International Journal of Industrial Engineering & Production Management (2011)

September 2011, Volume 22, Number 2
pp. 99-111

http://IJIEPM.iust.ac.ir/

Mobile Robot Path Planning Using a New GA-based Method with Variable Chromosome Length

Ellips Masehian*, Farnaz Barzinpour & Samira Saedi

** **Ellips Masehian**, Assistant Professor, Industrial Engineering Department, Tarbiat Modares University*
***Farnaz Barzinpour**, Assistant Professor, School of Industrial Engineering, Iran University of Science and Technology*
***Samira Saedi**, M.Sc. graduate, School of Industrial Engineering, Iran University of Science and Technology*

Keywords

Robot Motion Planning,
Genetic Algorithms,
Delaunay Triangulation,
Rapidly-exploring Random Trees

ABSTRACT

Being one of the major research fields in the robotics discipline, the robot motion planning problem deals with finding an obstacle-free start-to-goal path for a robot navigating among workspace obstacles. Such a problem is also encountered in path planning of intelligent vehicles and Automatic Guided Vehicles (AGVs). Traditional (exact) algorithms have failed to solve the problem effectively since it is proven that the complexity of the problem is NP-hard. Consequently, several heuristic algorithms have been developed for solving the problem, among which the Genetic Algorithms (GA) evolutionary approach has been increasingly utilized. In this paper, a new GA-based method has been developed to improve the drawbacks of current GA methods regarding fixed chromosome string size and limitations in the crossover and mutation operators. In this method, first the workspace is tessellated via the Delaunay triangulation, and a set of paths are created using the powerful Rapidly-exploring Random Trees (RRT) and smoothed. As a result, a population of variable-length chromosomes is obtained, which is a novelty in the field. Next, these solutions are improved further by using new selection, crossover and mutation operators, in which the triangulation plays a pivotal role. Extensive Simulations of the algorithm showed that the obtained path lengths are averagely 25% shorter than those found by the pure RRT method.

*
Corresponding author. Ellips Masehian
Email: *masehian@modares.ac.ir*

نشـــریه بین المللـــی مهنـــدسی صنـــایع و مدیـــــریت تولیـــد

شمـــــاره ۲، جلـــد ۲۲، تابستـــــان ۱۳۹۰
صفحـــــه ۱۰۰-۱۱۱

http://IJIEPM.iust.ac.ir/

نشریه بین المللی مهندسی صنایع و مدیریت تولید دانشگاه علم و صنعت ایران

ISSN: 2008-4870

مسیریابی روبات متحرک با استفاده از روشی جدید مبتنی بر الگوریتم ژنتیک با طول آرایهٔ متغیر

الیپس مسیحی*، فرناز برزین پور و سمیرا ساعدی

کلمات کلیدی

برنامه ریزی حرکت روبات،
الگوریتم ژنتیک،
مثلث بندی دِلونِی،
درخت کاوش تصادفی سریع

چکیده:

یکی از شاخه‌های مهم تحقیقاتی در علم روباتیک، برنامه‌ریزی حرکت روبات است که هدف آن یافتن مسیری بهینه از نقطه شروع به هدف و عاری از تصادم با موانع موجود در محیط روبات است. چنین مسئله ای در برنامه ریزی مسیر وسایل نقلیه هوشمند و AGV-ها نیز مطرح می شود. از آنجا که ثابت شده است این مسئله از نوع NP-Hard می‌باشد، الگوریتم‌های بهینه سازی سنتی موفقیت زیادی در حل آن کسب نکرده‌اند. در نتیجه، برای حل این مسائل روش های ابتکاری مختلفی توسعه داده شده که در این میان، روش الگوریتم ژنتیک به عنوان یکی از روشهای تکاملی مبتنی بر جمعیت با روندی افزایشی مورد استفاده قرار گرفته است. در این مقاله، جهت رفع کاستی‌های روش‌های ژنتیک موجود مانند ثابت بودن طول رشته جواب و محدودیت های عملگرهای جهش و تعویض، یک الگوریتم ژنتیک جدید برای حل مسئله برنامه‌ریزی حرکت روبات‌های متحرک در شرایط غیربهنگام با داشتن اطلاعات فضای جهانی روبات ارائه شده است، که در آن پس از مثلث بندی فضای روبات به روش Delaunay، دسته جواب هایی با روش درخت کاوش تصادفی سریع (RRT) ایجاد شده و هموارسازی می شوند که طول رشته جواب ها بر خلاف الگوریتم های ارائه شده در ادبیات متغیر است. سپس با استفاده از عملگرهای جدید جهش، تعویض و حذف، این جواب ها بهبود داده می شوند که در هر سه عملگر، مثلث بندی ایجاد شده نقشی کلیدی ایفا می کند. شبیه سازی گستردهٔ الگوریتم نشان داده است که طول مسیرهای به دست آمده با الگوریتم جدید به طور متوسط در حدود ۲۵٪ نسبت به جواب های به دست آمده از روش RRT محض کوتاه تر می‌باشند.

۱. مقدمه

روبات های نسل جدید برای آن که بتوانند به طور مستقل، خودکار و هوشمند عمل کنند، باید قابلیت ایجاد برنامه ای برای حرکت از یک نقطه به نقطه دیگر را دارا باشند، به طوری که این حرکت ترجیحاً در حداکثر سرعت ممکن و بدون برخورد با موانع در طول مسیر باشد و مسیر حاصل حتی الامکان کوتاه و هموار بوده و امکان پیمایش آن توسط روبات ممکن باشد.

مسئلهٔ *برنامه ریزی حرکت روبات*[۲] جهت پاسخگویی به همین نیاز مطرح شد و از اواخر دهه ۱۹۷۰ میلادی به عنوان یکی از زمینه های مهم در علم روباتیک مورد توجه محققان قرار گرفت [۱]. البته مسئلهٔ یافتن مسیرهای بهینه یا ایمن تنها محدود به علم روباتیک نشده است و در حقیقت بسیاری از مسایل مهندسی

تاریخ وصول: ۸۹/۳/۲۲
تاریخ تصویب: ۸۹/۸/۵

***نویسنده مسئول مقاله: دکتر الیپس مسیحی**، استادیار، بخش مهندسی صنایع، دانشکده فنی و مهندسی، دانشگاه تربیت مدرس، masehian@modares.ac.ir
دکتر فرناز برزین پور، استادیار، دانشکده مهندسی صنایع، دانشگاه علم و صنعت ایران، barzinpour@iust.ac.ir
سمیرا ساعدی، دانش آموخته کارشناسی ارشد، دانشکده مهندسی صنایع، دانشگاه علم و صنعت ایران، samira.saedi@gmail.com

[2] *Robot Motion Planning*

به طور مستقیم یا غیر مستقیم به برنامه ریزی حرکت روبات ارتباط دارند. به عنوان نمونه، چند کاربرد آن به شرح زیر می‌باشد:

- برنامه‌ریزی حرکت لیفت تراک ها یا AGV-ها در کف کارخانه و در حضور موانع ثابت (مانند دیوارها، ستون ها و دستگاه ها) و متحرک (مانند افراد و خودروهای در حال حرکت)؛
- برنامه ریزی مسیر ابزار فرزکاری یا مته زنی بر روی قطعه کار به منظور کمینه کردن زمان پردازش؛
- یافتن کوتاه ترین مسیرها جهت تردد در شهر برای وسایل نقلیه شخصی و عمومی و پیشنهاد دادن آن به رانندگان از طریق سیستم GPS؛
- طراحی مسیرهای اتصالی قطعات الکترونیکی بر روی بوردهای مدار چاپی به طوری که کمترین جا اشغال شود و میزان تقاطع های مسیرهای مدارها با یکدیگر به حداقل برسد.

به طور کلی روش‌های ارائه شده برای انجام برنامه‌ریزی حرکت روبات را از حیث نوع اطلاعات در دسترس می توان به دو بخش عمده تقسیم نمود:

نوع اول به صورت *غیربهنگام*[1] است که دانش مربوط به فضای اطراف روبات براساس نقشه‌های آماده یا دانش قبلی حاصل از یک برنامه‌ریزی قبلی موجود است. در نوع دوم، برنامه‌ریزی به صورت *بهنگام*[2] انجام می‌شود که دانش مربوط به فضای اطراف به کمک حسگرها و در حین انجام برنامه‌ریزی حرکت برای روبات مشخص می‌شود [۲]. در این مقاله برنامه‌ریزی حرکت در حالت غیربهنگام مورد بررسی قرار گرفته و روش ارائه شده با فرض داشتن اطلاعات اولیه در خصوص فضای جهانی برنامه‌ریزی حرکت را انجام می‌دهد. تاکنون روش های زیادی برای حل این دسته از مسائل مطرح شده است اما به دلیل پیچیدگی زمانی این مسائل، روش های سنتی بهینه سازی در این زمینه با اقبال مواجه نبوده‌اند و در نتیجه روش های ابتکاری و فراابتکاری فراوانی برای حل این مسائل مطرح شده اند. در این میان، الگوریتم های با رویکرد تکاملی به طور عام و الگوریتم ژنتیک به طور خاص از اوایل دهه ۱۹۹۰ میلادی به حجم فزاینده ای در زمینه های مختلف مهندسی مورد استفاده قرار گرفتند [۳].

در این پژوهش‌ها، طراحی نحوه در نظر گرفتن فضای جواب ها، نحوه تولید جواب های اولیه، نوع عملگرهای به کار رفته و نحوه به کارگیری آنها، گام های اساسی پیاده سازی الگوریتم ژنتیک بوده و همین عوامل ضعف‌ها و قوت‌های الگوریتم‌ها را مشخص می‌کنند.

اولین کاربرد الگوریتم ژنتیک در سال ۱۹۹۱ برای برنامه‌ریزی حرکت یک روبات متحرک در فضا ارائه شد که در آن جواب های اولیه بدون در نظر گرفتن شدنی بودن یا نبودن به صورت کاملاً تصادفی در فضای پیوسته تولید شده و دو عملگر جهش[3] و تعویض[4] بدون در نظر گرفتن طول گام روبات به کار گرفته شدند [۴]. در تحقیقاتی که در سال ۱۹۹۳ توسط شیباتا و فوکودا [۵] و در سال ۱۹۹۶ توسط وانگ و همکارانش [۶] انجام شد فضای آزاد روبات تبدیل به گرافی می شود که هر یک از گره های آن شماره بندی شده و در نتیجه طول رشته جواب مقداری ثابت و مشخص است و به هر یک از ژن ها مقداری برابر صفر یا یک تخصیص می یابد. آنگاه عملگر تعویض هر ژن انتخابی را با ژنی به صورت کاملاً تصادفی تعویض می‌کند. در الگوریتمی که توسط تادوکورو و همکاران [۷] ارائه شده است فضا به صورت مربع (grid) شبکه بندی شده است که در این حالت به هر یک از مربع ها عدد مشخصی تخصیص یافته و در نتیجه طول رشته جواب عددی مشخص است و مقدار هر یک از ژن ها صفر یا یک می باشد. در مقاله لی و همکارانش [۸] فضا مربع بندی شده و عملگر تعویض در سه حالت ارائه شده است و عملگر جهش نیز مانند دیگر مقالات، به صورت کاملاً تصادفی ژنی را با ژن دیگر تعویض می‌کند که در این حالت احتمال ایجاد جواب های غیرممکن (نشدنی) بالا رفته و از طرفی طول گام روبات نیز در نظر گرفته نمی‌شود.

محجوبی و همکاران [۹] فضا را به صورت مربعی تقسیم‌بندی کرده و عملگر جهش در سه حالت حذف نقطه، الحاق نقطه جدید و تعویض نقطه را به کار گرفته‌اند. در الگوریتم ارائه شده توسط مانیکاس و همکارانش [۱۰] فضا به صورت مربعی شبکه بندی شده و دو عملگر جهش و تعویض به صورت کاملاً ساده‌ای به کار رفته‌اند. هو و همکاران [۱۱] نیز در الگوریتمی که ارائه داده اند محیط را به صورت مربعی شبکه بندی کرده و چهار عملگر جهش، تعویض، حذف نقطه و اضافه کردن نقطه را ارائه داده و از طرفی تابع برازش را نیز اصلاح کردند تا پراکندگی را در فضای جواب ایجاد شود. در اغلب مقالات مرور شده چند نکته اساسی به عنوان ضعف مشاهده می شوند: اول آنکه در اکثر روش های ارائه شده، برنامه‌ریزی حرکت روبات در فضایی گسسته تصور شده و کل فضای جهانی به قسمت های کوچکتری تقسیم‌بندی و شبکه‌بندی می‌شود. این تقسیم‌بندی به روش های مختلف ایجاد گراف یا شبکه بندی فضا صورت می گیرد. مشکلی که در نتیجه این تقسیم بندی فضا به وجود می‌آید آن است که طول رشته جواب مقداری ثابت در نظر گرفته می‌شود که امر قابل قبولی نیست، چرا که نه تنها در هر بار برنامه‌ریزی حرکت روبات در یک فضای مشخص با نقاط شروع و پایان متفاوت جوابی متفاوت به دست می‌آید، بلکه در یک فضای ثابت با نقاط شروع و هدف ثابت نیز در هر بار اجرای برنامه مسیری متفاوت ایجاد می‌شود و ثابت

[1] *Offline*
[2] *Online*
[3] *Mutation*
[4] *Crossover*

بودن طول این مسیرهای متفاوت قابل قبول نیست. از طرفی در صورت تقسیم بندی فضا، روبات موظف است در هر حرکت از مرکز یک مربع به مرکز مربع دیگری برود و در واقع علی رغم آنکه روبات قادر است در هر نقطه ای از فضای آزاد قرار گیرد، تعداد نقاط مجاز آن محدود می‌شود. دومین مشکلی که در مقالات مرور شده وجود دارد نوع عملگر جهش است: در این الگوریتم‌ها در هنگام عملگر جهش، به محدودیت روبات در خصوص طول گام توجه نشده است و نقطه جدید انتخابی ممکن است منجر به حرکتی با طول بیشتر از گام روبات شود و همچنین در هنگام حذف یا الحاق نقطه جدید نیز در خصوص طول گام روبات توجهی به عمل نمی آید. مشکل سوم شکل موانع است: در تحقیقات گذشته، موانعی که در فضای کاری قرار گرفته‌اند دارای شکلهای ساده بوده و روش‌های حل ارائه شده قابلیت برنامه‌ریزی در فضاهای دارای موانع پیچیده را ندارند [۱۰]. همچنین در برنامه‌ریزی در فضاهای گسسته، شکل مانع بایستی به گونه‌ای باشد که مضرب صحیحی از تقسیم‌بندی‌های انجام شده در فضا گردد [۷، ۸، ۹، ۱۰] و اگر مانع دارای شکل پیچیده‌ای باشد به صورت تقریبی در فضا نشان داده می‌شود تا ضریبی از تقسیم‌بندی اعمال شده باشد [۹، ۱۰].

در مقاله حاضر با در نظر گرفتن ضعف های عمدهٔ روش‌های موجود، روشی ارائه گردیده است که در شرایط غیربهنگام، مسیری نزدیک به بهینه با تعداد نقاط متفاوت در فضای جهانی پیوسته و دارای موانع با اشکال مختلف ارائه دهد، و عملگر جهش نیز به گونه‌ای تنظیم شده است که جواب های به دست آمده از آن به صورت جواب شدنی و امکان‌پذیر باشد، و همچنین به طول گام روبات هم توجه شده است.

۲. بیان مسئله و رویکرد حل

در این مقاله به مسیریابی یک روبات نقطه ای با استفاده از الگوریتم ژنتیک جدیدی پرداخته شده است. مسیریابی روبات در ساده ترین مفهوم خود عبارتست از یافتن یک توالی حرکت برای روباتی که در محیطی با اشیای ثابت یا متحرک از یک نقطه‌ی آغازین شروع شده و بدون برخورد با موانع به یک نقطه‌ی پایانی ختم می‌گردد. در مسائل برنامه‌ریزی حرکت توصیف کاملی از هندسه روبات، محیط و موانع ارائه می‌شود، و هدف پیدا کردن مسیری است که روبات بتواند بدون برخورد با موانع موجود در محیط مسیری را از نقطه ابتدا به نقطه هدف طی کند [۱]. هر چند الگوریتم های برنامه‌ریزی حرکت از نظر جزئیات تفاوت‌های زیادی با هم دارند اما در بیشتر آن‌ها یک چارچوب کلی وجود دارد. در گام اول، روبات که ممکن است دارای یک شکل پیچیده هندسی باشد به نقطهٔ مرجع خود تبدیل می‌شود و فضای موجود نیز به یک فضای جدید تبدیل می‌شود که به آن *فضای پیکربندی* گفته می‌شود. این امر باعث می‌شود تا مسئله اصلی به یک مسئله مسیریابی برای یک نقطه تبدیل شود [۲].

با توجه به امکان رویارویی با حوزه وسیعی از فرضیات در طی حل هر مسئله، لازم است مسئله با توجه به فرضیات اولیه مشخص و مرزبندی شود. فرضیاتی که در ارائه این الگوریتم در نظر گرفته شده اند عبارتند از:

- برنامه‌ریزی حرکت برای یک روبات متحرک انجام می‌شود که برای تسهیل شبیه‌سازی به صورت نقطه‌ای در نظر گرفته شده است.
- روبات صلب بوده و تغییر شکل نمی‌دهد.
- روبات دارای محدودیت‌های سینماتیک و دینامیک نبوده و میتواند در جهات مختلف و با سرعت های متفاوت حرکت نماید.
- فضا به صورت پیوسته (شبکه بندی نشده) درنظر گرفته میشود.
- برنامه‌ریزی حرکت در محیط به صورت کلان[1] است.
- محیط فاقد تغییرات زمانی می‌باشد یعنی موانع دارای حرکت نبوده و ایستا هستند.
- موانع موجود در محیط می‌توانند به هر شکلی باشند و روبات باید مسیر خود را در فضای آزاد از موانع طی نماید.
- نحوه کسب اطلاعات محیطی به صورت غیربهنگام است.

در بخش بعدی مقاله برای تشریح الگوریتم ارائه شده، هر یک از مراحل الگوریتم و عملگرهای به کار رفته در آنها توضیح داده شده‌اند. در بخش چهارم پاسخ های به دست آمده از این روش با پاسخ های به دست آمده از روش های دیگر مقایسه شده و میزان کارایی این روش برای موانع گوناگون مورد بررسی قرار گرفته است. در انتها نتیجه‌گیری و پیشنهاداتی جهت پژوهش‌های آتی ارائه گردیده است.

۳. روش پیشنهادی مبتنی بر الگوریتم ژنتیک

سیستم‌های تکاملی[2] اولین بار بین سالهای ۱۹۵۰ و ۱۹۶۰ به عنوان ابزاری جهت بهینه‌سازی مسائل مهندسی توسط محققان علوم کامپیوتری مورد بررسی قرار گرفتند. ایده به کار رفته در تمام این سیستم‌ها بکارگیری عملگرهای الهام گرفته از تغییرات ژنتیکی طبیعی و انتخاب طبیعی جهت تکامل جمعیتی از حل‌های کاندیدای مسئله مورد نظر بود.

الگوریتم‌های ژنتیک اولین بار در سال ۱۹۶۰ توسط جان هالند ارائه شده و توسعه یافتند. طبق تعریف هالند، "*یک الگوریتم ژنتیک یک تکنیک جستجوی ریاضی است که براساس قواعد انتخاب طبیعی و*

[1] *Gross*
[2] *Evolutionary Systems*

ترکیب‌دهی مجدد ژنتیک عمل می‌کند" [۱۲]. با توجه به این تعریف هر الگوریتم ژنتیک در ساده‌ترین حالت دارای دو جز اصلی است: جمعیت جوابهای اولیه و عملگرهای ژنتیک. نحوه تعریف جواب‌های اولیه و عملگرها میزان کارایی الگوریتم ارائه شده را برای بهینه‌سازی مسئله مورد نظر مشخص می‌کنند. در ادامه جهت تشریح الگوریتم ارائه شده اجزای اصلی آن کاملاً تشریح شده و نحوه عملکرد آنها توضیح داده می شود.

۳-۱. پارامترها

با توجه به اینکه در هر الگوریتم فراابتکاری همواره پارامترهایی وجود دارند که تغییر هر یک از آن‌ها نتیجه الگوریتم را تحت تاثیر قرار می‌دهد، قبل از تشریح الگوریتم باید پارامترهای به کار رفته در آن مشخص شوند:

k: بزرگی طول گام روبات،

res: واحد تقسیم‌بندی برای مثلث‌بندی دِلونِی،

N_{it}: تعداد تکرارهای الگوریتم،

P_s: اندازه جمعیت اولیه،

C_s: دفعات انجام الگوریتم تعویض،

M_s: درصد مسیرهای انتخابی برای انجام عملگر جهش در هر تکرار الگوریتم،

δ: پارامتر عملگر جهش،

N_{rat}: نرخ نخبه‌گرایی.

۳-۲. مراحل الگوریتم پیشنهادی

الگوریتم ژنتیک پیشنهادی را می‌توان در چهار مرحله اصلی خلاصه نمود: (۱) پیش‌پردازش و تولید جواب‌های اولیه، (۲) اعمال عملگر تعویض، (۳) اعمال عملگر جهش، و (۴) انتخاب جمعیت برای نسل بعد (شکل ۱). ابتدا جواب‌های اولیه توسط روش RRT تولید شده و سپس به کمک روش جستجوی دایسترا (Dijkstra) مسیر اصلی شناسایی می‌شود. در ادامه این مرحله توسط عملگر حذف، مسیرهای ایجاد شده هموارسازی می‌شوند. برای انجام عملگرهای تعویض و جهش در الگوریتم لازم است فضای روبات تقسیم‌بندی شود. برای انجام این تقسیم‌بندی از روش *مثلث‌بندی دلونِی* استفاده می‌شود که قادر است فضای آزاد روبات را به مثلث‌هایی افراز کند.

در مرحله دوم برای انجام عملگر تعویض ابتدا از میان مثلث‌های ایجاد شده، یک مثلث توسط چرخ رولت انتخاب می‌شود به‌گونه‌ای که مثلث‌هایی که مسیرهای بیشتری از آنها گذشته‌اند احتمال بیشتری برای انتخاب شدن دارند. پس از انتخاب یک مثلث، دو مسیر از میان مسیرهایی که از مثلث مذکور می گذرند انتخاب شده و تحت عملگر حذف قرار می گیرند. عملگر حذف در صورت وجود دور (لوپ) در مسیر، نقاط موجود در دور را حذف کرده و طول مسیرهای والد را کاهش می‌دهد. پس از اعمال این عملگر، عملگر تعویض بر دو مسیر اصلاح شده انجام می‌گیرد.

در مرحله سوم به تعداد M_s مسیر از میان مسیرهای موجود در جمعیت انتخاب شده و پتانسیل جهش هر یک محاسبه می‌شود. در ادامه با توجه با میزان پتانسیل هر مسیر، تعدادی از نقاط آن انتخاب و مثلث دلونِی حاوی آن نقطه مشخص می‌شود. سپس نقطه‌ای تصادفی در این مثلث تولید شده و جایگزین نقطه انتخابی می‌شود. در مرحله آخر N_{rat} درصد از جمعیت موجود با استفاده از چرخ رولت انتخاب و برای نسل بعد نگهداری می‌شوند.

۳-۳. ایجاد جوابهای اولیه

برای تعیین مسیر یک روبات در فضای آزاد در حالت غیربهنگام روش‌های مختلفی در ادبیات برنامه‌ریزی حرکت وجود دارد اما با توجه به این‌که ایجاد جواب‌های اولیه برای مدل نباید زمان‌بر باشد، لازم است از میان روش‌های موجود روشی انتخاب شود که در کمترین زمان ممکن جواب هایی با پراکندگی مناسب در فضای جواب ایجاد کند. تکنیک پیشنهادی برای این قسمت از مدل روش *درخت کاوش تصادفی سریع*[1] (RRT) می‌باشد. روش RRT بر اساس تلفیق ایده‌هایی از کنترل بهینه، برنامه‌ریزی روبات‌های غیرهولونومیک و برنامه‌ریزی مسیر تصادفی[2] ایجاد شده است.

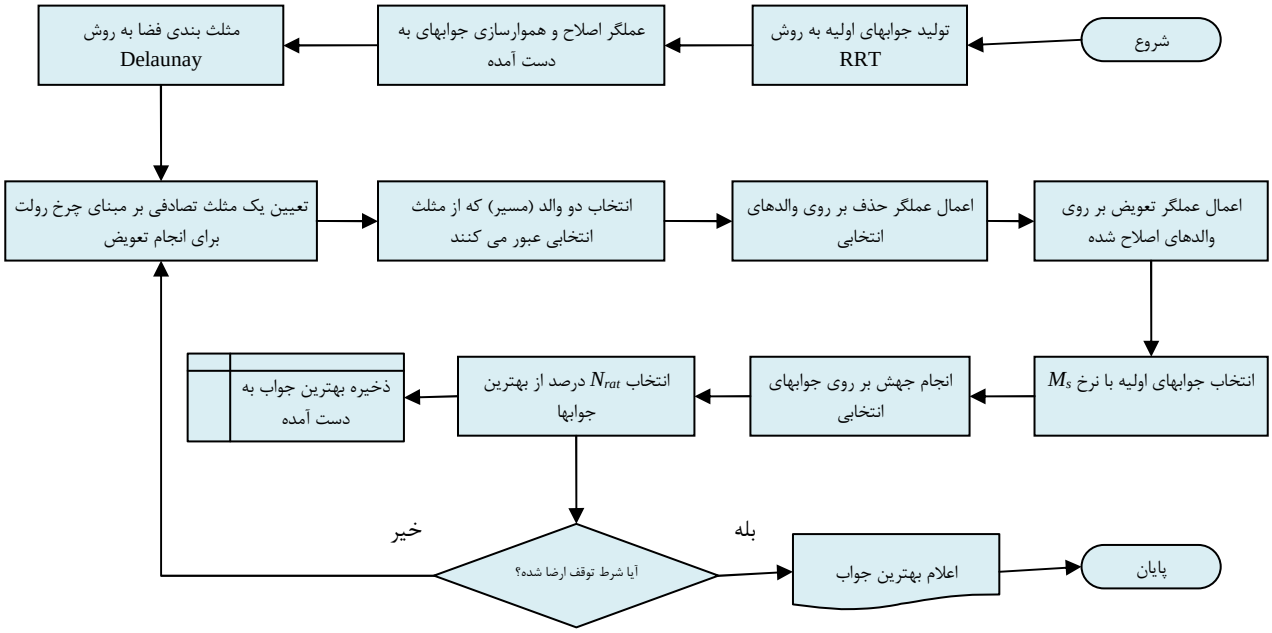
RRT یک ساختار داده و طرح نمونه‌برداری است که برای جستجوی سریع فضاهایی با ابعاد زیاد و دارای محدودیت‌های جبری (محدودیت‌های ایجاد شده بر اثر موانع) و محدودیت‌های دیفرانسیلی (محدودیت‌های دینامیکی و غیرهولونومیکی روبات) مورد استفاده قرار می‌گیرد.

در این روش یک درخت جستجو از مکان اولیه شروع به رشد می‌کند و با استفاده از داده‌های کنترلی به یک موقعیت جدید می‌رسد. هر رأس بیانگر یک موقعیت جدید است و هر بردار یک ورودی است که برای رسیدن به موقعیت جدید مورد استفاده قرار می‌گیرد. ایده اصلی این روش در ایجاد نقاطی تصادفی در فضا و کشیدن درخت جستجو به سمت آنهاست [۱۳]. نحوه انتخاب تصادفی نقاط موجب می‌شود فضای جواب در این روش به خوبی پوشش داده شده و شانس وجود هر نقطه از فضای آزاد با یکدیگر برابر باشد. نحوه عملکرد این روش در ایجاد یک نقطه جدید (عملیات بسط، extend) در شکل ۲ نشان داده است.

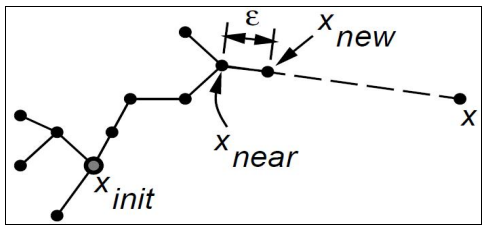
در عملیات بسط پس از ایجاد یک نقطه تصادفی در فضا (x)، از میان نقاط موجود در درخت، نزدیکترین نقطه به آن تعیین گردیده (x_{near}) و بر روی بردار متصل‌کننده دو نقطه x و x_{near} نقطه جدیدی در فاصله‌ای مشخص از x_{near} تعیین می‌شود (x_{new}).

[1] *Rapidly-exploring Random Trees*
[2] *Randomized Path Planning*

پس از ایجاد x_{new} ممکن است سه حالت برای این نقطه رخ دهد: در حالت اول این نقطه جدید در مانع قرار دارد (حالت در تله افتاده، *Trapped*) که در صورت رخ دادن این حالت، فرآیند تولید x_{new} باید مجدداً تکرار شود. حالت دوم آن است که x_{new} تولید شده بسیار نزدیک به نقطه تصادفی است، یعنی ($|x_{new} - x| < \varepsilon$) (حالت به هدف رسیده، *Reached*). در حالت سوم x_{new} نقطه‌ای جدید و قابل قبول است و به مسیر τ اضافه می‌شود (حالت پیشروی کرده، *Advanced*).

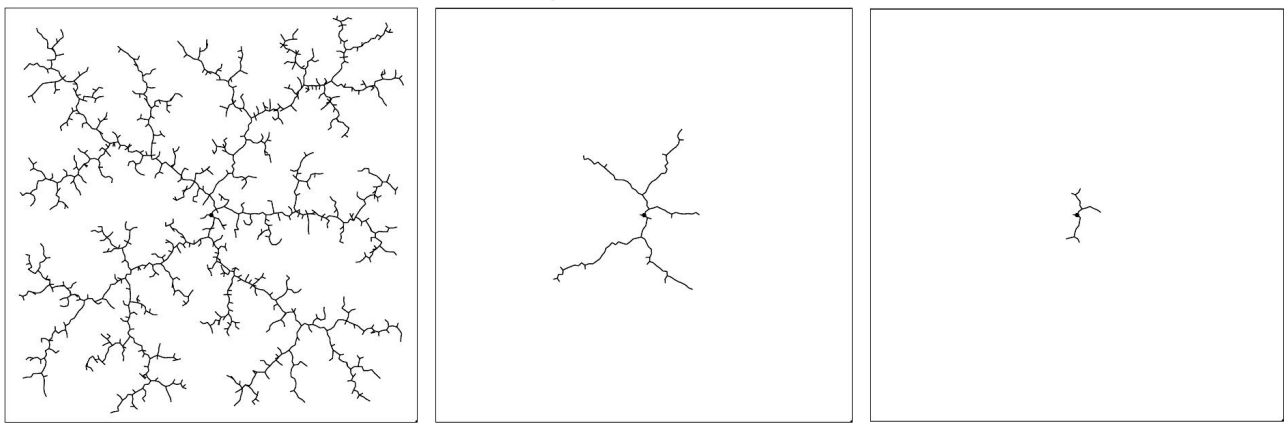


شکل ۱. فلوچارت روش پیشنهادی



شکل ۲. نحوه عملیات بسط در روش RRT

یکی از پارامترهای قابل توجه در این روش طول گام (k) است. طول گام می‌تواند با توجه به تابع محاسبه فاصله که در بررسی تصادم با مانع استفاده می‌شود به صورت پویا تعریف شود. اگر موانع دور از نقطه جدید قرار دارند، گام بعدی می‌تواند بلندتر انتخاب شود. شکل ۳ مراحل مختلف رشد و توسعه درخت جستجو را به روش RRT نشان می دهد، و الگوریتم پایه ای روش RRT در شکل ۴ بیان شده است. پس از شکل‌گیری یک درخت شدنی میان نقاط شروع و هدف، لازم است قسمت هایی از درخت که در فضا پراکنده شده و در شکل‌گیری مسیر میان نقاط شروع و هدف نقشی ندارند حذف شوند و سپس طی عملیات هموارسازی مسیر ایجاد شده، نقاط میانی گره های به دست آمده به یکدیگر متصل شده و مسیر هموارتری به دست می‌آید. جواب‌های به دست آمده از این مرحله وارد الگوریتم ژنتیک ارائه شده می‌شوند و به عنوان جمعیت اولیه تحت عملگرهای مختلف قرار می‌گیرند.



شکل ۳. مراحل ساخت درخت RRT

```
Build_RRT(x_init)
    τ.init(x_init)
    For k = 1 to K do
        x_rand ← Random_State();
        Extend(τ, x_rand);
    Return τ

Extend (τ, x)
    x_near ← Nearest_Neighbor(x, τ);
    If New_State(x, x_near, x_new) then
        τ.add-vertex(x_new);
        τ.add-edge(x_near, x_new);
        If x_new= x then
            Return Reached
        Else
            Return Advanced
    Return Trapped
```

شکل ۴. الگوریتم پایه RRT [۱۳]

با الهام از تکنیک‌های جستجوی دو جهته، منطقی است که برای بهبود روش RRT نیز بتوان آن را به صورت دو طرفه انجام داد. در این روش که شبه کُد آن در شکل ۵ نشان داده شده است دو مسیر τ_a و τ_b از دو نقطه x_{init} (نقطه شروع حرکت) و x_{goal} (نقطه هدف) بسط می‌یابند و زمانی که دو مسیر به هم می‌رسند، یک راه حل شکل گرفته است (که بعداً عضوی از جمعیت آرایه های الگوریتم ژنتیک خواهد شد).

```
RRT-BIDIRECTIONAL(x_init, x_goal)
    τ_a.init(x_init);τ_b.init(x_goal);
    For k = 1 to K do
        x_rand ← Random_State();
        If not (Extend(τ_a, x_rand) = Trapped) then
            If (Extend(τ_b, x_new) = Reached) then
                Return Path(τ_a, τ_b);
        Swap(τ_a, τ_b);
    Return Failure;
```

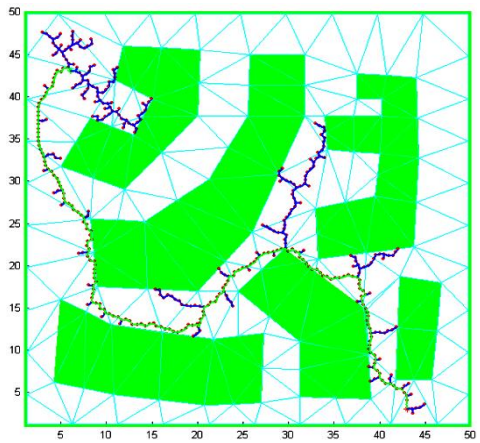
شکل ۵. الگوریتم RRT دو جهته[۱۳]

در شکل ۶ نمونه‌ای از یک مسیر ایجاد شده توسط روش RRT مشاهده می‌شود. با استفاده از روش RRT دوجهته، دو مسیر از نقاط شروع و هدف به طور همزمان توسعه می‌یابند (شکل ۶-الف) و پس از ایجاد کل مسیر لازم است شاخه‌های زائد مسیر حذف شوند. این کار با حرکت از سمت نقطه هدف به سمت نقطه شروع انجام می‌شود (شکل۶-ب). پس از به دست آمدن هر یک از مسیرهای اولیه، عملگر هموارسازی بر هر مسیر اعمال می‌شود. با اعمال این عملگر با اتصال نقاط میانی در هر یک از مسیرهای اولیه به دست آمده، شکستگی های مسیر کمتر شده و هموارتر می‌شود.

۳-۴. عملگر تعویض

با توجه به وجود موانع بایستی نوعی هوشمندی در عملگر جهش پیش‌بینی شود تا انتخاب والدین و نقاطی جهت انجام عملگر به درستی انجام شود و فرزندان به وجود آمده غیرممکن (نشدنی) نباشند. برای انجام این کار از مفهوم *مثلث‌بندی دلونی* که در ادبیات هندسه محاسباتی به دوگان *نمودار ورونویی*[1] معروف است، استفاده شده است. هر نمودار ورونویی یک دوگان به شکل تقسیم بندی مثلثی دارد که با نام *تقسیم بندی دلونی*[2] شناخته می شود و با داشتن هر کدام، دیگری نیز به دست می آید؛ بدین ترتیب که رئوس مثلث های ایجاد شده همان مراکز نواحی ورونویی بوده و گره های نمودار ورونویی معادل با محل تلاقی عمود منصف های مثلث متناظر با آن می باشد. شکل (۷-الف) چگونگی ارتباط میان نمودار ورونویی و مثلث بندی دلونی معادل با آن را نشان می دهد [۱۴].

(الف)



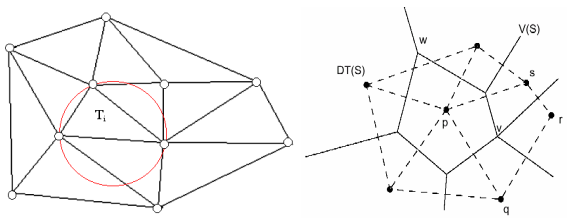
(ب)

شکل ۶. مراحل الگوریتم RRT: (الف) توسعه مسیر از دو نقطه شروع و هدف؛ (ب) حذف شاخه‌های اضافه

[1] *Voronoi Diagram*
[2] *Delaunay Tessellation (or Triangulation)*

روش مثلث بندی دلونی دارای خواص زیر می باشد:

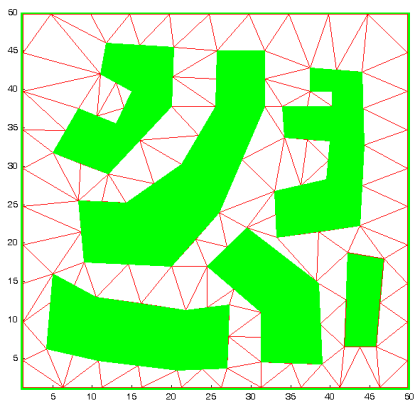
۱. نوع مثلث بندی یکتا و منحصر به فرد است و با تکرار الگوریتم تغییری نمی‌کند.
۲. از طریق مثلث بندی دلونی می توان کل فضا را به مجموعه‌ای از مثلث های بدون تداخل افراز نمود.
۳. دایره محاطی هر مثلث، رئوس مثلث های دیگر را دربر نمی‌گیرد (شکل (۷-ب)).
۴. مثلث بندی دلونی قابلیت تعمیم به فضاهای بالاتر را دارد، به طوری که در فضای سه بعدی از طریق دلونی فضا به مجموعه‌ای از هرم‌های به هم متصل افراز می‌گردد به طوری که کرهٔ محیطی گذرنده از رئوس هر هرم، رأس هرم دیگری را در بر نمی گیرد.



(ب) (الف)

شکل ۷. (الف) مثلث بندی دِلونِی (خط چین) به همراه دوگان وُرونویی آن؛ (ب) نمایش خاصیت سوم مثلث بندی دلونی

رواج روش دلونِی به دو دلیل است: اول آنکه مثلث هایی با شکل مناسب ایجاد می کند و روش دوگان آن یعنی دیاگرام ورونویی در ادبیات کاملاً پذیرفته شده و دارای کاربردهای بسیار گسترده ای است. از طرفی، روش های دیگر مثلث بندی مستلزم محاسبات زیادی بوده و دارای پیچیدگی زمانی بالایی هستند، در حالیکه پیچیدگی زمانی محاسبه مثلث های دلونی در مرتبه $O(n\log n)$ است [۱۴]. لذا با توجه به خواص فوق می توان از مثلث بندی دلونی در تجزیه سلولی فضا جهت برنامه ریزی مسیر استفاده نمود. نمونه‌ای از فضای مثلث‌بندی شده در شکل ۸ نشان داده شده است.

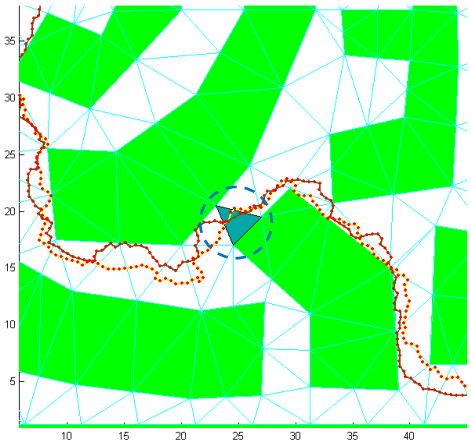


شکل ۸. نمونه‌ای از مثلث‌بندی فضای حرکت روبات

در صورت استفاده از مثلث‌بندی دلونی در تقسیم محیط روبات به همراه موانع، رئوس موانع همان رئوس مثلث ها خواهند بود که با انتخاب درست رئوس، مثلثهای ایجاد شده یا کاملاً داخل موانع یا کاملاً در فضای آزاد قرار دارند؛ یعنی می توان با استفاده از مثلث‌بندی دلونی فضای آزاد و فضای موانع را به طور دقیق از یکدیگر تفکیک نمود.

با استفاده از این روش تعویض در نقاطی انجام خواهد شد که در فضای آزاد قرار گرفته باشند. برای انجام این عملگر ابتدا با استفاده از چرخ رولت، مثلثی از میان مثلث های موجود در فضای آزاد انتخاب می‌شود. با توجه به قانون چرخ رولت، مثلث‌هایی که مسیرهای بیشتری از آن‌ها گذشته‌اند با احتمال بیشتری انتخاب خواهند شد. سپس از میان مسیرهایی که از مثلث انتخابی می‌گذرند، دو مسیر تصادفی انتخاب می‌شود.

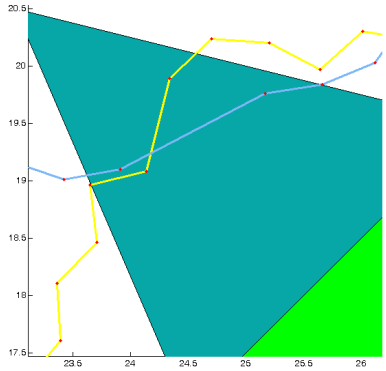
در ادامه روی والدین انتخابی عملگر حذف انجام می‌شود که در قسمت بعد توضیح داده شده است. پس از اعمال عملگر حذف، نقاط ورود و خروج مسیرهای والد به مثلث انتخابی شناسایی شده و عمل تعویض بر روی مسیرها در آن نقاط انجام می‌شود. نمونه‌ای از اعمال عملگر تعویض بر روی دو مسیر موجود در شکل ۹ را می توان در شکل ۱۰ مشاهده نمود.



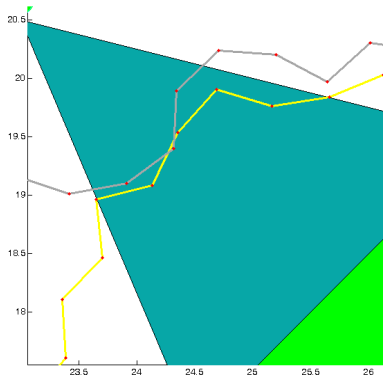
شکل ۹. انتخاب یک مثلث به صورت تصادفی

۵-۳. عملگر حذف

این عملگر بر روی والدین انتخابی در عملگر تعویض اعمال می‌شود و نقاطی از مسیر را که بیهوده طی شده حذف می‌کند و در نتیجه مسیر را هموارتر می کند. عملکرد این عملگر به این صورت است که نقاط ورود و خروج مسیرهای والد به مثلث انتخابی توسط چرخ رولت را شناسایی کرده و سپس در صورتی که این نقاط به گونه‌ای باشند که نشان دهد مسیر بین این دو نقطه از مثلث مورد نظر خارج شده است، نقاط میان این دو نقطه حذف خواهد شد. نمونه‌ای از اِعمال این عملگر بر یک مسیر در شکل ۱۱ نشان داده شده است.

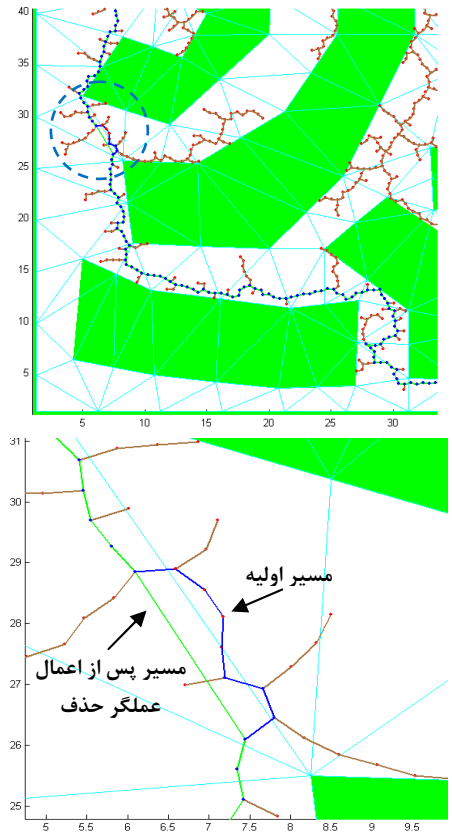


(الف)



(ب)

شکل ۱۰. جزئیات انجام عملگر تعویض: (الف) والدین و مثلث انتخاب شده برای انجام عملگر تعویض؛ (ب) فرزندان ایجاد شده



شکل ۱۱. نحوه عملکرد عملگر حذف

۳-۶. عملگر جهش

در این عملگر ابتدا تعدادی از مسیرها (به تعداد پارامتر M_s) به صورت تصادفی انتخاب می‌شوند. سپس لازم است برای هر مسیر نرخ جهش مشخص شود. نرخ جهش در برخی الگوریتم‌ها عدد ثابتی است، اما در این الگوریتم به منظور ایجاد پراکندگی[1] در این مرحله، نرخ جهش براساس رابطه (۱) مشخص می‌شود [۱۵]:

$$\sigma = e^{-\delta \cdot f} \qquad (۱)$$

در این رابطه f نشانگر میزان تابع برازش مسیر (عکس طول مسیر) است که بین صفر و یک نرمال گردیده (عکس طول مسیر) و δ پارامتری است که تحت کنترل اپراتور است. تعداد جهش در هر مسیر با طول L برابر با $L \times \sigma$ است. همانطور که از رابطه فوق مشخص است هر چه برازش مسیری بهتر باشد میزان جهش روی آن کمتر است.

پس از تعیین تعداد جهش در هر مسیر انتخابی، لازم است ژن هایی که به صورت تصادفی در این مسیرها انتخاب می‌شوند تحت جهش قرار گیرند. برای این کار مثلث دلونی مربوط به ژن انتخابی شناسایی شده و نقطه ای تصادفی درون ناحیه ای به شعاع تلاقی طول گام روبات با مثلث دلونی و مثلث های همسایه مشخص می شود. سپس نقطه انتخاب شده جایگزین ژن انتخابی می‌شود.

۳-۷. تحلیل حساسیت پارامترها

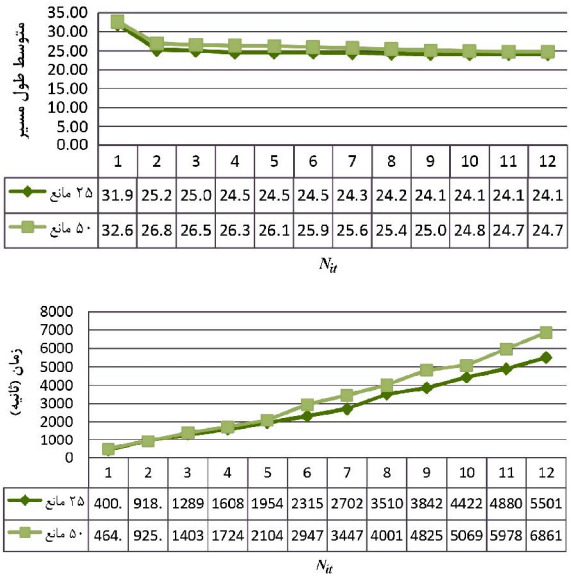
پیش از تعیین میزان کارایی هر الگوریتم متاهیوریستیک لازم است حساسیت الگوریتم نسبت به پارامترهای موجود در الگوریتم مشخص شود. سه پارامتر، N_{it} (تعداد تکرار الگوریتم یا شرط توقف)، δ (پارامتر مشخص‌کننده پتانسیل جهش) و C_s (پارامتر نرخ عملگر تعویض) به این منظور بررسی می‌شوند، چرا که الگوریتم نسبت به دیگر پارامترها حساسیت زیادی را نشان نمی‌دهد. برای این کار یک مسئله در فضای ۲۰×۲۰ با ۲۵ مانع و یک مسئله در همین فضا با ۵۰ مانع در نظر گرفته شده است.

اولین پارامتر بررسی شده تعداد دفعات اجرای الگوریتم (N_{it}) است. جواب های به دست آمده از اجراهای الگوریتم با دفعات مختلف اجرا در شکل ۱۲ نشان داده شده اند، که به ترتیب شامل تحلیل حساسیت الگوریتم با استفاده از بررسی طول مسیر و زمان حل است.

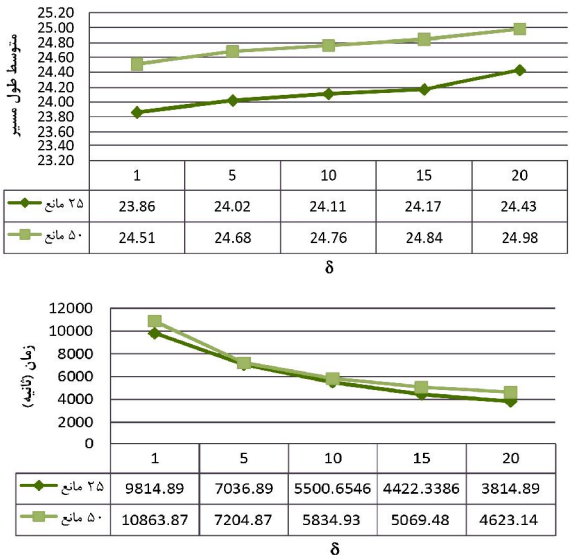
همان گونه که مشاهده می شود، در تکرار اول الگوریتم بهبود زیادی در جواب داده می شود اما با افزایش تعداد تکرارها و حصول به همگرایی، میزان بهبود روندی کاهشی داشته و از طرفی زمان حل در هر بار اجرا زیاد می شود. در نتیجه، جهت

[1] *Diversification*

حفظ تعادل میان این دو معیار متناقض، مقدار N_{it} = ۱۰ در نظر گرفته می شود. پارامتر دوم که در خصوص الگوریتم ژنتیک باید تعیین شود پارامتر δ است که در عملگر جهش الگوریتم نقش دارد. هر چقدر مقدار این پارامتر کمتر باشد پتانسیل جهش مسیرها بیشتر بوده و تعداد نقاط بیشتری تحت عملگر جهش قرار می‌گیرند. تحلیل حساسیت این پارامتر در الگوریتم ژنتیک در دو مسئله نمونه، در شکل ۱۳ نشان داده شده است. همان طور که در شکل ۱۳ مشاهده می‌شود با افزایش مقدار δ طول مسیر افزایش یافته و زمان حل کاهش می‌یابد. اما در مقدار δ = ۱۵ شیب طول مسیر به صورت محسوسی تغییر می‌کند و از طرفی زمان حل نیز با افزایش مقدار δ در حال کاهش است.

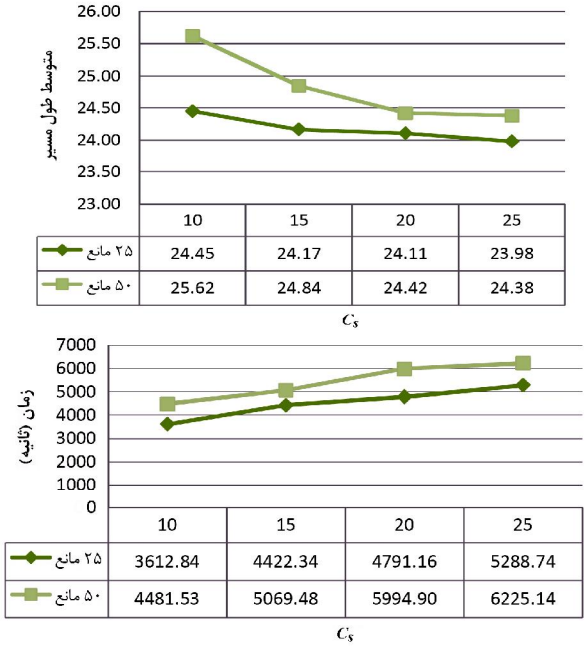


شکل ۱۲. تحلیل حساسیت کیفیت جواب، و زمان محاسبات الگوریتم نسبت به پارامتر N_{it}



شکل ۱۳. تحلیل حساسیت کیفیت جواب، و زمان محاسبات الگوریتم نسبت به پارامتر δ

پارامتر سوم C_s است که در عملگر تعویض در الگوریتم ژنتیک نقش بسیار کلیدی را دارد در نتیجه لازم است حساسیت الگوریتم نسبت به این پارامتر سنجیده شود. نتیجهٔ این کار در شکل ۱۴ نشان داده شده است. مشاهده می‌شود که با افزایش مقدار C_s طول مسیر کاهش می‌یابد و زمان حل افزایش می‌یابد. در نقطه C_s = ۱۵ تغییر شیب در شکل وجود دارد در نتیجه این نقطه به عنوان مقدار پیشنهادی در الگوریتم تثبیت می‌شود.



شکل ۱۴. تحلیل حساسیت کیفیت جواب، و زمان محاسبات الگوریتم نسبت به پارامتر C_s

پس از بررسی مقادیر متفاوت برای کلیهٔ پارامترهای این مدل به روش تحلیل سعی و خطا، مقادیر پارامترهای پیشنهادی به شرح جدول ۱ تعیین شده اند. لازم به ذکر است که پارامترهای k و res در فضای ۲۰×۲۰ پیشنهاد شده اند و در صورت تغییر ابعاد فضا، این پارامترها دستخوش تغییر می شوند.

جدول ۱. مقادیر پیشنهادی برای پارامترهای الگوریتم

پارامتر	k	res	N_{it}	P_s	C_s	M_s	δ	N_{rat}
مقدار پیشنهادی	۰/۱	۲	۱۰	۲۰	۱۵	۱۰	۱۵	۰/۳

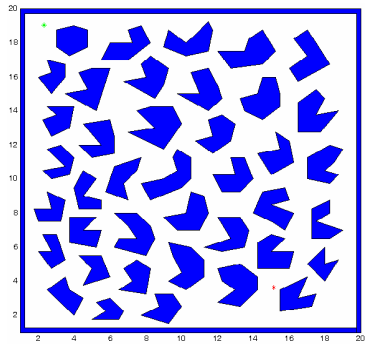
۴. نتایج محاسباتی

برای تعیین میزان کارایی الگوریتم ژنتیک ارائه شده لازم است نتایج حاصل از آن، یعنی متوسط طول تعیین شده توسط این روش را با طول مسیر تعیین شده توسط روش‌های دیگر موجود در ادبیات مقایسه نمود. به منظور انجام این مقایسه سه روش کلاسیک موجود در ادبیات انتخاب شده‌اند. روش اول روش *دیدنگار* (Visibility Graph) است که در شرایط غیربهنگام

مسیر بهینه را در فضای دو بعدی برای حرکت روبات مشخص می‌کند. این روش یکی از اولین روش‌های ارائه شده برای برنامه‌ریزی حرکت روبات بوده و معمولاً تنها برای فضاهای دو بُعدی قابل استفاده است. در این روش با اتصال رئوس موانع با یکدیگر و نقطه‌های شروع و پایان گرافی تشکیل می‌شود که لازم است در نهایت با یکی از روش‌های جستجو، کوتاهترین مسیر در آن مشخص شود. این روش دارای پیچیدگی زمانی بالایی بوده و پیچیدگی آن در صورت استفاده از روش جستجوی دایسترا برابر $O(n^2 \log n)$ می‌باشد و لذا استفاده از این روش چندان معمول نیست [۱]. از طرفی این روش در خصوص فضاهایی با ابعاد بالاتر از دو بعد بهینه نبوده و دیگر روش‌های موجود از روش دیدنگار بهتر عمل می‌کنند. روش دوم روش نمودار ورونویی است که فضای آزاد روبات را تقسیم نموده و با یافتن مسیری روی مرزبندی های انجام شده توسط روش دایسترا، مسیر حرکت مشخص می‌شود. این روش به علت آن که امن‌ترین مسیر روبات را ارائه می‌دهد به عنوان روش دوم جهت مقایسه در نظر گرفته شده است. روش سوم روش درخت کاوش تصادفی سریع (RRT) است که به علت احتمالی بودن مکانیزم انتخاب نقاط قادر است به سرعت جواب‌های پراکنده‌ای را در فضای جواب ایجاد کند و در نتیجه در مینیمم‌های محلی قرار نگیرد. جهت مقایسه چهار روش فوق، ۲۰ مسئلهٔ انتخابی با موانع پراکنده یا مازی شکل طراحی و حل شدند. شکل ۱۵ تصویر یک مسئله نمونه را نشان می دهد.

نتایج محاسبات در جدول ۲ و شکل ۱۶ آمده اند. همان گونه که از جدول ۲ پیداست الگوریتم ژنتیک به طور متوسط موجب بهبودی معادل ۲۵٪ در جواب‌های اولیه ایجاد شده توسط روش RRT می‌شود.

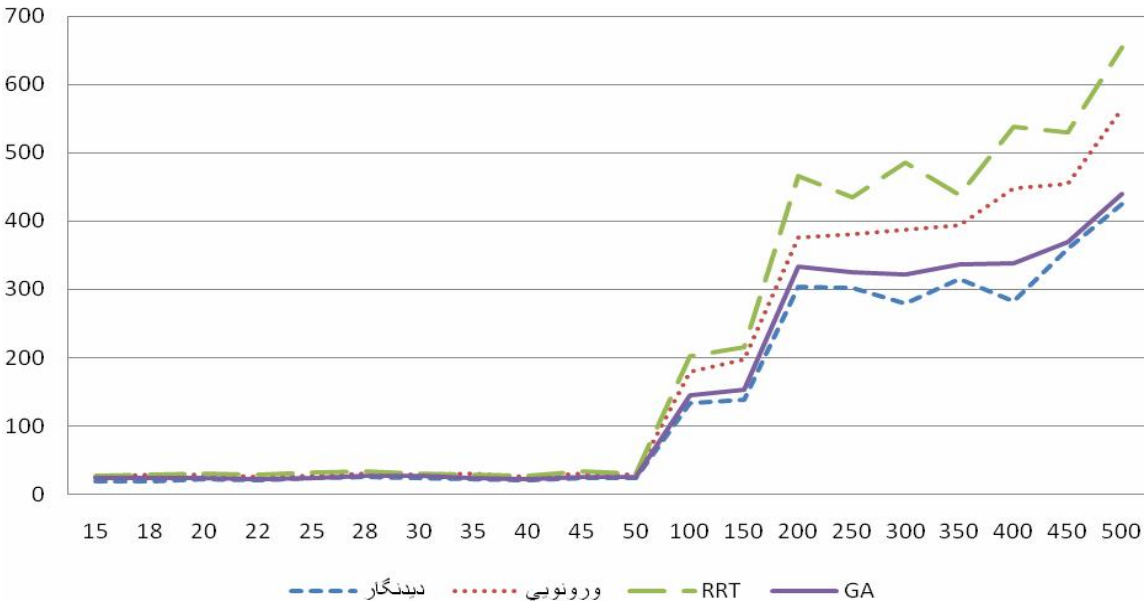
از طرفی الگوریتم ژنتیک نسبت به روش ورونویی که یکی از معمول ترین روش‌های برنامه‌ریزی حرکت می‌باشد کاهشی معادل ۱۷٪ طول مسیر را داراست. خطای روش جدید نسبت به جواب بهینه (جواب به دست آمده از روش دیدنگار در فضای دو بعدی) نیز در جدول ۲ نشان داده شده است، که به طور متوسط برابر ۱۲٫۵٪ است، و با توجه به سرعت الگوریتم جدید و قابلیت تعمیم آن به فضاهای بالاتر از دو بعد، خطای الگوریتم در مقایسه با مزایای آن قابل اغماض است.



شکل ۱۵. مسئله نمونه شماره ۹ در جدول ۲ با ۴۰ مانع

جدول ۲. مقایسه الگوریتم ژنتیک جدید با دیگر روش‌ها

مسئله	تعداد موانع	ابعاد فضای کار	متوسط طول مسیر				درصد بهبود نسبت به روش RRT	درصد بهبود نسبت به روش ورونویی	درصد خطا نسبت به مقادیر بهینه (روش دیدنگار)
			روش GA	روش RRT	روش ورونویی	روش دیدنگار			
۱	۱۵	۲۰×۲۰	۲۲٫۸۳	۲۷٫۵۴	۲۶٫۴۰	۱۸٫۳۱	۱۷٫۱۰	۱۳٫۵۲	۲۴٫۶۸
۲	۱۸	۲۰×۲۰	۲۳٫۶۱	۲۹٫۱۷	۲۸٫۸۳	۱۹٫۰۱	۱۹٫۰۶	۱۸٫۱۰	۲۴٫۱۹
۳	۲۰	۲۰×۲۰	۲۴٫۲۶	۳۰٫۵۹	۲۹٫۸۲	۲۱٫۰۵	۲۰٫۶۹	۱۸٫۶۴	۱۵٫۲۴
۴	۲۲	۲۰×۲۰	۲۲٫۵۳	۲۸٫۶۰	۲۶٫۳۵	۲۰٫۷۱	۲۱٫۲۲	۱۴٫۴۹	۸٫۷۸
۵	۲۵	۲۰×۲۰	۲۴٫۱۷	۳۱٫۹۸	۲۷٫۵۸	۲۲٫۴۵	۲۴٫۴۲	۱۲٫۳۶	۷٫۶۶
۶	۲۸	۲۰×۲۰	۲۶٫۹۹	۳۳٫۴۷	۳۰٫۴۱	۲۴٫۵۷	۱۹٫۳۶	۱۱٫۲۴	۹٫۸۴
۷	۳۰	۲۰×۲۰	۲۶٫۰۳	۳۱٫۰۲	۲۸٫۷۵	۲۳٫۱۳	۱۶٫۰۸	۹٫۴۶	۱۲٫۵۳
۸	۳۵	۲۰×۲۰	۲۳٫۸۱	۲۹٫۹۷	۳۰٫۲۴	۲۱٫۱۱	۲۰٫۵۵	۲۱٫۲۶	۱۲٫۷۹
۹	۴۰	۲۰×۲۰	۲۱٫۸۸	۲۸٫۳۰	۲۶٫۱۴	۲۰٫۲۵	۲۲٫۶۸	۱۶٫۲۹	۸٫۰۴
۱۰	۴۵	۲۰×۲۰	۲۵٫۳۰	۳۳٫۹۵	۳۰٫۷۲	۲۳٫۴۰	۳۱٫۳۶	۱۷٫۶۴	۸٫۱۱
۱۱	۵۰	۲۰×۲۰	۲۴٫۸۴	۳۱٫۱۵	۲۹٫۸۵	۲۳٫۵۲	۲۰٫۲۵	۱۶٫۷۸	۵٫۶۱
۱۲	۱۰۰	۵۰×۵۰	۱۴۴٫۳۸	۲۰۲٫۳۶	۱۷۸٫۹۲	۱۳۲٫۶۷	۲۸٫۶۵	۱۹٫۳۰	۸٫۸۲
۱۳	۱۵۰	۵۰×۵۰	۱۵۳٫۰۴	۲۱۴٫۹۷	۱۹۷٫۰۸	۱۳۸٫۶۳	۲۸٫۸۰	۲۲٫۳۵	۱۰٫۳۹
۱۴	۲۰۰	۱۰۰×۱۰۰	۳۳۳٫۲۶	۴۶۵٫۹۶	۳۷۶٫۱۵	۳۰۳٫۱۶	۲۸٫۴۸	۱۱٫۴۰	۹٫۹۲
۱۵	۲۵۰	۱۰۰×۱۰۰	۳۲۵٫۹۴	۴۳۵٫۰۲	۳۸۱٫۳۴	۳۰۱٫۹۶	۲۵٫۰۷	۱۴٫۵۳	۷٫۹۴
۱۶	۳۰۰	۱۰۰×۱۰۰	۳۲۲٫۱۵	۴۸۵٫۹۶	۳۸۶٫۳۹	۲۷۹٫۲۰	۳۳٫۷۱	۱۶٫۶۲	۱۵٫۳۸
۱۷	۳۵۰	۱۰۰×۱۰۰	۳۳۶٫۳۲	۴۳۷٫۹۶	۳۹۴٫۲۴	۳۱۵٫۶۰	۲۳٫۲۰	۱۴٫۶۹	۶٫۵۶
۱۸	۴۰۰	۱۰۰×۱۰۰	۳۳۷٫۹۴	۵۳۷٫۸۹	۴۴۸٫۳۸	۲۸۱٫۸۲	۳۷٫۱۷	۲۴٫۶۳	۱۹٫۹۱
۱۹	۴۵۰	۱۰۰×۱۰۰	۳۷۰٫۳۸	۵۲۸٫۷۵	۴۵۳٫۳۴	۳۵۹٫۴۸	۲۹٫۹۵	۱۸٫۳۰	۳۰٫۳۲
۲۰	۵۰۰	۱۰۰×۱۰۰	۴۴۰٫۰۸	۶۵۳٫۷۸	۵۶۳٫۵۷	۴۲۴٫۹۲	۳۲٫۶۹	۲۱٫۹۱	۳٫۵۶
میانگین درصد بهبود (یا خطا)							٪۲۵٫۰۲	٪۱۶٫۶۷	٪۱۲٫۵۱



شکل ۱۶. نمودار طول های به دست آمده از ۴ روش نسبت به تعداد موانع

لازم به ذکر است که با تغییر نحوهٔ تولید جواب های اولیه الگوریتم می توان کارایی الگوریتم را نسبت به حالت کنونی بیشتر کرد چرا که روش RRT جوابهای اولیهٔ نسبتاً خوبی را ارائه می دهد و در نتیجه اثربخشی الگوریتم ژنتیک پیشنهادی بر روی این جواب های اولیه خوب کمتر قابل مشاهده یا توسعه است تا بر روی جواب های اولیه ای که نسبت به مسیرهای RRT بدتر بوده ولی به صورت تصادفی و با هزینهٔ محاسباتی کمتری تولید می شوند.

۵. نتیجه گیری

در این مقاله روشی جدید برای برنامه ریزی حرکت روبات سیار در شرایط غیربهنگام توسعه داده شده است که از تلفیق چند ابزار قدرتمندِ شناخته شده، یعنی الگوریتم ژنتیک، درخت کاوش تصادفی سریع، و مثلث بندی دِلونِی ایجاد شده است. الگوریتم توسعه داده شده کاستی های روش های ژنتیک موجود در ادبیات مانند ثابت بودن طول رشته جواب و محدودیت های عملگرهای جهش و تعویض را رفع نموده و دارای عملگر جدیدی به نام حذف است که سبب هموارتر شدن مسیر می شود.

شبیه سازی الگوریتم پیشنهادی نشان داده است که جواب های به دست آمده در شرایط غیربهنگام نسبت به جوابهای به دست آمده از روش RRT محض به طور متوسط ۲۵٪ بهتر می باشد. همچنین، طول مسیرهای به دست آمده با این روش از مسیرهای تولید شده توسط روش نمودار ورونویی نیز به طور متوسط حدود ۱۷٪ کوتاه تر هستند.

به عنوان موضوع پژوهش آتی، از آنجا که در ادبیات برنامه ریزی حرکت روبات روش های کارای چندانی برای تعیین مسیر بهینه در فضای بیش از دو بعد وجود ندارد، توسعه الگوریتم های متاهیوریستیک در فضاهای بیش از دو بعد اهمیت به سزایی می یابد.

با توجه به نتایج شبیه سازی های انجام شده و روشن شدن قابلیت‌های الگوریتم ژنتیک توسعه یافته در فضای دوبعدی، و با توجه به امکان تولید جواب های اولیه به روش RRT در فضاهای بیش از دو بعدی، امکان مثلث بندی دلونی در فضاهای با ابعاد بالا، و عدم وابستگی عملگرهای الگوریتم به بُعد فضا، می توان الگوریتم ژنتیک ارائه شده را در فضاهای بالاتر نیز توسعه داد. این در حالی است که روش دیدنگار در فضاهای بیش از دو بعد علاوه بر زمان حل بسیار طولانی، تولید جواب بهینه را تضمین نمی‌کند. همچنین، می توان با تعریف شعاع ایمنی و کنترل فاصلهٔ نقاط مسیر تولید شده توسط روش RRT از موانع، روش حاضر را برای برنامه ریزی حرکت روبات های دیسکی نیز توسعه داد.

مراجع

[1] Latombe, J.C., *Robot Motion Planning*, Kluwer Academic Publishers, Boston, 1991.

[2] Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S., *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Boston, 2005.

[3] Masehian, E., Sedighizadeh, D., "*Classic and Heuristic Approaches in Robot Motion Planning: A Chronological Review*", in proceedings of World Academy of Science, Engineering and Technology 2007, pp. 101-106.

[4] Ahuactzin, J.M., Talbi, E.G., "*Using Genetic Algorithms for Robot Motion Planning*", Lecture Notes in Computer Science, Vol. 708. 1991.

[5] Shibata, T., Fukuda, T., "*Coordinative Behavior in Evolutionary Multi-Agent Systems*", in proceedings of IEEE International Conference on Intelligent Robots and Systems 1993 Vol. 1, pp. 448-453.

[6] Wang, J.C., Horng, J.T., Liu, B.J., Fan, K.C., "*A Genetic Algorithm for Structural Query Processing in Hypertext Systems*", in Proceedings of the International Conference on Evolutionary Computation 1996, pp. 506-511.

[7] Tadokoro, S., Hayashi, M., Manabe, Y., "*On Motion Planning of Mobile Robots Which Coexist and Cooperate with Human*", in proceedings of the IEEE International Conference on Intelligent Robots and Systems, Vol. 2, 1995, pp. 518-523.

[8] Li, Q., Zhang, W., Yin, Y., Wang, Z., Liu, G., "*An Improved Genetic Algorithm of Optimum Path Planning for Mobile Robots*", in Proceedings of the 6th International Conference on Intelligent Systems Design and Applications 2006.

[9] Mahjoubi, H., Bahrami, F., Lucas, C., "*Path Planning in an Environment with Static and Dynamic Obstacles Using Genetic Algorithm: A Simplified Search Space Approach*", in proceedings of IEEE Congress on Evolutionary Computation 2006.

[10] Manikas, T.W., Ashenayi, K., Wainwright, R.L., "*Genetic Algorithms for Autonomous Robot Navigation*", IEEE Instrumentation & Measurement Magazine, Vol. 10, No. 6, Dec. 2007, pp. 26-31.

[11] Hu, X., Xie, C., "*Niche Genetic Algorithm for Robot Path Planning*", in proceedings of the 3rd International Conference on Natural Computation ICNC 2007.

[12] Holland, J.H., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[13] LaValle, S.M., Kuffner, J.J., "*Rapidly-Exploring Random Trees: Progress and Prospects*". In B.R. Donald, K.M. Lynch, and D. Rus, editors, Algorithmic and Computational Robotics: New Directions, pp. 293-308. A K Peters, Wellesley, MA, 2001.

[14] Hjelle, *Ó*., Daehlen, M., *Triangulations and Applications*, Springer, Berlin Heidelberg, 2006.

[15] de Castro, L.N., Von Zuben, F.J., *Recent Developments in Biologically Inspired Computing*, Idea Group Publishing, 2004.