«علوم و فناوری‌های پدافند نوین»

سال چهاردهم، شماره ۴، زمستان ۱۴۰۲؛ ص ۲۳۳-۲۲۳

علمی - پژوهشی

استفاده از الگوریتم‌های غزال کوهستان و کپک مخاطی برای حل مسئله برنامه‌ریزی مسیر

سید ابوالفضل شاهزاده فاضلی[۱]، ندا طیبی[۲]، سعیده برخورداری فیروزآبادی[۳]، اسراء موسوی [۴]

۱- دانشیار ۲- کارشناسی‌ارشد ۳ و ۴- دانشجوی دکتری، دانشگاه یزد

(دریافت: ۱۴۰۲/۰۸/۲۵، بازنگری: ۱۴۰۲/۰۹/۳۰، پذیرش: ۱۴۰۲/۱۰/۱۴، انتشار: ۱۴۰۲/۱۱/۰۱)

DOR: https://dor.isc.ac/dor/20.1001.1.26762935.1402.14.4.1.7

چکیده

یکی از بخش‌های مهم رباتیک برنامه‌ریزی مسیر است، به‌طوری‌که مطالعه مسیر ربات یکی از موضوعات بسیار مهم تلقی می‌شود. ربات متحرک باید از موقعیت شروع به سمت موقعیت هدف حرکت کند، درحالی‌که در یک محیط حاوی موانع از موانع موجود اجتناب کند. مسیر باید بر اساس برخی از معیارها مانند کوتاهی طول مسیر، همواری مسیر و امنیت مسیر بهینه باشد. در این مطالعه، هدف اصلی حل مسئله برنامه‌ریزی مسیر برای یک ربات به‌صورت شبکه، ایستا و شناخته شده است که معیارهای کوتاه‌ترین فاصله، امنیت مسیر و همواری مسیر را برآورده می‌سازد. مسئله برنامه‌ریزی مسیر یک مسئله NP-کامل می‌باشد و برای این مسئله روش‌ها و الگوریتم‌های مختلفی پیشنهاد شده است که شامل روش‌های دقیق و فراابتکاری است. برای حل این مسئله با محاسباتی کمتر از الگوریتم‌های فراابتکاری می‌توان استفاده کرد که در این مطالعه از الگوریتم ژنتیک، الگوریتم غزال کوهستان و الگوریتم کپک مخاطی استفاده شده است. در پیاده‌سازی‌ها علاوه بر استفاده از عملگرهای خود الگوریتم‌ها از سه عملگر ساده‌سازی، بازبینی و جایگزینی استفاده شده است و همچنین یک تابع ارزیابی جدید و برای تولید جمعیت اولیه سه عملگر ترمیم گره، ترمیم پاره‌خط و بهبود گره برای ایجاد مسیرهای تاحدامکان شدنی ارائه شده است. نتایج نشان می‌دهند که این الگوریتم‌ها دارای کارایی بالایی هستند و همچنین از پیچیدگی محاسباتی کمتری برای حل این مسئله برخوردارند.

کلیدواژه‌ها: الگوریتم‌های فراابتکاری، برنامه‌ریزی مسیر، الگوریتم ژنتیک، بهینه‌ساز غزال کوهستان، الگوریتم کپک مخاطی.

Using Mountain Gazelle and Slime Mould Algorithms to Solve the Path Planning Problem

A. Shahzadeh Fazeli* , **N.Tayebi, S. Barkhordari Firozabadi,S.** Mosavi,

Yazd University

(Received:2023/11/16, Revised: 2023/12/21, Accepted: 2024/01/04, Published: 2024/01/21)

Abstract

Path planning is one of the important parts of robotics, so studying the path of the robot is considered one of the most important subjects. The mobile robot must move from the start position to the goal position, while avoiding obstacles in an obstacle environment. The route should be optimal based on some criteria such as shortness of the path, smoothness of the path and security of the path. In this study, the main goal is to solve the path planning problem for a robot in the form of a discrete, static and known, which meet the criteria of shortest distance, path security and path smoothness. Path planning problem is a NP-complete problem and various methods and algorithms have been proposed, which include exact and meta-heuristic algorithms. Meta-heuristic algorithms can be used to solve this problem with less computational load. In this study genetic algorithm, mountain gazelle algorithm, and Slime mould algorithm are used. In implementations, in addition to the operators of the algorithms themselves, three simplification, revision and replacement operators have been used, as well as a new evaluation function has been presented and after generating the initial population of three operators node repair, line segment repair and node improvement to create paths up to feasible path has been used. The results show that these algorithms have high efficiency and also have less computational complexity to solve this problem.

Keywords: Meta-heuristic Algorithms, Path Planning, Genetic Algorithm, Mountain Gazelle Optimizer, Slime Mould Algorithm.

*Corresponding Author E-mail: fazeli@yazd.ac.ir

۱. مقدمه

یکی از بخش‌های مهم برای ربات‌ها، برنامه‌ریزی مسیر است که به ربات‌ها اجازه می‌دهد کوتاه‌ترین مسیر یا یک مسیر بهینه را بین دونقطه پیدا کنند، در غیر این صورت مسیرهای بهینه می‌تواند مسیرهایی باشد که میزان چرخش، مقدار ترمز و یا هر آنچه که در یک کاربرد خاص نیاز است را به حداقل برساند. الگوریتم‌ها برای یافتن کوتاه‌ترین مسیر نه‌تنها در رباتیک، بلکه در مسیریابی شبکه، بازی‌های ویدئویی و غیره مهم هستند. برنامه‌ریزی مسیر نیازمند نقشه محیط و ربات است و ربات باید از موقعیت خود باتوجه‌به نقشه آگاه باشد. در چند دهه اخیر مسئله برنامه‌ریزی مسیر برای ربات‌ها موردتوجه قرار گرفته است. در این مطالعه به بررسی مسئله برنامه‌ریزی مسیر برای یک ربات پرداخته می‌شود. ربات نقطه‌ای و محیط ربات به‌صورت گسسته (شبکه)، ایستا و شناخته شده در نظر گرفته شده‌اند. برای این مسئله الگوریتم‌های فراابتکاری مانند الگوریتم ژنتیک، الگوریتم غزال کوهستان و الگوریتم کپک مخاطی پیاده‌سازی شده است. در این الگوریتم‌ها علاوه بر عملگرهای بهینه‌سازی مربوط به خود الگوریتم‌ها از عملگرهای جدیدی استفاده شده است که این عملگرها کارایی الگوریتم‌ها را بالا برده است و از افتادن در بهینه محلی جلوگیری می‌کنند. نتایج این راه کارها در محیط‌های مختلفی بررسی شده است که نشان از عملکرد خوب چهار الگوریتم برای مسئله برنامه‌ریزی مسیر می‌باشد.

۲. برنامه‌ریزی مسیر

تعداد زیادی از کاربردهایی مانند مین‌روبی، کشاورزی، امنیت و نظارت، نظارت بر محیط‌زیست و کاربردهای نظامی و غیره استفاده می‌شوند. براین‌اساس، جنبه‌های مختلفی از تحقیقات در زمینه ربات‌های متحرک در حال بررسی و بحث هستند که باتوجه‌به توانایی ربات‌ها برای هدایت در محیط‌های مختلف استفاده می‌شوند؛ بنابراین برنامه‌ریزی مسیر برای ربات‌ها یک جنبه اساسی در زمینه تحقیقات است. هدف اصلی برنامه‌ریزی مسیر، یافتن مسیر بدون برخورد، در یک محیط با موانع، از یک مکان شروع تا مقصد با برآورده‌کردن معیارهای خاص است. مسئله برنامه‌ریزی مسیر ربات‌ها یکی از پیچیده‌ترین مسائل در حوزه رباتیک شناخته می‌شود و به‌عنوان یک مسئله -NP کامل در ساده‌ترین شکل آن و در حضور موانع متعدد به‌عنوان یک مسئله -NP سخت طبقه‌بندی می‌شود. این پیچیدگی به دلیل چندین هدف متفاوت ناشی می‌شود، یکی از این دلایل این است که باید در حین برنامه‌ریزی مسیر اهدافی مانند کوتاهی طول مسیر، زمان طی‌کردن مسیر، انرژی مصرفی و هموارکردن مسیر نیز بهینه شوند [۱ و ۲]. یکی دیگر از جنبه‌های چالش‌برانگیز در مسئله برنامه‌ریزی مسیر این است که پیچیدگی محاسباتی آن به‌صورت نمایی با افزایش دامنه یا فضای مسئله افزایش می‌یابد [۳]. اهمیت مسئله برنامه‌ریزی مسیر و علاقه محققین به این مسئله در چند دهه اخیر باعث ایجاد روش‌های مختلفی برای حل این مسئله شده است. مسئله برنامه‌ریزی مسیر در چند دهه گذشته در کتاب‌های چست[1] و همکاران [۴]، دیبرگ[2] و همکاران [۵]، لاتمبه[3] [۶] و لاوال[4] [۷] و غیره مورد بررسی قرار گرفته است.

۱-۲. محیط های برنامه ریزی مسیر

مسئله برنامه‌ریزی مسیر از دیدگاه‌های مختلف بسته به نوع محیطی که ربات‌ها و موانع در آن قرار گرفته‌اند یا میزان اطلاعاتی که ربات‌ها از محیط اطرافشان دارند می‌تواند مورد بررسی قرار بگیرد. محیطی که ربات در آن قرار دارد می‌تواند شناخته شده، نیمه‌شناخته‌شده و یا شناخته نشده باشد. هم چنین می‌تواند پیوسته یا گسسته مانند (گراف یا مشبک) پویا یا ایستا باشد. در زیر توضیح مختصری در مورد هر کدام داده شده است [۸].

[1] Choset
[2] De Berg
[3] Latombe
[4] LaValle

محیط ایستا و پویا: محیط برنامه‌ریزی مسیر می‌تواند ایستا یا پویا باشد [۹]. اگر محیط و موانع موجود در آن ثابت و بدون هیچ تغییری در طول حرکت‌های ربات باشند، به این محیط ایستا گفته می‌شود. در مقابل اگر بدون اطلاع ربات موانع تغییر کند، جابه‌جا شوند یا شیء یا ربات دیگری در محیط حرکت کند، به این محیط پویا گفته می‌شود.

محیط شناخته شده، نیمه شناخته شده و یا شناخته نشده: در محیط‌های شناخته شده ربات موقعیت کامل موانع موجود در محیط را در اختیار دارد. درحالی‌که در محیط‌های شناخته نشده چنین نیست و ربات هیچ اطلاعاتی از محیط ندارد، مانند ربات‌های کاوشگر زمانی که می‌خواهند یک محیط شناخته نشده را کاوش کنند. در مقابل محیط‌های نیمه‌شناخته‌شده هم وجود دارند که ربات اطلاعات جزئی درمورد محیط دارد و یا مجهز به حسگرهایی است که هنگام حرکت، محیط را شناسایی می‌کنند [۱۰].

محیط‌های گسسته و پیوسته: محیط گسسته؛ مانند شبکه و محیط پیوسته؛ مانند صفحه که تمایز بین وضعیت گسسته و پیوسته می‌تواند به حالت محیط، اداره‌کردن زمان، و به ادراکات و فعالیت‌های ربات اعمال شود. محیط گسسته، محیطی است که تعداد حالات محدودی داشته باشند. به‌عنوان‌مثال، محیط گسسته مثل بازی شطرنج و رانندگی تاکسی یک مسئله حالت پیوسته و زمان پیوسته است [۱۰].

۲-۲. دسته بندی برنامه‌ریزی مسیر

برنامه‌ریزی مسیر[1] اغلب به دو مسئله برنامه‌ریزی مسیر کلی[2] و مسیر محلی[3] تقسیم می‌شود:

برنامه‌ریزی مسیر کلی: در برنامه‌ریزی مسیر کلی، محیط ربات ایستا، موانع از قبل تعیین شده و جزئیات محیط از قبل شناخته شده است. این روش معمولاً با اجتناب از موانع و دستیابی به مسیریابی مؤثر، مسیری از مکان شروع تا پایان در محیط ایجاد می‌کند. یک نقشه جامع را برای ربات در نظر می‌گیرد و الگوریتم مسیریابی، مسیر بهینه را حتی قبل از اینکه ربات حرکت خود را آغاز کند، تعیین می‌کند. اگر چه توصیف کامل محیط از قبل موجود است؛ اما برنامه‌ریزی مسیر کلی یک مسئله بسیار پیچیده است. مسئله برنامه‌ریزی مسیر کلی را می‌توان با استفاده از روش‌های جستجوی دقیق حل کرد. روش دقیق دارای مزیت کامل‌بودن است، اما زمان اجرا با اندازه مسئله افزایش می‌یابد. این مسئله با روش فراابتکاری نیز حل می‌شود، زیرا در اغلب موارد راه‌حل‌های خوبی با رویکرد محاسباتی کمتری نسبت به هر روش تکراری یا یک روش دقیق پیدا می‌کند [۱۱].

برنامه‌ریزی مسیر محلی: برنامه‌ریزی مسیر محلی زمانی که اطلاعات از محیط کامل نباشد و همچنین زمانی که محیط پویا باشد مورداستفاده قرار می‌گیرد. در واقع، ربات اطلاعات محیط و پیرامونش را از قبل ندارد. این روش، یک مسیر جدید با بازبینی مسیریابی کلی پایه برای پاسخ به تغییرات به وجود آمده در محیط، ایجاد می‌کند. هدف از برنامه‌ریزی مسیر محلی، تولید پیوسته یک مسیر به هنگام شده برای هدایت ربات است. در برنامه‌ریزی مسیر محلی، مسیر یا موانع از قبل شناخته نشده‌اند، بنابراین ربات باید اطلاعات محیط را قبل از این که تصمیمی برای حرکت و تعیین جهت به سمت مکان هدف بگیرد حس و دریافت کند. اگر مانعی پیدا کند، ربات از مسیر کنونی خود منحرف شده و مسیر جدیدی را در محیط به سمت مکان هدف جستجو می‌کند. روش میدان بالقوه یکی از ساده‌ترین روش‌های برنامه‌ریزی مسیر محلی است که بر اساس پتانسیل جذب و دفع عمل می‌کند. در این روش ربات در طول مسیر از موانع دفع و به سمت مکان هدف جذب می‌شود [۱۱].

۳. الگوریتم های برنامه ریزی مسیر

همانند سایر مسائل بهینه‌سازی، مسئله برنامه‌ریزی مسیر را می‌توان با الگوریتم‌های جستجوی دقیق یا فراابتکاری حل کرد.

۱-۳. الگوریتم‌های دقیق

الگوریتم‌های دقیق از مزیت کامل‌بودن برخوردار می‌باشند، به این معنی که آن‌ها جواب بهینه را در صورت وجود خواهند داشت. بااین‌حال، مقیاس‌پذیری این الگوریتم‌ها محدود است، زیرا زمان اجرای آن‌ها به طور نمایی با اندازه مسئله افزایش می‌یابد که نشان‌دهنده اندازه مدل محیطی در مورد برنامه‌ریزی مسیر است.

[1] Path planning
[2] Global Path Planning
[3] Local Path Planning

۲-۳. الگوریتم های فراابتکاری

مسئله برنامه‌ریزی مسیر با الگوریتم دقیق در محیط پیچیده دارای زمان اجرای بالا است، به‌این‌علت از الگوریتم‌های فراابتکاری برای حل مسئله استفاده می‌شود که الگوریتم‌های فراابتکاری ابزار هوشمند برای کشف فضای جستجو هستند که به طور قابل‌توجهی زمان اجرا را کاهش می‌دهند. الگوریتم‌های فراابتکاری شامل الگوریتم‌های احتمالاتی مانند نقشه راه احتمالاتی[1] [۱۲] درخت جستجوی تصادفی [۱۳] و الگوریتم هوش مصنوعی مانند شبکه عصبی و الگوریتم تکاملی هستند [۱۰]. الگوریتم‌های تکاملی مانند الگوریتم ژنتیک[2] [۱۴] از مکانیزم‌ها و عملیات ابتدایی برای حل مسئله استفاده می‌کنند و در طی یک سری از تکرارها به جواب مناسب برای مسئله می‌رسند. این الگوریتم‌ها غالباً از یک جمعیت حاوی جواب‌های تصادفی شروع می‌کنند و در طی هر مرحله تکرار سعی در بهتر کردن مجموعه جواب‌ها دارند [۱۵]. در این مطالعه به معرفی سه الگوریتم فراابتکاری می‌پردازیم که برای حل مسئله برنامه‌ریزی مسیر، پیاده‌سازی و نتایج بررسی شده است.

۳-۳. الگوریتم ژنتیک

الگوریتم ژنتیک یک الگوریتم جستجوی فراابتکاری بر اساس انتخاب طبیعی است [۱۴]. الگوریتم ژنتیک یک نوع از الگوریتم‌های تکاملی است که در مسائل بهینه‌سازی برای یافتن جواب تقریبی استفاده می‌شود. به همین ترتیب، در مسئله برنامه-ریزی مسیر، مسیر بر اساس واکنش محیط با اجتناب از موانع انتخاب می‌شود. از الگوریتم ژنتیک برای جلوگیری از مشکل بهینه محلی و مشکل‌های محاسباتی بالا استفاده می‌شود. الگوریتم‌های ژنتیک معمولاً به‌عنوان یک شبیه‌ساز کامپیوتر که در آن جمعیت یک نمونه انتزاعی (کروموزوم‌ها) از نامزدهای جواب یک مسئله بهینه‌سازی به جواب بهتری منجر شود، پیاده‌سازی می‌شوند. به طور سنتی جواب‌ها به شکل رشته‌هایی از ۰۰ و ۱ بودند، اما امروزه به گونه‌های دیگری هم پیاده‌سازی شده‌اند. فرضیه با جمعیتی کاملاً تصادفی منحصربه‌فرد آغاز می‌شود و در نسل‌ها ادامه می‌یابد. در هر نسل گنجایش تمام جمعیت ارزیابی می‌شود، چندین فرد منحصر در فرایندی تصادفی از نسل جاری بر اساس شایستگی‌ها انتخاب می‌شوند و برای شکل‌دادن نسل جدید، اصلاح می‌شوند (کسر یا دوباره ترکیب می‌شوند) و در تکرار بعدی الگوریتم به نسل جاری تبدیل می‌شود. معمولاً الگوریتم‌های ژنتیک یک عدد احتمال تقاطع دارند که بین ۰/۶ و ۱ است و احتمال به‌وجودآمدن فرزند را نشان می‌دهد. کروموزوم‌ها با این احتمال دوباره با هم ترکیب می‌شوند. اتصال ۲ کروموزوم، فرزند ایجاد می‌کند که به نسل بعدی اضافه می‌شوند. این کارها انجام می‌شوند تا این که کاندیدهای مناسبی برای جواب در نسل بعدی پیدا شوند. مرحله بعدی تغییردادن فرزندان جدید است. الگوریتم‌های ژنتیک یک احتمال تغییر کوچک و ثابت دارند. بر اساس این احتمال، کروموزوم های فرزند به طور تصادفی تغییر می‌کنند یا با جهش بیت‌ها در کروموزوم ساختمان داده جهش می‌یابند. این فرآیند باعث به‌وجودآمدن نسل جدیدی از کروموزوم هایی می‌شود که با نسل قبلی متفاوت است. کل فرآیند برای نسل بعدی هم تکرار می‌شود و جفت‌ها برای ترکیب انتخاب می‌شوند و جمعیت نسل سوم به وجود می‌آیند و این فرآیند تکرار می‌شود تا این که به آخرین مرحله برسیم [۱۴].

برنامه‌ریزی مسیر ربات بر مبنای الگوریتم ژنتیک: در این مطالعه الگوریتم ژنتیک با طول متغیر کروموزوم‌ها برای مسئله برنامه‌ریزی مسیر ربات معرفی می‌شود، این الگوریتم بر مبنای سیستم دسیمال و ساده‌سازی عملیات در عملگرهای انتخاب، تقاطع و جهش است. به‌علاوه می‌تواند طول کروموزوم‌های بهینه و زیر بهینه را بر طبق محیط داده شده بدون دادن طول دقیق کروموزوم بیابد. همان‌طور که مشاهده می‌شود، به‌دست‌آوردن جواب بهینه برای الگوریتم یک چالش بزرگ است، به این دلیل که طول اولیه کروموزوم به‌صورت احتمالی انتخاب می‌شود. اگر طول کروموزوم خیلی بزرگ باشد، میدان و زمان جستجو افزایش خواهد یافت. برای حل این مشکل، سه عملگر ساده‌سازی، بازبینی و جایگزینی به الگوریتم اضافه می‌شود که می‌تواند احتمال افتادن در بهینه محلی را نیز کاهش دهد. در الگوریتم بهبودیافته، یک تابع ارزیابی جامع استفاده شده است که نه‌تنها طول مسیر ربات، هموار بودن و امنیت مسیر به‌عنوان اهداف بهینه در نظر می‌گیرد، همچنین سازگاری الگوریتم را هم‌زمان افزایش می‌دهد. در این الگوریتم عملگر انتخاب مبنی بر انتخاب چرخ رولت و روش انتخاب نخبه است. عملگر تقاطع و عملگر جهش به ترتیب دونقطه‌ای و تک‌نقطه‌ای در نظر گرفته شده است [۱۶].

۴-۳. الگوریتم غزال کوهستان[3]

الگوریتم بهینه‌سازی غزال کوهستان، یک الگوریتم فراابتکاری مبتنی بر رفتار غزال‌های کوهستان است. این الگوریتم در سال ۲۰۲۲ توسط عبدالله‌زاده و همکاران ارائه شده است [۱۷]. بهبودهای این الگوریتم توسط محققان در سال ۲۰۲۳ نیز صورت‌گرفته است که اهمیت این الگوریتم را بیش‌ازپیش مشخص می‌سازد [۱۸-۱۹]. الگوریتم غزال کوهستان از رفتارهای اجتماعی و زندگی غزال‌های کوهستان الهام‌گرفته است و با استفاده از چهار عامل اصلی، گله‌های نر مجرد، گله‌های زایمان، نرهای منفرد منطقه و مهاجرین برای جستجوی غذا بهینه‌سازی را انجام می‌دهد. در الگوریتم

[1] Roadmap
[2] Genetic algorithm
[3] Mountain gazelle algorithm

غزال‌های کوهستان هر غزال می‌تواند در طول عملیات بهینه‌سازی عضو یکی از گله‌های زایمان، نر مجرد یا نرهای منفرد منطقه‌ای شود. بهترین جواب سراسری در الگوریتم غزال‌های کوهستان غزال نر بالغ در قلمرو گله است. مؤلفه‌های اصلی الگوریتم غزال کوهستان به شرح زیر است:

❖ گله نرهای منفرد منطقه[1]:

این گله‌ها یک قلمرو منفرد ایجاد می‌کنند و نبرد بین غزال‌های نر بالغ بر سر قلمرو یا مالکیت ماده صورت می‌گیرد که به‌صورت ریاضی مدل‌سازی شده است.

$$\mathrm{TSM} = \mathrm{male_{gazelle}} - |(\mathrm{ri_1} \times \mathrm{BH} - \mathrm{ri_2} \times \mathrm{X(t)}) \times \mathrm{F}| \times \mathrm{Cof_r}$$

$\mathrm{male_{gazelle}}$ بردار موقعیت از بهترین راه حل کلی و پارامترهای $\mathrm{ri_2}$ و $\mathrm{ri_1}$ به صورت اعداد صحیح تصادفی ۱ یا ۲ هستند.

BH بردار ضریب گله نرجوان، با استفاده از معادله زیر محاسبه می‌شود.

$\mathrm{BH} = \mathrm{x_{ra}} \times \lfloor \mathrm{r1} \rfloor \times \mathrm{M_{pr}} \times \lceil \mathrm{r2} \rceil, \mathrm{ra} = \left\{ \left\lceil \frac{\mathrm{N}}{3} \right\rceil \dots \mathrm{N} \right\}$ $\mathrm{x_{ra}}$ یک جواب تصادفی (غزال نر جوان) در فاصله ra است. $\mathrm{M_{pr}}$ که میانگین تعداد عوامل جستجو $\left\lceil \frac{\mathrm{N}}{3} \right\rceil$ است که به صورت تصادفی انتخاب می‌شود، همچنین N تعداد کل غزال ها است و $\mathrm{r_2}$ و $\mathrm{r_1}$ مقادیر تصادفی بین ۰ و ۱ هستند. F نیز با استفاده از فرمول زیر استفاده می‌شود.

$F = N_1(D) \times \exp\left(2 - Iter \times \left(\frac{MaxIter}{2}\right)\right)$ در ابعاد موضوع، N_1 یک عدد تصادفی از توزیع استاندارد است، تابع نمایی نیز به عنوان *exp* شناخته می‌شود.

$\mathrm{Cof_r}$ یک بردار ضریب انتخابی تصادفی است که در هر تکرار به روز می شود و برای افزایش قابلیت جستجو استفاده می‌شود با استفاده از معادله زیر محاسبه می‌شود.

$$\mathrm{Cof_r} = \begin{cases} (\mathrm{a}+1) + \mathrm{r_3}, \\ \mathrm{a} \times \mathrm{N_2(D)}, \\ \mathrm{N_4(D)}, \\ \mathrm{N_3(D)} \times \mathrm{N_4(D)^2} \times \mathrm{COS}((\mathrm{r_4} \times 2) \times \mathrm{N_3(D)}) \end{cases} \quad (4)$$

$\mathrm{N_2}$، $\mathrm{N_3}$، $\mathrm{N_4}$ اعداد تصادفی در محدوده نرمال و ابعاد مسئله هستند. در بعد مسئله، $\mathrm{r_3}$ و $\mathrm{r_4}$ نیز یک عدد تصادفی در فیلد ۰ و ۱ است. در نهایت، cos تابع کسینوس را نشان می‌دهد.

$MaxIter$ تعداد کل تکرارها و Iter تعداد تکرارهای فعلی است و a با استفاده از معادله زیر محاسبه می‌شود.

$$a = -1 + Iter \times \left(\frac{-1}{MaxIter}\right) \quad (5)$$

❖ گله‌های زایمان[2]: این گله‌ها غزال‌های نر قوی را به دنیا می‌آورند که به‌صورت ریاضی مدل‌سازی شده است:

$$\mathrm{MH} = \left(\mathrm{BH} + \mathrm{Cof_{1,r}}\right) + \left(\mathrm{ri_3} \times \mathrm{male_{gazelle}} - \mathrm{ri_2} \times \mathrm{X}_{rand}\right) \times \mathrm{Cof_{1,r}} \quad (6)$$

$\mathrm{ri_3}$و $\mathrm{ri_4}$ اعداد صحیح و تصادفی ۱ یا ۲ هستند. غزال نر بهترین راه حل در تکرار فعلی است. در نهایت، Xrand برداری که موقعیت یک غزال که به طور تصادفی از کل جمعیت انتخاب شده است.

❖ گله نر مجرد[3]:

غزال‌های نر جوان بر سر قلمرو و کنترل غزال‌های ماده وارد نبردهای سختی با غزال‌های نر می‌شوند که به‌صورت ریاضی مدل‌سازی شده است:

$$\mathrm{BMH} = (\mathrm{X(t)} - \mathrm{D}) + \left(\mathrm{ri_5} \times \mathrm{male_{gazelle}} - \mathrm{ri_6} \times \mathrm{BH}\right) \times \mathrm{Cof_r} \quad (7)$$

$\mathrm{X(t)}$ موقعیت بردار غزال در تکرار جاری است و $\mathrm{ri_5}$ و $\mathrm{ri_6}$ اعداد صحیح ۱ یا ۲ هستند که به طور تصادفی انتخاب می‌شوند. غزال نر بردار موقعیت بهترین راه حل است و $\mathrm{Cof_r}$ یک ضریب تصادفی انتخاب شده است که به عنوان بردار محاسبه و استفاده می‌شود.

D با استفاده از معادله زیر محاسبه می‌شود:

$$\mathrm{D} = \left(|\mathrm{X(t)}| + \left|\left|\mathrm{male_{gazelle}}\right|\right|\right) \times (2 \times -\mathrm{r_6} - 1) \quad (8)$$

$\mathrm{r_6}$ نیز یک عدد تصادفی بین ۰ و ۱ است.

❖ مهاجرین برای تأمین غذا[4] :

مهاجرین مسافت‌های طولانی را برای به‌دست‌آوردن غذا طی می‌کنند و سرعت دویدن بالا و قدرت پرش خوبی دارند که به‌صورت ریاضی مدل‌سازی می‌شود:

$$\mathrm{MSF} = (\mathrm{ub} - \mathrm{lb}) \times \mathrm{r_7} - \mathrm{lb}) \quad (9)$$

Ub و lb به ترتیب حد بالایی و پایینی مسئله هستند. در نهایت، $\mathrm{r_7}$ یک عدد صحیح بین ۰ و ۱ است که به طور تصادفی انتخاب شده است.

[1] Territorial Solitary Males

[2] Maternity Herds
[3] Bachelor Male Herds
[4] Migration to Search for Food

الگوریتم غزال کوهستان بهبودیافته برای حل مسئله برنامه‌ریزی مسیر

۱- ورودی N : اندازه جمعیت و T حداکثر تعداد تکرار.
خروجی‌ها: موقعیت بهترین غزال و بهترین مقدار تابع برازش.

۲- مقداردهی اولیه :
-جمعیت اولیه xi ، I={1,...,N} تولید شود.
-ترمیم گره
-ترمیم پاره‌خط
-بهبود گره
مراحل زیر را تا رسیدن به‌شرط توقف انجام دهید:
(۱-۳): برای هر غزال موجود در جمعیت مراحل زیر را انجام دهید:
الف) گله‌های نر منفرد منطقه را طبق معادله (۱) محاسبه کنید.
ب) گله‌های زایمان را طبق معادله (۶) محاسبه کنید. ج) گله‌های نر مجرد را طبق معادله (۷) محاسبه کنید.
د) مهاجرین برای جستجوی غذا را طبق معادله (۹) محاسبه کنید.
-ساده‌سازی
-بازبینی
۳-جایگزینی
۳-۱) مقدار تابع برازش را برای BMH، TSM،MH و MSF محاسبه کنید، سپس آنها را به زیستگاه اضافه کنید.
۳-۲): کل جمعیت را به ترتیب صعودی مرتب کنید. ۳-۳): موقعیت بهترین غزال را به‌روزرسانی کنید.

۳-۴): N غزال برتر را ذخیره کنید.

۴- با پایان الگوریتم بهترین مقدار تابع برازش را برگردانید.

در ادامه به توضیح الگوریتم کپک مخاطی پرداخته می‌شود.

۳-۲-۳. الگوریتم کپک مخاطی[1]

بهینه‌ساز کپک مخاطی یکی از روش‌های فراابتکاری که در سال ۲۰۲۰ معرفی شده است [۲۰]. این روش یک بهینه‌ساز قدرتمند مبتنی بر جمعیت که از رفتار هوشمندانه یک موجود تک‌سلولی بنام کپک مخاطی الهام‌گرفته و بر اساس حالت رفتار نوسانی این موجودات در طبیعت ارائه شده است. موجودات تک‌سلولی در این کپک، می‌توانند با استفاده از روش‌های هوشمندانه مسیرهای پر پیچ‌وخم را طی نموده و تا بتوانند راه خود را به مقصدشان پیدا کنند. باتوجه‌به این که اغلب، الگوریتم کپک مخاطی عملکردی رقابتی و برتر در مقایسه با الگوریتم‌های مختلف را ارائه می‌دهد، اخیراً موردتوجه بسیاری از محققان قرار گرفته شده است. یک الگوریتم کپک مخاطی برای مسئله برنامه‌ریزی مسیر ربات معرفی می‌شود، این الگوریتم بر مبنای سیستم دسیمال و ساده‌سازی عملیات در عملگرهای نزدیک‌شدن به غذا، بسته‌بندی غذا و نوسان است. به‌علاوه می‌تواند طول کپک‌های بهینه زیر بهینه را بر طبق محیط داده شده بدون دادن طول دقیق کپک‌ها بیابد. همان‌طور که مشاهده می‌شود، به‌دست‌آوردن جواب بهینه برای الگوریتم یک چالش بزرگ است به این دلیل که طول اولیه کپک‌ها به‌صورت احتمالی انتخاب می‌شود. اگر طول کپک‌ها خیلی بزرگ باشد، میدان و زمان جستجو افزایش خواهد یافت. برای حل این مشکل، سه عملگر ساده‌سازی، بازبینی و جایگزینی به الگوریتم اضافه می‌شود که می‌تواند احتمال افتادن در بهینه محلی را نیز کاهش دهد [۲۰]. در این الگوریتم، از یک تابع ارزیابی جامع استفاده شده است که نه‌تنها طول مسیر ربات، هموار بودن و امنیت مسیر به‌عنوان اهداف بهینه در نظر می‌گیرد، همچنین سازگاری الگوریتم را هم‌زمان افزایش می‌دهد. در ادامه برخی از عملکردهای کپک به‌صورت ریاضی مدل‌سازی شده است:

❖ نزدیک‌شدن به غذا:

$$X_{t+1} = \begin{cases} x_b(t) + v_b.(w.x_a(t) - x_B(t)), & r < p \\ v_c.x_t, & r \geq p \end{cases} \quad (10)$$

در اینجا t تکرار فعلی، X نشان‌دهنده محل کپک، W وزن کپک، X_A و X_B نشان دهنده دو مکان است که به طور تصادفی انتخاب شده اند، X_b نشان دهنده مکانی با بالاترین غلظت بو ویک پارامتر در محدوده [-a, a] و V_c یک پارامتر که از یک به صفربصورت خطی کاهش می یابد. متغیر p به صورت زیر بدست می‌آید:

$$p = \tanh|s(i) - DF| \quad (11)$$

S(i)معرف تناسب X و DF معرف بهترین تناسب به‌دست‌آمده در همه تکرارها است.

V_bبه صورت زیر بدست می‌آید:

$$V_b = [-a, a]$$
$$a = tanh^{-1}\left(-\left(\frac{t}{max_t}\right) + 1\right) \quad (12)$$

و وزن کپک از رابطه زیر محاسبه می‌گردد:

$$w(smellindex(i)) = \begin{cases} 1 + rlog\left(\frac{b_f - s(i)}{b_f - w_f}\right) + 1, & condition \\ 1 - rlog\left(\frac{b_f - s(i)}{b_f - w_f}\right) + 1, & others \end{cases} \quad (13)$$
$$smellindex = sort(s)$$

[1] Slime Mould Algorithms

در اینجا r یک مقدار تصادفی است که در بازه [1, 0] انتخاب شده است، condition نشان می‌دهد که S(i) دررتبه نیمه اول جمعیت قرار دارد، مقادیر تناسب مرتب می‌شوند و در متغیر $smellindex$ قرار می‌گیرند. b_f نشان‌دهنده تناسب بهینه به‌دست آمده در فرآیند تکراری فعلی است. بدترین مقدار تناسب در فرآیند تکراری با w_f نشان داده شده است.

❖ بسته‌بندی غذا

مکان کپک با فرمول زیر به‌روز می‌شود:

$$X^* = \begin{cases} rand(UB - LB) + LB, & rand < \\ x_b(t) + v_b\big(w.x_A(t) - x_B(t)\big), & r < p \\ v_c.X(t), & r \geq p \end{cases} \quad (۱۴)$$

در اینجا r یک مقدار تصادفی LB و UB به ترتیب نشان‌دهنده مرزهای پایین و بالایی محدوده جستجو هستند.

❖ نوسان

مقدار V_b در انتهای الگوریتم به صفر تمایل دارد و به طور تصادفی بین [a- , a] در نوسان است. مقدار V_c به تدریج با افزایش تکرارها و نوسان به صفر نزدیک می‌شود.

۱-۳-۲-۳. الگوریتم کپک مخاطی بهبودیافته برای حل مسئله برنامه‌ریزی مسیر

ورودی: N اندازه جمعیت اولیه و T حداکثر دفعات تکرار
خروجی: بهترین موقعیت و مقدار تابع مقصد
۱- مقداردهی اولیه: بهترین موقعیت‌ها و مقدار تابع مقصد برابر با صفر و مقدار تابع برازندگی برابر با بی نهایت و موقعیت کپک ها به طور تصادفی مقداردهی می‌شود.

-ترمیم گره

-ترمیم پاره‌خط

-بهبود گره

۲- تا زمان رسیدن به حداکثر تکرار، مراحل زیر باید انجام شود: بررسی شود، که موقعیت ها خارج از فضای جستجو نباشند. چنانچه جواب این شرایط را نداشت باید در محدوده فضای جستجو قرار داده شود.

۳- تابع برازندگی موقعیت کپک ها، که در مرحله قبل مقداردهی شده ، محاسبه شود.
۴- مقادیر توابع برازندگی به دست آمده از مرحله قبل باید مرتب سازی شود.
۵- وزن توابع برازندگی به دست آمده محاسبه شود.
۶- بهترین مقدار تابع برازندگی با مقدار تابع مقصد مقایسه شود چنانچه بهترین مقدار تابع برازندگی کمتر از مقدار تابع مقصد باشد، مقدار تابع مقصد را برابر با بهترین مقدار تابع برازندگی قرار دهید

۷- برای جلوگیری از گیر افتادن در بهینه محلی، موقعیت کپک‌ها، با استفاده از یک عدد تصادفی به روز خواهد شد.
۸- در صورتی که، شرط پایان برآورده شود الگوریتم خاتمه می‌یابد در غیر این صورت، برای تکرار بعدی به مرحله (۲) خواهدرفت.
۹- پایان

۴. تابع ارزیابی

انتخاب تابع ارزیابی یک گام بسیار مهم در الگوریتم‌های فراابتکاری است. مقدار برازش هر مسیر از جمعیت تعیین می‌کند که آیا مسیر در تکرار بعد باقی می‌ماند و یا حذف می‌شود. در تعیین تابع ارزیابی سه هدف کوتاه‌ترین مسیر، همواری و امنیت مسیر در نظر گرفته شده است. دو نوع تابع ارزیابی در این بخش استفاده شده است [۱۶]. تابع ارزیابی برای مسیر شدنی و مسیر نشدنی که به ترتیب آن‌ها را با $f_f(p)$ و $f_i(p)$ نمایش می‌دهیم. در تابع ارزیابی برای یک مسیر شدنی p طول مسیر با dis(p) هموار بودن مسیر را با smo(p) و فاصله امن برای مسیر با saf(p) نمایش داده می‌شود. تابع ارزیابی یک مسیر شدنی p به صورت زیر تعریف می‌شود:

$$\mathrm{f_f(p)} = \frac{1}{\mathrm{w_d}.\,\mathrm{dis(p)} + \mathrm{w_h}.\,\mathrm{saf(p)}} + \mathrm{w_s}.\,\mathrm{smo(p)} \quad (۱۵)$$

در فرمول فوق wh، ws و wd به ترتیب وزن‌های saf(p) ، smo(p)، و dis(p) هستند. dis(p) به‌صورت زیر تعریف می‌شود:

$$dis(p) = \sum_{i=1}^{N-1} \sqrt{(a_{i+1} - a_i)^2 + (b_{i+1} - b_i)^2} \quad (۱۶)$$

که در آن N طول مسیر است و a_i, b_i مختصات گره i را در محیط نشان می‌دهد. smo(p) به صورت زیر تعریف می‌شود:

$$smo(p) = \sum_{i}^{N-1} \beta(l_i + l_i + 1) \quad (۱۷)$$

که در آن زاویه بین خط $\beta(l_i + l_i + 1)$ در مسیر است. saf(p) به صورت زیر تعریف می‌شود:

$$saf(p) = \sum_{i}^{N-1} \alpha(l_i) \quad (۱۸)$$

که در آن $\alpha(l_i)$ نزدیکترین فاصله خط l_i با موانع است. در تابع ارزیابی برای یک مسیر نشدنی، p طول مسیر dis(p) نسبت

طول مسیری که از موانع عبور می‌کند به طول کل مسیر cro(p) و نسبت خط های نشدنی مسیر به تعداد کل پاره‌خط‌های مسیر lin(p) استفاده می شود، که می‌توان آن را به صورت زیر تعریف کرد:

$$f_i(p) = \frac{1}{\frac{dis(p)}{\max[dis(p)]} + w_l.lin(p) + w_c.cro(p)} \quad (۱۹)$$

که در آن w_c و w_l به ترتیب وزن‌های cro(p) و lin(p) هستند. lin(p) به صورت زیر تعریف می‌شود:

$$lin(p) = \frac{lin_{feasible}}{l_{total}} \quad (۲۰)$$

که در آن $lin_{feasible}$ تعداد خط های نشدنی و l_{total} تعداد کل خط های مسیر است. cro(p) به صورت زیر تعریف می‌شود:

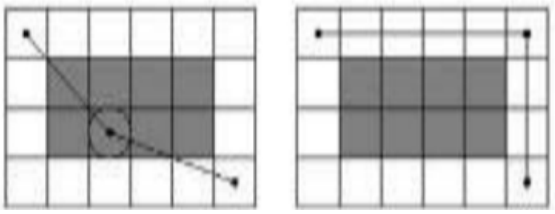
$$cro(p) = \sum_{i}^{N-1} \frac{obs(l_i)}{dis(p)} \quad (۲۱)$$

که در آن obs(l_i) طول موانعی است که مسیر از آن ها عبور می‌کند.

۵. عملگرهای پیشنهادی

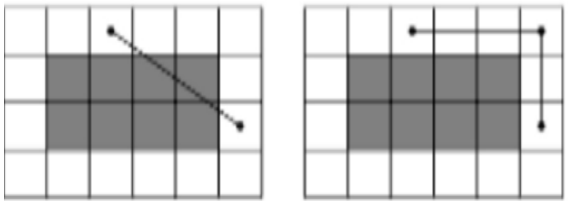
در الگوریتم‌های پیشنهادی فراابتکاری جمعیت اولیه به‌صورت کاملاً تصادفی انتخاب می‌شوند. در الگوریتم‌های پیشنهادی برای برنامه‌ریزی مسیر ربات، یک مسیر از تعدادی گره (سلول‌های شبکه) تشکیل می‌شود که از آن‌ها عبور می‌کند. طول مسیرها در مسئله متغیر است، به این دلیل که طول یک مسیر شدنی ممکن است متفاوت باشد و طول مسیر بهینه (به‌اصطلاح تعداد گره‌ها) نامعلوم است، بنابراین مسیرهای جمعیت‌های مختلف، ممکن است طول‌های متفاوتی داشته باشند. هر مسیر در یک جمعیت باید تاحدامکان یک مسیر شدنی در محدوده محیط باشد و از موانع عبور نکند. تولید جمعیت اولیه با ایجاد یک مسیر که نقطه شروع و پایان آن مشخص است، شروع می‌شود. در الگوریتم‌های پیشنهادی پس از ایجاد تعداد مشخص جمعیت به‌صورت تصادفی، از سه عملگر جدید که بر مبنای اطلاعات موجود در هر مسیر است برای ایجاد مسیر بهتر استفاده می‌شود. باتوجه‌به این که مسیر شدنی یا نشدنی باشد می‌توان سه عملگری که در ادامه تعریف خواهیم کرد در جهت بهبود و یا شدنی نمودن مسیر اعمال کرد که برای رسیدن به جواب نزدیک به بهینه و یا بهینه از تعداد تکرار بسیار کمتری نسبت به الگوریتم بهبودیافته ژنتیک استفاده شده است. بعد از اینکه جمعیت اولیه تولید شد، در دو حالت مسیر نشدنی به وجود می‌آید: ۱- گره‌های میانی از سلول‌هایی که مانع در آن‌ها قرار دارند، انتخاب شوند. ۲- پاره‌خط‌های مسیر از موانع عبور کنند. در این دو حالت از دو عملگر ترمیم گره و ترمیم پاره‌خط استفاده می‌شود. اگر مسیر شدنی باشد از عملگر بهبود گره برای ایجاد مسیر بهتر استفاده می‌شود. در ادامه به معرفی این سه عملگر می‌پردازیم.

ترمیم – گره: ترمیم - گره برای انتقال یک گره که بر روی مانع قرار گرفته، به یک سلول آزاد حول مانع استفاده می‌شود. برای پیداکردن بهترین سلول، یک جستجوی محلی کوچک در مجاورت مانع اعمال می‌شود (شکل ۱).



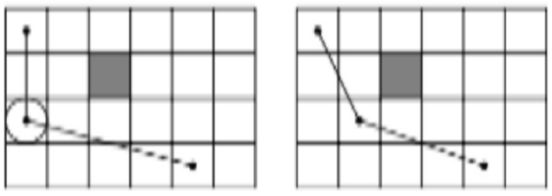
شکل ۱. تبدیل یک مسیر نشدنی به مسیر شدنی با عملگر ترمیم - گره

- **ترمیم – پاره‌خط:** برای ترمیم یک پاره‌خط نشدنی با قراردادن یک گره مناسب بین دو گره، پاره‌خط نشدنی را به دو پاره‌خط شدنی تغییر می‌دهد. مانند عملگر ترمیم گره برای یافتن بهترین گره، جستجوی محلی مشابهی در تمام سلول‌های مجاور مانع اعمال می‌شود، پس از یافتن گره مناسب پاره‌خط نشدنی حذف و دو پاره‌خط شدنی به مسیر اضافه می‌شود (شکل ۲)..



شکل ۲. تبدیل یک مسیر نشدنی به مسیر شدنی با عملگر ترمیم - پاره‌خط

- **بهبود – گره:** بهبود - گره برای مسیرهای شدنی طراحی شده است. عملگر یک گره را انتخاب کرده، یک جستجوی محلی را در شبکه‌های مجاور گره انجام می‌دهد و به بهترین سلول حرکت می‌کند (شکل ۳)..



شکل ۳. کوتاه‌شدن یک مسیر شدنی با حذف‌شدن گره با عملگر بهبود - گره

در ادامه استفاده از سه عملگر دیگر برای جلوگیری از افتادن در بهینه محلی می‌پردازیم.

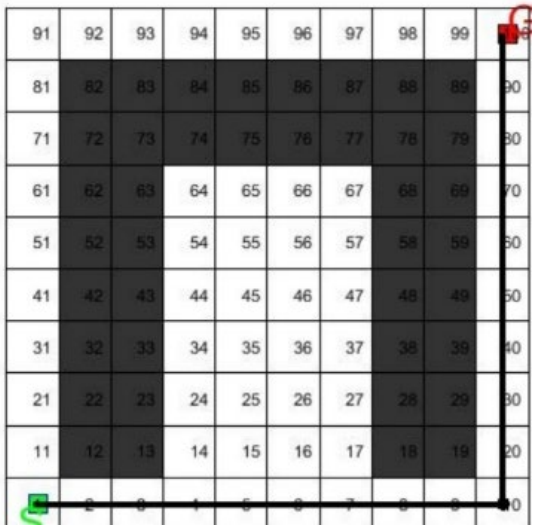
- **ساده‌سازی مسیر:** ساده‌سازی مسیر برای هر دو پاره‌خط شدنی و نشدنی استفاده می‌شود. مانند عملگر ساده‌سازی در بخش قبل، طول مسیر را با حذف گره‌های مشابه کاهش می‌دهد. اگر پاره‌خطی که دو گره pi و pi+۲ را به هم وصل می‌کند از مانعی نگذرد، گره pi+۱ را حذف می‌کند.
- **عملگر بازبینی:** باتوجه‌به ایجاد جمعیت اولیه در الگوریتم پیشنهادی که گره‌های مسیر در سلول‌های آزاد قرار می‌گیرند و تاحدامکان از پاره‌خط‌های نشدنی جلوگیری می‌کند، استفاده از این عملگر را کاهش می‌دهد. بااین‌وجود ممکن است در عملگرهای تقاطع و جهش مسیر نشدنی به دست آید. برای حل این مشکل از عملگر بازبینی استفاده می‌شود.
- **عملگر جایگزینی:** برای تضمین این که تمام مسیرهای هر نسل مشابه نباشند، لازم است تا درصدی از مسیرهای احتمالی تولید شده را به جمعیت اضافه کنیم؛ بنابراین یک عملگر جایگزینی در این الگوریتم استفاده شده است که می‌تواند احتمال همگرایی را کاهش دهد.

۶. آزمایشات تجربی

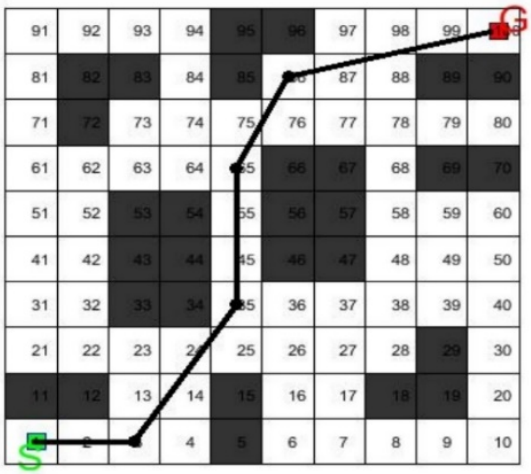
در این بخش برخی از مطالعات انجام شده برای الگوریتم‌های فراابتکاری آورده شده است. هر سه الگوریتم در محیط‌های مشابهی، با نرم‌افزار متلب پیاده‌سازی و اجرا شده‌اند، تا کارایی (سازگاری و شدنی بودن) الگوریتم‌ها نشان داده شود. در این پیاده‌سازی، الگوریتم‌ها را برای محیط‌های مختلف با موانع متفاوت اجرا می‌کنیم. فرضیات مسئله به این صورت می‌باشد که ربات به‌صورت نقطه‌ای و ربات صلب بوده و تغییر شکل نمی‌دهد. فضا به‌صورت گسسته و محیط فاقد تغییرات زمانی می‌باشد، یعنی موانع دارای حرکت نبوده و ایستا هستند و همچنین موانع موجود در محیط می‌توانند به هر شکلی باشند و ربات باید مسیر خود را در فضای آزاد از موانع طی نماید و محیط شناخته شده است که در محیط‌های ساده و محیط‌های پیچیده با تعداد تکرار مشخص شده در جدول (۱)، الگوریتم‌ها به‌راحتی می‌توانند به یک جواب شدنی نزدیک به بهینه و حتی به جواب بهینه دست یابند. شکل (۴) آزمون را برای محیط‌های ساده برای الگوریتم ژنتیک نمایش می‌دهند. شکل (۵) نشانگر مسیری بهینه برای الگوریتم ژنتیک در محیط پیچیده است. شکل (۶) مسیری بهینه برای الگوریتم غزال‌های کوهستان و شکل (۷) مسیری بهینه برای کپک مخاطی را نشان می‌دهند. الگوریتم بهبودیافته ژنتیک با تعداد جمعیت ۸۰ و تعداد تکرار ۳۰۰ مسیر با طول ۱۵/۱۸۴ را مشخص نمود، درحالی‌که با الگوریتم پیشنهادی ژنتیک با تعداد جمعیت ۳۰ و تعداد ۳۰ تکرار، مسیری با طول ۱۴/۳۰۶ را مشخص نمود. همچنین طبق اشکال (۶ و ۷)، الگوریتم‌های غزال کوهستان و کپک مخاطی با تعداد جمعیت ۳۰ و تعداد تکرار ۵ به ترتیب مسیرهای با طول ۱۵/۳۱۹ و ۱۴/۴۸۲ را به دست آوردند.

جدول ۱. نتایج پیاده سازی الگوریتم‌ها

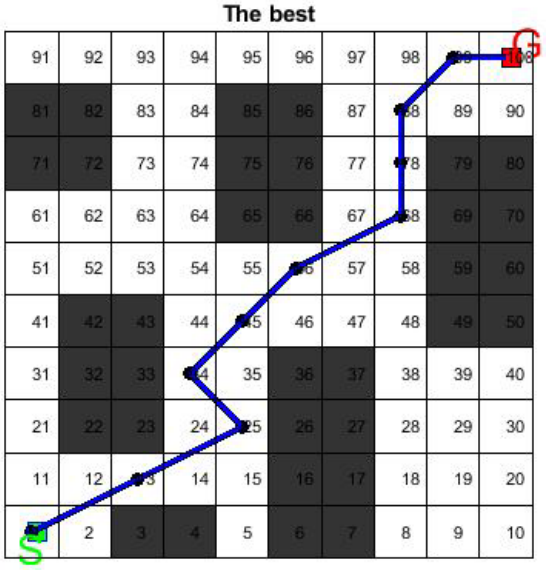
الگوریتم /پارامتر	تعداد جمعیت	تکرار	طول
بهبودیافته ژنتیک	۸۰	۳۰۰	۱۵/۱۸۴
پیشنهادی ژنتیک	۳۰	۳۰	۱۴/۳۰۶
غزال کوهستان	۳۰	۵	۱۵/۳۱۹
کپک مخاطی	۳۰	۵	۱۴/۴۸۲



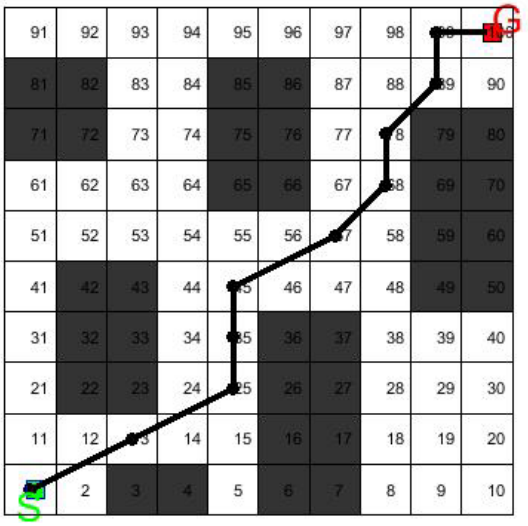
شکل ۴. مسیر بهینه در محیط ساده با استفاده از الگوریتم ژنتیک



شکل ۵. مسیر بهینه در محیط پیچیده با استفاده از الگوریتم ژنتیک



شکل ۶. مسیر بهینه با استفاده از الگوریتم غزال کوهستان



شکل ۷. مسیر بهینه با استفاده از الگوریتم کپک مخاطی

۷. نتیجه گیری

در این تحقیق، مسئله برنامه‌ریزی مسیر علاوه بر الگوریتم ژنتیک با دو الگوریتم جدید غزال کوهستان و کپک مخاطی پیاده‌سازی گردید و کارایی این دو الگوریتم موردمطالعه قرار گرفت. در این مطالعه حل مسئله طراحی مسیر برای یک ربات در محیط گسسته (به‌صورت شبکه)، ایستا و شناخته شده با تعیین مسیر بدون برخورد مورد بررسی قرار گرفت که معیارهای انتخاب شده برای کوتاه‌ترین فاصله، امنیت مسیر و هموارسازی مسیر را برآورده می‌سازد. بررسی مسئله برنامه‌ریزی مسیر برای یک ربات با سه الگوریتم فراابتکاری انجام گردید. در این الگوریتم‌ها یک تابع ارزیابی جدید و سه عملگر جدید شامل عملگر ساده‌سازی، عملگر بازبینی و عملگر جایگزینی پیشنهاد شده‌اند. همچنین در این الگوریتم‌ها برای تولید جمعیت اولیه از سه عملگر ترمیم گره، ترمیم پاره‌خط و بهبود گره استفاده گردید. این سه عملگر به تولید مسیرهای شدنی کمک می‌کنند. باتوجه‌به اینکه دو الگوریتم غزال کوهستان و کپک مخاطی دارای پیچیدگی محاسباتی تقریباً یکسان هستند از مقایسه این دو الگوریتم از نظر پیچیدگی محاسباتی صرف‌نظر گردید و صرفاً تولید جواب شدنی باتوجه‌به تعداد جمعیت، تعداد تکرار و طول مسیر، موردنظر قرار گرفت.

نتایج به‌دست‌آمده در شکل‌های (۵ تا ۷) نشان داده شده است. همچنین تعداد جمعیت یکسان با تعداد تکرار و طول مسیر بدست آمده متفاوت الگوریتم ها، تفاوت آنها را مشخص می کند. نتایج نشان می‌دهد که در محیط های ساده این الگوریتم ها به راحتی می توانند جواب بهینه را بیابند. برای مثال در محیط های ساده در حداکثر ۳۰ تکرار به یک جواب شدنی نزدیک بهینه و حتی به جواب بهینه دست می یابند.

الگوریتم کپک مخاطی و غزال کوهستان در صورت گیرافتادن در بهینه محلی نسبت به الگوریتم ژنتیک راه‌حل سریع‌تر و بهتری ارائه می‌کنند که به جواب شدنی نزدیک‌تر است. نتایج حاکی از برتری نسبی الگوریتم پیشنهادی ژنتیک و سپس الگوریتم کپک مخاطی نسبت به دو الگوریتم دیگر دارد. این مطلب کارایی بالای الگوریتم‌های پیشنهادی ژنتیک، کپک مخاطی و غزال کوهستان را نشان می‌دهد. کار آینده بررسی مسئله برنامه‌ریزی مسیر برای ربات‌ها در محیط پویا است.

۸. مراجع

[1] Canny, J.; Reif, J. “New lower bound techniques for robot motion planning problems”; Int. J. Found. Comput. Sci. 28th Annual Symp. on, 1987, 49–60. DOI: 10.1109/SFCS.1987.42

[2] Nearchou, A. C. “Path planning of a mobile robot using genetic heuristics”; Cambridge University Press, 1998, 16, 575–588. DOI: 10.1017/S0263574798000289

[3] Hussein, A.; Mostafa, H.; Badrel-din, M.; Sultan, O.; Khamis, A.; “Metaheuristic optimization approach to mobile robot path planning”; Int. Conf. on, 2012, 1–6. DOI: 10.1109/ICEngTechnol.2012.6396150

[4] Choset, H.; Lynch, K.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.; Thrun, S. “Principles of robot motion: Theory, algorithms, and implementation errata”; 2007, 1–150.

[5] Berg, M. d.; Cheong, O.; Kreveld, M.v.; Overmars, M. “Computational geometry: algorithms and applications”; Springer-Verlag TELOS, 2008. DOI: 10.1007/978-3-540-77974-2

[6] Latombe, J.-C. “Robot motion planning”; Springer Science & Business Media, 2012, 124. DOI: 10.1007/978-1-4615-4022-9

[7] LaValle, S. M. “Planning algorithms”; Cambridge university press, 2006.

[8] Van, J. P.; Berg, D. “Path planning in dynamic environments”; PhD thesis, Utrecht University, 2007.

[9] Liao, W. “Genetic Algorithm based robot path planning”; In Intelligent Computation Technology and Automation, Int. Conf. on, 2008, 1, 56–59. DOI: 10.1109/ICICTA.2008.216

[10] Leena, N.; Saju, K. “A survey on path planning techniques for autonomous mobile robots”; IOSR J. of Mechanical and Civil Eng. 2014, 8, 76–79.

[11] Victerpaul, P.; Saravanan, D.; Janakiraman, S.; Pradeep, J. “Path planning of autonomous mobile robots: A survey and comparison”; Journal of Advanced Research in Dynamical and Control Systems, 2017, 9, 1535–1565.

[12] Svestka, P.; Latombe, J.; Overmars Kavraki, L. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”; IEEE T. Robotics Autom. 1996, 12, 566–580. DOI: 10.1109/70.508439

[13] LaValle, S. M. “Rapidly-exploring random trees: A new tool for path planning. 1998”, 1998.

[14] Ismail, A.; Sheta, A.; Al-Weshah, M. “A mobile robot path planning using genetic algorithm in static environment”; Journal of Computer Science, 2008, 4, 341–344.

[15] Buckley, S. J. “Fast motion planning for multiple moving robots”; IEEE Int. Conf. Robot. Autom. 1989, 322–326.

[16] Ni, J., Wang, K., Huang, H., Wu, L. and Luo, C., 2016, August. Robot path planning based on an improved genetic algorithm with variable length chromosome. In 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD) (pp. 145-149). IEEE.

[17] Abdollahzadeh, B.; Gharehchopogh, F. S.; Khodadadi, N.; Mirjalili, S. “Mountain gazelle optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems”; Adv. Eng. Softw. 2022, 174, 103282. DOI :10.1016/j.advengsoft.2022.103282

[18] Seini, T., Yussif, A.S. and Katali, I.M. Enhancing Mountain Gazelle Optimizer (MGO) with an Improved F-Parameter for Global Optimization. Int. Res. J. Eng. Technol, 2023, 10(6), pp.921-930.

[19] Sarangi, P. and Mohapatra, P.,. Evolved opposition-based mountain gazelle optimizer to solve optimization problems. Journal of King Saud University-Computer and Information Sciences, 2023, 35(10), p.101812.

[20] Houssein, E. H.; Mahdy, M. A.; Shebl, D.; Manzoor, A.; Sarkar, R.; Mohamed, W.M. “An effcient slime mould algorithm for solving multi-objective optimization problems”; Expert Syst.- Appl. 2021, 187, 115870.DOI: 10.1016/j.eswa.2021.115870