اول برای مقایسه پنج الگوریتم **DFS**، **BFS** ، **UCS** ، **Greedy** و **A*** از نظر عملکرد، بهینه‌سازی، زمان اجرا و معیارهای دیگر، به بررسی جزئیات هر الگوریتم می‌پردازیم:

Depth-First Search (DFS)

- **روش کار** DFS :به صورت عمقی کار می‌کند، یعنی ابتدا تا جایی که ممکن است یک شاخه از درخت جستجو را به عمق طی می‌کند و سپس به سراغ شاخه‌های دیگر می‌رود.
- **تعداد گره‌های گسترش‌یافته** DFS :ممکن است گره‌های زیادی را گسترش دهد، به‌ویژه اگر مسیر بهینه در شاخه‌های بالایی قرار نداشته باشد.
- **عمق مسیر** :این الگوریتم در صورت وجود حلقه، ممکن است به بی‌نهایت برود. عمق مسیر وابسته به ساختار گراف و هدف است.
- **زمان اجرا** DFS :معمولاً سریع است و از حافظه کمتری نسبت به BFS استفاده می‌کند.
- **بهینه‌سازی** DFS :هیچ تضمینی برای یافتن کوتاه‌ترین مسیر یا بهینه‌ترین مسیر ندارد و ممکن است به بن‌بست برسد.

Breadth-First Search (BFS)

- **روش کار** BFS :به صورت عرضی کار می‌کند، یعنی همه گره‌های یک سطح را بررسی می‌کند قبل از اینکه به سراغ سطح بعدی برود.

- **تعداد گره‌های گسترش‌یافته** BFS :در بدترین حالت تمام گره‌های فضای جستجو را گسترش می‌دهد.
- **عمق مسیر** :این الگوریتم تضمین می‌کند که کوتاه‌ترین مسیر (از نظر تعداد گره‌ها) پیدا شود.
- **زمان اجرا** BFS :زمان اجرای بالایی دارد زیرا تمام گره‌های ممکن را بررسی می‌کند.
- **بهینه‌سازی** BFS :تضمین می‌کند که کوتاه‌ترین مسیر را پیدا کند، اما لزوماً بهینه‌ترین مسیر از نظر هزینه نیست.

Uniform Cost Search (UCS)

- **روش کار** UCS :یک نسخه تعمیم‌یافته از BFS است که به جای تعداد گره‌ها، هزینه مسیرها را در نظر می‌گیرد. این الگوریتم گره‌هایی با کمترین هزینه را اولویت می‌دهد.
- **تعداد گره‌های گسترش‌یافته** UCS :تمام گره‌ها را بررسی می‌کند، اما فقط مسیرهایی را که از نظر هزینه کمتر هستند گسترش می‌دهد.
- **عمق مسیر** UCS :معمولاً مسیرهای بهینه از نظر هزینه را پیدا می‌کند، اما تعداد گره‌های زیادی را ممکن است گسترش دهد.
- **زمان اجرا** UCS :ممکن است زمان بیشتری نسبت به BFS نیاز داشته باشد، زیرا باید هزینه‌ها را مقایسه کند.
- **بهینه‌سازی** UCS :تضمین می‌کند که بهینه‌ترین مسیر (از نظر هزینه) را پیدا کند.

Greedy Search

- **روش کار** :این الگوریتم فقط به فاصله تخمینی از گره جاری به هدف) تابع heuristic)نگاه می‌کند و هر بار گره‌ای که نزدیک‌تر به هدف است را گسترش می‌دهد.
- **تعداد گره‌های گسترش‌یافته** Greedy :معمولاً تعداد کمی از گره‌ها را گسترش می‌دهد زیرا تنها روی هدف متمرکز است.
- **عمق مسیر** :ممکن است مسیر طولانی‌تری نسبت به الگوریتم‌های دیگر پیدا کند، زیرا بهینه‌سازی هزینه مسیر را در نظر نمی‌گیرد.
- **زمان اجرا** Greedy :بسیار سریع‌تر از BFS و UCS است زیرا فقط گره‌های مرتبط با هدف را بررسی می‌کند.
- **بهینه‌سازی** Greedy :تضمینی برای یافتن مسیر بهینه ندارد و ممکن است گره‌های نامناسبی را انتخاب کند.

A* (A-Star) Search

- **روش کار** A* :ترکیبی از UCS و Greedy است. این الگوریتم هم هزینه طی‌شده را در نظر می‌گیرد و هم از تخمین فاصله تا هدف استفاده می‌کند.
- **تعداد گره‌های گسترش‌یافته** A* :تعداد گره‌های متوسطی را گسترش می‌دهد، زیرا هم به هزینه و هم به تخمین نگاه می‌کند.
- **عمق مسیر** A* :معمولاً بهینه‌ترین مسیر را پیدا می‌کند که کمترین هزینه را دارد.
- **زمان اجرا** :بسته به پیچیدگی تابعheuristic ، زمان اجرای A* ممکن است بین UCSو Greedy باشد.
- **بهینه‌سازی** A* :تضمین می‌کند که بهترین مسیر را (از نظر هزینه و تخمین فاصله) پیدا کند.

|| نود های اکسپند شده ||

تعداد نودهای اکسپند شده در الگوریتم‌های مختلف جستجو به عواملی همچون ساختار فضای حالت، نوع مسئله و روش جستجو بستگی دارد. در ادامه، توجیهاتی برای تعداد نودهای اکسپند شده برای هر یک از الگوریتم‌هایDFS ، BFS، Greedy، UCSو A* آورده می‌شود:

DFS (Depth-First Search)

- **تعداد اکسپند نودها** :در بدترین حالت، DFSممکن است تمام نودهای یک درخت یا گراف را اکسپند کند.
- **توجیه** :زیرا DFS به صورت عمق‌گرا و تنها با تکمیل یکی از شاخه‌های درخت جستجو می‌کند، در صورتی که درخت عمیق باشد، ممکن است همه‌ی نودها را گسترش دهد، حتی اگر جواب در عمق کمتری باشد.

BFS (Breadth-First Search)

- **تعداد اکسپند نودها** BFS :در بدترین حالت ممکن است تمام نودهای سطح‌های بالاتر را اکسپند کند.
- **توجیه** BFS :به صورت سطحی کار می‌کند و همه نودهای یک سطح را قبل از رفتن به سطح بعدی اکسپند می‌کند. این باعث می‌شود که در صورت وجود فضای جستجو با تعداد نودهای زیاد در سطوح بالاتر، تعداد نودهای اکسپند شده زیاد باشد.

Greedy

- **تعداد اکسپند نودها** :بستگی به کیفیت تابع ارزیابی (heuristic) دارد.

- **توجیه** :اگر تابع ارزیابی خوبی انتخاب شود که نزدیک‌ترین نود به هدف را شناسایی کند، تعداد اکسپند نودها ممکن است کاهش یابد. اما اگر تابع ارزیابی ضعیف باشد، ممکن است نیاز به اکسپند کردن نودهای زیادی باشد.

UCS (Uniform Cost Search)

- **تعداد اکسپند نودها** UCS :نیز در بدترین حالت ممکن است تمام نودها را اکسپند کند.
- **توجیه** UCS :به هزینه مسیر تا نود استفاده می‌کند و اولویت‌هایی بر اساس هزینه‌های کم‌تر دارد. اگر گراف دارای تعداد زیادی نود با هزینه‌های نزدیک به هم باشد، UCSممکن است تعداد زیادی نود را اکسپند کند تا هزینه بهینه را پیدا کند.

A* (A-Star)

- **تعداد اکسپند نودها** :بستگی به دقت تابع ارزیابی (ترکیبی از هزینه مسیر و پیش‌بینی هزینه باقی‌مانده) دارد.
- **توجیه** A* :با استفاده از یک تابع هوریستیک ($h(n)$) و هزینه واقعی ($g(n)$) تلاش می‌کند تا بهتر از سایر الگوریتم‌ها، نودهای بهینه را انتخاب کند. اگر تابع هوریستیک دقیق باشد، تعداد نودهای اکسپند شده کمتر خواهد بود. اما اگر تابع هوریستیک خوب نباشد، ممکن است نیاز به اکسپند کردن نودهای بیشتری باشد.

خلاصه

تعداد نودهای اکسپند شده در هر الگوریتم بستگی به نوع جستجو، ساختار فضای حالت و کیفیت تابع ارزیابی (در الگوریتم‌های مرتبط با هزینه) دارد. به همین دلیل هر الگوریتم نقاط قوت و ضعف خاص خود را دارد و در شرایط خاص به بهترین نحو عمل می‌کند.

-	زمان اجرا	گره های بررسی شده	عمق طی شده
BFS	0.37	1595	63
DFS	0.06	255	85
UCS	0.37	1594	63
Greedy	0.06	212	76
A*	3.76	۶۷۹۰	62

نتیجه‌گیری:

DFS **: برای مسائل ساده یا زمانی که حافظه کم است، کارآمد است اما بهینه‌سازی را تضمین نمی‌کند.**

BFS : برای یافتن کوتاه‌ترین مسیر از نظر تعداد گره‌ها مفید است اما کند و حافظه‌بر است.

UCS : برای یافتن مسیر بهینه از نظر هزینه عالی است، اما ممکن است زمان‌بر باشد.

Greedy : سریع است اما بهینه‌ترین مسیر را پیدا نمی‌کند.

A* : کمترین عمق را طی میکند و بهترین ترکیب از سرعت و بهینه‌سازی است و مسیر بهینه را با توجه به هزینه و تخمین پیدا می‌کند.

۳-۲- اثبات Consistent یا Admissible بودن راه حل

تعریف:Admissible

- یک تابع تخمین‌زن **پذیرفتنی** است اگر هرگز هزینه مسیر واقعی را بیش از حد تخمین نزند، یعنی $h(n) \leq h^*(n)$ که در آن:
  - $h(n)$ مقدار تخمینی تابع تخمین‌زن برای رسیدن از گره n به هدف است.
  - $h^*(n)$ هزینه واقعی بهینه از گره n تا هدف است.

تعریف:Consistent

- یک تابع تخمین‌زن **سازگار** است اگر برای هر جفت گره‌های n و n' که n' فرزند n است، تابع تخمین‌زن از رابطه رو به رو تبعیت کند $h(n) \leq c(n,n') + h(n')$

که در آن:

  - $c(n,n')$ هزینه واقعی بین گره n و گره n' است.
  - $h(n)$ مقدار تخمینی برای رسیدن از گره n به هدف است.
  - $h(n')$ مقدار تخمینی برای رسیدن از گره n' به هدف است.

برای بررسی اینکه آیا هر یک از پنج الگوریتم **DFS** ، **BFS** ، **UCS** ، **Greedy** و **A*** **Consistent هست یا Admissible** ، نیاز داریم تا به ویژگی‌های هر الگوریتم و نحوه عملکرد آن‌ها بپردازیم. الگوریتم‌های جستجو بر اساس تابع **تخمین‌زن(Heuristic)** ، **هزینه مسیر** و **استراتژی جستجو** ارزیابی می‌شوند. در ادامه، برای هر الگوریتم بررسی می‌کنیم که آیا Consistent یا Admissible است یا خیر.

Depth-First Search (DFS)

- **روش کار** DFS :به صورت عمقی جستجو می‌کند، ابتدا تا آخرین شاخه ممکن می‌رود و سپس به سراغ شاخه‌های دیگر می‌رود. این الگوریتم تابع تخمین‌زن یا هزینه مسیر ندارد و فقط به ساختار درخت یا گراف توجه دارد.

Admissible بودن:

- DFSهیچ تابع تخمین‌زن یا اطلاعاتی از هزینه مسیر را در نظر نمی‌گیرد، بنابراین نمی‌توان آن را به عنوان **Admissible** در نظر گرفت. این الگوریتم ممکن است مسیری طولانی‌تر از بهینه‌ترین مسیر پیدا کند.

Consistent بودن:

- DFS به دلیل عدم استفاده از هزینه‌های بین گره‌ها $c(n,n')$ و نبود تابع تخمین‌زن، **Consistent** هم نیست. زیرا تخمینی از هزینه‌ها ارائه نمی‌دهد که پایدار یا سازگار باشد.

نتیجه DFS : نه Admissible است و نهConsistent

Breadth-First Search (BFS)

- **روش کار** BFS :جستجو را به صورت عرضی انجام می‌دهد و همه گره‌های یک سطح را قبل از حرکت به سطح بعدی بررسی می‌کند. این الگوریتم نیز هیچ تابع تخمین‌زن ندارد، اما هزینه مسیر برای هر گره برابر است (همه لبه‌ها هزینه یکسان دارند).

Admissible بودن:

- BFS اگر گراف یا مسئله جستجو **یکنواخت (unweighted)** باشد، **Admissible** است. زیرا همیشه کوتاه‌ترین مسیر (از نظر تعداد گره‌ها) را پیدا می‌کند، به این معنی که مسیر انتخاب‌شده هرگز طولانی‌تر از بهینه‌ترین مسیر نیست.

Consistent بودن:

- در گراف‌هایی که هزینه یکنواخت است (همه لبه‌ها وزن یکسان دارند)، **BFS می‌تواند Consistent باشد** .زیرا هزینه بین گره‌ها پایدار و همگن است، و بنابراین هیچ‌گونه تغییر ناگهانی در هزینه وجود ندارد.

نتیجه BFS :در گراف‌های یکنواخت Admissible و Consistent است.

Uniform Cost Search (UCS)

- **روش کار** UCS :مانند BFS است، اما هزینه لبه‌ها را در نظر می‌گیرد و گره‌هایی با کمترین هزینه تا کنون را گسترش می‌دهد. این الگوریتم هزینه واقعی مسیر $g(n)$ را در نظر می‌گیرد، اما از تابع تخمین‌زن استفاده نمی‌کند.

Admissible بودن:

- UCS به دلیل اینکه همیشه کمترین هزینه واقعی را گسترش می‌دهد، **Admissible است** .زیرا همیشه بهینه‌ترین مسیر را از نظر هزینه واقعی پیدا می‌کند و هیچ‌گاه هزینه‌ای بیشتر از هزینه واقعی را برای یک مسیر تخمین نمی‌زند.

Consistent بودن:

- چون UCS فقط به هزینه واقعی بین گره‌ها توجه می‌کند و از تخمین استفاده نمی‌کند، **Consistent نیز هست** .هزینه بین هر دو گره پایدار است و به تدریج افزایش می‌یابد.

نتیجه UCS :هم Admissible و هم Consistent است.

Greedy Search

- **روش کار** Greedy :فقط از تابع تخمین‌زن $h(n)$ برای ارزیابی فاصله تا هدف استفاده می‌کند و هر بار گره‌ای را که کمترین مقدار $h(n)$ دارد، انتخاب می‌کند. این الگوریتم هزینه واقعی مسیر $g(n)$ را نادیده می‌گیرد.

Admissible بودن:

- به طور کلی، تابع تخمین‌زن در Greedy **Admissible نیست** .زیرا Greedy ممکن است مسیری را انتخاب کند که از نظر $h(n)$ بهترین به نظر برسد، اما در واقع هزینه واقعی آن بیشتر از مسیری دیگر باشد.

Consistent بودن:

- تابع تخمین‌زن در Greedy معمولاً **Consistent نیست** .زیرا این الگوریتم هزینه واقعی مسیر را در نظر نمی‌گیرد و ممکن است در طول مسیر دچار تغییرات ناپایدار در تخمین هزینه‌ها شود.

نتیجه Greedy :نه Admissible است و نهConsistent

A* (A-Star)

- **روش کار** A* :ترکیبی از UCS و Greedy است. این الگوریتم هم هزینه واقعی طی‌شده g(n) و هم تخمین هزینه تا هدف h(n) را در نظر می‌گیرد.

Admissible بودن:

- اگر تابع تخمین‌زن h(n) **پذیرفتنی (Admissible)** باشد (یعنی هزینه واقعی را بیش از حد تخمین نزند)، الگوریتم A* **نیز پذیرفتنی** است. در این حالت، A* همیشه بهینه‌ترین مسیر را پیدا می‌کند.

Consistent بودن:

- اگر تابع تخمین‌زن h(n) **Consistent** باشد (یعنی رابطه مثلث نابرابری $h(n) \leq c(n,n') + h(n')$ را رعایت کند A* **Consistent** است و هر گره فقط یک بار گسترش می‌یابد. اگر h(n) Consistent باشد، به طور ضمنی Admissibleهم هست.

نتیجه: A* Admissible است اگر h(n) پذیرفتنی باشد و Consistent است اگر h(n)پایدار و سازگار باشد.

Consistent	Admissible	-
بله اما در گراف یکنواخت	بله اما در گراف یکنواخت	BFS

DFS	نه	نه
UCS	بله	بله
Greedy	نه	نه
A*	بله با h(n) پذیرفتنی	بله با h(n) سازگار

فصل چهارم
نتیجه‌گیری و جمع بندی

نتیجه‌گیری و جمع بندی:

در این مقاله به بررسی و بهبود الگوریتم *A برای مسیریابی وسایل نقلیه خودران پرداخته شده است. الگوریتم *Aکه از پرکاربردترین روش‌های جستجو در مسائل پیدا کردن مسیر است، با استفاده از تخمین فاصله، مسیری بهینه با کمترین هزینه را برای رسیدن از مبدا به مقصد می‌یابد. مقاله نوآوری‌هایی از جمله استفاده از راهنما، نقاط کلیدی و گام‌های متغیر را برای بهبود این الگوریتم پیشنهاد داده که بهبود عملکرد و کارایی الگوریتم در شرایط پیچیده و واقعی مانند مسیرهای پرمانع را به همراه داشته است.

در نهایت، به مقایسه نتایج الگوریتم‌های جستجو مختلف (BFS، UCS، Greedy و *A) پرداخته شده و نشان داده شده که الگوریتم بهبودیافته *A با استفاده از نقاط کلیدی و گام‌های متغیر، بهترین عملکرد را از نظر کاهش زمان اجرا و تعداد گره‌های گسترش‌یافته دارد. این بهبودها باعث افزایش ایمنی و دقت در مسیریابی خودروهای خودران می‌شود و الگوریتم را به یک انتخاب مناسب برای کاربردهای عملی در این حوزه تبدیل می‌کند.

منابع و مراجع

https://www.geeksforgeeks.org/a-search-algorithm/

https://stackoverflow.com/questions/56444216/how-can-i-improve-performance-of-the-a-star-algorithm

"Artificial Intelligence: A Modern Approach"

Stuart Russell و Peter Norvig