Iranian Journal of Wargaming
AJA Command and Staff University
http://www.ijwg.ir/
Online ISSN: 2676-7708
Print ISSN: 2676-5284

Analyzing and comparing routing algorithms to find the shortest path in graph transformable problems

Ahmad Allahyari[1]✉

1. Faculty of Industrial Engineering, Islamic Azad University – South Tehran Branch, Tehran, Iran
E-mail: st_a_allahyari@azad.ac.ir

Article Info

Article type:

Research Article

Article history:

Received 20 February 2023
Received in revised form 25 April 2023
Accepted 23 June 2023
Published online 4 September 2023

Keywords:

Pathfinding problem, Dijkstra, IDA, IDA*,*

ABSTRACT

Objective: Search is a problem solving technique in artificial intelligence. Graph search problems are often modeled as graph games between multiple agents. Graph search algorithms are designed with two main applications of graph traversal and finding the shortest path between two vertices of a graph.

Method: In this article, first a comprehensive study on search methods has been conducted and then, various route search algorithms have been investigated and compared in order to achieve the most optimal algorithm in finding the shortest route.

Findings: These algorithms include Dijkstra, A*, and IDA* search algorithms, which have been compared with each other using the three parameters of execution time, time complexity, and space complexity.

Conclusion: To carry out investigations, a hypothetical game state (field) was used in two conditions, with and without obstacles, and programming was done using Python programming language. The results show that the Dijkstra algorithm and the A* algorithm have relatively the same time complexity, and the IDA* algorithm is faster than both; Also, the IDA* method occupies less memory than the A* method.

Cite this article: Allahyari. Ahmad, (2023). Analyzing and comparing routing algorithms to find the shortest path in graph transformable problems. *Journal Title*, 56 (1), 1-20. DOI: http//doi.org/000000000000000000

Publisher: Command and Staff University
DOI: https://orcid.org/0000-0002-0718-5998

دوفصلنامه علمی بازی جنگ
دانشگاه فرماندهی و ستاد آجا
http://www.ijwg.ir/
شاپای چاپی: ۵۲۸۴-۲۶۷۶ شاپای الکترونیکی: ۷۷۰۸-۲۶۷۶

تحلیل و مقایسه‌ی الگوریتم‌های مسیریابی برای یافتن کوتاه‌ترین مسیر در مسائل قابل تبدیل به گراف

احمد الهیاری[۱]✉

۱. دانشجوی دکتری، دانشکده مهندسی صنایع، دانشگاه آزاد اسلامی واحد تهران جنوب، تهران، ایران، رایانامه: st_a_allahyari@azad.ac.ir

اطلاعات مقاله

نوع مقاله:
مقالهٔ پژوهشی

تاریخ دریافت:
۱۴۰۱/۱۲/۰۱

تاریخ بازنگری:
۱۴۰۲/۰۲/۰۵

تاریخ پذیرش:
۱۴۰۲/۰۴/۰۲

تاریخ انتشار:
۱۴۰۲/۰۶/۱۳

کلیدواژه‌ها:
مسئله مسیریابی، دایکسترا، A، IDA**

چکیده

هدف: جستجو تکنیک حل مسئله در هوش مصنوعی است. مسائل جستجو در گراف غالباً بصورت بازی‌های روی گراف بین چند عامل مدل‌سازی می شود. الگوریتم‌های جستجو بر روی گراف با دو کاربرد عمده پیمایش گراف و یافتن کوتاه‌ترین مسیر در بین دو رأس یک گراف طراحی می‌گردند.

روش: در مقاله حاضر، ابتدا مطالعه جامعی بر روی روش‌های جستجو انجام و سپس، به بررسی و مقایسه الگوریتم‌های مختلف جستجوی مسیر جهت دستیابی به بهینه‌ترین الگوریتم در یافتن کوتاه‌ترین مسیر پرداخته شده است.

یافته‌ها: این الگوریتم‌ها شامل الگوریتم‌های جستجوی دایکسترا[۱]، A* و IDA*[۲] می‌باشد که با استفاده از سه پارامتر زمان اجرا، پیچیدگی زمانی و پیچیدگی فضا با یکدیگر مورد مقایسه قرار گرفته شده‌اند.

نتیجه‌گیری: برای انجام بررسی‌ها از یک حالت (زمین) بازی فرضی در دو شرایط با وجود مانع و بدون وجود مانع استفاده شده و برنامه‌نویسی‌ها با استفاده از زبان برنامه‌نویسی پایتون انجام شده است. نتایج نشان می‌دهد که الگوریتم دایکسترا و الگوریتم A* دارای پیچیدگی زمانی نسبتاً یکسانی هستند و الگوریتم IDA* از نظر زمانی سریعتر از هر دو است؛ همچنین روش IDA* حافظه کمتری نسبت به روش A* اشغال می‌کند.

استناد: الهیاری، احمد؛ (۱۴۰۲). تحلیل و مقایسه‌ی الگوریتم‌های مسیریابی برای یافتن کوتاه‌ترین مسیر در مسائل قابل تبدیل به گراف. *دوفصلنامه بازی جنگ*، ۲ (۴)، ۲۰-۱. DOI: https://orcid.org/0000-0002-0718-5998

ناشر: دانشگاه فرماندهی و ستاد ارتش جمهوری اسلامی ایران

DOI:

[1] Dijkstra
[2] Iterative Deepening A Star

مقدمه

امروزه تنوع و گستردگی مسائل در شاخه‌های گوناگون علوم تا حد بسیار زیادی افزایش یافته است و یافتن پاسخ چنین مسائلی در مدت زمان کم، چالشی اساسی محسوب می‌شود. در حال حاضر یکی از موضوعات داغ حوزه تکنولوژی ربات‌های مجهز به هوش مصنوعی هستند و مواجهه ربات‌های متحرک خودران به مسائل برنامه‌ریزی مسیر اجتناب‌ناپذیر است. حوزه تحقیقاتی برنامه‌ریزی مسیر حرکت ربات و مسئله مسیریابی، حرکت بهینه ربات با اجتناب از برخورد با موانع فیزیکی را پشتیبانی می‌کند (H. Wang et al., 2020). در حال حاضر الگوریتم‌های مؤثری در برنامه‌ریزی مسیر تولید شده‌اند مانند دایکسترا، الگوریتم کلونی مورچه‌ها، الگوریتم ژنتیک و الگوریتم ازدحام ذرات و غیره (Qiong et al., 2019). همچنین الگوریتم A* به دلیل عملکرد آسان و رلندمان جستجوی بالا به طور گسترده در برنامه‌ریزی مسیر استفاده می شود. با این حال، خود الگوریتم A* نیز دارای کاستی‌هایی از جمله زوایای جستجوی کوچک، نقاط عطف زیاد و غیره است که باعث می‌شود گاهی مسیر جستجو شده بهینه نباشد (Peng et al., 2015a). از این رو الگوریتم‌های مختلفی بر پایه الگوریتم A* توسعه یافته‌اند که هرکدام مزایا و معایب خود را دارند که از این میان می‌توان به الگوریتم IDA* اشاره نمود (Keshuai & Pingqing, 2021; Peng et al., 2015b; Yu et al., 2014a, 2014b).

برای بررسی کارایی الگوریتم‌های مسیریابی از فاکتورهای زمان، حافظه و سرعت در پاسخ استفاده می‌گردد. در ادامه در این مقاله برای بررسی کارایی، یک مسئله بر مبنای تئوری گراف در یک محیط گسسته تعریف و کارایی ۳ الگوریتم شاخص مسیریابی مورد بررسی عملی قرار گرفته است.

مبانی نظری و پیشینه‌های پژوهش

مبانی نظری

جستجو تکنیک فراگیر حل مسئله در هوش مصنوعی است. دو نوع رویکرد کلی در تکنیک جستجوی هوش مصنوعی جستجوی ناآگاهانه[1] و جستجوی آگاهانه[2] است (Russell & Norvig, 2009).

[1] Uninformed Search
[2] Informed Search

جستجوی ناآگاهانه

جستجوی ناآگاهانه که جستجوی کور[1] نیز نامیده می‌شود، یک کلاس از الگوریتم‌های جستجوی عمومی است که به روش جامع و فراگیر[2] عمل می‌کنند و هیچ اطلاعاتی اضافی یا دانشی در مورد شرایط مسئله ندارند. این جستجوها یک کاوش سیستماتیک و جامع در فضای جستجو را تا زمانی که راه حلی پیدا شود، انجام می‌دهند. از این روش اغلب زمانی استفاده می‌شود که مشکل خیلی پیچیده باشد یا اطلاعات لازم برای اعمال ابتکاری[3] (اکتشافی) یا تکنیک‌های آگاهانه در دسترس نباشد. چند نوع رایج از استراتژی‌های جستجوی ناآگاهانه به شرح زیر می‌باشد:

جستجوی اول عرض[4] (BFS)، جستجوی اول عمق[5] (DFS)، جستجوی عمقی تکراری[6] (IDDFS)، جستجوی هزینه یکنواخت[7] (UCS)، جستجوی دو طرفه[8]، جستجوی محدود به عمق[9] (DLS)، جستجوی تکراری عمق محدود[10] (ID-DLS)، حریص بهترین-اولین جستجو[11] و جستجوی تصادفی[12].

توجه داشته باشید که برخی از این جستجوها ممکن است الگوریتم اصلی را برای دستیابی به عملکرد بهتر یا تطبیق با شرایط مشکل خاص اصلاح یا ترکیب کنند (Fredman & Tarjan, 1987). در واقع، در این روش اکثراً همان الگوریتم‌های حریصانه شناخته شده با استفاده از اطلاعات تعدادی از رئوس شناخته شده و بهره‌برداری از آنها برای شناسایی مسیر کم هزینه، از رأس مبدأ به رأس مقصد کار می‌کند. با این حال،

---

[1] Blind search
[2] Brute-Force
[3] Heuristic
[4] Breadth-First Search
[5] Depth-First Search
[6] Iterative Deepening Depth-First Search
[7] Uniform-Cost Search
[8] Bidirectional Search
[9] Depth-Limited Search
[10] Iterative Deepening Depth-Limited Search
[11] Greedy Best-First Search
[12] Random Search

ایراداتی به عملکرد آنها در هنگام برخورد با گراف‌های بزرگتر وارد شـــده و از وزن منفی پشتیبانی نمی‌کنند.

جستجوی آگاهانه

در جستجوی آگاهانه، از یک برآورد ابتکاری (اکتشافی) به عنوان راهنمایی استفاده می‌شود که منجر به عملکرد کلی بهتر در رسیدن به وضعیت هدف می‌شود (Fredman & Tarjan, 1987) جستجوهای آگاهانه که به عنوان جستجوهای ابتکاری (اکتشافی) نیز شناخته می‌شوند، الگوریتم‌های جستجویی هستند که از اطلاعات یا دانش اضافی برای هدایت فرآیند جستجو به سمت وضعیت هدف استفاده می‌کنند. این الگوریتم‌ها دارای یک تابع اکتشافی هستند که هزینه رسیدن به حالت هدف را از هر حالت ممکن تخمین می‌زند، که به اولویت‌بندی امیدوارکننده‌ترین گره‌ها در طول جستجو کمک می‌کند. به این ترتیب، جستجوهای آگاهانه اغلب می‌توانند راه حل را کارآمدتر و با گسترش گره کمتر نسبت به جستجوهای ناآگاهانه پیدا کنند. اطلاعات اکتشافی معمولاً بر اساس مسئله‌ای است که حل می‌شود و می‌تواند به الگوریتم کمک کند تا تصمیمات آگاهانه‌ای در مورد اینکه کدام گره‌ها را در ادامه بررسی کند، بگیرد. الگوریتم‌های جستجوی آگاهانه معمولاً در برنامه‌های هوش مصنوعی و علوم رایانه مانند مسیریابی، بازی کردن و پردازش زبان طبیعی استفاده می‌شوند (Kapi, 2020). چند نوع رایج از استراتژی‌های جستجوی آگاهانه به شرح زیر می‌باشد:

جســـتجوی A*، جســـتجوی اول-بهترین حریصـــانه[1]، جســـتجوی تعمیق تکراری A* (IDA*)، جســـتجوی تپه نوردی[2]، جســـتجوی شـــبیه‌ســـازی تبرید[3]، جســـتجوی پرتو[4]، جستجوی بازگشتی اول-بهترین[5] و جستجوی SMA*. توجه داشته باشید که برخی از

[1] Greedy Best-First Search
[2] Hill Climbing Search
[3] Simulated Annealing Search
[4] Beam Search
[5] Recursive Best-First Search

این الگوریتم‌ها ممکن است الگوریتم اصلی را برای دستیابی به عملکرد بهتر یا سازگاری با شرایط مشکل خاص ترکیب یا اصلاح کنند (Pemmaraju & Skiena, 2003)

استراتژی‌های تکامل

چهار استراتژی مختلف وجود دارد که تکامل یک الگوریتم جستجو را مشخص می‌نماید که عبارتند از:

۱. کامل بودن[1]: به الگوریتمی کامل گفته می‌شود که حداقل یک راه حل ارائه دهد.

۲. بهینه بودن[2]: به الگوریتمی بهینه گفته می‌شود که تضمین نماید راه حل یافت شده بهترین یا کمترین هزینه را دارا است.

۳. پیچیدگی زمانی[3]: حد بالایی زمان مورد نیاز برای یافتن راه حل تابعی از پیچیدگی مسئله است.

۴. پیچیدگی فضا[4]: حد بالایی فضای ذخیره‌سازی (حافظه) مورد نیاز در هر لحظه از جستجو تابعی از پیچیدگی فضا است.

الگوریتم دایکسترا

در نظریه گراف، الگوریتم دایکسترا یکی از الگوریتم‌های پیمایش گراف است که توسط دانشمند هلندی علوم رایانه، اِدْسْخِر دیْکْسْترا[5] در سال ۱۹۵۹ ارائه شد. این الگوریتم یکی از الگوریتم‌های پیمایش گراف است که مسئله کوتاه‌ترین مسیر از مبدأ واحد را برای گراف‌های وزن‌داری که یال با وزن مثبت دارند، حل می‌کند و در نهایت، با ایجاد درخت کوتاه‌ترین مسیر، کوتاه‌ترین مسیر از مبدأ به همه رأس‌های گراف را به دست می‌دهد (Dijkstra, 1959).

[1] Completeness
[2] Optimality
[3] Time Complexity
[4] Space Complexity
[5] Edsger Wybe Dijkstra

هم‌اکنون از الگوریتم دایکسترا برای یافتن خودکار سمت حرکت در مکان‌های فیزیکی و یافتن مسیرهای رانندگی در نرم‌افزارهای کاربردی مانند مپ کوئست[1] یا گوگل مپ استفاده می‌شود (Lanning et al., 2014). در نرم‌افزارهای کاربردی شبکه یا مخابرات، از الگوریتم دایکسترا برای حل مسئله یافتن مسیر با حداقل تاخیر استفاده می‌شود. به عنوان مثال، در مسیریابی شبکه انتقال اطلاعات، هدف یافتن مسیری است که بسته‌های داده از طریق یک شبکه سوئیچینگ با حداقل تاخیر عبور کنند. همچنین در مسائل کاربردی مختلفی که به نوعی یافتن کوتاه‌ترین مسیر اهمیت دارد مانند چیدمان تجهیزات کارخانه و تأسیسات، رباتیک، حمل‌ونقل و طراحی یکپارچه‌سازی کلان‌مقیاس[2] (VLSI)، کاربرد دارد. الگوریتم دایکسترا در انواع مختلفی وجود دارد (Whiting & Hillier, 1960). نوع اصلی آن کوتاه‌ترین مسیر را بین دو گره پیدا می‌کند، اما نوع رایج‌تر یک گره را به عنوان گره «منبع و مبدأ» تثبیت می‌کند و کوتاه‌ترین مسیرها را از مبدأ به همه گره‌های دیگر در گراف پیدا و کوتاه‌ترین درخت مسیر را تولید می‌کند (Skiena, 1991; Vamja et al., 2017; Whiting & Hillier, 1960).

جستجوی A*

الگوریتم A*، الگوریتمی است که به طور گسترده برای «مسیریابی[3]» و «پیمایش گراف[4]» مورد استفاده قرار می‌گیرد. پیمایش گراف، فرایند پیدا کردن مسیر بین رئوس مختلف یک گراف است. الگوریتم A* به دلیل کارایی و صحتی که دارد، به طور گسترده در موارد کاربردی گوناگون مورد استفاده قرار می‌گیرد (Candra et al., 2021). الگوریتم A* مبتنی بر استفاده از روش‌های ابتکاری (اکتشافی) برای دستیابی به پاسخ بهینه و کامل است و نوعی از الگوریتم اول-بهترین[5] است (Gibert & Nofrarias, 2021). هنگامی که یک الگوریتم جستجو خاصیت بهینه بودن را دارد، به این معنی

[1] Map quest
[2] Very Large-Scale Integration
[3] Pathfinding
[4] Graph Traversal
[5] best-first search

است که یافتن بهترین راه حل ممکن تضمین شده است، در مورد الگوریتم A* این به معنی تضمین یافتن کوتاه‌ترین مسیر بین مبدأ و مقصد است. هنگامی که یک الگوریتم جستجو ویژگی کامل بودن را دارد، به این معنی است که اگر راه حلی برای یک مشکل معین وجود داشته باشد، این الگوریتم یافتن آن را تضمین می‌کند (Guruji et al., 2016).

برای اولین‌بار، در سال ۱۹۶۳ میلادی «پیتر هارت[1]»، «نیلز نیلسون[2]» و «برترام رافائل[3]» از «موسسه پژوهشی استنفورد» مقاله‌ای پیرامون الگوریتم A* منتشر کردند. این الگوریتم را می‌توان به عنوان افزونه‌ای از «الگوریتم دایکسترا» در نظر گرفت که در سال ۱۹۵۹ ارائه شده است. الگوریتم A* با بهره‌گیری از «یک تخمین ابتکاری (اکتشافی)» برای هدایت فرایند جستجو به جواب بهینه، با کارایی بهتر عمل می‌کند (Hart et al., 1968).

الگوریتم A* برخلاف دیگر روش‌های پیمایش گراف، دارای «حافظه» است. این یعنی، A* الگوریتم بسیار هوشمندی است و همین ویژگی، موجب تمایز آن از دیگر الگوریتم‌های مرسوم می‌شود. شایان ذکر است که بسیاری از بازی‌ها و نقشه‌های مبتنی بر وب، از الگوریتم A* برای پیدا کردن کوتاه‌ترین مسیر استفاده می‌کنند (Meng & Zhang, 2019; Zhang et al., 2020).

روش کار الگوریتم A* به این صورت است که فرض کنید در یک شبکه گرافی یک خانه به عنوان خانه شروع (مبدأ) و یک خانه به عنوان خانه هدف (مقصد) در نظر گرفته شده و می‌خواهیم (در صورت امکان) از خانه شروع آغاز به حرکت نموده و با سریع‌ترین حالت ممکن به خانه مقصد برسیم. کاری که الگوریتم A* انجام می‌دهد آن است که در هر گام، گره بعدی را با توجه به مقداری بنام «f» که پارامتری مساوی با مجموع دو پارامتر دیگر g و h است انتخاب می‌کند. در هر گام، گره/خانه‌ای که دارای کمترین مقدار f

[1] Peter Hart
[2] Nils Nilsson
[3] Bertram Raphael

است را انتخاب و آن گره/خانه را پردازش می‌کند، g وh به روش ساده‌ای که در ادامه بیان شده محاسبه می‌شوند.

هر بار که A* یک حللت را بررسی می‌کند، تابع هزینه‌ای به نام f(n) را محاسبه می‌نماید که هزینه حرکت به n گره مجاور (همسایه) است، بعد از محاسبه هزینه حرکت به تمام گره‌های مجاور، گره‌ای با کمترین مقدار f(n) به عنوان مسیر حرکت انتخاب و در لیست مسیر حرکت قرار می‌گیرد. این مقدار با فرمول زیر محاسبه می‌شود:

$$f(n) = g(n) + h(n) \qquad (۰.۱)$$

g، هزینه حرکت از نقطه مبدأ به یک گره خاص در شبکه، با دنبال کردن مسیری که برای رسیدن به آن تولید شده است. بنا به تعریفی دیگر g(n) مقدار کوتاه‌ترین مسیر از گره شروع (مبدأ) به گره n است.

h، هزینه تخمین زده شده برای حرکت از یک خانه داده شده در شبکه به مقصد نهایی است. به بیان دیگر، h(n) یک تقریب ابتکاری (اکتشافی) از مقدار فاصله گره تا مقصد است. مقدار اکتشافی 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 عبارتند از روش اکتشافی دقیق و روش‌های اکتشافی تخمین که دارای انواع فاصله منهتن[1]، فاصله قطری[2] و فاصله اقلیدسی [3] می‌باشد (H. Wang et al., 2021). کارایی A* بسیار به مقدار اکتشافی h(n) وابسته است و بسته به نوع مسئله، ممکن است برای یافتن راه‌حل بهینه نیاز به استفاده از یک تابع اکتشافی خاص برای آن داشته باشیم (Phaneendhar Reddy Vanam, 2011). به طور مکرر گره‌ای را با کمترین f(n) انتخاب و به دنبال آن مسیر گسترش می‌یابد.

---

[1] Manhattan Distance
[2] Diagonal Distance
[3] Euclidean Distance

الگوریتم *IDA

تعمیق تکراری *A[1] (*IDA) یک الگوریتم پیمایش گراف و جستجوی مسیر است که می‌تواند کوتاه‌ترین مسیر را بین یک گره شروع تعیین‌شده و هر عضوی از مجموعه‌ای از گره‌های هدف در یک گراف وزن‌دار پیدا کند. این جستجو نوعی از جستجوی اول عمق با تعمیق تکراری است که ایده استفاده از یک تابع ابتکاری برای ارزیابی هزینه باقیمانده برای رسیدن به هدف از الگوریتم جستجوی *A بهره می‌گیرد. با این حال، الگوریتم *IDA با جستجوی استاندارد *A متفاوت است زیرا از یک فرآیند تکراری برای کاوش گره‌ها استفاده می‌کند و به تدریج عمق جستجو را افزایش می‌دهد تا زمانی که گره هدف پیدا شود. الگوریتم *IDA تمام عملیاتی را که *A انجام می‌دهد را انجام می‌دهد و دارای ویژگی‌های بهینه برای مکان‌یابی کوتاه‌ترین مسیر است، اما حافظه کمتری را اشغال می‌کند(Q. Wang et al., 2021) . از آنجایی که این الگوریتم از نوع جستجوی اول عمق است، استفاده از حافظه آن کمتر از *A است، اما بر خلاف جستجوی عمیق تکراری معمولی، روی کاوش بر روی گره‌هایی با نتایج بهتر تمرکز می‌کند و بنابراین، در همه جای درخت جستجو عمق یکسانی را بررسی نمی‌کند. برخلاف *A، *IDA از برنامه‌نویسی پویا استفاده نمی‌کند و به همین علت اغلب به کاوش در گره‌های یکسان ختم می‌شود(Phaneendhar Reddy Vanam, 2011; Russell & Norvig, 2009).

الگوریتم *IDA با یک جستجوی عمق شروع می‌شود. در ابتدا، حداکثر هزینه برای رسیدن به گره هدف را تعیین می‌کند تا تخمین اکتشافی فاصله گره شروع شده باشد. سپس، الگوریتم گراف را بررسی می‌کند و ارزیابی می‌کند که آیا هزینه کل مسیر فعلی از حداکثر هزینه تعیین شده برای این عمق بیشتر است یا خیر. اگر این اتفاق بیفتد، الگوریتم به عقب برمی‌گردد و حداکثر هزینه را افزایش می‌دهد و جستجو را از گره والد از سر می‌گیرد. این فرآیند تا زمانی تکرار می‌شود که یا گره هدف پیدا شود یا حداکثر هزینه از حداقل هزینه‌ای که تاکنون پیدا شده است بیشتر شود. الگوریتم *IDA یک

[1] Iterative Deepening A* algorithm

الگوریتم جستجوی کارآمد و کامل از نظر حافظه است که تضمین می‌کند که راه حل بهینه را در زمان نمایی پیدا کند، حتی زمانی که فضای جستجو نامحدود یا ناشناخته است (Russell & Norvig, 2009).

پیشینه‌های پژوهش

الگوریتم‌های کوتاه‌ترین مسیر در جستجوی گراف بدلیل کاربردهای فراوان آن در دنیای واقعی از جمله شبکه‌های حمل‌ونقل، شبکه‌های ارتباطی و شبکه‌های اجتماعی با ایده‌ها و تکنیک‌های جدیدی که به طور منظم توسعه می‌یابند پیوسته در حال رشد است (Gross & Yellen, 2005; Hassan et al., 2023). مطالب علمی منتشر شده در مورد این موضوع در سال‌های اخیر به دلیل پیشرفت در قدرت محاسباتی و رشد داده‌های بزرگ افزایش یافته است. علاوه بر این، ظهور برنامه‌های کاربردی جدید نیز به افزایش فعالیت‌های تحقیقاتی کمک کرده است.

یوکسی لی[1] و همکارانش در سال ۲۰۰۷ «الگوریتم‌های کوتاه‌ترین مسیر چند محدودیتی سریع و دقیق» را طراحی کردند. آن‌ها دو الگوریتم A* و Fringe MCSP (کوتاه‌ترین مسیر با محدودیت چندگانه) را آزمایش و کارایی الگوریتم‌ها در دو وضعیت مختلف بررسی نمودند که الگوریتم Fringe MCSP، که خود نیز بر اساس منطق الگوریتم A* طراحی شده است در هر دو مورد بررسی شده عملکرد بهتری دارد. هر دو الگوریتم جزو مسائل سخت NP بودند (Li et al., 2007).

آنکیت باهادوریا[2] و همکاران در سال ۲۰۱۴، الگوریتم A* را برای جستجوی مسیر بهینه فراگیر (جهانی) بر اساس ارزیابی گره همسایه بهینه کردند. در این مقاله مسیر بهینه با رویکرد ابتکاری جدیدی بر روی الگوریتم A* برنامه‌ریزی شده است (Bhadoria & Singh, 2014)

[1] Yuxi Li
[2] Ankit Bhadoria

دوشان[1] و همکارانش در سال ۲۰۱۴ در مقاله‌ای با عنوان "برنامه‌ریزی مسیر با الگوریتم *A اصلاح شده برای یک ربات متحرک" الگوریتم *A را به دو صورت مجزا تنظیم می‌نمایند که نوع اول سرعت در پاسخ‌گویی را در اولویت قرار می‌دهد و گونه دوم رسیدن به کوتاه‌ترین مسیر را تضمین می‌نماید (Duchon et al., 2014).

پاوان جیندال[2] و همکاران در سال ۲۰۱۰ میلادی الگوریتم کوتاه‌ترین مسیر تک مبدأ و همه مسائل زوج کوتاه‌ترین مسیر مانند الگوریتم‌های بلمن فورد[3]، دایکسترا و فلوید وارشال[4] را تجزیه و تحلیل کرده‌اند و پیچیدگی زمان و مکان را به دست آوردند (Jindal & Kumar, 2010).

تپاریت سینتامرونگروک[5] و همکاران، در سال ۲۰۱۳ میلادی، الگوریتم مسیریابی *A و ناوبری نقطه راه[6] را در سیستم عامل اندروید و آی او اس[7] بر اساس زمان جستجو مقایسه کرده‌اند. با توجه به نمونه آزمایشی اجرا شده، *A بهتر از الگوریتم نقطه راه عمل کرد (Sinthamrongruk et al., 2013).

آبشیک گویال[8] و همکارانش در سال ۲۰۱۴ *A و دایکسترا را برای یافتن کوتاه‌ترین مسیر بین مبدأ و مقصد بر اساس زمان اجرا مقایسه و ثابت کرده‌اند که *A تقریباً با صرف نصف زمان کمتر از الگوریتم دایکسترا مسئله را حل می‌کند (Goyal et al., 2014).

[1] Duchon, František
[2] Pawan Jindal
[3] Ball man ford
[4] Floyd warshall
[5] Thepparit Sinthamrongruk
[6] Waypoint Navigator
[7] IOS
[8] Abhishek Goyal

شروان شارما[1] و پال[2]در سال ۲۰۱۵، در مقاله‌ای با عنوان "کوتاه‌ترین مسیر جستجوی شبکه جاده‌ای با استفاده از الگوریتم *A" تجزیه و تحلیل کاملی بر روی الگوریتم دایکسترا و *A برای یافتن کوتاه‌ترین مسیرها در محیط‌های دارای موانع و بدون مانع با استفاده از تکنیک جستجوی دوطرفه انجام داده‌اند. در نهایت نشان داده شده که الگوریتم *A بهترین مسیر را پیدا می‌کند (Sharma, 2015).

هاریتا وامجا[3] و همکارانش در سال ۲۰۱۷ به تجزیه و تحلیل الگوریتم‌های دایکسترا، *A و *IDA پرداختند و این الگوریتم‌ها را در نظرگرفتن پارامترهای مختلفی با یکدیگر مقایسه کردند. با این حال، مهم‌ترین پارامتر که زمان اجرا است را مورد توجه و بررسی قرار ندادند (Vamja et al., 2017).

تانگ[4] و همکارانش در سال ۲۰۲۱ در مقاله‌ای با عنوان "الگوریتم هندسی *A: یک الگوریتم *A بهبود یافته برای برنامه‌ریزی مسیر AGV در یک محیط بندری" الگوریتم *A را برای برنامه‌ریزی حرکت خودروهای بدون سرنشین صنعتی زمینی برای محیط باراندازهای بندری تنظیم نموده‌اند (Tang et al., 2021).

سونگ[5] و همکارانش در سال ۲۰۲۱ در مقاله‌ای با عنوان "بررسی برنامه‌ریزی حرکت ربات متحرک بر مبنای الگوریتم *A بهبود یافته" الگوریتم *A سنتی را برای برنامه‌ریزی مسیر با استفاده از پردازش درون‌یابی کانولوشن کامل برای گسترش ناحیه جستجو و کاهش نقطه عطف مسیر بهبود دادند (Song & Ma, 2021).

روش‌شناسی پژوهش

بیان مسئله

[1] Sharwan .Kr. Sharma
[2] B.L. Pal
[3] Harita vamja
[4] Tang, Gang
[5] Yu Song

هم‌اکنون در بازار تکنولوژی‌های جدید زمان مهمترین عامل برای الگوریتم جستجوی مسیر است. هزینه‌های مرتبط با حافظه روز به روز کاهش می‌یابد و سرعت وسایل نقلیه رباتیک با کاربری‌های مختلف روز به روز افزایش می‌یلبد. بنابراین، لازم است تجزیه و تحلیل شود که کدام الگوریتم نتیجه بهتری را برای جستجوی مسیر در سناریوهای مختلف و متفاوت، اساساً با مانع و بدون وجود مانع ارائه می‌دهد.

اهداف تحقیق

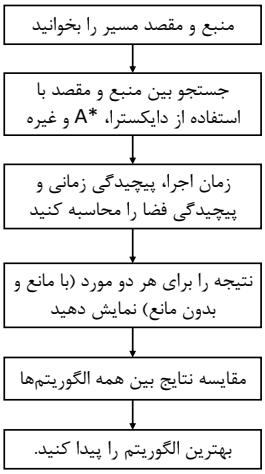
در این مقاله، پس از انجام مطالعه‌ای کامل در رابطه با روش‌های جستجوی و یافتن کوتاه‌ترین مسیر، با استفاده از زبان برنامه‌نویسی پایتون نسبت به پیاده‌سازی روش‌های فوق اقدام گردیده است. سپس، با تعریف یک مسأله در برنامه‌نویسی انجام شده به مقایسه سه الگوریتم پرکاربرد در یافتن کوتاه‌ترین مسیر پرداخته شده است. این مقایسه با توجه به معیارهای پیچیدگی زمانی، پیچیدگی فضا و زمان اجرا انجام می‌شود. در نهایت نیز به تحلیل و توسعه نتایج به دست آمده پرداخته می‌شود.

سناریو آزمون

برای بررسی کارایی الگوریتم‌های مختلف جستجوی کوتاه‌ترین مسیر به شرح زیر عمل گردیده است:

ابتدا یک محیط بررسی (زمین بازی) طراحی و به صورت چهارضلعی شبکه‌بندی گردیده است. سپس یک خانه (سلول) به عنوان مبدأ و یک خانه به عنوان مقصد بطور تصادفی انتخاب گردیده است. سپس، مسئله یافتن کوتاه‌ترین مسیر بین مبدأ و مقصد توسط الگوریتم‌های دایکسترا، A* و IDA* مورد آزمون قرار گرفته و سه پارامتر زمان اجرا، پیچیدگی زمانی و پیچیدگی فضا به عنوان خروجی محاسبه و استخراج گردیده است. همچنین، شرایط مسئله با انتخاب برخی خانه‌ها به عنوان مانع تغییر یافته و مجددا اثر وجود موانع مورد آزمون قرار گرفته و نتایج استخراج گردیده است. در نهایت، جداول و نمودارهایی برای مقایسه‌ی نتایج تهیه و بهترین الگوریتم در شرایط طراحی شده برای

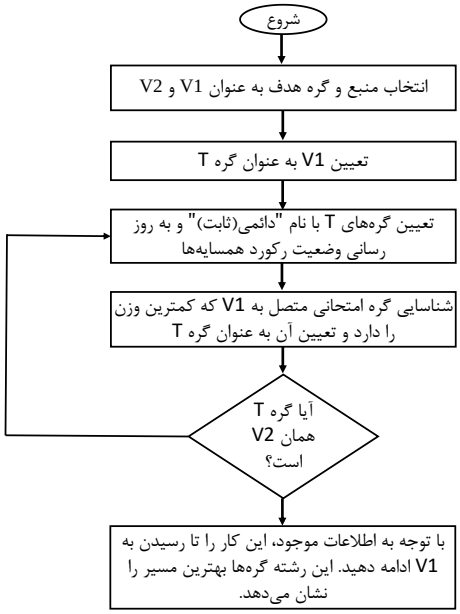
تحقیق انتخاب گردیده است. شکل (۱) سناریو آزمون را به صورت فلوچارت نمایش می‌دهد.



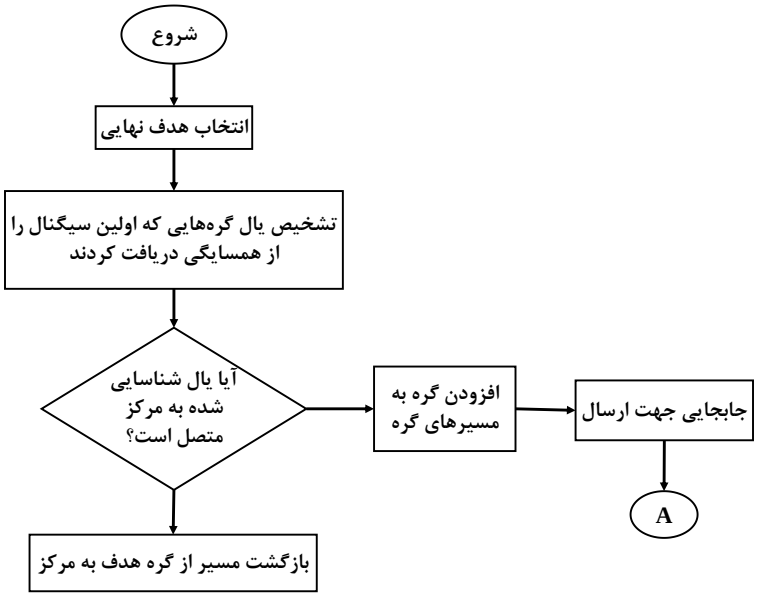
شکل (۱) سناریو آزمون

همچنین، شکل(۲)، شکل(۳) و شکل(۴) بطور خلاصه روش عملکرد سه الگوریتم مهم که مورد بررسی قرار گرفته‌اند را به ترتیب به صورت فلوچارت نشان می‌دهد. این الگوریتم‌ها توسط ویرایش سوم زبان برنامه‌نویسی پایتون[۱] پیاده‌سازی گردیده است. در برنامه نوشته شده، ساخت شبکه و یافتن کوتاه‌ترین مسیر با استفاده کتابخانه‌های نتورک-ایکس[۲] و سای‌پای[۳] انجام شده است. در بخش بعد، نتایج به دست آمده از برنامه‌نویسی که شامل سرعت و تعداد عملیات انجام شده توسط برنامه با توجه به الگوریتم انتخابی و نوع مسیر آورده شده و با یکدیگر مورد مقایسه قرار گرفته شده است.

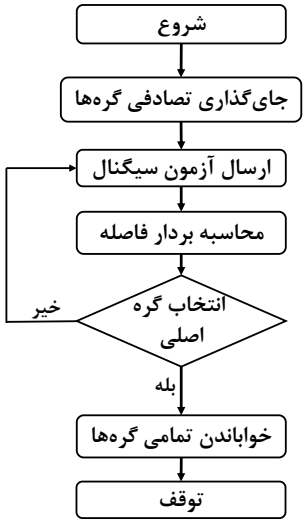
[1] Python
[2] Networkx
[3] Scipy



شکل (۲) فلوچارت الگوریتم Dijkstra



شکل (۳) نمودار جریان الگوریتم *A



شکل (۴)نمودار جریان الگوریتم *IDA

تجزیه و تحلیل یافته‌های پژوهش

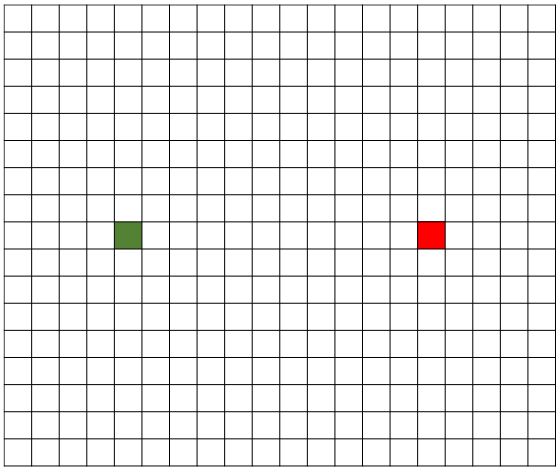
برای محاسبه و مقایسه روش‌های مختلف یافتن کوتاه‌ترین مسیر ابتدا با استفاده از برنامه نوشته شده در نرم‌افزار بهترین مسیر یافته می‌شود. سپس، با توجه میزان زمان حل مسئله و با استفاده از کتابخانه‌های موجود در پایتون سرعت یافتن کوتاه‌ترین مسیر بدست می‌آید. در نهایت، در هر الگوریتم با توجه به مسیرهای پیموده شده جهت رسیدن از شبکه مبدأ به شبکه مقصد تعداد شبکه‌های پیموده شده بدست می‌آید که نشان‌دهنده تعداد عملیات انجام شده در هر الگوریتم می‌باشد. تعداد عملیات انجام شده در واقع رابطه‌ی مستقیمی با میزان اشغال حافظه دستگاه دارد و هزینه محاسباتی را افزایش می‌دهد. روابط مورد استفاده جهت بدست آوردن طول مسیر، زمان و تعداد عملیات عبارتند از:

طول = تعداد کل شبکه

زمان = زمان پایان - زمان شروع

عملیات = شبکه شروع + تعداد شبکه باز + تعداد شبکه جایگزین + هدف

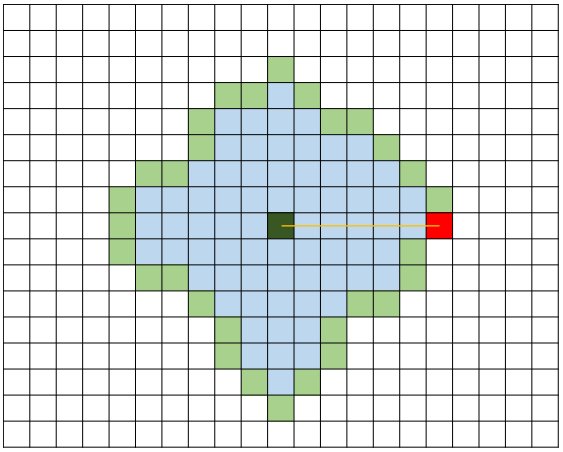
انجام محاسبات نیازمند یک محیط مناسب می‌باشد که در این مقاله از یک زمین بازی مربعی با شبکه‌بندی چهار ضلعی استفاده شده است. شکل (۵) تصویر محیط (زمین) بازی انتخاب شده برای ابزار رابط کاربری را نشان می‌دهد که در آن شبکه سبز رنگ نشان‌دهنده نقطه شروع حرکت و شبکه قرمز رنگ نماینده نقطه هدف در مسیریابی است.



شکل (۵) ابزار رابط کاربری

بررسی مسیر یافتن کوتاه‌ترین مسیر در تصویر محیط بدون مانع:

ابتدا با استفاده از الگوریتم دایکسترا محاسبات جهت یافتن کوتاه‌ترین مسیر انجام شد. شکل (۶) نمایی از مسیرهای پیموده شده توسط این الگوریتم برای رسیدن به شبکه مقصد را نشان می‌دهد.



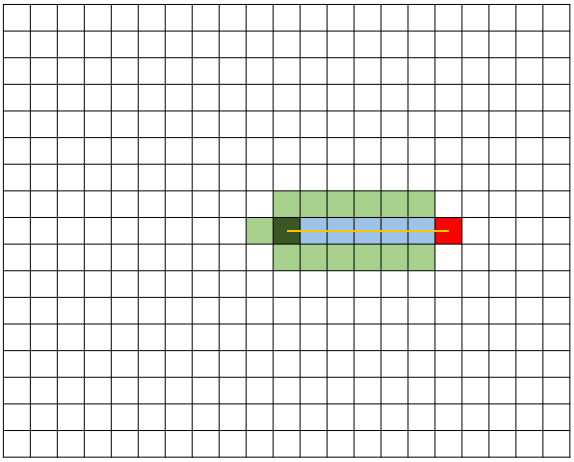
شکل (۶) الگوریتم دایکسترا

همانطور که ملاحظه می‌شود این الگوریتم برای یافتن کوتاه‌ترین مسیر تعداد شبکه‌های قابل ملاحظه‌ای مورد بررسی قرار داده است. تعداد عملیات انجام شده توسط این الگوریتم ۱۷۴ بار می‌باشد که از طریق رابطه زیر بدست می‌آید.

عملیات=۱(شروع) +۳۲(باز) +۱۴۰(جایگزین/پس‌نیاز) +۱(هدف) =۱۷۴

همچنین، مقدار زمان در محاسبات با توجه به زمان پردازنده کامپیوتر و از طریق نرم‌افزار پایتون بدست آمده است که مقدار آن حدود ۵.۳۵ ثانیه می‌باشد.

پس از الگوریتم دایکسترا، الگوریتم A* جهت یافتن کوتاه‌ترین مسیر رسیدن به مقصد مورد بررسی قرار گرفته شد. شکل (۷) تصویر مسیرهای پیموده شده توسط این الگوریتم را نشان می‌دهد.



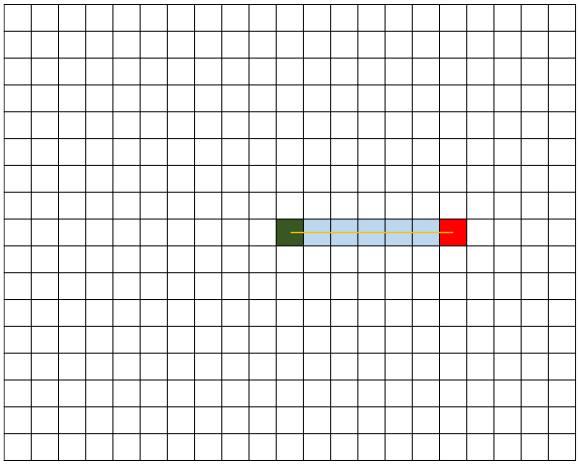
شکل (۷) الگوریتم *A

تعداد عملیات انجام شده توسط نرم‌افزار با توجه به رابطه اشاره شده در قسمت‌های قبل به صورت زیر بدست می‌آید.

عملیات=۱(شروع) +۱۸(باز) +۷(جایگزین/پس‌نیاز) +۱(هدف) =۲۷

همانطور که ملاحظه می‌شود تعداد عملیات انجام شده نسبت به الگوریتم دایکسترا به شکل قابل ملاحظه‌ای کاهش می‌یابد. همچنین، زمان انجام محاسبات در نرم‌افزار حدود ۴.۰۸ ثانیه می‌باشد که حدود ۱.۲۷ ثانیه سریعتر از روش دایکسترا به نتیجه می‌رسد.

در نهایت نیز از الگوریتم IDA* جهت انجام محاسبات استفاده شد که نتایج آن در شکل (۸) ارائه شده است.



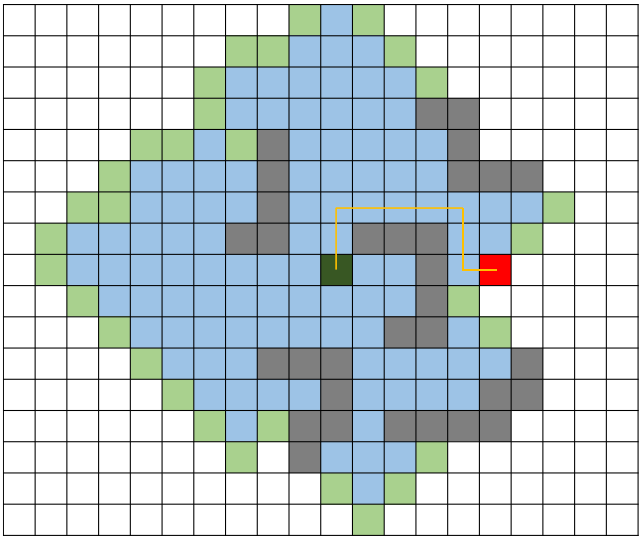
شکل (۸) الگوریتم *IDA

تعداد عملیات انجام شده توسط این الگوریتم ۱۸ عدد در زمان ۱.۴۴ ثانیه می‌باشد که به صورت زیر محاسبه شده است.

عملیات=۱(شروع) +۱۰(باز) +۶(جایگزین/پس‌نیاز) +۱(هدف) =۱۸

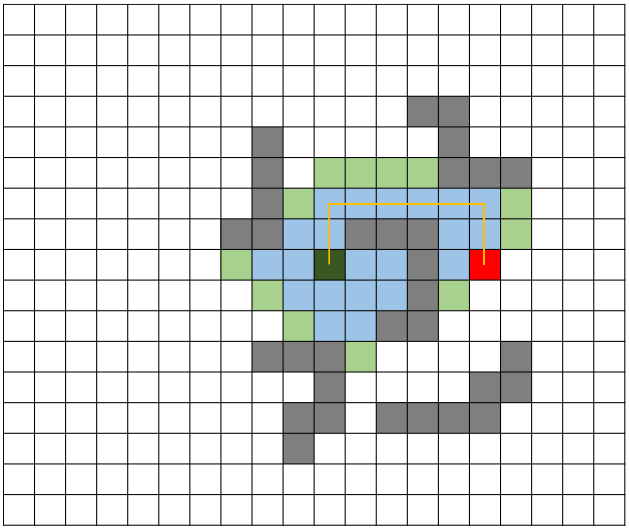
بررسی مسیر یافتن کوتاه‌ترین مسیر در تصویر محیط با مانع

در این قسمت جهت مقایسه هرچه بهتر کارایی الگوریتم‌های جستجو عبور از چند خانه زمین بازی را ممنوع فرض کرده و با وجود موانع ثابت محاسبات انجام شده است. مسیریابی انجام شده توسط الگوریتم دایکسترا با بررسی تعداد ۲۴۴ خانه شبکه بهترین و کوتاه‌ترین مسیر را در مدت زمان ۹.۳۱ ثانیه بدست آورد. شکل (۹) تصویری از پیمایش انجام شده با الگوریتم دایکسترا را نشان می‌دهد.



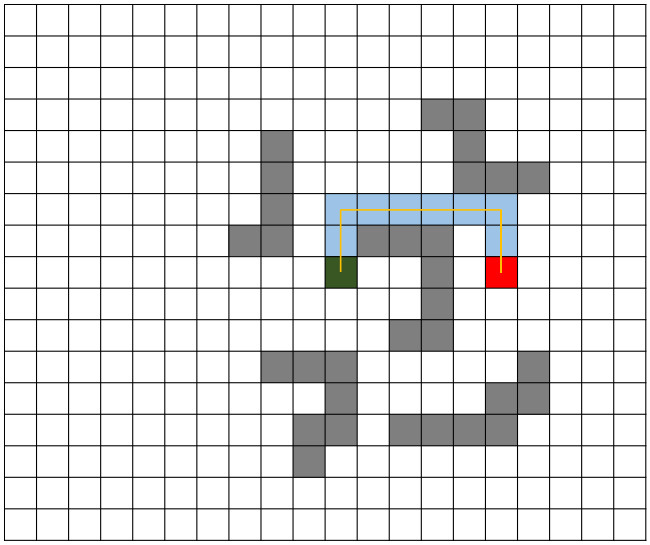
شکل (۹) الگوریتم دایکسترا با وجود مانع

شکل (۱۰) تصویر مسیریابی انجام شده توسط الگوریتم A* را نمایش می‌دهد. این الگوریتم با انجام ۵۸ عملیات در مدت زمان ۲.۸۷ ثانیه مسیر مورد نظر را جهت رسیدن به شبکه مقصد بدست آورد.



شکل (۱۰) الگوریتم A* با وجود مانع

الگوریتم *IDA نیز با انجام ۲۷ عملیات در مدت زمان ۱.۰۳ ثانیه کوتاه‌ترین مسیر را جهت عزیمت از شبکه مبدا به شبکه مقصد بدست آورد که در شکل (۱۱) تصویر نحوه پیمایش آن قابل مشاهده است.



شکل (۱۱) الگوریتم *IDA با وجود مانع

آنالیز نتایج

در قسمت قبل، مسیریابی در زمین بازی با وجود و بدون وجود مانع با استفاده از الگوریتم‌های مختلف مسیریابی انجام گرفت. نتایج نشان داد که الگوریتم مورد استفاده تاثیر بسیار زیادی در هزینه‌های محاسباتی شامل زمان انجام محاسبات و حافظه اشغال شده توسط کامپیوتر که اثر تعداد عملیات محاسباتی را نشان می‌دهد دارد. در این قسمت به تحلیل و مقایسه الگوریتم‌های ارزیابی شده جهت یافتن بهترین مسیر پرداخته می‌شود.

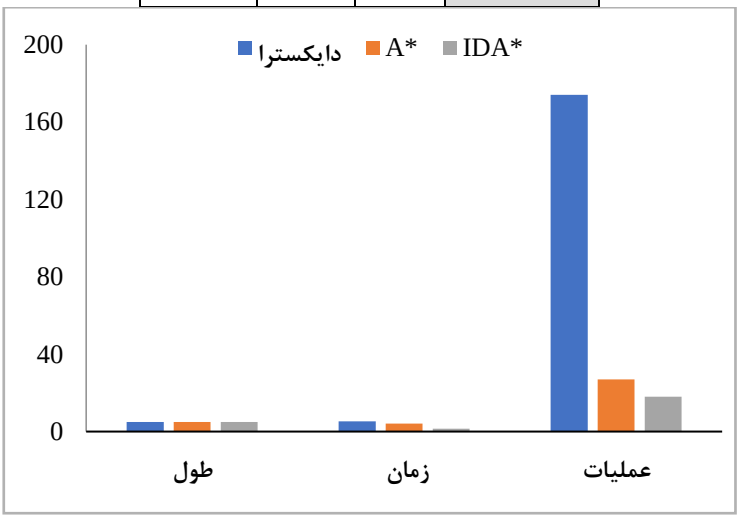
تجزیه و تحلیل نتایج بدون وجود مانع:

در زمین بازی بدون وجود مانع، نزدیک‌ترین فاصله بین نقطه شروع و نقطه هدف پنج خانه شبکه می‌باشد. جدول (۱) نتایج بدست آمده برای تمامی الگوریتم‌های بررسی شده را نشان می‌دهد. همانطور که ملاحظه می‌شود، تعداد عملیات محاسباتی در الگوریتم

دایکسترا نسبت به دیگر الگوریتم‌ها بسیار بالا می‌باشد که این عامل باعث کاهش سرعت محاسبات و افزایش حافظه کامپیوتر می‌شود. از سوی دیگر، سرعت این الگوریتم نسبت به دیگر الگوریتم‌ها بسیار کند می‌باشد. با استفاده از الگوریتم A* بهبود قابل توجهی در تعداد عملیات انجام شده نسبت به الگوریتم دایکسترا نشان می‌دهد که مقدار آن بیش از ۸۴٪ کاهش می‌یابد. همچنین زمان محاسبات نیز حدود ۲۴٪ کاهش یافته است. در نهایت، الگوریتم IDA* با کاهش ۹۰ و ۳۳ درصدی در تعداد عملیات نسبت به الگوریتم دایکسترا و A* بهینه‌ترین مدل مسیریابی ارزیابی شده در زمین بازی بدون وجود مانع را نمایش می‌دهد. همچنین، لازم به ذکر است که زمان انجام محاسبات در الگوریتم IDA* حدود یک سوم روش A* می‌باشد که به موجب آن هزینه محاسباتی به شدت کاهش یافته است.

جدول (۱) تحلیل مقایسه‌ای در زمین بازی بدون وجود مانع

الگوریتم	طول	زمان	عملیات
دایکسترا	۵	۵.۳۵	۱۷۴
A*	۵	۴.۰۸	۲۷
IDA*	۵	۱.۴۴	۱۸



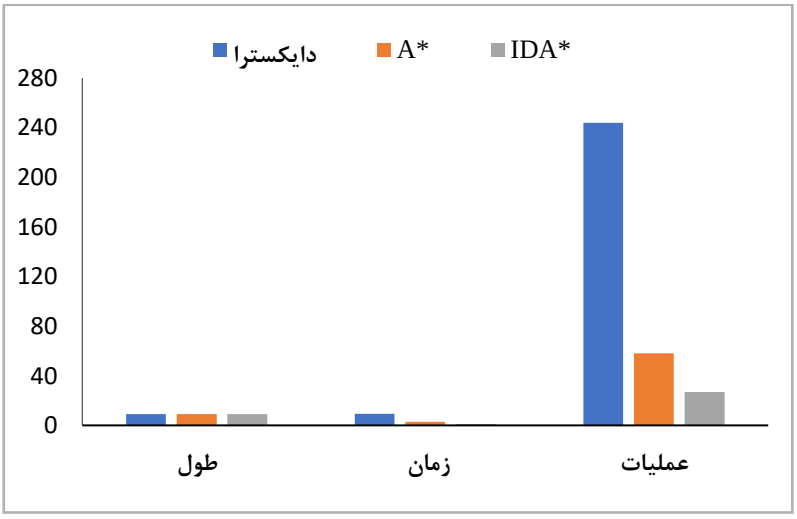
شکل(۱۲) نمودار تحلیل مقایسه‌ای در زمین بازی بدون وجود مانع

تجزیه و تحلیل نتایج با وجود مانع

در نتایج بدست آمده در بخش زمین با مانع نزدیکترین فاصله بین نقطه شروه و نقطه هدف ۹ خانه شبکه می‌باشد. با توجه به نتایج آورده شده در جدول (۲) و نمودارهای آورده شده در شکل (۱۳)، در این مسئله نیز الگوریتم IDA* از دو روش دیگر و الگوریتم A* نسبت به روش دایکسترا نتایج بهینه‌تری را نشان می‌دهد. به طور دقیق‌تر، زمان انجام محاسبات در روش دایکسترا حدود ۳ برابر روش A* و بیش از ۹ برابر الگوریتم IDA* می‌باشد. از طرف دیگر، تعداد عملیات انجام شده در الگوریتم IDA* به طور قابل ملاحظه‌ای نسبت به دو روش دیگر کاهش یافته است.

جدول (۲) تحلیل مقایسه‌ای در زمین بازی با وجود مانع

الگوریتم	طول	زمان	عملیات
دایکسترا	۹	۹.۳۱	۲۴۴
A*	۹	۲.۸۷	۵۸
IDA*	۹	۱.۰۳	۲۷



شکل(۱۳) نمودار تحلیل مقایسه‌ای در زمین بازی با وجود مانع

نتیجه‌گیری و پیشنهادها

بررسی نتایج نشان می‌دهد که از نظر پیچیدگی زمانی، الگوریتم دایکسترا و الگوریتم*A دارای پیچیدگی زمانی تقریباً یکسانی هستند. با این حال، به‌طور معمول زمان اجرای الگوریتم *A کمتر از الگوریتم دایکسترا است که این خروجی به دلیل استفاده الگوریتم *A از یک تابع اکتشافی برای هدایت جستجوی خود به سمت گره هدف حاصل شده است. اگر تابع اکتشافی در الگوریتم *A به خوبی طراحی شده و تخمین‌های دقیقی از فاصله تا هدف ارائه دهد، می‌تواند با کاهش (هرس) تعداد گره‌هایی که در طول فرآیند جستجو نیاز به گسترش دارند به طور قابل توجهی از الگوریتم دایکسترا از نظر زمان اجرا بهتر عمل کند.

همچنین در مسأله بررسی شده، الگوریتم *IDA از نظر زمان اجرا دارای سرعت بالاتر و نیاز به حافظه کمتر نسبت به الگوریتم *A می‌باشد. باید به این نکته توجه داشت که الگوریتم *IDA یک جستجوی عمقی را که گاهی به کاوش گره‌های اضافی نیاز دارد انجام می‌دهد و در صورت تعریف یک تابع اکتشافی بهینه برای *A ممکن است دیرتر از آن به جواب برسد. ولی در هر صورت چون الگوریتم *A برخلاف *IDA که مسیر در حال بررسی را پیگیری می‌کند باید لیست باز را پیگیری کند به حافظه بیشتری نسبت به الگوریتم*IDA نیاز دارد.

به طور خلاصه، با توجه به نتایج آزمایش روی مسأله طراحی شده، الگوریتم*IDA در هر دو حالت (یعنی با مانع و بدون مانع) بهتر از الگوریتم دایکسترا و*A عمل می‌کند.

قدر دانی

از جناب آقای جادی میرمیرانی که برنامه‌نویسی به زبان پایتون را از طریق کلاس‌های آموزشی ایشان فراگرفته‌ام کمال تشکر و قدردانی را دارم.

منابع

- Bhadoria, A., & Singh, R. K. (2014). Optimized angular a star algorithm for global path search based on neighbor node evaluation. *International Journal of Intelligent Systems and Applications*, *6*(8), 46.

- Candra, A., Budiman, M. A., & Pohan, R. I. (2021). Application of A-Star Algorithm on Pathfinding Game. *Journal of Physics: Conference Series*, *1898*(1), 012047. https://doi.org/10.1088/1742-6596/1898/1/012047

- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/BF01386390/METRICS

- Duchon, F., Babinec, A., Kajan, M., Beno, P., Florek, M., Fico, T., & Jurišica, L. (2014). Path planning with modified A star algorithm for a mobile robot. *Procedia Engineering*, *96*, 59–69. https://doi.org/10.1016/J.PROENG.2014.12.098

- Fredman, M. L., & Tarjan, R. E. (1987). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, *34*(3), 596–615. https://doi.org/10.1145/28869.28874

- Gibert, F., & Nofrarias, M. (2021). *Application of A-Star Algorithm on Pathfinding Game Application of A-Star Algorithm on Pathfinding Game*. https://doi.org/10.1088/1742-6596/1898/1/012047

- Goyal, A., Mogha, P., Luthra, R., & Sangwan, N. (2014). Path finding: A* or Dijkstra's? *International Journal in IT & Engineering*, *2*(1), 1–15.

- Gross, J. L., & Yellen, J. (2005). *Graph theory and its applications*. CRC press.

- Guruji, A. K., Agarwal, H., & Parsediya, D. K. (2016). Time-efficient A* Algorithm for Robot Path Planning. *Procedia Technology*, *23*, 144–149. https://doi.org/10.1016/J.PROTCY.2016.03.010

- Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE*

Transactions on Systems Science and Cybernetics, *4*(2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

- Hassan, F., Sunar, N., Mohd Basri, M. A., Mahmud, M. S. A., Ishak, M. H. I., & Mohamed Ali, M. S. (Eds.). (2023). *Methods and Applications for Modeling and Simulation of Complex Systems*. *1912*. https://doi.org/10.1007/978-981-99-7243-2
- Jindal, P., & Kumar, A. (2010). Analysis of shortest path algorithms. *Global Journal of Computer Science and Technology*, *10*(8).
- Kapi, A. (2020). A Review on Informed Search Algorithms for Video Games Pathfinding. *International Journal of Advanced Trends in Computer Science and Engineering*, *9*, 2756–2764. https://doi.org/10.30534/ijatcse/2020/42932020
- Keshuai, Z., & Pingqing, F. (2021). Improved A~* algorithm and artificial potential field algorithm for mobile robot path planning [J]. *Electronic Devices*, *44*(02), 368–374.
- Lanning, D. R., Harrell, G. K., & Wang, J. (2014). Dijkstra's Algorithm and Google Maps. *Proceedings of the 2014 ACM Southeast Regional Conference, ACM SE 2014*. https://doi.org/10.1145/2638404.2638494
- Li, Y., Harms, J., & Holte, R. (2007). No Title. *IEEE International Conference on Communications*, 123–130. https://doi.org/10.1109/ICC.2007.29
- Meng, Q., & Zhang, J. (2019). Optimization and application of artificial intelligence routing algorithm. *Cluster Computing*, *22*(4), 8747–8755. https://doi.org/10.1007/s10586-018-1963-z
- Pemmaraju, S., & Skiena, S. (2003). *Computational discrete mathematics: Combinatorics and graph theory with mathematica®*. Cambridge university press.
- Peng, J., Huang, Y., & Luo, G. (2015a). Robot path planning based on improved A* algorithm. *Cybernetics and Information Technologies*, *15*(2), 171–180.

- Peng, J., Huang, Y., & Luo, G. (2015b). Robot path planning based on improved A* algorithm. *Cybernetics and Information Technologies*, *15*(2), 171–180.
- Phaneendhar Reddy Vanam. (2011). *Shortest path using A∗ Algorithm*. http://cs.indstate.edu/~pvanam/phani.pdf
- Qiong, W., Meiwan, L., Weijian, R., & Tianren, W. (2019). Overview of common algorithms for UAV path planning. *Journal of Jilin University (Information Science Edition)*, *37*(01), 58–67.
- Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence A Modern Approach* (M. Pompili, Ed.). Alan Apt.
- Sharma, Cs. K. (2015). *Shortest path searching for road network using a* algorithm*.
- Sinthamrongruk, T., Mahakitpaisarn, K., & Manopiniwes, W. (2013). A Performance Comparison between A* Pathfinding and Waypoint Navigator Algorithm on Android and iOS Operating System. *International Journal of Engineering and Technology*, *5*(4), 498.
- Skiena, S. (1991). *Implementing discrete mathematics: combinatorics and graph theory with Mathematica*. Addison-Wesley Longman Publishing Co., Inc.
- Song, Y., & Ma, P. (2021). Research on mobile robot path planning based on improved A-star algorithm. *2021 International Conference on Electronic Information Engineering and Computer Science (EIECS)*, 683–687.
- Tang, G., Tang, C., Claramunt, C., Hu, X., & Zhou, P. (2021). Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment. *IEEE Access*, *9*, 59196–59210. https://doi.org/10.1109/ACCESS.2021.3070054
- Vamja, H., Shaikh, A., & Rami, S. (2017). Comparative Analysis of Different Path Finding Algorithms to Study Limitations and Progress. *The Fourteenth International Conference on Networks*, *7*(9), 68–75.
- Wang, H., Qi, X., Lou, S., Jing, J., He, H., & Liu, W. (2021). An Efficient and Robust Improved A* Algorithm for Path Planning.

Symmetry 2021, Vol. 13, Page 2213, *13*(11), 2213. https://doi.org/10.3390/SYM13112213

- Wang, H., Yin, P., Zheng, W., Wang, H., & Zuo, J. (2020). Mobile robot path planning based on improved A* algorithm and dynamic window method. *Robot*, *42*(03), 346–353.
- Wang, Q., Hao, Y., & Chen, F. (2021). Deepening the IDA* algorithm for knowledge graph reasoning through neural network architecture. *Neurocomputing*, *429*, 101–109. https://doi.org/https://doi.org/10.1016/j.neucom.2020.12.040
- Whiting, P. D., & Hillier, J. A. (1960). A method for finding the shortest route through a road network. *Journal of the Operational Research Society*, *11*, 37–40.
- Yu, X., Huahua, L., Mingbo, D. U., Tao, M., & Zhiling, W. (2014a). An improved A* algorithm that can search infinite neighborhoods [J]. *Robot*, *36*(05), 627–633.
- Yu, X., Huahua, L., Mingbo, D. U., Tao, M., & Zhiling, W. (2014b). An improved A* algorithm that can search infinite neighborhoods [J]. *Robot*, *36*(05), 627–633.
- Zhang, C., Ao, L., Yang, J., & Xie, W. (2020). An Improved A* Algorithm Applying to Path Planning of Games. *Journal of Physics: Conference Series*, *1631*(1), 012068. https://doi.org/10.1088/1742-6596/1631/1/012068
- Zikky, Moh. (2016). Review of A* (A Star) Navigation Mesh Pathfinding as the Alternative of Artificial Intelligent for Ghosts Agent on the Pacman Game. *EMITTER International Journal of Engineering Technology*, *4*(1), 141–149. https://doi.org/10.24003/EMITTER.V4I1.117