ISSN: 2476-6909; Modares Mechanical Engineering. 2023;23(03):161-172

A New Incremental Search Method for Multi-robot Path Planning

ARTICLE INFO

Article Type
Original Research

Authors
Esmaeel Khanmirza.[1*],
Morteza Haghbeigi.[1],
Mohammad Farzan.[1]

How to cite this article
Beharvan A. H., Seyedkashi S. M. H., Sheikhi Azqandi, M, Optimum Design and Construction of Cylindrical Energy Absorber under Internal Pressure using Time Evolutionary Optimization Algorithm. Modares Mechanical Engineering. 2023; 23 (03):161-172.

[1] Iran University of Science and Technology

Correspondence
Address: University of Science and Technology of Iran, University St., Hengam St Tehran, Iran
khanmirza@iust.ac.ir

Article History
Received: July 27, 2022
Accepted: November 27, 2022
ePublished: March 15, 2023

ABSTRACT

Multi-robot path planning problem involves some challenges. One of them is the exponential increase in the size of the search space as a result of increasing the number of robots in the operating environment. Therefore, there is a need for algorithms with high computational performance to plan optimal and collision-free paths in a limited time. In this article, a centralized path planning algorithm is presented. The algorithm is a heuristic incremental search, in which the D* Lite algorithm has been adapted for the multi-robot case. The concept of occupancy time has been embedded into the environment model to avoid path interference. A centralized function has been designed to update the environment model and robot data. To evaluate the method, two groups of simulations of static and dynamic types were carried out. The static simulations focused on studying the effect of algorithm parameters, and it was shown that the algorithm can plan paths for up to 40 robots in an environment having 55 percent free space. The dynamic simulations were carried out in Gazebo, a real-time and dynamic physical simulator. The results were compared to a baseline method based on potential fields. The number of robots was increased to 14, and it was demonstrated that for 9 robots and more, the potential field approach either fails or has a rapid increase in computation time, while the proposed method can find feasible solutions in a limited time.

Keywords Path Planning, Multi-Robot Systems, Incremental Search, Autonomous Mobile Robots

CITATION LINKS

1- A comparative study of deterministic and probabilistic mobile robot path planning algorithms 2- Path planning and trajectory planning algorithms 3- A potential field approach to path planning 4- Numerical potential field techniques for robot path planning 5- based path planning-using the voronoi diagram for a clearance-based shortest path 6- A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices .7- Engineering route planning algorithms 8- Incremental A* 9- D*lite 10- Fast replanning for navigation in unknown terrain 11- Planning algorithms 12- Cloud-based multi-robot path planning in complex and ...13- Subdimensional expansion for multirobot path planning. Artificial intelligence 14- Dynamic prioritization for conflict-free path planning of .... 15- Multi-robot path planning using an improved self-adaptive particle swarm optimization 16- Contribution to the path planning of a multi-robot system 17- Finding optimal solutions to cooperative pathfinding problems 18- Partial-expansion A* with selective node generation 19- Planning optimal paths for multiple robots on graphs. 20- A complete multirobot path planning algorithm with performance bounds 21- optimal multirobot path planning in low dimensional search spaces 22- A. The increasing cost tree search for optimal multi-agent pathfinding. Artificial intelligence 23- A, Sturtevant NR. Conflict-based search for optimal multi-agent pathfinding. 24- Efficient and complete centralized multi-robot path planning 25- Push and rotate: cooperative multi-agent path planning 26- Cooperative multi-robot path planning by heuristic priority adjustment 27- Anytime graph-based path planning with deep reinforcement learning for an autonomous UAV 28- A polynomial-time algorithm for non-optimal multi-agent pathfinding .29- Distributed path planning of a multi-robot system based on the neighborhood artificial potential field approach

Copyright© 2020, TMU Press. This open-access article is published under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License which permits Share (copy and redistribute the material in any medium or format) and Adapt (remix, transform, and build upon the material) under the Attribution-NonCommercial terms.

روش جستجوی تدریجی جدید برای طراحی مسیر سیستم‌های چندرباتی

اسماعیل خان میرزا[1]*، مرتضی حق بیگی[1]، محمد فرزان[1]
۱ دانشگاه علم و صنعت ایران

چکیده

یکی از این چالش‌های مساله طراحی مسیر چندرباتی، افزایش ابعاد فضای جستجو به صورت نمایی همراه با افزایش تعداد ربات‌ها در محیط عملیات است. بنابراین، به الگوریتم‌هایی نیاز است که دارای کارایی محاسباتی بوده و بتوانند مسیرهای بهینه و بدون برخورد ربات‌ها را در زمان محدود طراحی کنند. در این مقاله یک الگوریتم طراحی مسیر مرکزی برای هدایت ربات‌ها در محیط عملیات مشترک ارائه شده است. این الگوریتم یک روش جستجوی اکتشافی تدریجی است که در آن الگوریتم D* Lite به منظور تطبیق با حالت چندرباتی توسعه داده شده است. هماهنگی در طراحی مسیر برای تمام ربات‌ها بر اساس مفهوم زمان تصرف بوده که در ساختار مدل مفهومی محیط پیاده‌سازی شده است. همچنین، یک تابع مرکزی جهت به‌روزرسانی اطلاعات مدل مفهومی محیط و حرکت تدریجی ربات‌ها توسعه داده شده است. به منظور ارزیابی روش پیشنهادی، دو گروه شبیه‌سازی‌های استاتیک و پویا انجام شده است. در دسته اول، تمرکز بر مطالعه اثر عوامل الگوریتم است. نتایج نشان می‌دهد که الگوریتم پیشنهادی قابلیت طراحی مسیر برای ۴۰ ربات در محیطی با ۵۵ درصد فضای آزاد را دارد و نیز رابطه زمان محاسباتی و تعداد ربات‌ها غیر نمایی است. دسته دوم شبیه‌سازی‌ها در محیط سه‌بعدی Gazebo انجام شده که به صورت برخط و پویا است. نتایج روش پیشنهادی با روشی بر اساس میدان‌های پتانسیل مصنوعی برای تعداد ۱۴ ربات مورد مقایسه قرار گرفته است. نتایج نشان می‌دهد که با افزایش تعداد ربات‌ها از ۹ عدد، زمان انجام عملیات برای روش مبتنی بر میدان پتانسیل افزایش زیادی پیدا کرده و یا غیرممکن می‌شود.

کلیدواژه‌ها: طراحی مسیر، سیستم‌های چندرباتی، جستجوی تدریجی، ربات خودکار

تاریخ دریافت: ۱۴۰۱/۰۵/۰۵
تاریخ پذیرش: ۱۴۰۱/۰۹/۰۶
*نویسنده مسئول: khanmirza@iust.ac.ir

۱- مقدمه

یکی از چالش‌های فنی مهم در طراحی ربات‌های متحرک، مساله طراحی مسیر می‌باشد. این مساله به صورت یافتن مسیر بدون برخورد با موانع برای حرکت ربات از پیکربندی اولیه به پیکربندی هدف تعریف می‌شود[1]. این الگوریتم‌ها بر اساس پیشنهاد گاسپارتو و همکاران، به سه دسته تقسیم می‌شوند: میدان‌های پتانسیل مصنوعی، نقشه راه (Roadmap) و روش‌های مبتنی بر جست و جوی شبکه (Grid)[2]. در روش‌های مبتنی بر میدان پتانسیل که در سال ۱۹۹۲ در دو مقاله مستقل توسط هوانگ و آهونجا[3]، و باراکواند و همکاران[4] پیشنهاد شد، در محل موانع نیروی دافعه و در محل هدف نیروی جاذبه در نظر گرفته می‌شود. سپس با شبیه‌سازی حرکت ربات تحت تاثیر این نیروهای فرضی، مسیر مطلوب تولید می‌شود. در روش‌های مبتنی بر نقشه راه، ابتدا به طور تصادفی تعدادی نقطه در نقشه انتخاب می‌شود. سپس با ارزیابی مسیرهای ممکن بین این نقاط و انتخاب بهترین آن‌ها، مسیر مورد نظر به دست می‌آید. در برخی نسخه‌های این روش، مراحلی نیز جهت افزایش دقت اضافه شده است که کیفیت مسیر خروجی را بهبود می‌دهد. این روش در سال ۲۰۰۸ توسط باتاچاریا و گاوریلوا معرفی شد[5]. در روش‌های مبتنی بر شبکه، ابتدا فضا به منطقه‌های کوچک منتظم تقسیم می‌شود. این منطقه‌ها در یک گراف نمایش داده می‌شوند که هر گره با یک منطقه متناظر است. بدین ترتیب، مساله یافتن بهترین مسیر به کمک روش‌های رایج در نظریه گراف قابل حل خواهد بود.

ساده‌ترین روش برای یافتن مسیر بهینه بین دو گره در گراف، الگوریتم دایجسترا (Dijkstra's Algorithm) است که در سال ۱۹۵۶ توسط دایجسترا پیشنهاد شد[6]. نسخه‌های متنوعی از الگوریتم دایجسترا پیشنهاد شده است. الگوریتم A* که در سال ۱۹۶۸ توسط هارت و همکاران ارائه شد، با مدل نمودن فاصله یک گره تا مقصد با یک تابع ابتکاری (Heuristic)، سرعت یافتن پاسخ روش دایجسترا را افزایش می‌دهد[7]. محدودیت روش دایجسترا و همچنین A* این است که در صورت بروز تغییر در گراف، مسیر باید از نو طراحی شود. در کاربردهای واقعی، موقعیت موانع همواره در معرض تغییر است و یا گاهی در آغاز حرکت ربات از وجود بعضی موانع آگاهی ندارد. همچنین، حرکت ربات در گراف خود به معنای تغییر نقطه شروع مساله است که در روش‌های دایجسترا و A* مستلزم از سرگیری کامل محاسبات است. برای رفع این محدودیت، الگوریتم‌های جستجوی تدریجی (Incremental Search) پیشنهاد شده‌اند.

الگوریتم LPA* یک روش جستجوی تدریجی بر پایه روش A* می‌باشد که در سال ۲۰۰۱ توسط کوئینگ و لیخاچوف مطرح شد. در این روش هنگام بروز تغییرات در گراف، مسیر جدیدی با اصلاح پاسخ قبلی تولید شده و صف هزینه گره‌های بررسی شده پس از پیدا شدن پاسخ نیز در حافظه نگه‌داری می‌شود. هنگامی که در گراف تغییری ایجاد شود، تنها گره‌هایی که مسیر آن‌ها از این تغییر متاثر شده به روز می‌شوند. بدین ترتیب، نیازی به بررسی مکرر تمام گره‌ها نیست و بار محاسباتی بسیار کمتر خواهد بود[8]. با این حال، این الگوریتم برای ربات متحرک مزیتی ایجاد نمی‌کند زیرا در طراحی آن فرض شده که نقطه شروع و پایان ثابت هستند. برای رفع مشکل جابجایی نقطه شروع، روش D* Lite در سال ۲۰۰۲ توسط کوئینگ و لیخاچوف ارائه شد[9]. این روش بر پایه LDA* است و در آن محاسبات از نقطه مقصد آغاز شود. در نتیجه با حرکت ربات در گراف، مقادیر فاصله شروع گره‌ها (g-values) تغییر نکرده و نتیجه محاسبات قبلی قابل استفاده است. همچنین در این روش با نمایش وجود یک حد بالا برای تغییرات مقادیر فاصله پایان (h-values)، مرتب‌سازی پشته نقاط بهینه شده است[10].

مساله طراحی مسیر چندرباتی نسبت به حالت طراحی برای ربات‌های منفرد با چالش‌های بیشتری مواجه است. یکی از این

چالش‌ها افزایش ابعاد فضای جستجو به صورت نمایی با افزایش تعداد ربات‌ها است[11]. همچنین، زمان واکنش یا طراحی محدودیت شدیدتری دارد. زیرا با وجود تعداد زیادی ربات، پویایی محیط افزایش زیادی پیدا می‌کند[12]. الگوریتم‌های طراحی مسیر چندرباتی به دو دسته کلی قابل تقسیم هستند: الگوریتم‌های مبتنی بر حل کوپل شده و الگوریتم‌های غیر کوپل شده (یا با کوپلینگ ضعیف)[13,16] . در روش‌های کوپل شده، مساله طراحی مسیر ربات‌ها در قالب یک مساله واحد در یک پردازشگر مرکزی حل می‌شود. این روش‌ها دارای مشخصه کامل بودن و بهینگی پاسخ هستند، اما پیچیدگی محاسباتی بالایی داشته و قابل استفاده برای تعداد بالایی از ربات‌ها نیستند. جستجوی مسیر بهینه سراسری در فضای جستجوی مشترک ربات‌ها صورت می‌گیرد که به صورت نمایی با افزایش ربات‌ها افزایش پیدا می‌کند. یک روش ابتدایی برای حل این مساله، استفاده از الگوریتم A* برای یافتن مسیرهای بهینه است که کارایی محاسباتی پایینی دارد. برای ارتقای عملکرد این روش، در سال ۲۰۱۰ تکنیک تجزیه عملگر (Operator Decomposition) توسط استندلی معرفی شد. در این روش، هنگامی که دو ربات نزدیک هم هستند، زوج مرتب حرکات ممکن آن‌ها، به طور هم‌زمان مورد بررسی قرار می‌گیرد تا از برخورد جلوگیری شود. این روش همواره خروجی بهینه به دست می‌دهد[17]. روش دیگری که در سال ۲۰۱۲ توسط فلنر معرفی شد بر این اساس است که در روش جستجوی A* برای چند ربات، از تولید حالت‌هایی که هزینه محلی آن‌ها بالاتر از هزینه بهترین مسیر پیدا شده است صرف نظر شود. این روش در مسائلی که تعداد حرکات ممکن در هر گام بزرگ است، کارکرد محاسباتی را بهبود می‌دهد[18]. روش دیگر در این دسته، تبدیل مساله طراحی مسیر در گراف به مساله برنامه‌ریزی خطی (Linear Programming) است که در کاربردهایی با تعداد پایین ربات قابل استفاده است[19].

محققان دسته دیگری از روش‌ها با کوپلینگ متغیر را نیز پیشنهاد داده‌اند. در این روش‌ها، طراحی مسیر در فضای جستجوی انفرادی ربات‌ها انجام شده و سپس در صورت وجود برخورد، مسیر ربات‌های برخورد کننده در فضای جستجوی مشترک بازطراحی می‌شود. برای مثال، در روش M* که در سال ۲۰۱۱ توسط واگنر و چوست ابداع شد، ابتدا مسیریابی تمام ربات‌ها به صورت مجزا انجام می‌شود. سپس ربات‌هایی که مسیر آن‌ها تلاقی دارد مشخص شده و برای هر دسته ربات مساله مجددا به صورت کوپل شده حل می‌شود[20]. در سال ۲۰۱۳، روش M* توسط فرنر و همکاران بدین گونه توسعه داده شد که فضای جستجو فقط در اطراف نقاط برخورد افزایش پیدا کند و بدین ترتیب راندمان پردازشی بهتری به دست آمد[21].

در سال ۲۰۱۳، روش جستجوی درخت هزینه افزایشی (Increasing Cost Tree Search) توسط شارون و همکاران پیشنهاد شد. در این روش، ابتدا هزینه مسیرهای بهینه برای هر ربات با فرض وجود نداشتن ربات‌های دیگر محاسبه می‌شود. این هزینه‌ها در یک بردار n مولفه‌ای قرار داده شده و به عنوان ریشه یک درخت جستجو در نظر گرفته می‌شوند. بدیهی است که این مسیرها ممکن است تلاقی کنند و پاسخ مساله نباشند. بنابراین، یک جستجوی سطح-اول روی این درخت انجام می‌شود؛ فرزندان هر گره با افزایش وزن یکی از مولفه‌های بردار تولید می‌شوند. برای بررسی مناسب بودن هر گره، تمام ترکیبات مسیرهای ممکن در فضای پیکربندی با قید هزینه کل برابر با وزن‌های موجود در گره بررسی می‌شوند تا مشخص شود که آیا ترکیبی وجود دارد که در آن ربات‌ها برخورد نداشته باشند یا خیر. در صورت وجود چنین پاسخی، جستجو به پایان می‌رسد و در غیر این صورت، قیدها دوباره افزایش می‌یابند.[22] در سال ۲۰۱۵، روش جستجوی مبتنی بر تعارض (Conflict-Based Search) نیز توسط شارون و همکاران پیشنهاد شد که در آن، جستجوی سطح-اول به جای درخت هزینه افزایشی روی درخت تعارض (Conflict Tree) انجام می‌شود. در این درخت، هر گره نماینده مجموعه‌ای از قیود است. گره ریشه متناظر با حالتی است که هیچ قیدی وجود نداشته باشد. سپس برای انجام جستجو، مسیرها به طور مجازی اجرا شده و اولین برخورد آن‌ها پیدا می‌شود و برای آن دو گره جدید به وجود می‌آید که در هر یک از آن‌ها یک قید اضافه شده است تا یکی از دو ربات متلاقی، در زمان مربوط به برخورد از گره برخورد استفاده نکند. بدین ترتیب، مسیرها با یک قید جدید بازطراحی می‌شوند و جستجو تا جایی تکرار می‌شود که هیچ دو مسیری برخورد نداشته باشند[23].

روش‌های مبتنی بر قانون از یکسری قواعد از پیش تعیین شده جهت حرکت ربات‌ها استفاده می‌کنند. روش Push and Swap که در سال ۲۰۱۱ توسط لونا و بکریس ارائه شد، در این دسته قرار دارد. در این روش، ربات‌ها روی مسیرهای بهینه خود با فرض وجود نداشتن ربات‌های دیگر به طور مجازی حرکت داده می‌شوند و در هر گام اگر مسیرها با هم تداخل کنند، بر اساس اولویت ربات‌ها یکی از ربات‌ها به کنار زده می‌شود (Push) یا ربات‌ها از کنار هم دور زده و موقعیت آن‌ها با یکدیگر تعویض می‌شود (Swap)[24] .در سال ۲۰۱۳ توسط ویلده و همکاران نشان داده شد که در روش Push and Swap چند موقعیت خاص در نظر گرفته نشده است که این روش در موقعیت‌های مذکور شکست می‌خورد. برای مثال، در حالت ساده‌ای که دو ربات در یک راهروی باریک قرار داشته باشند، یا به بیان دیگر گره‌ها دو اتصالی باشند، امکان تعویض موقعیت دو ربات وجود ندارد. با ارتقای این مشکلات، روش Push and Rotate توسط ویلده و همکاران ارائه شد که در آن به مشکلات این چنینی در قالب زیرمساله‌های از پیش تعریف شده پرداخته شده است[25].

در روش‌های غیرکوپل، ابتدا مساله به طور مستقل برای هر ربات حل می‌شود. سپس مسیرهای طراحی شده در صورت وجود برخورد،

اصلاح می‌شوند. این دسته از روش‌ها کامل بودن و یا بهینگی مسیرها را تضمین نمی‌کنند، اما در کاربردهایی با تعداد بالایی ربات کارایی محاسباتی خوبی دارند. در روش‌های مبتنی بر اولویت، ربات با اولویت بالاتر به عنوان مانع برای ربات‌هایی با اولویت پایین‌تر در نظر گرفته می‌شود. یک نمونه از این روش‌ها در سال ۲۰۰۶ توسط رگل و لیوی توصیف شده است که در آن هنگام برخورد دو ربات، یکی از آن‌ها بر اساس اولویت‌بندی تعیین شده توقف می‌کند تا ربات دیگر عبور کند[26]. روش دیگری نیز در سال ۲۰۲۱ توسط ماو و همکاران پیشنهاد شده است که با سپردن مسیریابی محلی به یک الگوریتم یادگیری تقویتی عمیق، از برخورد جلوگیری کرده است[27].

در برخی الگوریتم‌هایی که در آن‌ها محیط به صورت گراف مدل می‌شود، برای طراحی مسیر از روش‌های ابتکاری استفاده شده است. یک نمونه از این الگوریتم‌ها در سال ۲۰۱۱ توسط خورشید و همکاران ارائه شده است که در آن از روشی ابتکاری به نام تبادل موقعیت بر پایه درخت (Tree-based Swapping) استفاده شده است که از طریق یک اثبات سازنده (Constructive Proof) نشان داده شده است که همیشه یک جواب مناسب را پیدا می‌کند. با این حال، این روش لزوما کوتاه‌ترین مسیرهای ممکن را به دست نمی‌دهد[28].

دسته دیگر، روش‌های محلی با قابلیت پیاده‌سازی سریع و برخط هستند. در سال ۲۰۱۹ توسط ماتوی و همکاران روش مسیریابی بر پایه میدان پتانسیل مصنوعی پیشنهاد شد و بین ربات‌ها نیروی دافعه تعریف شد تا از برخورد جلوگیری شود[16]. در همان سال، نسخه توزیع شده این روش توسط ماتوی و همکاران منتشر شد که در آن محاسبات هر ربات مستقل و بدون نیاز به سرور مرکزی انجام می‌شود[29].

در اغلب روش‌های بررسی شده، طراحی مسیر به صورت سراسری در ابتدای عملیات انجام شده و در صورت تغییر در محیط عملیات و یا تعداد ربات‌ها، طراحی باید از نو صورت بگیرد. این مساله یک چالش جدی در کاربردهای رباتیکی است. در این پژوهش، یک الگوریتم مرکزی برای طراحی مسیر چندرباتی در محیط پویا پیشنهاد شده است. روش پیشنهادی شامل دو بخش است. بخش اول، الگوریتم مرکزی است که مدیریت عملیات و به‌روزرسانی اطلاعات گراف و عوامل الگوریتم را بر عهده دارد. بخش دیگر، تابع درونی است که بر پایه D* Lite توسعه داده شده و طراحی مسیر هر ربات را بر عهده دارد. در این تابع با استفاده از مفهوم زمان تصرف (Occupancy Time)، اطلاعات زمانی مسیر در گراف ذخیره می‌شود.

نوآوری روش پیشنهادی در استفاده از یک نقشه مفهومی مشترک و توسعه یک روش جستجوی ابتکاری تدریجی در یافتن مسیر چندین ربات به صورت برخط در محیطی پویا است. این روش، دارای کارامدی محاسباتی روش‌های محلی و غیر کوپل شده است. در نتیجه، همان‌طور که در شبیه‌سازی‌ها نشان داده شده ، قابل پیاده‌سازی به صورت برخط است. همچنین، برخلاف روش‌های کوپل شده، قابل اجرا در تمام مراحل عملیات بوده و در صورت ایجاد تغییرات در محیط، افزایش ربات‌های دیگر به عملیات، و یا تغییر موقعیت هدف ربات‌ها، مسیرهای مورد نیاز را به صورت برخط به‌روزرسانی می‌کند. همچنین، همان‌طور که در شبیه‌سازی‌های فیزیکی و پویا نشان داده شده است، هماهنگی بیشتری در طراحی مسیر و مدیریت ترافیک نسبت به روش‌های محلی با پیچیدگی محاسباتی کمتر نظیر روش‌های مبتنی بر میدان پتانسیل دارد. این موارد مزیت استفاده از روش پیشنهادی در کاربردهای رو به رشد سیستم‌های چندرباتی در صنعت را نشان می‌دهد.

۲- الگوریتم پیشنهادی

ایده مفهومی الگوریتم طراحی مسیر چندرباتی پیشنهادی مبتنی بر طراحی مسیر در یک سرور مرکزی است. با استفاده از یک پروتکل دلخواه شبکه، مسیر طراحی شده از این سرور به ربات‌ها ارسال شده و بدین ترتیب هماهنگی مسیر ربات‌ها امکان‌پذیر می‌گردد. وضعیت ربات‌ها و موانع جدید به طور پیوسته توسط سرور پایش شده و در صورت نیاز به اصلاح هر یک از مسیرها، نتیجه به ربات‌ها ارسال می‌شود. در مقاله پیش رو برای این معماری الگوریتمی پیشنهاد شده است که در سرور اجرا شده و وظایف ذکر شده را انجام می‌دهد.

الگوریتم در دو سطح طراحی شده است. سطح بالاتر که حلقه بیرونی را شامل می‌شود، وظیفه اولویت‌بندی ربات‌ها، به‌روزرسانی نقشه مشترک و رفع برخوردها را دارد. در سطح پایین‌تر، الگوریتم مسیریابی ربات‌های منفرد قرار دارد که توسط حلقه بیرونی فراخوانی می‌شود. حلقه درونی در حقیقت نسخه توسعه یافته‌ای از روش D* Lite است که در ادامه شرح داده خواهد شد. همچنین، الگوریتم پیشنهادی در قالب دو نسخه متفاوت ارائه شده است که یکی روی کاهش طول مسیر و نرمی آن، و دیگری روی کاهش زمان عملیاتی تمرکز دارد.

۲-۱ روش D* Lite توسعه یافته

در این قسمت نسخه توسعه داده شده روش D* Lite شرح داده شده است. روش D* Lite یک الگوریتم مسیریابی در گراف است که کوتاه‌ترین مسیر بین دو راس گراف را به دست می‌دهد.

در این مقاله، الگوریتم D*Lite به منظور هماهنگی در طراحی مسیر همزمان برای مجموعه‌ای از ربات‌های خودکار تطبیق داده شده است. ابتدا محیط عملیات در قالب گراف وزن‌دار $Gr(V, E)$ مدل می‌شود. این گراف از نوع ۸-اتصالی (8-connected) است. برای هر گره در صف یک مشخصه تعریف می‌شود که هزینه رسیدن به آن را نگهداری می‌کند. این عدد برابر جمع وزن یال‌های مسیر بوده و g-value نام دارد. همچنین، برای هر راس گراف اصلی یک هزینه

ابتکاری موسوم به h-value تعریف می‌شود. این مشخصه، تقریبی از فاصله گره تا مقصد است. شرط کافی برای این که تابع ابتکاری بهینگی پاسخ را از بین نبرد به صورت زیر است:

$$\begin{cases} \forall\, s' \in P_s & h(s) \le \text{cost}(s, s') + h(s') \\ & h(s_{\text{goal}}) = 0 \end{cases} \quad (۱)$$

که در معادله فوق P_s مجموعه گره‌هایی است که از s به آن‌ها مسیری وجود دارد. متغیرهای میانی فوق شامل صف گره‌های مورد بررسی و همچنین مقدار g-value گره‌های خارج از مسیر، پس از به دست آمدن پاسخ از حافظه پاک نمی‌شوند. در روش D* جستجو از مقصد آغاز می‌شود. بدین ترتیب، g-value هزینه هر گره تا مقصد است و h-value معیاری ابتکاری از فاصله هر گره تا مبدا می‌باشد. پس از اعمال این تغییر، قانون جستجوی A* به همان صورت قابل اعمال خواهد بود؛ با این تفاوت که در صورت حرکت ربات، همچنان می‌توان از مقادیر بدست آمده قبلی برای محاسبه مسیر جدید استفاده کرد. برای این کار، پس از کشف تغییرات در وزن یال‌ها یا تابع ابتکاری، برای هر گره مانند s، عامل زیر تعریف می‌شود:

$$\text{rhs}(s) = \min_{s' \in P_s} g(s') + \text{cost}(s, s') \quad (۲)$$

مفهوم عامل rhs از طریق مقایسه آن با $g(s)$ مشخص می‌شود. در واقع، اگر داشته باشیم:

$$g(s) < rhs(s) \quad (۳)$$

به این معنا خواهد بود که به ازای یک گره همسایه s مانند s' داریم:

$$g(s) - g(s') < \text{cost}(s, s') \quad (۴)$$

که نشان می‌دهد حرکت از s' به s بهتر از حرکتی است که در مسیر فعلی طراحی شده است. در این حالت، به گره s یک گره Over-consistent گفته می‌شود. از آنجایی که مسیر منتهی به این گره دیگر مطلوب نیست، مسیر باید باز طراحی شده و این گره باید مانند سایر گره‌ها به درخت جستجو بازگردد. اولویت بندی و ترتیب جستجو در ادامه همین قسمت به طور مفصل شرح داده شده است.

همچنین، حالت مقابل رابطه (٤) به صورت زیر است:

$$g(s) > rhs(s) \quad (۵)$$

این حالت نشان می‌دهد که مسیر فعلی همچنان بهینه است اما هزینه آن کاهش یافته است. در این حالت نیازی به طراحی مسیر جدید نبوده و تنها کافیست مقدار g-value برای گره s و گره‌های قبل آن به روز شود. به این حالت، Under-consistent گفته می‌شود.

در D* Lite برای مشکل تغییر تابع ابتکاری حین کار ربات (که ناشی از حرکت خود ربات و تغییر فاصله هندسی آن با گره‌ها است)، تغییری در نحوه اولویت‌دهی به جستجوی گره‌ها داده شده است. بر اساس این تغییر، به جای استفاده از معیار حاصل جمع مقدار g-value و h-value، از حد پایین این عامل استفاده می‌شود. برای این کار، اولویت‌بندی گره‌ها بر اساس جزء اول و سپس جزء دوم زوج مرتب زیر انجام می‌شود:

$$K = \begin{cases} \min\big(g(s), rhs(s)\big) + h(start, s) \\ \min\big(g(s), rhs(s)\big) \end{cases} \quad (۶)$$

بدین ترتیب هنگامی که ربات از گره a به b حرکت می‌کند، نیازی به محاسبه h-valueها و مرتب کردن مجدد صف اولویت‌بندی نیست؛ زیرا تغییرات هزینه ابتکاری گره‌ها یک حد بالا دارد که برابر با تفاوت مقدار ابتکاری گره a و b است. لازم به ذکر است که این ویژگی تنها زمانی صحت دارد که تابع ابتکاری نامنفی بوده و شرط اضافی زیر را نیز ارضا کند:

$$h(a, c) \le h(a, b) + h(b, c) \quad (۷)$$

شرط فوق برای اغلب توابع ابتکاری رایج، صادق است. مزیت قضیه ذکر شده در این است گام محاسباتی سنگین مربوط به مرتب‌سازی صف گره‌ها از حلقه حذف می‌شود.

گفته شد که g-value هزینه یک گره تا مقصد را نشان می‌دهد. بنابراین تغییر وزن یک یال روی مقدار g-value مربوط به گره‌های بعدی تاثیر نمی‌گذارد. بدین ترتیب، بازطراحی مسیر فقط از محل کشف تغییر تا مبدا (نقطه کنونی ربات) انجام می‌شود.

مسیرهای طراحی شده برای هر ربات به صورت زوج مرتب به گره‌های مسیر نسبت داده می‌شود. این مقدار را با V_s نمایش می‌دهیم و مقدار آن به صورت زیر است:

$$V_s = \begin{cases} (\text{rid}, t_{\text{ocp}}) & \text{گره در مسیر یک ربات است} \\ (0,0) & \text{گره در مسیر هیچ رباتی نیست} \\ (-1, \infty) & \text{گره یک مانع است} \end{cases} \quad (۸)$$

در عبارت فوق، rid کد id ربات است. همچنین، t_{ocp} زمان تصرف (Occupancy time) را نشان می‌دهد. زمان تصرف هر گره نشان می‌دهد که چند گام زمانی تا لحظه‌ای که ربات وارد آن گره می‌شود باقی مانده است (اگر آن گره در مسیر پیش‌بینی شده برای ربات مورد نظر قرار گرفته باشد). با گذشت هر گام زمانی، مقدار زمان تصرف تمام گره‌های درون مسیر یک واحد کاهش می‌یابد؛ مگر در مواردی که مسیر جدیدی طراحی شده باشد. جهت جلوگیری از تداخل مسیر ربات‌ها، وزن یال‌های گراف به صورت زیر تعریف می‌شود:

$$E_{s-s'} = \begin{cases} \sqrt{\Delta x^2 + \Delta y^2} & \text{مسیر آزاد است} \\ \infty & \text{مسیر بسته است} \\ \infty & t_{ocp_s} > 0 \text{ یا } t_{ocp_{s'}} > 0 \end{cases} \quad (۹)$$

برای هر ربات متغیری از جنس RobotData برای نگهداری اطلاعات تعریف می‌شود که متشکل از موارد زیر است:

- rid: کد id ربات
- $s_{i_{\text{start}}}$: موقعیت اولیه ربات
- s_{start}: موقعیت فعلی ربات (که طراحی مسیر از آن آغاز می‌شود)
- s_{last}: مقدار قبلی s_{start}

- s_{goal}: گره هدف ربات
- Path: گره‌هایی که ربات از آن‌ها گذشته است.
- PathComplete: مقدار بولی که مشخص می‌کند آیا ربات به هدف رسیده است یا خیر.
- MoveFlag: مقدار بولی که مشخص می‌کند آیا ربات باید حرکت کند یا خیر.

همچنین به ازای هر ربات، مقادیر مورد استفاده در الگوریتم D* Lite نظیر مقادیر g-value گره‌ها نگه‌داری می‌شود. این مقادیر را نمی‌توان به طور مشترک برای تمام ربات‌ها استفاده کرد؛ چرا که برای هر ربات، مسیر ربات‌های دیگر یک مانع تلقی شده و باعث می‌شود وزن یال‌های گراف از دیدگاه هر ربات متفاوت باشد.

الگوریتم D* Lite توسعه یافته در اولین قدم به متغیرهای فوق مقدار اولیه می‌دهد. این مرحله آماده‌سازی نامیده می‌شود. مقادیر اولیه g-valueها و rhsها بی‌نهایت تعریف می‌شود تا گره هدف Over-consistent شود. همچنین، مکان اولیه ربات‌ها به عنوان گره اشغال شده تعریف می‌شود. به عبارتی، برای هر ربات خواهیم داشت:

$$s_{start} = (rid, 1) \quad (۱۰)$$

پس از آماده‌سازی، نوبت به تعریف تابع محاسبه کوتاه‌ترین مسیر می‌رسد.این تابع دقیقا مانند آنچه در D* Lite معمولی به کار می‌رود است؛ با این تفاوت که اگر مسیری به گره مقصد پیدا نشود، این امر با صفر شدن متغیری به نام فلگ‌موفقیت اعلان می‌شود. روش کار این تابع در شبه کد شکل ۱ آمده است. همچنین، تابع به‌روزرسانی‌گره برای به‌روزرسانی صف گره‌ها و کلیدهای آن‌ها تعریف شده است که جزییات آن در شبه کد شکل ۲آمده است.

بدین ترتیب، تابع اصلی الگوریتم D* Lite تعریف می‌شود. این تابع به ترتیب برای تمام ربات‌ها اجرا می‌شود و اجرای آن بر عهده حلقه خارجی است. این تابع در شکل ۳ به صورت فلوچارت نمایش داده شده است. تفاوت‌های این تابع با آنچه در روش D* Lite استاندارد وجود دارد عبارت است از:

- مدیریت حالت‌های استثنا با استفاده از فلگ موفقیت
- تعریف مسیر ربات‌های قبلی به عنوان موانع گذرا برای ربات‌های بعدی

ورودی‌های تابع اصلی که هنگام فراخوانی توسط تابع مرکزی مقداردهی می‌شوند عبارت‌اند از:

- ساختمان اطلاعات رباتی که تابع برای آن اجرا می‌شود (RobotData).
- *اطلاعات* گراف که مسیرهای طراحی شده در آن نگهداری می‌شود.
- ماتریس وزن یال‌ها
- لیست گره‌هایی که باید به‌روزرسانی شوند

الگوریتم ۱ - محاسبه کوتاه‌ترین مسیر

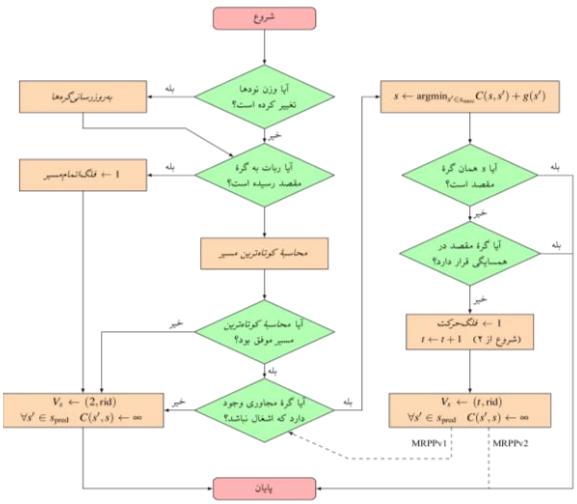
یادداشت: Qصف گره‌ها بوده و $Q[0]$ اولین گره صف است.
متد محاسبه کوتاه‌ترین مسیر
فلگ موفقیت ← ۱
تکرار تا $K_{old}^{Q[0]} < K_{start}$یا $g(start) \neq rhs(start)$
اگر $K_{old}^{Q[0]} < K^{Q[0]}$
مقدار $K^{Q[0]}$ را به‌روزرسانی کن.
وگرنه‌اگر $rhs(s^{Q[0]}) < g(s^{Q[0]})$
$rhs(s^{Q[0]}) \rightarrow g(s^{Q[0]})$
وگرنه
$\infty \rightarrow g(s^{Q[0]})$
پایان‌اگر
به‌روزرسانی گره برای $Q[0]$ و گره‌های منتهی به آن
حذف $Q[0]$ از صف
اگر Q خالی است
فلگ موفقیت ← ۰
پایان‌اگر
پایان‌تکرار
پایان‌متد

شکل ۱) شبه کد الگوریتم ۱

الگوریتم ۲ - به‌روزرسانی گره

متد به‌روزرسانی گره
تکرار روی گره‌های هدف مانند s
اگر $s \neq s_{goal}$
$\min_{s' \in P_s} C(s,s') + g(s') \rightarrow rhs(s)$
پایان‌اگر
اگر rid $>$ 0 و rid $\neq$ robot.id
s را از صف Q حذف کن.
وگرنه
مقدار K_s را به‌روزرسانی کن.
s را به صف Q اضافه کن.
پایان‌اگر
پایان‌تکرار
پایان‌متد

شکل ۲) شبه کد الگوریتم ۲



شکل ۳) تابع اصلی روش D* Lite تغییریافته

همچنین، خروجی‌های این تابع عبارت‌اند از:

- فلگ‌پایان‌مسیر: نشان می‌دهد که آیا ربات به مقصد رسیده است یا خیر.
- فلگ‌حرکت: نشان می‌دهد که آیا حرکتی برای ربات در گام زمانی فعلی طراحی شده است یا خیر.
- به‌روزرسانی اطلاعات گراف

همانطور که در شکل ۳ دیده می‌شود، در تابع اصلی ابتدا بررسی می‌شود که آیا وزن هیچ‌یک از گره‌ها تغییر کرده است یا خیر. این اطلاعات هنگام فراخوانی تابع اصلی توسط تابع مرکزی ارائه می‌شود. در صورتی که گره‌هایی دچار تغییر وزن شده باشند، این گره‌ها به تابع به‌روزرسانی‌گره‌ها فرستاده می‌شوند که در شبه کد شکل ۲ تعریف شده است. وظیفه تابع مذکور این است که لیست گره‌های مورد بررسی را به‌روز کند. به طور دقیق‌تر، اگر وزن یک گره اشغال نشده یا گره‌های پیش از آن تغییر کرده باشد، گره به لیست گسترش (Expansion List) اضافه می‌شود تا در الگوریتم محاسبه کوتاه‌ترین مسیر مورد بررسی قرار گیرد.

پس از به‌روزرسانی گره‌ها، ابتدا بررسی می‌شود که آیا ربات به مقصد رسیده است یا خیر. در صورتی که ربات به مقصد رسیده باشد، حرکت بیشتری برای ربات طراحی نشده و پس از تعیین نقطه فعلی ربات به عنوان مانع برای ربات‌های دیگر، متد اصلی به پایان می‌رسد.

در صورتی که ربات به مقصد نرسیده باشد، تابع محاسبه کوتاه‌ترین مسیر فراخوانی می‌شود. در این تابع در صورت لزوم مقادیر g-value و rhs گره‌ها به‌روزرسانی شده و مسیر بهینه جدید محاسبه می‌شود. هرگاه این تابع در محاسبه مسیر موفق نباشد (مسیری به نقطه هدف وجود نداشته باشد)، حرکتی برای ربات طراحی نمی‌شود و ربات درجا می‌زند؛ یعنی نقطه فعلی ربات اشغال شده در نظر گرفته شده و الگوریتم برای آن گام زمانی به پایان می‌رسد. به طور مشابه، اگر تمام گره‌های اطراف ربات اشغال باشند، ربات درجا می‌زند.

پس از طراحی مسیر، باید گذشتن مسیر از روی گره‌ها در گراف اشتراکی ثبت شود. همان‌طور که گفته شد، این کار با استفاده از مفهوم زمان تصرف انجام می‌شود (معادله (۸)). در نسخه اول الگوریتم (MRPPv1) برای این کار، تمام نقاط درون مسیر ربات در یک حلقه با شناسه ربات و گام زمانی که ربات آن‌ها را اشغال خواهد کرد، مقداردهی می‌شوند. حلقه هنگامی که ربات به مقصد برسد خاتمه می‌یابد. در نسخه دوم (MRPPv2)، ثبت زمان تصرف فقط برای یک گره بعدی در مسیر هر ربات انجام می‌شود. بدین ترتیب، در نسخه دوم در هر لحظه گره‌های اشغال نشده بیشتری وجود دارد.

۲-۲ تابع مرکزی طراحی مسیر چندرباتی

تابع مرکزی طراحی مسیر چندرباتی، مدیریت فراخوانی D* Lite برای ربات‌ها را بر عهده دارد که شامل اولویت‌بندی ربات‌ها، به‌روزرسانی داده‌های مشترک و مدیریت حالت‌های استثنا می‌باشد. این تابع الزاما به صورت متمرکز (Centeralized) اجرا می‌شود و ورودی‌های آن عبارت‌اند از:

- ماتریس وزن گراف
- اطلاعات ربات‌ها
- مقادیر وضعیت گره‌های گراف

الگوریتم عملکرد این تابع در شکل ۴ نشان داده شده است. همان‌طور که دیده می‌شود، در اولین گام، اطلاعات ربات‌ها مقداردهی اولیه می‌شود. این مرحله تنها یک بار انجام می‌شود و در آن اطلاعات ربات‌ها آماده می‌شود. این اطلاعات شامل وضعیت فعلی ربات، هدف آن و همچنین یک نسخه از گراف است که در آن مقادیر مورد استفاده در الگوریتم D* Lite شامل مقادیر g-value و rhs نگهداری می‌شود.

پس از آماده‌سازی، حلق اصلی آغاز می‌شود. این حلقه تا رسیدن تمام ربات‌ها به مقصد ادامه می‌یابد. در هر بار اجرای حلقه اصلی، ابتدا ربات‌ها بر اساس یک تابع خارجی اولویت‌بندی می‌شوند و سپس در یک حلقه داخلی به ازای هر ربات تابع D* Lite تغییریافته اجرا می‌شود. اگر فلگ‌پایان‌مسیر که از تابع اصلی دریافت می‌شود برابر با یک باشد، ربات از فهرست حذف می‌شود و در حلقه‌های بعدی مورد بررسی قرار نخواهد گرفت.

در پایان هر اجرای حلقه بیرونی، یک بار تابع به‌روزرسانی گراف اجرا می‌شود. در این تابع وظایف زیر انجام می‌شود:

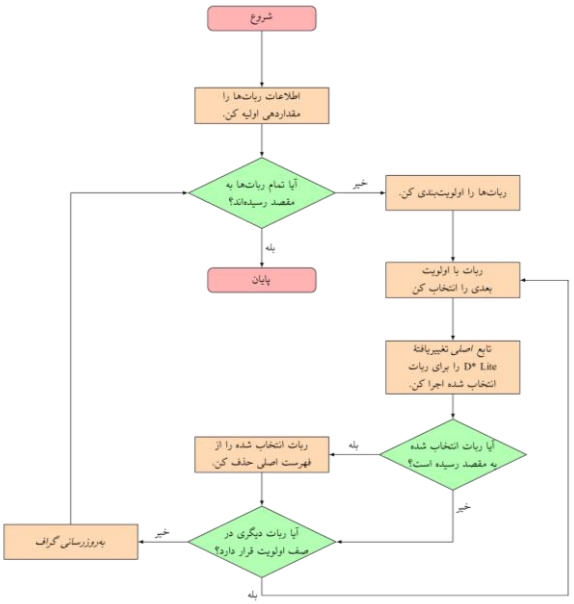
- مقادیر t_{ocp} یک گام زمانی به جلو برده می‌شود:

$$\forall\, s \text{ where } t_{ocp} > 0 \qquad t_{\text{ocp}} \leftarrow t_{\text{ocp}} - 1$$

- موانع خارجی که جدیداً کشف شده‌اند، در ماتریس وزن گراف اعمال می‌شوند:

$$\forall s \in \text{obstacle nodes} \qquad C(s_{\text{pred}}, s) \leftarrow \infty$$

در نهایت، با رسیدن آخرین ربات به مقصد تعیین شده، اجرای حلقه اصلی متوقف خواهد شد.



شکل ۴) تابع مرکزی طراحی مسیر چندرباتی

۳- شبیه سازی غیر برخط

در این بخش الگوریتم در محیط دو بعدی شبیه‌سازی شده و از دینامیک محلی ربات‌ها صرف نظر شده است. برای تعیین محل شروع و پایان ربات‌ها و همچنین محل موانع، از دو روش منظم و همچنین روش جایگذاری تصادفی استفاده شده است.

۳-۱ معیارهای ارزیابی

الف) طول متوسط مسیر ($\bar{L}$)

این معیار، جمع هزینه یال‌های پیمایش شده ربات‌ها می‌باشد که به صورت مقدار متوسط توسط رابطه ۱۱ بیان می‌شود. P_{rid} لیست گره‌های تشکیل دهنده مسیر ربات rid است.

$$\bar{L} = \frac{1}{N}\sum_{rid=1}^{N}\sum_{(s,s')\in P_{\text{rid}}} C(s,s') \qquad (۱۱)$$

ب) زمان متوسط مسیر ($\bar{T}$)

زمان متوسط مسیر بیانگر تعداد گام زمانی متوسط مسیرها است. اهمیت این عامل به این دلیل است که در کاربردهای صنعتی، علاوه بر کاهش هزینه جاری، کاهش زمان تولید نیز مطلوب است.

ج) زمان طولانی‌ترین مسیر ($T_{\max}$)

این عامل زمان انجام ماموریت را برای آخرین ربات که به مقصد می‌رسد را نشان می‌دهد.

د) ناصافی متوسط مسیر ($\bar{R}$)

ناصافی متوسط مسیر مقدار متوسط تغییرات زاویه ربات‌ها را نشان می‌دهد که توسط معادله ۱۲ بیان شده است. $\theta(s,s')$ زاویه ربات در صفحه دوبعدی در ابتدای یال $s-s'$ است.

$$\bar{R} = \frac{1}{N}\sum_{\text{rid}=1}^{N}\left(\sum_{(s,s'),(s',s'')\in P_{\text{rid}}} |\theta(s,s') - \theta(s',s'')|\right) \qquad (۱۲)$$

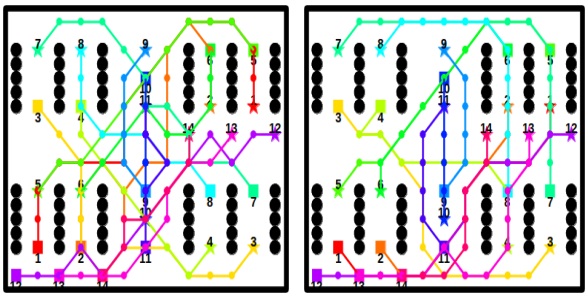
ه) زمان پردازش به ازای ربات (pTime)

این عامل مقدار متوسط زمان پردازش را بر حسب ثانیه نشان می‌دهد. این عامل با آزمایش روی سامانه‌ای دارای یک پردازنده Intel Core i5 و ۴ گیگابایت حافظه RAM گزارش شده است. پیاده‌سازی کد مسیریابی چندرباتی با استفاده از نرم‌افزار MATLAB انجام شده است.

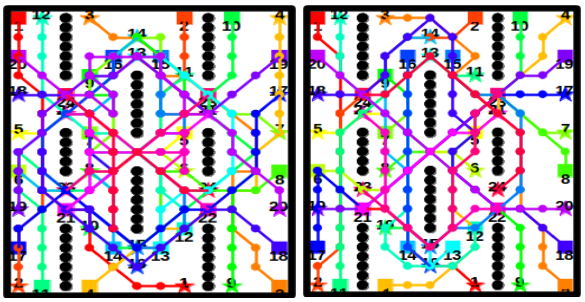
۳-۲ تحلیل نتایج

شکل ۵ مسیرهای طراحی شده در دو نقشه با آرایش منتظم ربات‌ها با ۱۴ و ۲۴ ربات را نشان می‌دهد. مشخصات این مسیرها به طور کمی در جدول ۱ و ۲ ارائه شده است. نتایج نشان می‌دهد که روش MRPPv1 مسیرهایی با طول متوسط کمتر و صافی بیشتر طراحی می‌کند. اما از لحاظ زمانی عملکرد ضعیفی دارد که خود را در زمان متوسط و بیشینه مسیرهای طراحی شده نشان می‌دهد. زمان طولانی‌ترین مسیر در این روش حدود سه برابر زمان طولانی‌ترین مسیر در روش MRPPv2 می‌باشد که تفاوت چشمگیری است.

در روش MRPPv1، زمان تصرف تمام گره‌های تشکیل دهنده مسیر بهینه افزایش پیدا می‌کند. در نتیجه، این گره‌ها تا زمان عبور ربات مربوطه از آن‌ها برای ربات‌های دیگر غیر قابل دسترس خواهند بود و مسیرهای بهینه دچار تغییر نمی‌شود. درحالیکه در روش MRPPv2 امکان تصرف گره‌های مسیر بهینه یک ربات توسط ربات‌های دیگر وجود دارد که منجر به تغییر مسیر ربات خواهد شد. این موضوع باعث افزایش طول مسیر و نیز ناصافی مسیرهای طراحی شده توسط روش MRPPv2 در مقایسه با MRPPv1 می شود. اما از آنجایی که تعداد گره‌های آزاد در روش MRPPv1 کمتر است، ربات‌ها باید زمان بیشتری را منتظر آزاد شدن گره‌ها و یافتن مسیر بهینه باشند. در واقع، طول کم و صافی زیاد مسیرهای طراحی شده در روش MRPPv1 به دلیل ها توقف بیشتر ربات‌ها در این روش و تداخل هندسی کمتر مسیرهای طراحی شده است. بدین ترتیب، مسیرهای صاف‌تری طراحی می‌شوند اما این امر به قیمت از دست رفتن همزمانی است.



(الف) نقشه ۱ (۱۴ ربات) MRPPv1 (ب) نقشه ۱ (۱۴ ربات) MRPPv2



(ج) نقشه ۲ (۲۴ ربات) MRPPv1 (د) نقشه ۲(۲۴ ربات) MRPPv2

شکل ۵) نمایش بصری مسیرهای طراحی شده برای نقشه‌های با مانع

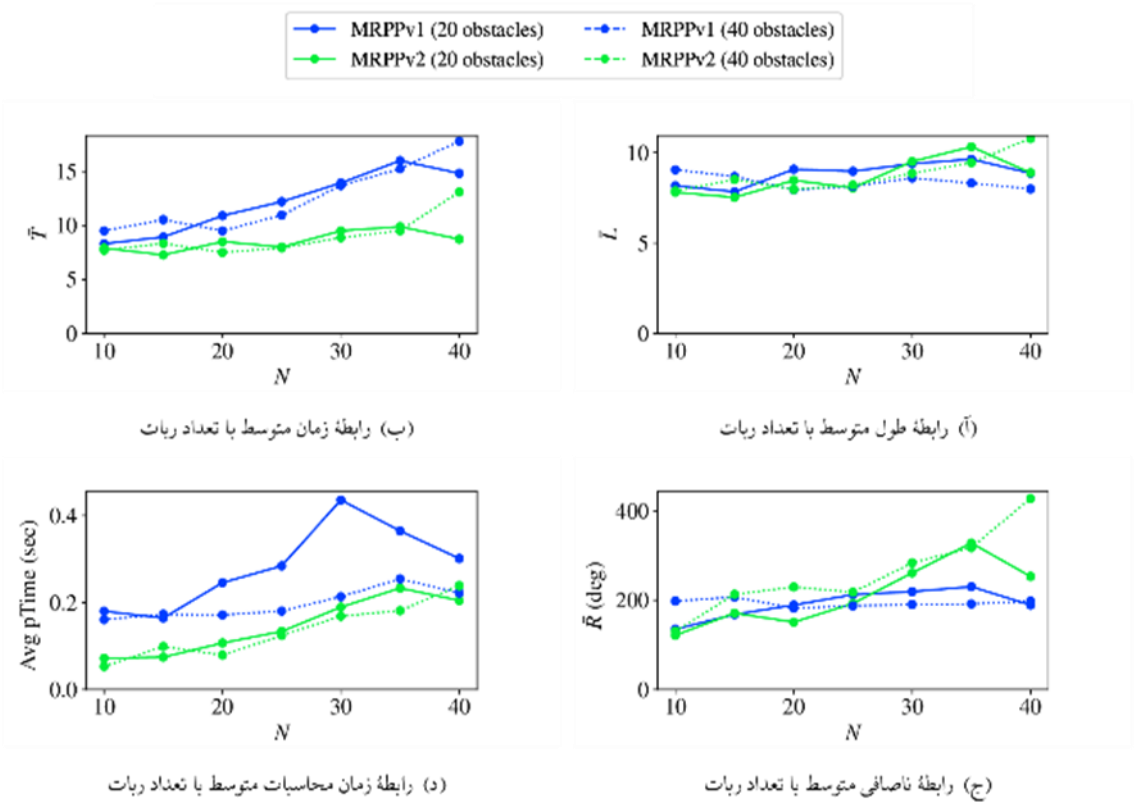
جدول ۱) مشخصات مسیرهای طراحی شده برای نقش ۱

تعداد	نسخه	$\bar{L}$	$\bar{T}$	$T_{\max}$	$\bar{R}$	$\overline{\text{pTime}}$
۱۴	MRPPv1	۱۰/۸	۱۶/۹	۴۲	۱۹۲/۲	۰/۱۵۶
۱۴	MRPPv2	۱۱/۳	۱۰/۹	۱۷	۲۷۳/۲	۰/۱۱۰

جدول ۲) مشخصات مسیرهای طراحی شده برای نقشه ۲

تعداد	نسخه	$\bar{L}$	$\bar{T}$	$T_{\max}$	$\bar{R}$	$\overline{\text{pTime}}$
۲۴	MRPPv1	۱۴/۶	۳۵/۲	۷۳	۲۶۲/۵	۰/۶۶۹
۲۴	MRPPv2	۱۷/۸	۱۷/۳	۲۷	۵۶۶/۲	۰/۴۳۸

در آزمون‌های تصادفی، تعداد ربات‌ها با گام مساوی از ۱۰ تا ۴۰ تغییر داده شده است. شکل ۶ روند عملکرد الگوریتم مسیریابی را در نقشه‌های تصادفی نشان می‌دهد که محور افقی تعداد ربات‌ها



شکل ۶) عملکرد دو الگوریتم پیشنهادی بر حسب تعداد ربات در نقشه با مانع

و محور عمودی مشخصه عملکردی را نشان می‌دهد. منحنی‌های نقطه‌چین نتایج را برای آزمون در محیطی با ۴۰ مانع نشان داده و منحنی دیگر نتایج را برای آزمون در محیطی با ۲۰ مانع نمایش می‌دهد. تعداد کل سلول‌های محیط ۱۴۴ بوده و محل موانع و نقاط شروع و پایان به طور تصادفی تولید می‌شوند.

همان‌طور که در شکل ۶-آ مشاهده می‌شود، طول متوسط مسیرهای طراحی شده توسط دو الگوریتم مشابه یکدیگر می‌باشد و اختلاف معنی‌داری مشاهده نمی‌شود. اما در انتهای سمت راست نمودار مشاهده می‌شود هنگامی که تعداد ربات بالا می‌رود روش MRPPv1 اندکی مسیرهای کوتاه‌تری را به دست می‌دهد. زیرا با افزایش تعداد گره‌های رزرو شده، احتمال تغییر مسیر از مسیر بهینه کاهش پیدا می‌کند.

در مقابل، در شکل ۶-ب ملاحظه می‌شود که زمان متوسط مسیر با افزایش تعداد ربات افزایش می‌یابد. این افزایش در مورد روش MRPPv1 چشمگیرتر است. همچنین، افزایش تعداد موانع اثر معنی‌داری روی افزایش زمان طی‌کردن مسیر ندارد؛ زیرا موانع ثابت هستند و نیازی به محاسبات پیشگیری از برخورد نیست. در نهایت، مشاهده می‌شود که به طور کلی روش MRPPv2 از لحاظ زمانی عملکرد بهتری دارد که این امر با افزایش تعداد ربات بیشتر خود را نشان می‌دهد. دلیل این موضوع، کمبود تعداد گره‌های آزاد در روش اول نسبت به روش دوم است که منجر به انتظار تعدادی از ربات‌ها تا آزاد شدن گره‌ها توسط دیگر ربات‌ها در محیط است.

ناصافی مسیر با افزایش تعداد ربات زیاد می‌شود که این مورد در شکل ۶-ج قابل ملاحظه است. همچنین مشاهده می‌شود که ناصافی مسیر عموما در روش MRPPv2 بیشتر است و روند افزایشی آن نیز در این روش سریع‌تر است. زیرا جز گره اول در مسیر بهینه، زمان تصرف گره‌های دیگر افزایش نیافته و امکان دسترسی توسط ربات‌های دیگر وجود دارد. این موضوع منجر به تغییر مسیر ربات‌ها و در نتیجه افزایش ناصافی می‌شود.

در نهایت، شکل ۶-د نشان می‌دهد که زمان محاسبات با افزایش تعداد ربات روند صعودی دارد. همچنین مشاهده می‌شود که روش MRPPv1 زمان محاسبات بیشتری احتیاج دارد. دلیل این موضوع، کاهش تعداد گره‌های آزاد در روش اول است. در این روش، زمان تصرف تمامی گره‌های موجود در مسیر بهینه طراحی شده افزایش می‌یابد و در نتیجه، تعداد گره‌های با زمان تصرف صفر نسبت به روش دوم کمتر است. کاهش تعداد گره‌های آزاد منجر به پیچیدگی در یافتن مسیر بهینه و نیز تکرار بیشتر این فرایند می‌شود.

۴- شبیه‌سازی برخط و مقایسه

به منظور مقایسه روش پیشنهادی با یکی از روش‌های برخط موجود یعنی روش طراحی مسیر چندرباتی مبتنی بر میدان‌های پتانسیل مصنوعی پیشنهاد شده توسط متوعی و همکاران[16]، روش پیشنهادی در بستر سامانه عاملی رباتی (ROS) پیاده‌سازی شده است.

شبیه‌سازی‌ها نیز در شبیه‌ساز Gazebo صورت گرفته و در نتیجه دینامیک ربات‌ها در نظر گرفته شده و طراحی مسیر نیز به صورت برخط انجام می‌شود. همچنین از مدل ربات Turtlebot3 به عنوان ربات زمینی استفاده شده است. شکل ۷ محیط عملیات سه بعدی به همراه موانع و نیز ربات‌ها را نمایش می‌دهد. نرخ صدور فرمان‌های کنترلی برابر با ۱۰ هرتز در نظر گرفته شده است

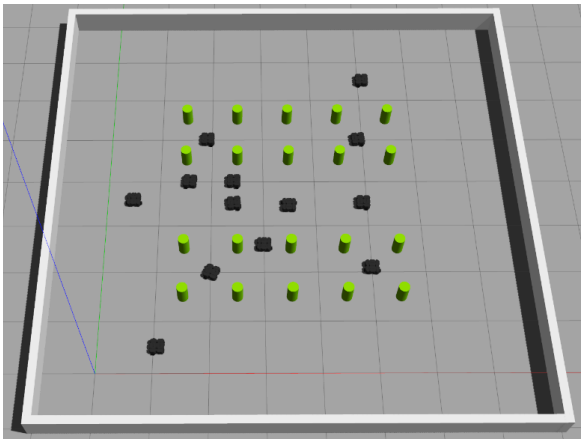
جدول ۳ نتایج این شبیه‌سازی را برای روش پیشنهادی و نیز روش میدان‌های پتانسیل مصنوعی با توجه به شاخص‌های متوسط طول مسیر $\bar{L}$، زمان اتمام عملیات T و متوسط صافی مسیر $\bar{R}$ برای تعداد مختلف ربات ارائه می‌دهد. موقعیت شروع و هدف ربات‌ها به صورتی انتخاب شده است که ارتباط مستقیم میان آن‌ها از منطقه میانی محیط عملیات عبور کند. مقایسه نتایج نشان می‌دهد که روش میدان پتانسیل در صورت موفقیت، مسیرهایی با طول کمتر ایجاد می‌کند. دلیل این موضوع، عدم محدودیت روش میدان پتانسیل به حرکت میان گره‌های شبکه مانند روش‌های جستجوی شبکه است.

اما مقایسه زمان انجام ماموریت نشان می‌دهد که با افزایش ربات‌ها به تعداد ۷ و بیشتر، زمان اجرای ماموریت روش پیشنهادی کمتر از روش میدان پتانسیل است. این موضوع به دلیل عدم هماهنگی در طراحی مسیر در روش میدان پتانسیل و ترافیک زیاد به علت نزدیک شدن ربات‌ها به میانه محیط است. اما در روش پیشنهادی، مسیرها از قبل به گونه‌ای طراحی شده که برخوردی در مسیر هیچ یک از ربات‌ها وجود نداشته باشد. همچنین به علت حرکت ربات‌ها در مسیرهای نسبتا صاف و بدون تغییر جهت‌های مداوم، روش پیشنهادی منجر به مسیرهای صاف‌تری نسبت به روش میدان پتانسیل شده است.

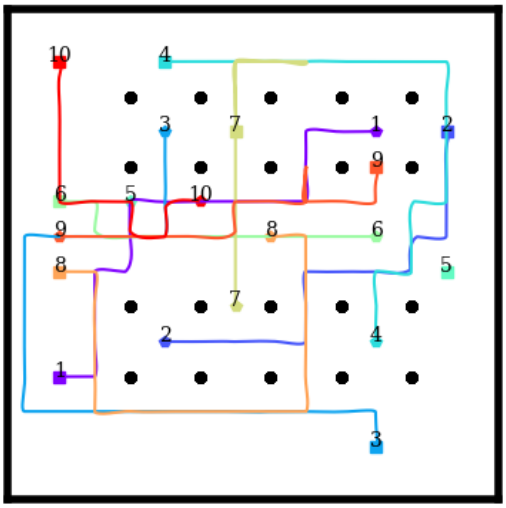
در سناریوهایی با تعداد ربات بیشتر از ۱۱، روش میدان پتانسیل قادر به اتمام عملیات به صورت موفقیت آمیز نمی‌باشد. شکل ۸ به ترتیب موقعیت ثبت شده ربات‌ها در محیط Gazebo در طول پیمایش مسیرهای طراحی شده توسط الگوریتم پیشنهادی و روش میدان‌های پتانسیل را برای سناریو با ۱۰ ربات نمایش می‌دهد. این شکل نیز عدم هماهنگی در طراحی مسیر در روش میدان پتانسیل را نشان می‌دهد که منجر به ترافیک سنگین ربات‌ها در بخش میانی محیط شده است.

۵- بحث و نتیجه‌گیری

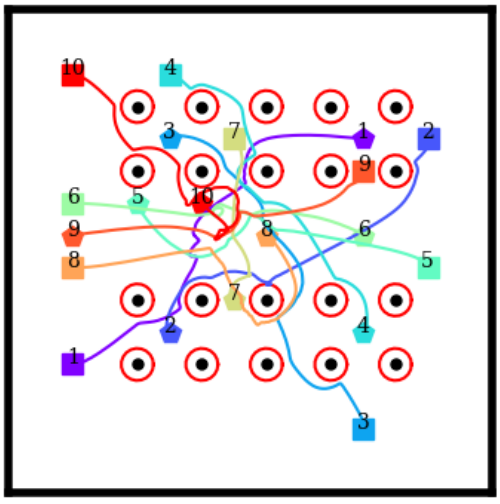
در این پژوهش، یک روش طراحی مسیر چندرباتی با قابلیت عملیات برخط در محیط پویا ارائه شد. این روش بر پایه الگوریتم D* Lite طراحی شده و با توسعه الگوریتم و همچنین تعریف یک حلقه کنترل‌گر بیرونی، برای کاربردهای چندرباتی بهینه‌سازی شد. این روش قادر به به‌روزرسانی مسیرهای طراحی شده در صورت وجود مسیر بهتر و یا تغییر در آرایش محیط عملیات در زمان محدود می‌باشد. بنابراین، این روش در کاربردهای رباتیکی صنعتی مانند انبارهای خودکار که زمان تصمیم‌گیری محدود بوده و محیط عملیات دارای پویایی بالا است قابل استفاده است. دو نسخه مختلف از این روش ارائه شد که تفاوت آن‌ها در نحوه افزایش زمان



شکل ۷) نمایی از شبیه‌سازی در محیط Gazebo



الف) مسیرهای ۱۰ ربات در طراحی مسیر با روش پیشنهادی



ب) مسیرهای ۱۰ ربات در طراحی مسیر با روش میدان پتانسیل

شکل ۸) مسیرهای ثبت شده ربات‌ها در طول شبیه‌سازی در محیط Gazebo

جدول ۳) مشخصات مسیرهای طراحی شده در شبیه سازی برخط

تعداد	روش پیشنهادی			روش میدان پتانسیل مصنوعی		
	$\bar{L}$	$\bar{T}$	$\bar{R}$	$\bar{L}$	$\bar{T}$	$\bar{R}$
۳	۷٫۵	۶۰٫۱	۱۱٫۴	۵٫۹	۵۳	۳۵٫۵
۴	۷	۶۱	۱۳٫۵	۵٫۸	۵۸٫۸	۷۲٫۶
۵	۷٫۱	۷۷	۱۱٫۸	۵٫۶	۷۱٫۸	۸۳٫۱
۶	۶٫۲	۷۳٫۸	۱۳٫۵	۵٫۲	۷۰٫۱	۵۳٫۶
۷	۶٫۹	۸۷٫۹	۱۲٫۸	۵٫۱	۸۹٫۵	۶۵٫۶
۸	۷٫۸	۱۰۷٫۸	۱۴٫۷	۵٫۶	۱۲۶٫۲	۹۱٫۱
۹	۶	۹۴٫۶	۱۲٫۱	۵٫۴	۱۱۱٫۹	۹۲٫۷
۱۰	۸	۱۱۹	۱۷٫۷	۵٫۷	۱۹۳٫۳	۱۱۷٫۱
۱۱	۷٫۴	۱۳۰٫۳	۱۵٫۱	۶٫۱	۱۸۱٫۲	۱۰۰٫۴
۱۲	۶٫۸	۱۰۷	۱۳٫۴	-	-	-
۱۳	۷٫۵	۱۳۷	۱۷٫۵	-	-	-

تصرف گره‌ها در مسیرهای بهینه است. رفتار الگوریتم پیشنهاد شده ابتدا در محیط دو بعدی بدون در نظر گرفتن دینامیک ربات‌ها مورد بررسی قرار گرفت. هر دو نسخه الگوریتم پیاده‌سازی شده و در سناریوهای مختلف با آرایش و تعداد مختلف ربات‌ها و موانع مورد آزمون قرار گرفتند. نتایج شبیه‌سازی‌ها نشان داد که یکی از روش‌ها (MRPPv1) قادر است مسیرهای کوتاه‌تر و صاف‌تری طراحی کند که این مزیت در محیط‌های با مانع خود را نشان می‌دهد. با این حال، مسیرهای طراحی شده توسط این روش زمان بیشتری برای طی‌کردن نیاز دارند. به عبارتی، مسیرها کوتاه‌تر هستند ولی هم‌زمانی آن‌ها باعث می‌شود ربات‌ها بیشتر مجبور به توقف شوند و زمان اتمام مسیر بالا برود. این زیاد بودن بازه زمانی در مشخصه عملکردی زمان طولانی‌ترین مسیر بیشترین نمود را دارد که در مواردی عملکرد روش MRPPv2 سه برابر بهتر است. همچنین، این الگوریتم به صورت برخط در بستر سامانه عامل رباتیکی پیاده‌سازی شده و در محیط Gazebo مورد آزمون قرار گرفت. نتایج این شبیه‌سازی‌ها با نتایج شبیه‌سازی با روش شاخص مبتنی بر میدان‌های پتانسیل مورد مقایسه قرار گرفت. نتایج نشان داد که با افزایش تعداد ربات‌ها، روش میدان‌های پتانسیل بر خلاف روش پیشنهادی، توانایی مدیریت ترافیک و هماهنگی در طراحی مسیرها را نداشته و در نتیجه زمان عملیات افزایش یافته و یا با شکست مواجه می‌شود.

۶- محدودیت‌ها

از آنجایی که پژوهش پیش رو الگوریتمی را برای عملکرد ربات‌ها ارائه می‌دهد، آزمایش عملی الگوریتم پیشنهادی می‌توانست سودمند باشد. اما با توجه به محدودیت در دسترسی به تعداد زیادی ربات زمینی، آزمون عملی غیرممکن بود. در نتیجه نزدیک‌ترین روش به واقعیت به منظور ارزیابی و آزمون، یعنی استفاده از مدل ربات‌ها در شبیه‌ساز Gazebo به کار گرفته شد.

همچنین، پیاده‌سازی و انتشار الگوریتم پیشنهادی در یک محیط ابری به همراه پروتکل‌های شبکه می‌تواند این امکان را به دست بدهد که پژوهشگران و صنعتگران دیگر از دستاوردهای این پژوهش استفاده کرده و بازخوردهای خود را به اشتراک بگذارند. این موضوع در اهداف پژوهشی نویسندگان به منظور توسعه کامل روش پیشنهادی قرار دارد.

تاییدیه اخلاقی: محتویات علمی این مقاله حاصل پژوهش نویسندگان است و در هیچ نشریه ایرانی و غیر ایرانی منتشر نشده است.

تعارض منافع: مقاله حاضر هیچ گونه تعارض منافعی با طرح پژوهشی ندارد.

منابع مالی: حمایت مالی از هیچ سازمانی دریافت نشده است.

منابع

1-Khanmirza E, Haghbeigi M, Nazarahari M, Doostie S. A comparative study of deterministic and probabilistic mobile robot path planning algorithms. In2017 5th RSI international conference on robotics and mechatronics (ICRoM) 2017 (pp. 534-539). IEEE.

2- Gasparetto A, Boscariol P, Lanzutti A, Vidoni R. Path planning and trajectory planning algorithms: A general overview. Motion and Operation Planning of Robotic Systems: Background and Practical Approaches. 2015:3-27.

3- Hwang YK, Ahuja N. A potential field approach to path planning. IEEE transactions on robotics and automation. 1992 ;8(1):23-32.

4- Barraquand J, Langlois B, Latombe JC. Numerical potential field techniques for robot path planning. IEEE transactions on systems, man, and cybernetics. 1992 (2):224-41.

5- Bhattacharya P, Gavrilova ML. Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path. IEEE Robotics & Automation Magazine. 2008;15(2):58-66.

6- Barbehenn M. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. IEEE transactions on computers. 1998 (2):263.

7- Delling D, Sanders P, Schultes D, Wagner D. Engineering route planning algorithms. Springer Berlin Heidelberg; 2009.

8- Koenig S, Likhachev M. Incremental A*. In: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. Cambridge, MA, USA: MIT Press; 2001. p. 1539–46. (NIPS'01).

9- Koenig S, Likhachev M. D*lite. In: Eighteenth national conference on Artificial intelligence. USA: American Association for Artificial Intelligence; 2002. p. 476–83.

10- Koenig S, Likhachev M. Fast replanning for navigation in unknown terrain. IEEE Transactions on Robotics. 2005;21(3):354-63.

11- LaValle SM. Planning algorithms. Cambridge university press; 2006.

12- Zagradjanin N, Pamucar D, Jovanovic K. Cloud-based multi-robot path planning in complex and crowded

environment with multi-criteria decision making using full consistency method. Symmetry. 2019;11(10):1241.
13- Wagner G, Choset H. Subdimensional expansion for multirobot path planning. Artificial intelligence. 2015;219:1-24.
14- Rathi A, Vadali M. Dynamic prioritization for conflict-free path planning of multi-robot systems. arXiv preprint arXiv:2101.01978. 2021
15- Tang B, Xiang K, Pang M, Zhanxia Z. Multi-robot path planning using an improved self-adaptive particle swarm optimization. International Journal of Advanced Robotic Systems. 2020;17(5):1729881420936154.
16- Matoui F, Boussaid B, Metoui B, Abdelkrim MN. Contribution to the path planning of a multi-robot system: centralized architecture. Intelligent Service Robotics. 2020;13(1):147-58.
17- Standley T. Finding optimal solutions to cooperative pathfinding problems. InProceedings of the AAAI Conference on Artificial Intelligence 2010 (Vol. 24, No. 1, pp. 173-178).
18- Felner A, Goldenberg M, Sharon G, Stern R, Beja T, Sturtevant N, Schaeffer J, Holte R. Partial-expansion A* with selective node generation. InTwenty-Sixth AAAI Conference on Artificial Intelligence 2012.
19- Yu J, LaValle SM. Planning optimal paths for multiple robots on graphs. In2013 IEEE International Conference on Robotics and Automation 2013 (pp. 3612-3617). IEEE.
20- Wagner G, Choset H. M*: A complete multirobot path planning algorithm with performance bounds. In2011 IEEE/RSJ international conference on intelligent robots and systems 2011 (pp. 3260-3267). IEEE.
21- Ferner C, Wagner G, Choset H. ODrM* optimal multirobot path planning in low dimensional search spaces. In2013 IEEE International Conference on Robotics and Automation 2013 (pp. 3854-3859). IEEE.
22- Sharon G, Stern R, Goldenberg M, Felner A. The increasing cost tree search for optimal multi-agent pathfinding. Artificial intelligence. 2013;195:470-95.
23- Sharon G, Stern R, Felner A, Sturtevant NR. Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence. 2015;219:40-66.
24- Luna R, Bekris KE. Efficient and complete centralized multi-robot path planning. In2011 IEEE/RSJ International Conference on Intelligent Robots and Systems 2011 (pp. 3268-3275). IEEE.
25- De Wilde B, Ter Mors AW, Witteveen C. Push and rotate: cooperative multi-agent path planning. InProceedings of the 2013 international conference on Autonomous agents and multi-agent systems 2013 (pp. 87-94).
26- Regele R, Levi P. Cooperative multi-robot path planning by heuristic priority adjustment. In2006 IEEE/RSJ International Conference on Intelligent Robots and Systems 2006 (pp. 5954-5959). IEEE.
27- Maw AA, Tyan M, Nguyen TA, Lee JW. iADA*-RL: Anytime graph-based path planning with deep reinforcement learning for an autonomous UAV. Applied Sciences. 2021;11(9):3948.
28- Khorshid MM, Holte RC, Sturtevant NR. A polynomial-time algorithm for non-optimal multi-agent pathfinding. InFourth Annual Symposium on Combinatorial Search 2011.
29- Matoui F, Boussaid B, Abdelkrim MN. Distributed path planning of a multi-robot system based on the neighborhood artificial potential field approach. Simulation. 2019;95(7):637-57.