



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده ریاضی و علوم کامپیوتر

پروژه
علوم کامپیوتر

بررسی الگوریتم A^* و کاربرد آن

نگارش
مهدی عباسعلی پور

استاد راهنما
جناب آقای دکتر قطعی

مهرماه ۱۴۰۲

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

چکیده

در این گزارش قصد بررسی یکی از الگوریتم های جست و جو به نام الگوریتم سرچ A^* را داریم . پس از بیان برخی از مقدمات و پیشینه ، به برخی از شیوه های بهبود این الگوریتم متناسب با فضای مسائل متفاوت می پردازیم . این مسائل غالباً مربوط به وسایل نقلیه خودران و بحث های مسیریابی آنان می باشد . تمرکز بیشتر مقاله به بررسی گستردگی ها و تنوع تغییراتی که می توان بر روی الگوریتم A^* اعمال نمود .

واژه های کلیدی:

هوش مصنوعی ، حمل و نقل هوشمند، مسیریابی هوشمند، بررسی الگوریتم A^*

فهرست مطالب

صفحه

عنوان

۲	۱ مقدمه ای بر الگوریتم A*
۳	۱-۱ اهمیت مسیریابی بهینه در سیستم های حمل و نقل خودران
۳	۲-۱ تاریخچه
۳	۳-۱ شیوه ی کار
۴	۴-۱ ورژن ها
۴	۱-۴-۱ هندسی
۴	۲-۴-۱ بهبودیافته
۵	۳-۴-۱ همیشه اصلاح شونده
۶	۴-۴-۱ پویا ساده
۷	۵-۴-۱ همیشه پویا
۸	۵-۱ جمع بندی
۱۰	مراجع

شکل	فهرست تصاویر	صفحه
۱-۱	بهینه سازی مسیر با استفاده از فیلتر $P(x,y)$	۵
۲-۱	بهینه سازی مسیر با استفاده از فیلتر $W(x,y)$	۵
۳-۱	مسیر یابی توسط الگوریتم کلاسیک A^*	۶
۴-۱	مسیر یابی توسط الگوریتم روش هندسی A^*	۷
۵-۱	شیوه کار A^* بهبود یافته	۸

فصل اول

مقدمه ای بر الگوریتم A^*

۱-۱ اهمیت مسیریابی بهینه در سیستم های حمل و نقل خودران

توسعه سیستم های اتومات مانند هواپیماهای بدون سرنشین، وسایل نقلیه هدایت شونده خودکار و ربات های خودکار مزایای بسیاری را برای انسان داشته اند. توسعه وسایل نقلیه خودران منجر به افزایش ایمنی جاده ها و بهبود مصرف انرژی شده است. برای خودران سازی وسایل نقلیه باید نوعی سیستم داشت تا مسیرهای خود را مطابق با محیطی که قرار است در آن حرکت کنند برنامه ریزی کند. خواسته ی ما در این گونه مسائل این است که این مسیرها تا حد امکان کوتاه باشند و وسیله نقلیه به راحتی حرکت کند و از همه مهمتر اینکه بدون مانع باشند. با این حال، تحقیق در مورد برنامه ریزی حرکتی سیستم های خودران جدید نیست و به دهه ۱۹۵۰ برمی گردد، با الگوریتم هایی مانند جستجوی عرضی و جستجوی عمقی در مرحله اولیه تحقیقات برنامه ریزی حرکتی فرموله شده است. از آن زمان تاکنون چندین پیشرفت بزرگ در توسعه الگوریتم های برنامه ریزی حرکت صورت گرفته است. [۲]. یکی از الگوریتم های مهم برای هوشمندسازی و توانمند سازی این وسایل برای مسیریابی الگوریتم جست و جوی A^* می باشد.

۲-۱ تاریخچه

پیتر هارت (Peter Hart)، نیلز نیلسون (Nils Nilsson) و برترام رافائل (Bertram Raphael) از موسسه پژوهشی استنفورد (Stanford Research Institute) که اکنون با عنوان اس آر آی اینترنشنال^۱ فعالیت می کند، برای اولین بار، مقاله ای پیرامون الگوریتم A^* را در سال ۱۹۶۳ منتشر کردند. این الگوریتم را می توان به عنوان افزونه ای از «الگوریتم دیکسترا»^۲ در نظر گرفت که توسط «ادسخر دیکسترا»^۳ در سال ۱۹۵۹ ارائه شده است. الگوریتم A^* با بهره گیری از «الگوریتم جستجوی کاشف» (جستجوی هیوریستیک Heuristics Search) برای هدایت فرایند جستجو، به کارایی بهتری دست پیدا می کند [۱].

۳-۱ شیوه ی کار

کاری که الگوریتم A^* انجام می دهد آن است که در هر گام، گره را متناسب با مقدار f که پارامتری مساوی با مجموع دو پارامتر دیگر g و h است انتخاب می کند. در هر گام، گره/خانه ای که دارای کمترین مقدار f است را انتخاب و آن گره را پردازش می کند. g و h به روش ساده ای که در زیر بیان شده است محاسبه می شوند.

^۱ SRI International

^۲ Dijkstra's Algorithm

^۳ Edsger Dijkstra

- g هزینه حرکت از نقطه آغاز به یک مربع خاص در شبکه، با دنبال کردن مسیری که برای رسیدن به آن تولید شده است.
- h هزینه تخمین زده شده برای حرکت از یک خانه داده شده در شبکه به مقصد نهایی است. از h معمولاً با عنوان هیوریستیک یاد می‌شود. هیوریستیک چیزی به جز نوعی حدس هوشمندانه نیست. کاربر واقعاً فاصله واقعی را تا هنگام یافتن مسیر نمی‌داند، زیرا هر مانعی (دیوار، آب و سایر موانع) ممکن است در مسیر باشد. راه‌های زیادی برای محاسبه h وجود دارد که در ادامه به آن‌ها اشاره شده است.

۴-۱ ورژن‌ها

این الگوریتم به شیوه‌های متفاوت متناسب با شرایط فضای مسئله گسترش یافته است. در ادامه به برخی از این ورژن‌ها می‌پردازیم:

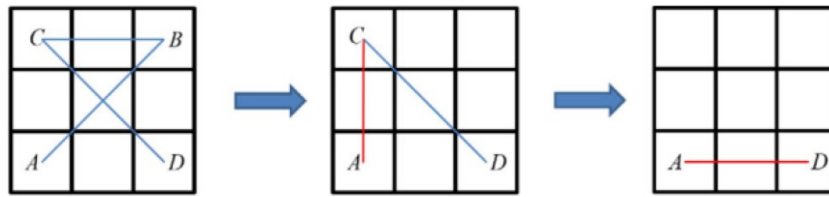
۱-۴-۱ هندسی

الگوریتم هندسی A^* نخستین بار در مسائل مسیریابی در محیط‌های بندری بیان شد؛ در این محیط‌ها عامل علاوه بر وظیفه‌ی حمل و نقل کالا، بایستی خود را در زمان مناسب به ایستگاه شارژ می‌رساند. این الگوریتم اساساً برای رسیدگی به مسائلی مانند زوایای چرخش بزرگ، گره‌های متعددی که معمولاً در مسیرهای متقاطع وجود دارند و مسیرهای دندان‌اره‌ای که توسط الگوریتم کلاسیک A^* تولید می‌شوند، توسعه داده شد.

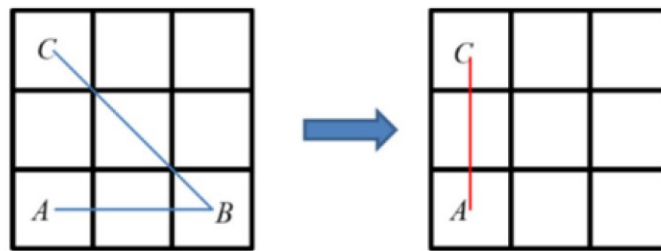
A^* هندسی ابتدا یک نقشه شبکه‌ای از محیط ایجاد می‌کند و موانع غیر ضروری را از بین می‌برد و اشکال نامنظم را منظم می‌کند. پس از این، الگوریتم کلاسیک A^* برای به دست آوردن یک مسیر بدون مانع از موقعیت شروع تا موقعیت نهایی اعمال می‌شود. چنین مسیرهایی به عنوان لیستی از نقاط به دست می‌آیند. سپس الگوریتم هندسی A^* با استفاده از توابع فیلتر $P(x, y)$ (۱-۱) و $W(x, y)$ (۲-۱) گره‌های نامعتبر را از این لیست فیلتر می‌کند. نتایج حاصل از کار این فیلتر نشان می‌دهد که تعداد گره‌های بررسی شده‌ی حاصل از این ورژن نسبت به حالت کلاسیک کاهش قابل ملاحظه‌ای یافته است و در یک نمونه مسئله از ۲۲۴۶ به ۱۰۹ مورد رسیده است. در ۳-۱ و ۴-۱ نقاط آبی گره‌های بررسی شده توسط الگوریتم و نقاط قرمز مسینهایی الگوریتم می‌باشند.

۲-۴-۱ بهبودیافته

این الگوریتم ابتدا هدف و موقعیت شروع را توسط یک خط مستقیم به هم متصل می‌کند. سپس فهرستی از مراکز، که نقاطی روی خط هستند که برخورد کردن به آن‌ها ممنوع می‌باشد را ایجاد می‌کند. برای هر مرکز، دایره‌ای به شعاع معین r در اطراف این نقاط می‌سازد، و سپس مختصات محل‌های



شکل ۱-۱: بهینه سازی مسیر با استفاده از فیلتر $P(x,y)$ [۲]

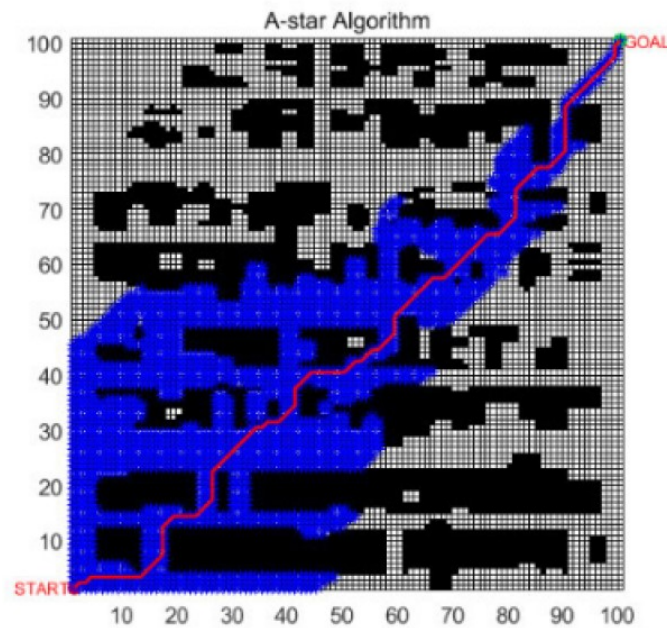


شکل ۱-۲: بهینه سازی مسیر با استفاده از فیلتر $W(x,y)$ [۲]

برخورد این دایره ها را با خط پیدا می کند به عنوان نمونه شکل ۵-۱ را در نظر بگیرید . سپس با انتخاب هر یک از این جفت ها به عنوان نقطه شروع و هدف محلی، با استفاده از آستانه ϵ بررسی می کند که آیا هدف محلی E و شروع محلی بعدی I به یکدیگر نزدیک هستند یا خیر. اگر نزدیک باشند، هدف محلی فعلی به هدف محلی بعدی تغییر می کند، یعنی برنامه ریز باید مسیری را از D به K ، همانطور که در شکل نشان داده شده است، برنامه ریزی کند. پس از انجام این کار، الگوریتم با استفاده از الگوریتم A^* ، مسیرهای موضعی را در اطراف همه جفت های (D,E) و (D,K) مسیریابی می کند. رد نهایت مسیری حاصل می شود که بخشی از آن شامل خطی راست است که کوه ترین مسیر است و موانع با استفاده از الگوریتم A^* دور زده می شوند .

۳-۴-۱ همیشه اصلاح شونده

در عمل، مقدار زمان در دسترس برای محاسبه راه حل بدون برخورد با توجه به محیط ارائه شده، محدود است و این محدودیت زمان بر دشواری حل می افزاید . الگوریتم های Anytime دسته ای از الگوریتم ها هستند که می توانند با محاسبه سریع یک راه حل که لزوماً بهینه نیست شروع کنند و این جواب را به تدریج به سمت بهینگی اصلاح نمایند الگوریتم ARA^* راه حلی برای محدودیت زمان ارائه می دهد، بلکه بسیار کارآمد است، زیرا از تاریخچه ی جستجوی قبلی خود استفاده می کند. این موضوع اجازه می دهد تا از محاسبه مجدد حالت های قبلی جلوگیری شود . الگوریتم از یک اکتشافی وزنی استفاده می کند، بنابراین، $f(n) = g(n) + h(n)$ با تنظیم $\epsilon = 1$ ، الگوریتم کلاسیک A^* راه حل های بهینه را ارائه می دهد. برای $\epsilon > 1$ ، راه حل ها ممکن است بهینه نباشند . نتایج تجربی روی یک ربات نشان می دهد که



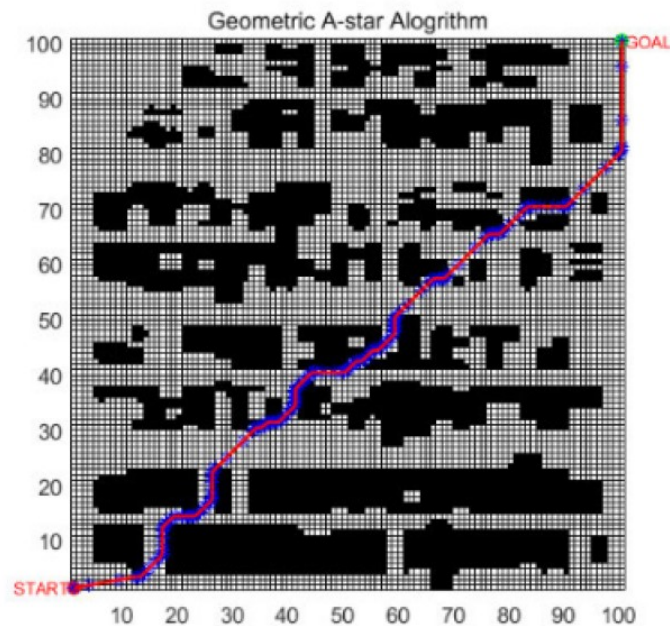
شکل ۱-۳: مسیر یابی توسط الگوریتم کلاسیک A* [۲]

رباتی که از 4D ARA* Planner استفاده می کند حدود ۶۰ ثانیه طول می کشد تا به یک طرح تقریباً بهینه دست یابد.

۴-۴-۱ پویا ساده

الگوریتم D* Lite، مانند الگوریتم A* کلاسیک با شروع از یک گره سعی می کند تا به نقطه ی هدف برسد اما با این فرض که گراف بتواند تغییر کند. تشخیص یک مانع همان اثری را بر روی نمودار خواهد داشت که با حذف یال بین دو گره که مانع بین آنها تشخیص داده می شود، و این معادل تغییر وزن یال به ∞ است. الگوریتم D* Lite چنین تغییرات وزنی را در نمودار محاسبه می کند و به طور مداوم مسیر خودروی خودران را دوباره برنامه ریزی می کند. الگوریتم D* Lite دو امتیاز را برای هر گره از گراف در نظر می گیرد G و RHS. که همانند A* امتیاز G هزینه رفتن از گره شروع به گره فعلی در گراف می باشد. امتیاز

RHS به صورت $RHS(n) = \min(G(n) + C(n, n'))$ تعریف می شود که در آن، n گره فعلی، n' گره قبلی است. و $C(n', n)$ هزینه جابجایی از n به n'. برای یافتن گره بهینه بعدی استفاده می شود. اگر گره بعدی مسدود شود، C برای آن لبه روی ∞ تنظیم می شود، در نتیجه، RHS نیز روی ∞ تنظیم می شود. با استفاده از این امتیازها، الگوریتم مسیری را از هدف تا شروع برنامه ریزی می کند که در نهایت یک مسیر بهینه را برای دنبال کردن به دست می دهد. در صورت مواجهه با یک مانع ناشناخته قبلی، امتیاز RHS با هزینه های جدید اتصال برای گره های آسیب دیده، دوباره محاسبه می شود. نتایج

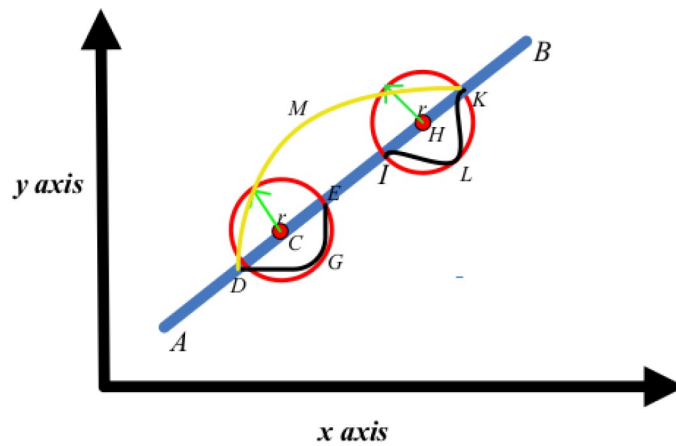


شکل ۱-۴: مسیر یابی توسط الگوریتم روش هندسی A* [۲]

نشان می‌دهد که D* Lite منجر به جست و جوی راس اضافی کمتر، گسترده شدن کمتری می‌شود، که ثابت می‌کند که D* Lite در واقع الگوریتم خوبی برای برنامه‌ریزی مجدد و حرکت در محیط‌های ناشناخته است.

۵-۴-۱ همیشه پویا

محیط‌های پویا را به حساب می‌آورد، که یک مشکل عمده است که باید در برنامه‌ریزی وسایل نقلیه خودران مورد توجه قرار گیرد. از آنجایی که فرد می‌خواهد کل زمان سفر به حداقل برسد، باید برنامه‌ریز راه‌حل‌ها را با بیشترین سرعتی که می‌تواند محاسبه کند. در صورت برخورد با یک مانع جدید، زمان سفر با زمان اضافی برای محاسبه یک مسیر جدید افزایش می‌یابد. برای مقابله با این مشکل، الگوریتم * [۱۱] با ترکیب ایده‌های دو الگوریتم قبلاً مورد بحث، * [۱۲] و * [۱۱] توسعه یافت. مشابه *، راه حل‌هایی با ضریب تورم کاهشی را جستجو می‌کند که به طور مستمر مرز زیر بهینه راه حل را بهبود می‌بخشد. هر تغییری که در محیط شناسایی شود (از طریق سنسور یا دوربین)، به عنوان تغییر در نموداری که مسیر در آن برنامه‌ریزی شده است، منعکس خواهد شد. گره‌های آسیب‌دیده در فهرست باز مانند الگوریتم * ارسال می‌شوند، با اولویت‌های آنها در میان مقدار کلید قبلی و مقدار کلید به‌روزرسانی. سپس گره‌های موجود در لیست پردازش می‌شوند تا زمانی که راه حل فعلی تضمین شود - باشد. بنابراین، * که یک الگوریتم در هر زمان است، امکان برنامه‌ریزی مجدد سریع راه حل‌ها را با محدود کردن بهینه بودن آنها فراهم می‌کند. شکل ۱۳ هزینه‌های



شکل ۱-۵: شیوه کار A^* بهبود یافته [۲]

راه حل متحمل شده و بسط حالت انجام شده توسط H^* و G^* را در برابر احتمال ظاهر شدن مانع نشان می دهد (این آزمایش شامل شبیه سازی یک بازوی رباتیک با ۳ درجه آزادی بود که یک افکتور انتهایی را از طریق یک محیط پویا دستکاری می کرد). [۱۳]. شکل ۱۲ نمونه ای از یک ربات ناوبری ساده را نشان می دهد که از سلول پایین سمت راست شبکه شروع می شود و قرار است به سلول سمت چپ بالای شبکه حرکت کند. سلول های خاکستری نشان دهنده حالت های گسترش یافته توسط هر الگوریتم در موقعیت مربوطه خود هستند. ربات دو قدم به جلو حرکت می کند و شکاف دیوار بالایی را مشاهده می کند. شش الگوریتم برای برنامه ریزی مسیرهای بدون برخورد مورد استفاده قرار گرفت: H^* وزن دار H^* با $H = 5.2$ ، G^* وزن دار G^* با $G = 5.2$ ، H^* و G^* جدول نتایج را برای مثال ربات ناوبری ساده از

۵-۱ جمع بندی

حوزه خودروهای خودمختار پیشرفت چشمگیری داشته است، اما هنوز هم دامنه پیشرفت زیادی در آینده دارد. این مقاله در ابتدا تکنیک های مختلف برنامه ریزی حرکتی را ارائه کرد که تمرکز اصلی آن بر روی الگوریتم جستجوی A^* و تغییرات آن بود. چندین گونه از الگوریتم A^* مورد بررسی قرار گرفت و نشان داده شد که چگونه این تغییرات به حل مسائل با الگوریتم کلاسیک A^* کمک می کند. بسیاری از تغییرات الگوریتم A^* را می توان در آینده با بهینه سازی الگوریتم کلاسیک یا انواع آن توسعه داد و آزمایش کرد. یکی از این تغییرات بالقوه، اصلاح تابع اکتشافی مورد استفاده در این الگوریتم ها است. نویسنده امیدوار است که این مقاله در ارائه یک نمای کلی از انواع مختلف الگوریتم A^* موفق باشد، که می تواند به عنوان مرجع در حین کار بر روی انواع جدید برای ارائه عملکرد بهتر نسبت به الگوریتم های برنامه ریزی حرکتی موجود استفاده شود.

مراجع

- [1] Hesaraki, Elham. A* algorithm in a simple way.
- [2] Paliwal, Pulkit. A survey of a-star algorithm family for motion planning of autonomous vehicles. In *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, pages 1–6. IEEE, 2023.