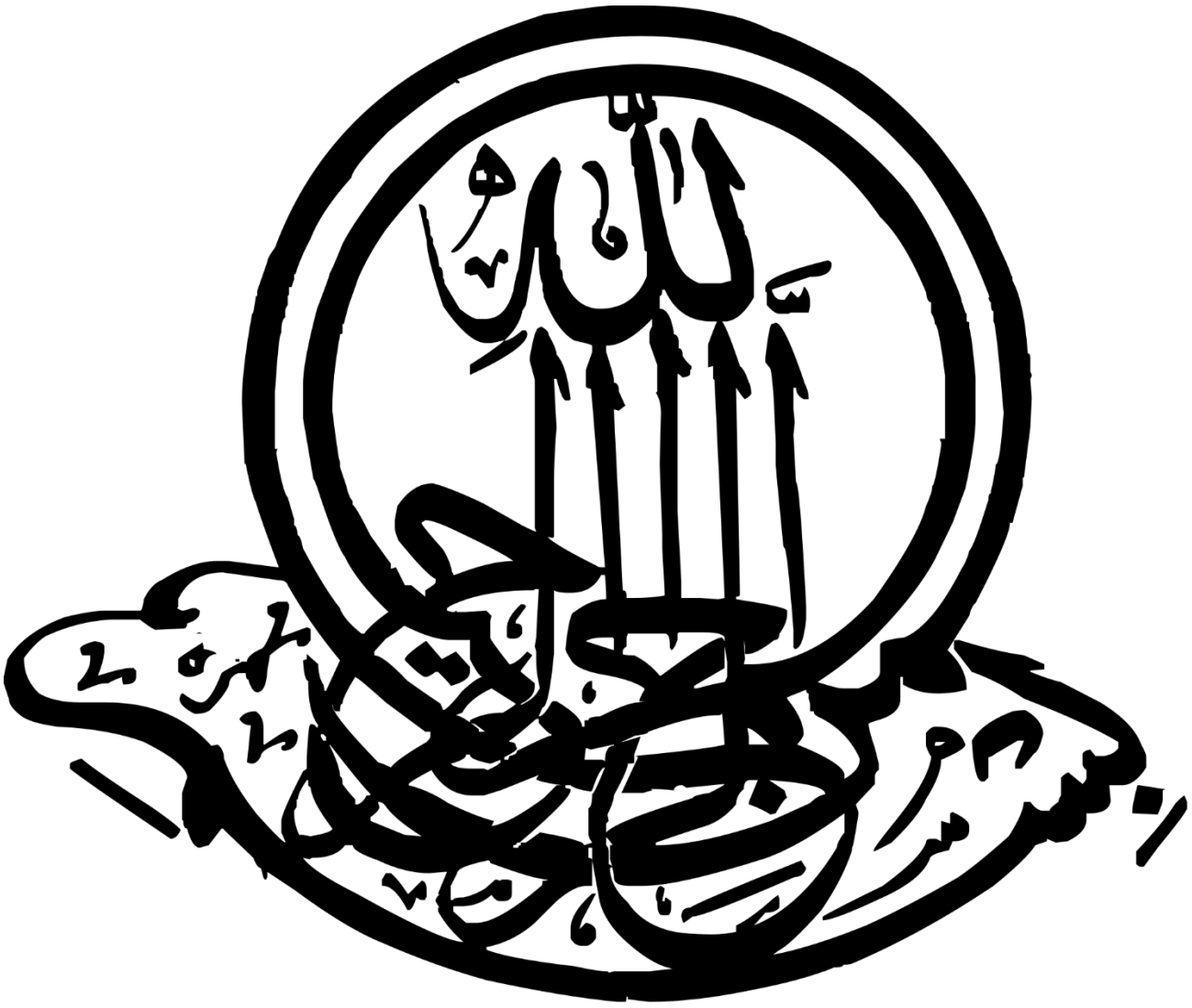



# آشنایی با رویکردهای فرا ابتکاری (تکاملی)، برای مسئله برنامه‌ریزی مسیر ربات‌های متحرک

رضا عباسی

شماره دانشجویی

۹۵۳۰۱۰۳

استاد راهنما

دکتر حسین فلسفین

**بهمن ماه سال ۱۳۹۹**

# فهرست مطالب

# چکیده

با پیشرفت روزافزون عمل و تکنولوژی، میزان استفاده از ربات در زمینه‌های مختلف رشد چشم گیری‌داشته است. از ربات‌ها میتوان در زمینه‌های گوناگون مثل جست‌وجو برای یافتن افراد و اشیا در هنگام وقوع حوادث طبیعی، جست‌وجستو در محیط برای یافتن نقاط مناسب انجام یک‌کارخاص، جابه‌جایی افراد بین دو نقطه، جابه‌جایی اشیا در کارخانه‌جات، جست‌وجو برای کسب اطلاعات از محیط و... استفاده کرد. مسیریابی ربات‌های متحرک یکی از مباحث اساسی در پژوهش‌های علمی در حوزه رباتیک می‌باشد. در این پروژه در مورد انواع مسائل مسیریابی و الگوریتم‌های مسیریابی برای ربات‌ها صحبت شده است. تمرکز بیشتر این پروژه بر روی الگوریتم‌های برگرفته شده از طبیعت می‌باشد. الگوریتم‌هایی که درعین سادگی و انجام یک سری کار ساده در یک حلقه تکرار، نتایج فوق العاده ای را ارائه می‌دهند.در این پروژه بر روی سه نمونه از این دسته الگوریتم‌ها به طور ویژه بحث صورت گرفته‌است و به طور کامل شرح‌داده شده اند. همچنین چندمحیط تعریف گردیده است و این سه الگوریتم بر روی این محیط‌ها آزمایش شده‌اند و مقایسه‌هایی بر روی آنها صورت گرفته شده تا با مزایا و معایب هریک نسبت به دیگری، آشنایی صورت گیرد.

# فصل اول: مقدمه و مرور فصل های پروژه

## مسیریابی ربات

با توجه به استفاده روزانه از انواع ربات‌های مسیریاب زمینی، پرنده و شناگر در کاربردهای متفاوت، بحث مسیریابی[1] این ربات‌ها جهت انجام ماموریت‌های مختلف از مهمترین مباحث پژوهشی و عملی در این حوزه است. از آنجا که بسیاری از ربات‌ها بدون کمک انسان کار می‌کنند یا این که بسته به شرایط و محیطی که ربات در آن کار می‌کند، عامل انسانی نمیتواند در مسیریابی ربات دخالت کند، نیاز است که ربات‌ها به صورت خوکار مسیریابی را از نقطه مبدا تا مقصد انجام دهند. همچنین گاهی اوقات با توجه کاربرد ربات نیاز داریم تا بهترین مسیر ممکن یا لااقل یکی از بهترین مسیرهای ممکن را برای ربات پیداکنیم. به عنوان مثال نیاز داریم تا ربات‌های شناگر در دریا، کم خطر ترین و در عین حال کوتاه ترین مسیر ممکن را پیدا کرده و پیمایش کنند. زیرا هم در دریا خطراتی مثل گرداب‌ها وجود دارد که میتواند باعث نابودی ربات شود و هم سوخت ربات محدود است و باید قبل از اتمام سوخت، وظیفه اش را انجام داده باشد.

تاکنون پژوهش‌های مختلفی در این زمینه صورت گرفته است و الگوریتم‌های مختلفی برای طراحی مسیر ربات ها به‌دست آمده است که ما در این پروژه قصد داریم روی سه مورد از آنها تمرکزکرده و آنها را بیان کرده و با یکدیگر مقایسه کنیم.

## مروری بر فصل های پروژه

در این بخش، شرح مختصری از هر یک از فصل‌های پروژه به صورت جداگانه ارائه گردیده است.

در فصل دوم انواع مسائل مسیریابی بررسی شده‌است و براساس معیارهای مختلف، دسته‌بندی‌های گوناگون مسائل مسیریابی مطرح شده‌است.

در فصل سوم یک دسته‌بندی کلی از الگوریتم‌های مسیریابی مخصوص ربات‌ها و محرک‌ها بیان شده است و برای هر دسته، چند نمونه الگوریتم ذکر و توضیحاتی ارائه شده است.

در فصل چهارم وارد بخش اصلی پروژه شده و متناسب با انواع مسائل و انواع الگوریتم‌هایی که در فصل‌های گذشته در مورد آنها توضیحاتی ارائه شد، مسئله و الگوریتم‌هایی که قرار است در این پروژه روی آنها کار شود، بررسی و شرح داده شده است. یک تعریف مسئله بیان شده و سپس در مورد معیارهایی که قرار است در این پروژه، الگوریتم‌ها را بر اساس آنها، بررسی کنیم توضیحاتی ارائه شده است.

[1] Path planning

در فصل پنجم ، توضیحاتی در رابطه با نحوه تولید مسیر و شیوه محاسبه تولید مسیر و اعمال جریمه برای مسیرهای که با موانع برخورد دارند، ارائه شده است.

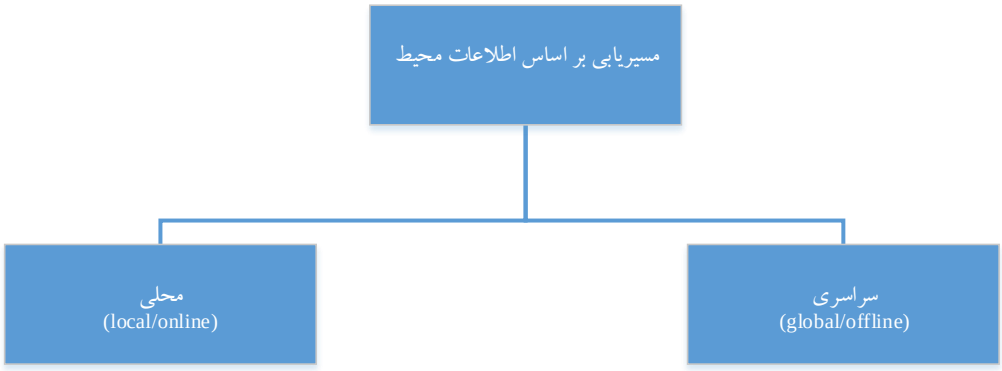
در فصل ششم، در مورد سه الگوریتمی که در این پروژه قرار است به آنها پرداخته شود، توضیحات کاملی ارائه می‌شود.

در فصل هفتم، پارامتر های تخصصی الگوریتم های بیان شده در فصل ششم مرور شده و در مورد مقادیر مناسبی که در مقالات معتبر در مورد آنها ثبت شده، توضیحاتی ارائه می‌شود. همچنین در بخش دوم این فصل بعد از معرفی مختصر الگوریتم ژنتیک به عنوان یک معیار مناسب ، نتایج آزمایش‌هایی که برای تنظیم پارامتر برای مسئله مسیریابی برای سه الگوریتم انجام داده‌ایم بیان می‌شود. همچنین کلیه نتایج را در پیوست گزارش ثبت شده است.

در فصل هشتم نیز، سه الگوریتم بر روی محیط های مختلف اجرا و آزمایش شده است و کار مقایسه سه الگوریتم صورت گرفته است.

# فصل دوم: دسته‌بندی مسائل مسیریابی ربات

برای تقسیم بندی انواع مسائل مسیریابی، دسته‌بندی‌های متفاوتی را میتوان در نظر گرفت. اولین سطح از دسته‌بندی مسائل مسیریابی را میتوان دسته‌بندی براساس اطلاعات محیط در نظر گرفت. در واقع بر این اساس، همان طور که از نمودار شکل ۲-۱مشخص است، دو مدل مسیریابی داریم. مسیر یابی سراسری[1] و مسیریابی محلی[2] که با نام های برون خط[3] و برخط[4] نیز شناخته می‌شوند. در مسیر یابی سراسری یا برون خط، اطلاعات کامل و دقیقی از محیط مد نظر موجود است و بر اساس آن یک مسیر از نقطه شروع تا انتها، به‌دست می‌آید و ربات یا محرک، این مسیر را طی خواهد کرد. در مسیریابی به صورت محلی یا برخط، اطلاعات کاملی از محیط موجود نیست و ربات مد نظر، اطلاعات خود را به صورت لحظه‌ای و به کمک سنسورهایی که به آنها مجهز است به‌دست می‌آورد و در نتیجه مسیر خود را به صورت لحظه‌ای و برای مسافت های کم به‌دست می‌آورد و سپس دوباره اطلاعاتی را به کمک سنسورها از محیط اطراف خود به دست آورده و دوباره مسیریابی خواهد کرد[۱].

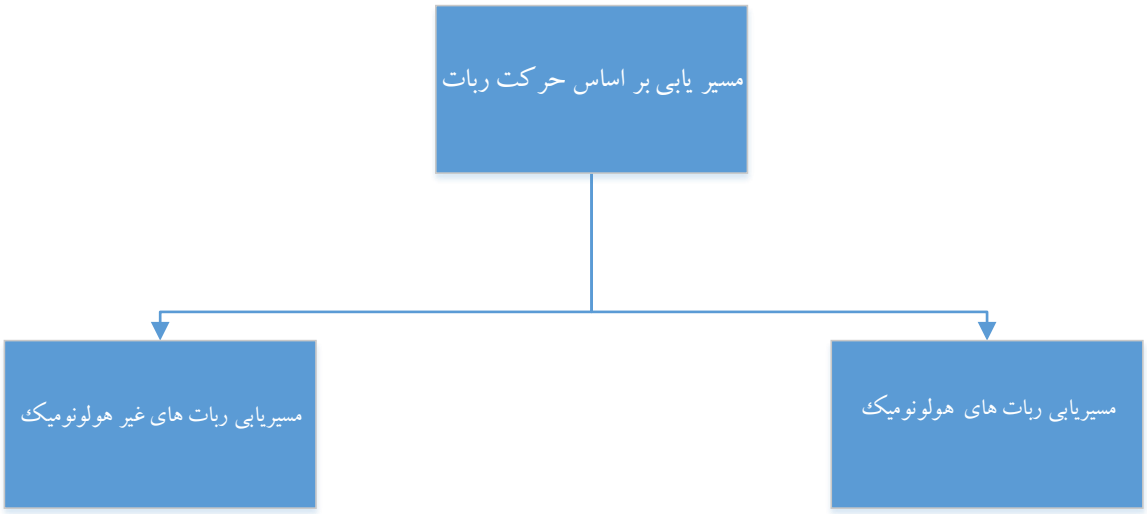



شکل ۲-۱: دسته‌بندی بر اساس اطلاعات در دسترس

یکی دیگر از مواردی که باعث ایجاد تمایز در مسائل مسیریابی ربات می‌شود، میزان آزادی ربات در حرکت خود می‌باشد. همان طور که از نمودار شکل ۲-۲ مشخص است، از لحاظ میزان آزادی ربات‌ها در حرکت، آنها به دو نوع هولونومیک[5] و غیر هولونومیک[6] دسته‌بندی می‌شوند. در دسته هولونومیک، ربات یا محرک مورد نظر، به صورت آزادانه میتواند در هر جهت حرکت کند. مانند ربات‌ها با بازوهای محرک در خط تولید کارخانه‌ها. در دسته غیر هولونومیک،

[1] Global
[2] Local
[3] Offline
[4] Online
[5] holonomic
[6] non-holonomic

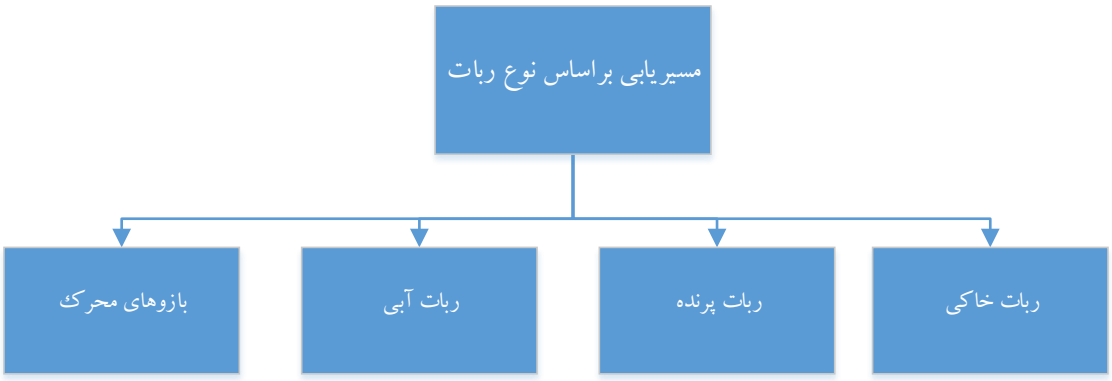
ربات دارای حرکت محدود می‌باشد. یعنی نمی‌تواند به صورت مستقیم در هرجهتی که می‌خواهد حرکت کند. برای مثال، یک اتوموبیل می‌تواند رو به جلو و عقب و نیز تا حداکثر زاویه‌ی خاصی به سمت چپ و یا راست حرکت کند و نمی‌تواند به صورت مستقیم در جهت چپ یا راست حرکت کند. مسیریابی برای این دو مدل ربات‌ها متفاوت است. زیرا اگر ربات غیر هولونومیک باشد، باید در مسیریابی به این نکته توجه کرد که نمی‌توان مسیرهایی را انتخاب کرد که دارای شکستگی های تند هستند. زیرا ربات ما توانایی دنبال کردن این مسیر را ندارد و حتی در صورت دنبال کردن، ممکن است به حالت ناپایدار در بیاید[۲].




شکل ۲-۲: دسته‌بندی مسئله مسیریابی بر اساس حرکت ربات

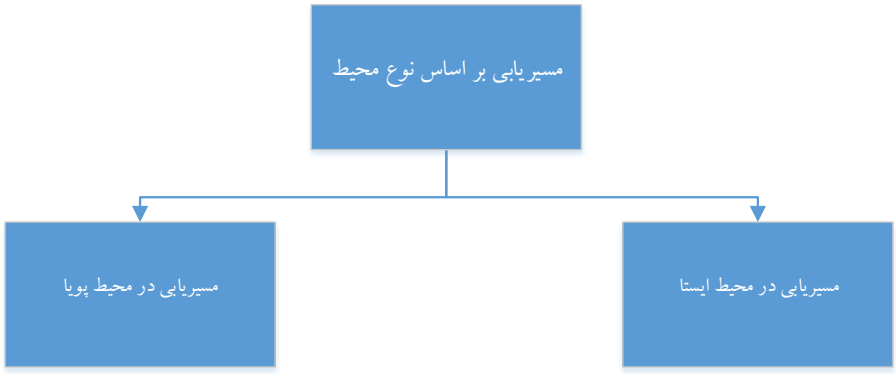
یک دسته‌بندی دیگر برای مسائل مسیریابی، دسته‌بندی بر اساس نوع محرک می‌باشد. همان‌گونه که از نمودار شکل ۲-۳ مشخص است، بر این اساس ربات‌ها به دسته‌های ربات‌های بازو محرک، ربات‌های پرنده، ربات‌های آبی و ربات‌های خاکی تقسیم بندی می‌شوند که هرکدام در محیطی با شرایط خاص حرکت می‌کنند. برای مثال، ربات های آبی مانند ربات های که روی سطح آب و یا داخل آب حرکت می‌کنند، در محیطی در حال حرکت هستند که این محیط دارای گرداب‌ها و چاله‌های دریایی است که علاوه بر این که باید سعی کنند که به موانع برخوردی نداشته باشند، باید مراقب باشند که از این موارد نیز دوری کنند. علاوه بر آن، امواج و باد ها بسیار روی حرکت آنها تاثیر می‌گذارد و می‌تواند آنها را از مسیرشان دور کند. به عنوان مثالی دیگر، ربات‌های پرنده مثل پهپادهای نظامی نیز شرایط خاص خود را دارند. برای مثال، آنها علاوه بر تشخیص موانع و جلوگیری از برخورد با آنها، باید تلاش کنند که در رادرهای که دشمن

قرار نگیرند و شناسایی نشوند[۲]. همچنین باید سعی کنند که در یک ارتفاع خاص پرواز کنند تا قابل ردگیری نباشند. این پهبادها دارای یک حداقل سرعت می‌باشند که نباید سرعتشان از آن میزان کمتر گردد یا این که با یک زوایه ی خاصی می‌توانند اوج بگیرند و یا فرود بیایند. همه این موارد باعث میشود که در مسیریابی شان یک سری محدودیت لحاظ کنند و نتوانند هر مسیری را برای حرکت انتخاب کنند[۳].

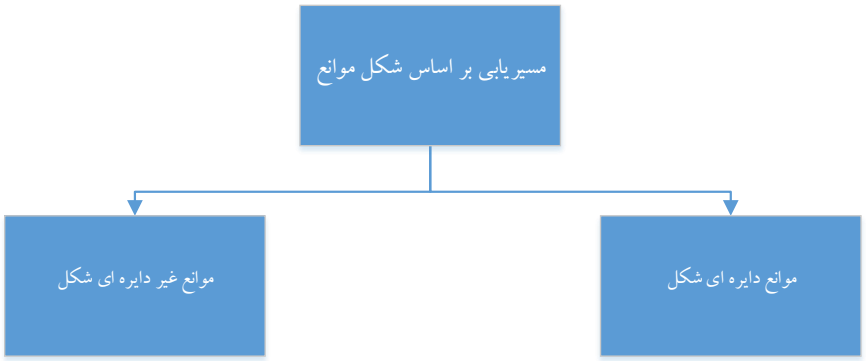



شکل ۲-۳: دسته بندی مسئله مسیریابی براساس نوع ربات

یک مدل دسته‌بندی دیگر برای مسائل مسیریابی، دسته‌بندی براساس نوع محیط است. همان‌طور که از نمودار شکل ۲-۴ مشخص است، میتوان محیط و به طبع آن مسائل مسیریابی را به دو دیگر دسته تقسیم بندی کرد. در واقع این که محیط ایستا باشد و یا پویا، این که موانع در این محیط در حال حرکت هستند و یا این که در جای خود ثابت می‌مانند[۱]. حتی نوع و شکل موانع نیز در نوع مسیریابی تاثیر می‌گذارد. با توجه به دسته‌بندی که در نمودار شکل ۲-۵ نیز میتوان مشاهد کرد، موانع می‌توانند به شکل دایره تصور گردند و یا این که به صورت اشکالی منتظم و غیر منتظم درنظر گرفته شوند. اگر موانع به شکل دایره در نظرگرفته‌شوند، برای جلوگیری از برخورد به موانع کافیست مختصات مرکز مانع و اندازه شعاع آن موجود باشد و سعی شود که فاصله ربات از فاصله مرکز مانع کمتر شعاع نشود و در واقع یک نسخه ساده شده از حالت غیر دایره است. اگر موانع شکلی غیره دایره در نظر گرفته شوند، به تبع مسیریابی و جلوگیری از برخورد به موانع کار سخت‌تری خواهد بود. البته این موردی که گفته شد بیشتر برای مسیریابی آفلاین و بدون سنسور می‌باشد. در مسیریابی آنلاین و همراه با سنسور ، ربات به کمک سنسور از برخورد با موانع جلوگیری می‌کند و شکل مانع برای ربات مهم نخواهد بود[۱] [۴].

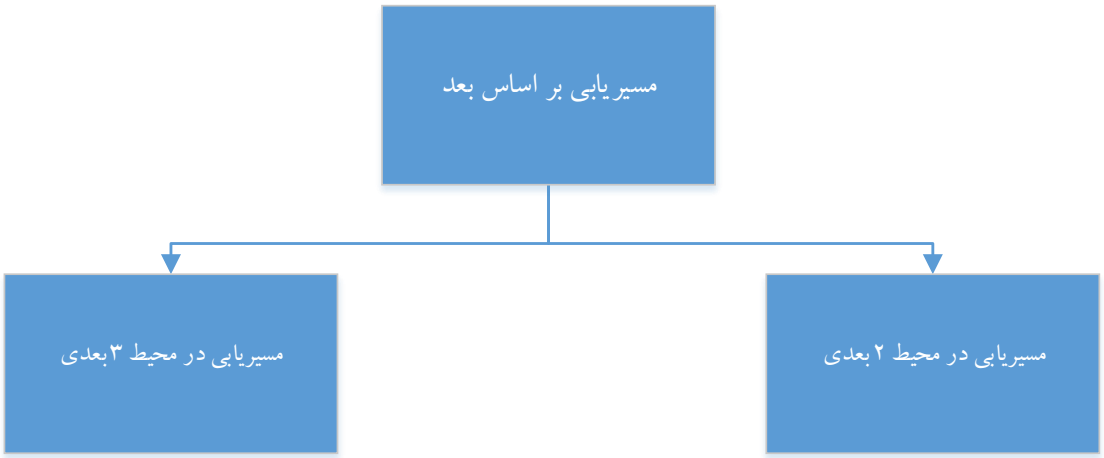



شکل ۲-۴: دسته بندی مسائل مسیریابی بر اساس نوع محیط




شکل۲-۵: دسته بندی مسائل مسیریابی بر اساس شکل موانع

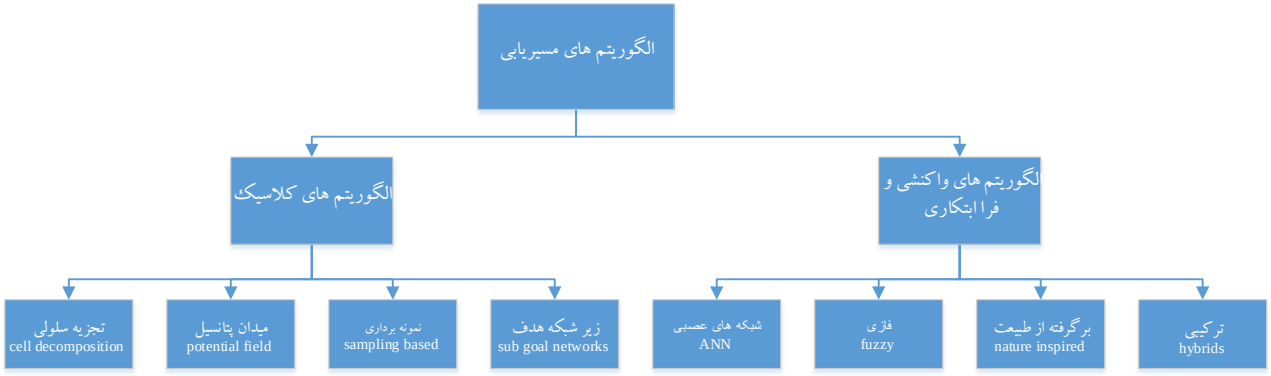
نوع دیگر دسته‌بندی مسیریابی ربات‌ها بر اساس این است که مسیریابی را در محیط ۲ بعدی در نظر بگیریم و یا در یک محیط ۳ بعدی که در نمودار شکل ۲-۶ نیز این دسته بندی قابل مشاهده است. این نوع مسائل بیشتر برای ربات‌های پرنده مثل بهبادها مطرح می‌شود که بحث ارتفاع ربات نیز مدنظر و مهم می‌باشد. اگر در مسیر یابی، ارتفاعی که ربات در آن پرواز می کند مهم باشد، مسئله در فضای سه بعدی در نظر گرفته می‌شود. و اگر ارتفاع مهم نباشد، میتوان مسئله را در فضای دو بعدی در نظر گرفت و حل کرد. البته می‌تواند مسائل در محیط سه بعدی را به دو بعدی تبدیل کرد. به این صورت که در هر مرحله از حرکت ربات ، یکی از محور ها را ثابت در نظر گرفت[۲] [۳].




شکل ۲-۶: دسته بندی مسائل مسیریابی براساس بعد محیط

# فصل سوم: انواع الگوریتم‌های مسیریابی ربات

الگوریتم‌های مسیریابی ربات به صورت کلی به دو دسته واکنشی یا فرا ابتکاری[1] و کلاسیک تقسیم بندی می‌شوند. الگوریتم‌های کلاسیک، دارای محاسبات سنگینی هستند و بهترین مسیر را به دست می‌آورند . در حالی که الگوریتم های فرا ابتکاری داری محاسبات سبک‌تری هستند ولی همیشه بهترین مسیر را به‌دست نمی‌آورند. با این حال مسیری نزدیک به بهترین مسیر را به‌دست می‌آورند. این الگوریتم‌ها برای مواقعی که نیاز است تا در مدت زمان کم و با یک تقریب خوب یک مسیر بهینه پیدا شود، مورد استفاده قرار می‌گیرند. نمودار شکل ۳-۱ زیر یک دسته بندی کلی از این الگوریتم ها را به نمایش می‌گذارد. البته در این دسته‌بندی تنها موارد مهم و کاربردی بیان شده است و همه زیر دسته‌های هر کدام از دسته ها عنوان نشده است[۴] .




شکل ۳-۱: دسته بندی الگوریتم های مسیریابی

## الگوریتم‌های کلاسیک

الگوریتم‌های کلاسیک ، روش‌هایی هستند که بهترین مسیر ممکن را پیدا می‌کنند و یا اثبات می‌کنند که مسیر مناسبی وجود ندارد. یکی از اشکالات این دسته از الگوریتم‌ها، محاسبات سنگین آنها می‌باشد. همچنین این الگوریتم‌ها توانایی لازم برای مسیریابی در محیط‌های نامطمئن را ندارند. در ادامه، هر کدام از این الگوریتم‌های کلاسیک را به صورت مختصر شرح داده شده است.

[1] metaheuristic

## روش تجزیه سلولی

ایده اصلی روش تجزیه سلولی[1] تقسیم فضای حرکت ربات به بخش‌های کوچک‌تر و در نتیجه آسان‌تر کردن مسیریابی می‌باشد. در این روش، فضای آزاد ربات را به چندین قسمت به نام سلول تقسیم می‌شود. سپس مسیر بین هر دو نقطه از یک سلول ترسیم شده و در نهایت یک گراف متصل تعریف می‌شود که روابط بین سلول ها را مشخص می‌کند. سپس بین هر دو نقطه از دو سلول مختلف، زمانی مسیر تشکیل می‌شود که درگراف مرتبط، دو سلول همسایه یکدیگر باشند. در این صورت مسئله مسیریابی به یک مسئله جست و جو در فضای گراف تبدیل می‌شود[۱] [۴] .

به طور خلاصه مراحل این روش به صورت زیر است.

- تقسیم فضای جست‌وجو به مناطقی به نام سلول
- ساخت یک گراف متصل بر اساس سلول‌ها
- مشخص کردن دو سلول به عنوان نقطه شروع و نقطه پایان
- مسیریابی بر روی گراف تشکیل شده

## روش میدان پتانسیل

در روش میدان پتانسیل[2] ربات یک محرک در نظر گرفته می‌شود. از طرفی مقصد مورد نظر به عنوان یک نیروی پتانسیل جذب کننده و موانع به عنوان یک نیروی پتانسیل دفع کننده در نظر گرفته می‌شوند. در نتیجه مجموع این نیرو های پتانسیل به عنوان یک نیرو روی حرکت ربات تاثیر می‌گذارد تا ربات به مقصد خود برسد. این روش برای سناریوهایی که در آن به جای یک ربات ، چندین ربات وجود دارد نیز مناسب است [۱].

## روش نمونه‌برداری

الگوریتم های مبتنی بر نمونه برداری[3] به دلیل توانایی شان در محیط های پیچیده و آنی[4]بودن آنها ، بسیار مورد توجه قرارگرفته اند. پاسخ‌گویی در زمان کوتاه، عدم نیاز به تعریف دقیق محیط و قابلیت به کارگیری در ربات‌ها

[1] Cell Decomposition
[2] Potential Method
[3] Sampling Based
[4] Real-time

مختلف ازجمله مزایای این مدل الگوریتم‌هاست. دو نمونه از این الگوریتم ها، نقشه راه احتمالی[1] و درخت جست‌وجوی تصادفی سریع[2] می‌باشند.

ایده اصلی الگوریتم نقشه راه احتمالی، نمونه گیری به صورت رندم از فضا جست جو و بررسی نمونه ها از لحاظ آزاد بودن یا در مانع قرار داشتن آنها و استفاده از یک جست‌وجوگر محلی برای اتصال این نمونه‌ها به یک دیگر است. در واقع این الگوریتم از دو فاز تشکیل می‌شود. در فاز اول ابتدا نمونه ها به صورت رندم تولید می‌شود. سپس برای هرنمونه، k نمونه نزدیک به آن و یا تمام نمونه های با فاصله کمتر از n از نمونه فعلی به یک دیگر متصل می‌شوند. سپس نقاط ابتدا و پایان مسیر نیز به این نمونه ها اضافه می‌شوند و در نهایت به کمک الگوریتم‌های جست‌وجوی گراف، بهترین مسیر بین نقطه شروع و پایان پیدا خواهد شد. در این روش برای تولید گراف اولیه از مسیرها از دو روش نمودار ورنوی[3] و گراف پدیداری[4] استفاده می‌شود. مسیری که به کمک گراف پدیداری به دست می آید، بهترین و کوتاه ترین مسیر است ولی یک مشکل دارد. آن هم این است که این مسیر بیش از حد به موانع نزدیک است که این باعث پایین آمدن امنیت مسیر برای ربات می‌شود. همچنین در محیط های بزرگ دارای پیچیدگی محاسباتی بسیار بالا می‌باشد [۵]. در روش نمودار ورنوی این مشکل برطرف گردیده است و مسیر به دست آمده امنیت بیشتری نسبت به روش اول دارد زیرا بر اساس فاصله بین موانع، مسیر را تولید می‌کند. با توجه به این که در این روش سعی می‌شود که مسیر تا حد ممکن از موانع دور باشد، در نتیجه مسیر به‌دست آمده بهینه نخواهد بود.[۶] [۷].

الگوریتم درخت جست و جوی تصادفی سریع با استفاده از جست و جوی تصادفی به حل مسائل مسیریابی می‌پردازد. در این روش یک نقطه به صورت رندم به عنوان ریشه درخت انتخاب میشود. اگر این نقطه در فضای موانع نبود، سپس یک نقطه دیگر انتخاب خواهد شد. با انتخاب نقطه از فضای آزاد ، در راستای خط واصل بین این نقطه و ریشه درخت، نقطه جدیدی تحت عنوان Xnew که به اندازه گام ربات از ریشه فاصله دارد، انتخاب می‌شود. اگر بتوان Xnew را با پاره خطی که تماما به فضای آزاد تعلق دارد، به ریشه وصل کرد، Xnew به عنوان گره جدید به درخت اضافه می‌شود[۸] .

[1] Probabilistic roadmap
[2] Rapidly-exploring-random-trees
[3] Voronoi Diagram
[4] Visibility Graph

## الگوریتم‌های واکنشی یا فرا ابتکاری

این الگوریتم‌ها به دوگروه کلی روشهای مبتنی بریک جواب و مبتنی برجمعیت تقسیم می‌شوند. الگوریتم های مبتنی بریک جواب درحین فرایند جستجو یک جواب را تغییر میدهند. درحالیکه الگوریتم های مبتنی برجمعیت درحین جست‌وجو یک جمعیت ازجواب‌ها رادرنظر می گیرند. در ادامه به شرح مختصر از این الگوریتم‌ها والگوریتم های مشابه آنها پرداخته شده است.

### شبکه عصبی مصنوعی

شبکه های عصبی مصنوعی[1]به دلیل توانایی در ارائه راه‌حل‌های ساده و بهینه، به شکل گسترده ای در مسائل بهینه سازی، یادگیری و تشخیص الگو به کار گرفته می‌شود. شبکه عصبی در واقع شامل یک سری گره یا نورون[2]با توانایی یادگیری و ذخیره تجربه و به کارگیری آن می‌باشد. عنصر اصلی شبکه عصبی درواقع نورون‌ها هستند که پردازش‌های ساده ای انجام می‌دهند. این نورون ها به یک دیگر متصل شده و شبکه عصبی را تشکیل می‌دهند. روش های مبتنی بر شبکه عصبی را میتوان بر اساس عوامل مختلف دسته بندی کرد. برای مثال دسته‌بندی بر اساس ساختمان لایه ها که به تک لایه و چند لایه تقسیم می‌شوند. یا بر اساس روش آموزش آنها که به مواردی همچون یادگیری تحت نظارت، یادگیری بدون نظارت، وزن ثابت و خود نظارت‌گر تقسیم بندی میشوند.روش اتصال نورن ها، روش مورد استفاده برای تعیین وزن ها و همچنین توابع فعال ساز آنها، از موارد دیگر برای دسته بندی می‌باشد. شبکه عصبی در مسیریابی ربات به سه بخش تفسیرداده‌های گرفته شده از سنسورها، مسیریابی و جلوگیری از برخورد به موانع تقسیم می‌شود. برای مسیر یابی ربات، استفاده از شبکه های عصبی در کنار الگوریتم های دیگر مانند fuzzy logic و الگوریتم های تکاملی بسیار خوب و کارآمد جواب می‌دهد.

### منطق فازی

انسان این توانایی را دارد که بدون شناخت دقیق وکامل از محیط خود، مسیریابی کند و به مقصد خود برسد. این توانایی بسیار مطلوبی است که میتوان از آن در مسیریابی ربات خود از آن استفاده کرد. در الگوریتم منطق فازی، مسیریابی ربات بر اساس مجموعه از اگر-آنگاه[3]ها انجام می‌شود. همچنین برای پیاده سازی ساده‌ترآن، مسئله حرکت ربات به مجموعه‌ای از کار ها و رفتارهای ساده‌تر تقسیم می‌شود. مدل های مختلفی برای حل مسئله مسیریابی ربات به کمک منطق فازی ارائه شده است. برای مثال دو محقق به نام‌های چانگ[4] جین[5] یک مدل از منطق فازی برای حل مسئله

[1] Artificial neural network
[2] Neuron
[3] If-then
[4] H. Chang
[5] T. Jin

مسیر یابی را ارائه داده اند که در این مدل، سه هدف دنبال می‌شود: جهت‌گیری ربات ، نزدیکی به هدف و حرکت چرخشی که این سه هدف در تابع هزینه مسئله گنجانده شده است که با بهینه سازی این تابع، سعی در یافتن مسیری بهینه برای ربات می‌شود. در واقع در اکثر راه‌حل‌های این روش، یک سری توابع برای مسیریابی تعریف می‌شود که سعی می‌شود به کمک الگوریتم هایی ، این توابع را بهینه کنند که منجر به یافت مسیر بهینه نیز می‌شود[۴] .

## الگوریتم های برگرفته شده از طبیعت

از الگوریتم ها برگرفته شده از طبیعت[1]برای بهینه سازی توابعی که برای طرح مسئله مسیریابی تعریف شده‌است ، استفاده می‌شود. پیش از پیاده سازی هر الگوریتم، ابتدا باید مسئله‌هایی که قرار است بهینه سازی شود، برای آن تعریف گردند. در مسئله طراحی مسیر، اطلاعات مورد نیاز الگوریتم، مثل مشخصات محیط، تابع هدف، تابع هزینه و قید مسئله باید تعیین شوند. از تابع هزینه به منظور ارزیابی پاسخ به دست آمده توسط الگوریتم، استفاده می‌شود. درمسئله مسیریابی، هزینه همان طول مسیر است.

ویژگی‌های محیط حرکت ربات، از پایه‌ای‌ترین اطلاعاتی است که باید در اختیار الگوریتم قرار گیرد. زیرا اساس کارکرد الگوریتم‌های هوشمند، برمبنای جستجوی تصادفی در یک فضای جستجو با هدف پیدا کردن پاسخ بهینه می‌باشد. الگوریتم های برگرفته از طبیعت کمک می‌کنند تا تابع هدف را ماکزیمم و یا تابع هزینه مینیمم مقدار خود را پیدا کنند. الگوریتم‌های برگرفته شده دارای ویژگی‌هایی هستند که در ادامه شرح داده شده‌است.

- سادگی و غیرمنتظره بودن: استراتژی و محاسبات غالباً بسیار ساده هستند ولی نتایج آنها بسیار چشمگیر است که نشان‌دهنده غیرمنتظره بودن آنها می‌باشد.
- مقاوم بودن و استحکام: این الگوریتم‌ها از استحکام و مقاومت زیادی در برابر تغییرات محیط، پارامترها و وظایف برخوردار هستند. به عبارت دیگر، این الگوریتم‌ها قابل اجرا و انعطاف‌پذیرند.
- خودسازماندهی: این الگوریتم‌ها تطبیق، خودآموزی و خودسازماندهی دارند و میتوانند مسیر تکامل را تشخیص دهند[۸].

الگوریتم‌هایی مانند الگوریتم تراکم ذرات، کلونی زنبور ها و حشرات شب‌تاب از الگوریتم‌های مهم این دسته هستند که در فصل های آینده به صورت مفصل در مورد آنها توضیحاتی ارائه داده خواهد شد.

[1] nature inspired

## الگوریتم های ترکیبی

در مسئله مسیریابی، هر الگوریتمی مزایا و معایب خود را دارد. برای مثال برخی از آنها دارای محاسبات سنگینی هستند ولی در عوض بهترین راه ممکن را ارائه می‌دهند. برخی نیز محاسبات سبک تری دارند ولی تضمین نمی‌دهند که بهترین جواب را برمی‌گردانند. برخی برای محیط های ایستا مناسب هستند و برخی برای محیط پویا. برخی برای جست وجوی محلی مناسب هستند و برخی برای جست‌وجوی سراسری خوب عمل می‌کنند. از این رو ، برخی محققان تصمیم گرفتند تا از ترکیبی از این الگوریتم‌ها استفاده کنند تا بتوانند معایب و کاستی‌های هر الگوریتم را به کمک دیگری برطرف کرده و پاسخ‌های بهتری به‌دست بیاورند. از جمله این الگوریتم‌های ترکیبی[1]میتوان به ترکیب شبکه عصبی و منطق فازی، ژنتیک و منطق فازی، ژنتیک و تراکم ذرات ، منطق فازی و تراکم ذرات ، منطق فازی و کلونی مورچگان و ... اشاره کرد که هرکدام در شرایط متفاوت ، پاسخ هایی به مراتب بهتر از الگوریتم های جداگانه می‌دهد[۴].

[1] hybrids

# فصل چهارم: تعریف مسئله و طراحی محیط مسئله

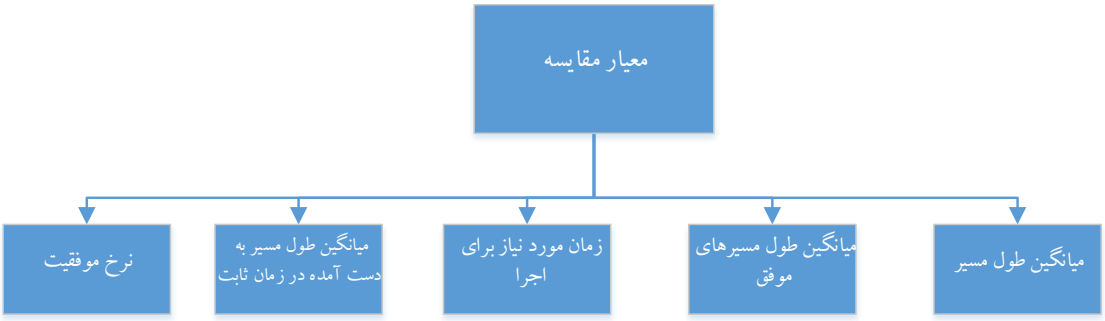
دسته‌بندی‌های مختلف از انواع مسائل مسیریابی مطرح شد. همچنین انواع الگوریتم‌ها برای حل این نوع مسائل نیز مطرح گردید. در این پروژه، قصد داریم که مدل مسیریابی سراسری یا همان برون خط را مد نظر قراربدهیم. در واقع قصد داریم در محیطی مسیریابی خود را انجام دهیم که این محیط برای ربات یا محرک ما شناخته شده است. همچنین قرار است موانع را به صورت دایره ای شکل در نظر گرفته و جست‌وجو خود را در فضای دو بعدی انجام دهیم.در ادامه بیشتر در مورد محیطی که قرار است مسیریابی در آن صورت بگیرد، صحبت خواهیم کرد.

همان طور که قبلا گفته شد، الگوریتم‌های برگرفته شده از طبیعت ویژگی‌های جذابی مانند سادگی، غیر منتظره بودن و مقاوم بودن در برابر تغییرات را دارند که به دلیل وجود همین ویژگی‌های جذاب ، از این دست الگوریتم ها برای حل مسائل مسیریابی استفاده می‌شود. تعداد بسیار زیادی الگوریتم برگرفته شده از طبیعت وجود دارد که در این پروژه به بررسی و مقایسه سه الگوریتم ازدحام ذرات[1] الگوریتم کلونی زنبور ها[2] و الگوریتم حشرات شب‌تاب[3] می‌پردازیم و آنها را از جنبه های مختلف مورد بررسی قرار می‌دهیم. در واقع ما قرار است بر روی یک سری محیط مشخص، به تعداد ثابت الگوریتم های ذکر شده را اجرا کنیم و مسیرهایی که این الگوریتم ها به عنوان بهترین مسیر به ما می دهند را مورد بررسی قرار دهیم. در این آزمایش یک سری پارامتر مشترک بین الگوریتم ها وجود دارد. مانند تعداد جمعیت اولیه پاسخ ها، تعداد تکرار ها[4] محدوده‌ی تغییرات x و y و ... که در همه آنها اعدادی مشخص و ثابت در نظر گرفته می‌شود تا بتوان مقایسه را عادلانه تر انجام داد. همچنین، هر کدام از این الگوریتم ها یک سری پارامتر منحصر به فرد دارند. این پارامترها یا به کمک مقالات معتبر که در این زمینه کار کرده‌اند و یا به صورت دستی و تجربی تنظیم می‌شوند که فصل‌های آینده به بررسی آنها خواهیم پرداخت.

## معیار های مقایسه الگوریتم‌ها

همان‌طور که از نمودار شکل ۴-۱ مشخص است، برای مقایسه الگوریتم ها در این پروژه از ۵ معیار مختلف استفاده شده است که در ادامه به شرح آنها می‌پردازیم.

[1] PSO
[2] ABC
[3] FA
[4] iteration




شکل ۴-۱: معیار های مقایسه الگوریتم ها

طول کوتاه‌ترین مسیر: کوتاه‌ترین مسیر تعیین شده توسط هر الگوریتم در آزمون های انجام شده، نشان دهنده کیفیت آن در همگرایی به پاسخ بهینه سراسری می‌باشد که مورد ارزیابی و مقایسه قرار خواهد گرفت.

میانگین طول همه مسیرهای موفق: میانگین طول مسیرهای موفق طراحی شده توسط هر الگوریتم، نشان دهنده کیفیت بهینه سازی الگوریتم در دستیابی به پاسخ بهینه می‌باشد. مسیر موفق، مسیری است که هیچ برخوردی با موانع موجود در محیط نداشته باشد.

میانگین طول مسیر به‌دست آمده در زمان ثابت: میانگین طول مسیر به‌دست آمده در زمان ثابت نشان دهنده این است که کدام الگوریتم میتواند در محدودیت بیشتر، عملکرد بهتری داشته باشد. برای این کار، در یک زمان ثابت مثلا یک دقیقه الگوریتم‌ها با یکدگیر مقایسه می‌شوند تا مشخص شود که کدام الگوریتم در زمان کمتر، پاسخ بهتری ارائه می‌دهد.

میانگین زمان محاسبات: زمان محاسبات برای اجرای n تکرار هر الگوریتم از پارامترهای مورد مقایسه می‌باشد که میانگین این زمان از نتایج اجرای الگوریتم ها برای هر محیط، محاسبه و ارزیابی خواهد شد.

نرخ موفقیت: درصد موفقیت هر الگوریتم در دستیابی به مسیری که هیچ برخوردی با موانع محیط در n بار اجرای الگوریتم نداشته باشد، به عنوان نرخ موفقیت آن الگوریتم تعریف می‌شود و این پارامتر نیز، به عنوان معیار مقایسه سه الگوریتم در نظر گرفته شده‌است. به عبارت دیگر، این معیار، درصد عدم برخورد مسیر طراحی شده به موانع را نمایش می‌دهد.

## تعریف مسئله طراحی مسیر

پیش از پیاده سازی هر الگوریتم، ابتدا باید مسئله ای که قرار است بهینه سازی شود، برای آن الگوریتم تعریف گردد. در مسئله طراحی مسیر، اطلاعات مورد نیاز الگوریتم، شامل مشخصات محیط، تابع هدف یا تابع هزینه و قید مسئله باید تعیین شوند. در ادامه، به تعریف هر یک پرداخته خواهد شد.

## محیط حرکت ربات

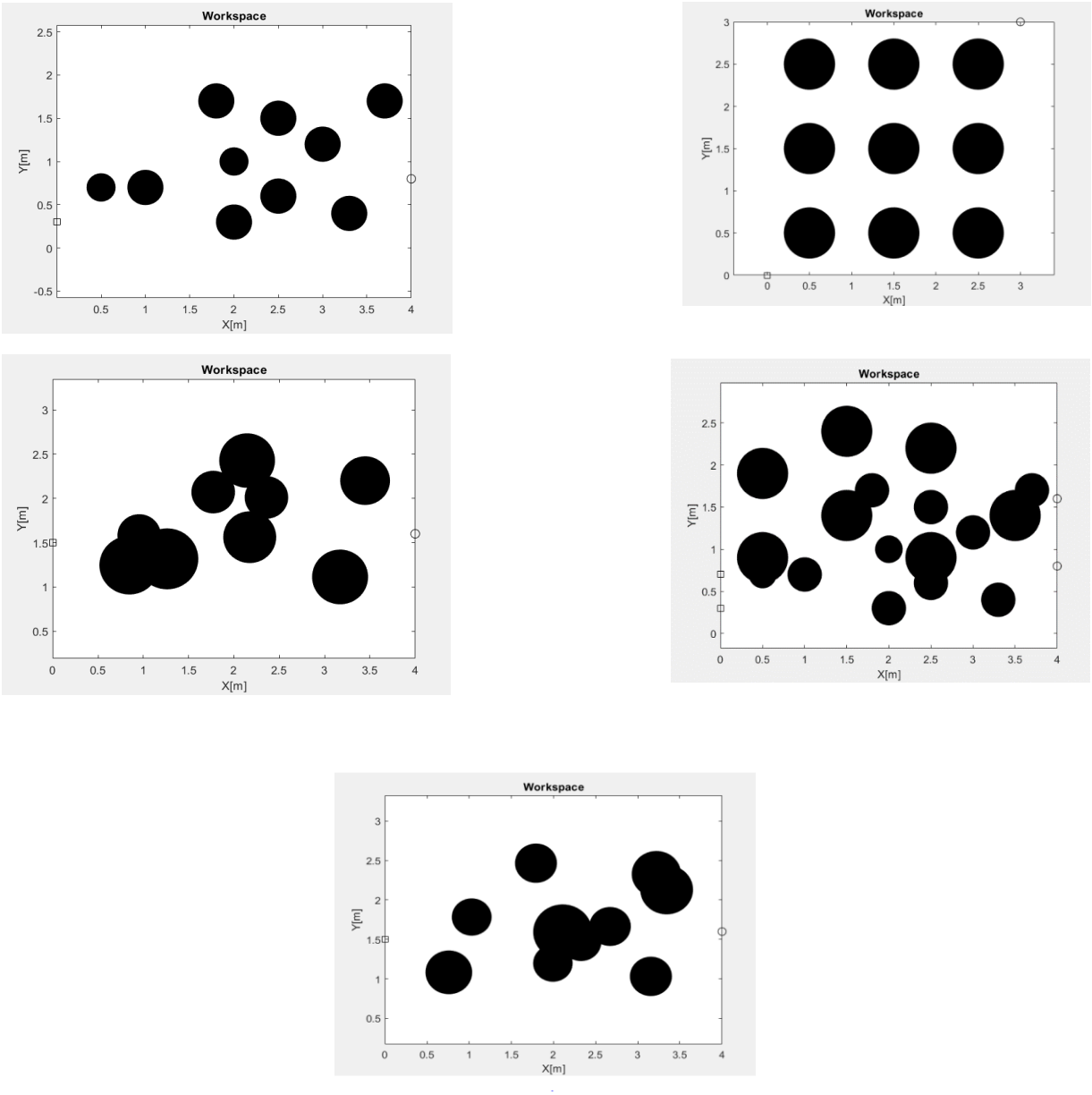
ویژگی‌های محیط حرکت ربات، از پایه ای ترین اطلاعاتی است که باید در اختیار الگوریتم قرار گیرد. زیرا اساس کارکرد الگوریتم‌های هوشمند، برمبنای جستجوی تصادفی در یک فضای جستجو با هدف یافتن پاسخ بهینه می‌باشد. بنابراین بازه تغییرات متغیرهای مسئله باید محدود و مشخص باشد. در مسئله طراحی مسیر، متغیرها مؤلفه های x و y موقعیت ربات هستند که باید بازه تغییرات آنها که همان ابعاد محیط آزمایش است، برای الگوریتم تعریف شود. علاوه بر آن، اطلاعاتی همچون نقطه آغاز و نقطه پایان مسیر در محیط کار نیز باید برای الگوریتم تعیین گردد. در جدول ۴-۱ مشخصات محیط کار ربات مشخص شده است.

جدول ۴-۱ مشخصات محیط کار ربات

| ردیف | مشخصه محیط | تعریف |
|---|---|---|
| ۱ | مولفه x مسیر | $x=\{x_0,x_1,x_2,\dots\}$ |
| ۲ | مولفه y مسیر | $y=\{y_0,y_1,y_2,\dots\}$ |
| ۳ | محدوده مولفه x | $[x_{min},x_{max}]$ |
| ۴ | محدوده مولفه y | $[y_{min},y_{max}]$ |
| ۵ | شعاع موانع | $r_{obs}=\{r_{obs0},r_{obs1},r_{obs2},\dots\}$ |
| ۶ | مولفه x مرکز موانع | $x_{obs}=\{x_{obs0},x_{obs1},x_{obs2},\dots\}$ |
| ۷ | مولفه y مرکز موانع | $y_{obs}=\{y_{obs0},y_{obs1},y_{obs2},\dots\}$ |
| ۸ | نقطه آغاز | $[x_s,y_s]$ |
| ۹ | نقطه پایان | $[x_f y_f]$ |

همان طور که از جدول نیز مشخص می‌باشد، مشخصات محیط کار ربات شامل: مؤلفه های مختصات مسیر و محدوده تغییر آنها، شعاع موانع و مختصات نقاط مرکز آنها و همچنین نقاط آغاز و پایان مسیر می‌باشد.

برای تولید محیطی که قرار است ربات در آن مسیریابی کند، کدی در محیط متلب نوشته‌ایم. همان طور که از شکل ۴-۲ مشخص است، در اینجا ۵ محیط مختلف ساخته‌ایم که در برخی از آنها، موانع منظم و با فاصله یکسان از یکدیگر چیده شده‌اند. در برخی موانع به صورت نامنظم در کنار یک دیگر قرار گرفته اند. نقطه شروع را یک مربع در نظر گرفته ایم و نقطه پایان را نیز به شکل یک دایره نشان داده ایم.




شکل ۴-۲ پنج محیط شبیه سازی شده برای آزمایش سه الگوریتم

## تابع هزینه

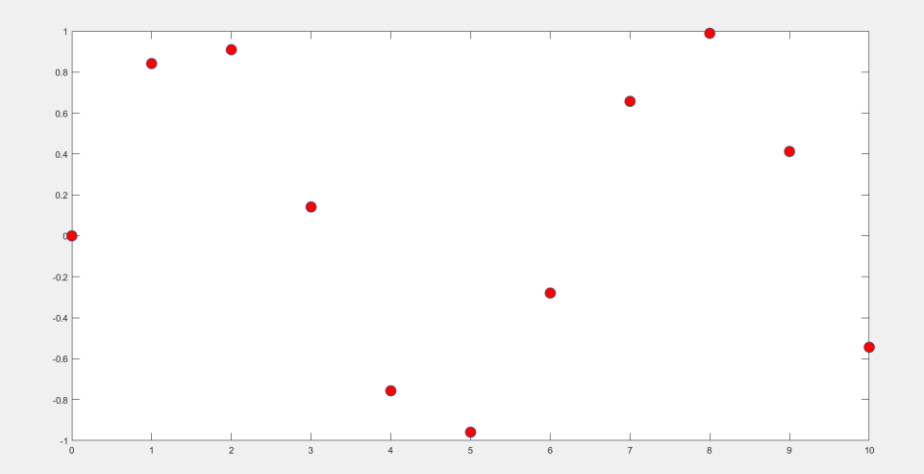
از تابع هزینه به منظور ارزیابی پاسخ به دست آمده توسط الگوریتم، استفاده می‌شود. در اینجا هزینه، همان طول مسیر است. اگر هیچ مانعی در محیط نباشد، کم هزینه‌ترین مسیر، همان مسیر مستقیم از نقطه آغاز تا پایان است. اما در صورت وجود مانع در مسیر حرکت ربات، باید عدم برخورد با موانع نیز، به عنوان قید به تابع هزینه اعمال گردد که برای اعمال این قید، میتوان فاصله همه نقاط مسیر را از موانع به کمک مختصات مرکز مانع و مختصات نقطه موجود در مسیر به‌دست آورد و سپس آن را به عددی در بازه ۰ تا ۱ تبدیل کرد به نحوی که ۱ به معنی قرار گرفتن آن نقطه از مسیر روی مرکز مانع و ۰ به معنی عدم برخورد آن نقطه از مسیر با مانع است. سپس برای هر مسیر به‌دست آمده میانگینی از این اعداد که مربوط به نقاط تشکیل دهنده مسیر هستند، به دست آورد و به نحوی آن را در تابع هزینه اعمال کرد. جزییات بیشتر را در مورد نحوه به‌دست آوردن تابع هزینه فصل آینده بیان خواهد شد.

# فصل پنجم: نحوه محاسبه طول مسیر و تابع هزینه

در این فصل، در مورد ارتباط مسئله مسیریابی و الگوریتم‌های بهینه‌سازی و نحوه طراحی مسیر و به‌دست آوردن تابع هزینه توضیحاتی ارائه خواهد شد.

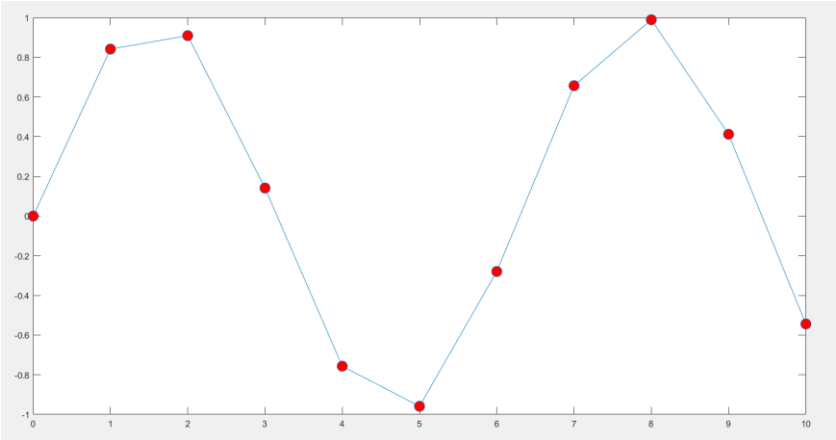
در طراحی یک مسیر در یک محیط دو بعدی، اولین مولفه‌های لازم، مختصات نقطه شروع و پایان است. برای طراحی مسیر، مسیر به صورت یک تابع در نظر گرفته می‌شود که تعدادی از نقاط این تابع موجود است و قرار است به کمک این نقاط، تابع اصلی را به دست آید. در این پروژه از روش درونیابی اسپلاین استفاده می‌کنیم. برای درک بهتر این مسئله، به مثال زیر توجه کنید.

فرض کنید که تعدادی از نقاط تابع $y = \sin(x)$ را داریم. این نقاط که در این مثال $x$ ها هستند شامل مقادیر $0,1,2,\ldots,10$ است و $y$ نیز برابر مقدار sin در نقاط ذکر شده می باشد. تصویر شکل ۵-۱ نشان دهنده این نقاط روی محور مختصات می‌باشد.



شکل ۵-۱ مقدار تابع سینوس در نقاط ۱ تا ۱۰

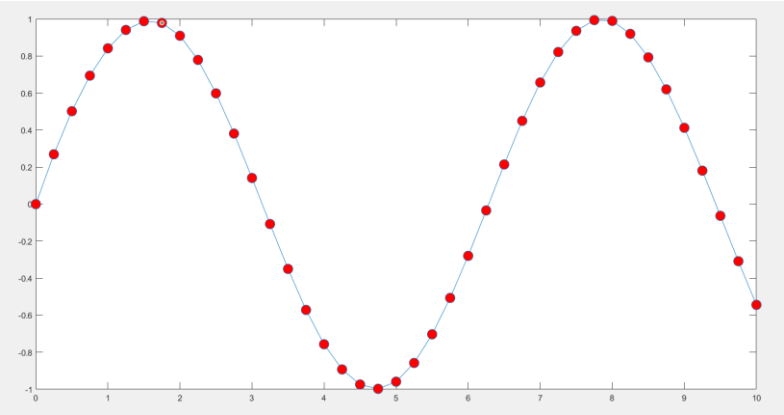
حال اگر بخواهیم به کمک نقاطی که داریم، به نزدیک ترین نمودار به $y = \sin(x)$ برسیم، یک روش ساده آن است که این نقاط را به صورت خطی به هم وصل کنیم که نتیجه این کار نموداری است که در تصویر شکل ۵-۲ مشاهده می کنیم.



شکل ۵-۲ اتصال خطی ده نقطه از تابع سینویس

همان طور که قابل مشاهده است، نتیجه مدنظر به اندازه کافی مطلوب نیست و منحنی مناسبی به‌دست نیامده‌است. روش دیگر که روش مدنظر ما در این پروژه می‌باشد و در مسیریابی خود از آن استفاده کرده‌ایم ، استفاده از درون‌یابی اسپلاین[1]است. این روش درون‌یابی، دارای روابط ریاضی پیچیده‌ای است که قصد ما بیان و شرح دقیق آنها و درون یابی اسپلاین نیست. فقط در همین حد بیان می کنیم که در روش مدنظرما، ابتدا همین نقاط قرمز رنگ بالا را مشخص می کنیم. سپس به کمک تابع اسپلاین در متلب، تعدادی نقطه بین هر دو نقطه قرمز به‌دست می‌آوریم و در نهایت این نقاط را بهم وصل می کنیم. برای مثال اگر برای مورد بالا، مشخص کنیم که بین هر دو نقطه قرمز، ۴ نقطه به کمک تابع اسپلاین به‌دست بیاید و نقاط بهم وصل گردند، نتیجه‌ای مانند تصویرشکل ۵-۳ به‌دست خواهد آمد که همان طور که مشاهده می کنیم، نتیجه بسیار مطلوبی را به‌دست آمده است.

[1] spline



شکل ۵-۳ به دست آوردن نمودار سینوس به کمک ۱۰ نقطه اصلی و تابع اسپلاین

در این پروژه، مسیرهایی که قرار است به‌دست بیاوریم، چیزی شبیه به مثال قبل می‌باشد. در واقع ما تعدادی نقطه در مسیر تعیین می‌کنیم و سپس به کمک نقاط و درون یابی اسپلاین ، مسیر را طراحی می‌کنیم.

برای این که بتوانیم از الگوریتم‌های بهینه سازی در جهت یافتن بهترین مسیر استفاده کنیم، نیاز است که یک تابع هدف یا هزینه تعریف کنیم که به هر مسیر تولید شده یک عدد نسبت بدهد و ما تلاش کنیم که مسیری را بیابیم که عدد مربوط به آن ماکزیمم و یا مینیمم باشد. در مسئله طراحی مسیر می‌توانیم تابع هدف را برابر طول مسیر در نظر بگیریم و به کمک الگوریتم‌های بهنیه سازی سعی کنیم که مسیری با کمترین طول ممکن را پیدا کنیم. همان طور که بیان کردیم ، مسیر ما از تعدادی نقطه با مولفه x و y تشکیل شده است. به کمک رابطه(۴-۱) میتوانیم فاصله هر دو نقطه را به‌دست بیاوریم.

$$\sqrt{(x_i - x_j)^2 - (y_i - y_j)^2} \qquad (۴-۱)$$

در نهایت با جمع این فواصل ، طول مسیر را به‌دست می آوریم. در واقع در نهایت تابع هدف ما به صورت رابطه (۴-۲) خواهد بود.

$$L(x.y) = \sum_{i=1}^{f} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \qquad (۴-۲)$$

درنتیجه الگوریتم‌های بهینه سازی، بر روی مینیمم کردن مقدار L کار خواهند کرد.

اما مسئله به همین سادگی نیست. ما در طول مسیر خود، موانعی نیز داریم که مسیر ما نباید با این موانع برخوردی داشته‌باشد. در نتیجه باید عدم برخورد به مانع را به صورت قیدی در مسئله و در تابع هزینه خود قرار دهیم. برای این که بتوانیم مسئله را، همچنان در قالب یک مسئله بهینه سازی نگه داریم، به صورت کلی می‌توانیم با هر برخورد مسیر با موانع ، مقدار تابع هزینه را افزایش دهیم. به نحوی که اگر مسیر دارای برخوردی بود، مقدار تابع هزینه طوری زیاد شود که دیگر به عنوان بهترین جواب انتخاب نشود. در ادامه به نحوه‌ی اعمال این قید به تابع هزینه می‌پردازیم.

همان طور که قبلا بیان شد، هر مانع در مسیر دارای دو مشخصه هست. یکی مختصات مرکز مانع و دیگری شعاع مانع. اگر مسیری با یک مانع برخورد داشت، به این معنی است که یک یا چند نقطه از نقاط تشکیل دهنده مسیر، فاصله‌شان با مرکز مانع از شعاع مانع کمتر است. بنابراین ابتدا به کمک رابطه (۴-۳)، فاصله تک تک نقاط مسیر را با مرکز موانع به‌دست می آوریم.

$$d_{ij} = \sqrt{\left(x_i - x_{obs\ j}\right)^2 + \left(y_i - y_{obs\ j}\right)^2} \qquad (۳-۰)$$

در رابطه (۴-۳)، $x_i$ و $y_i$ مختصات نقطه i ام مسیر و $x_{obs}$ و $y_{obs}$ مختصات مانع j ام و $d_{ij}$ فاصله نقطه i ام از مسیر با مانع j است.

حال باید میزان برخورد هر نقطه با هر مانع را به‌دست بیاوریم. برای این کار از رابطه (۴-۴)،استفاده می‌کنیم.

$$C_{obs\ j} = \max(1 - \frac{d_{ij}}{r_{obs\ j}}. 0) \qquad (۴-۰)$$

در رابطه (۴-۴)، $r_{obs\ j}$ شعاع مانع j ام است. طبق رابطه بالا اگر نقطه i ام مسیر، فاصله اش با مرکز مانع j ام کمتر از شعاع باشد، یعنی برخورد صورت گرفته و مقدار C بیشتر از ۰ خواهد شد و اگر فاصله اش با مرکز مانع بیشتر از شعاع باشد ، C برابر صفر خواهد بود. اگر برای همه نقاط مسیر رابطه بالا را استفاده کنیم در نهایت $C_{obs\ j}$، یک بردار از مقادیر برخورد نقاط تشکیل دهنده مسیر با مانع j ام به‌دست می‌آید. ما از میانگین این مقادیر به عنوان نماینده برخورد مسیر طراحی شده با مانع j استفاده می‌کنیم و به رابطه (۴-۵) می‌رسیم.

$$V_j = mean([C_{obs}]) \qquad (۵-۰)$$

و در آخر، به ازای وجود n مانع در مسیر، از $V_c$ که c دارای مقادیر ۱و۲و...n است ، به عنوان بردار نماینده برخورد مسیر با همه موانع استفاده می‌کنیم.

برای اعمال این مقادیر به عنوان جریمه به تابع هدف که طول مسیر است، از بین روش های اعمال جریمه به تابع هدف ، از روش جریمه ضرب شونده که به فرم رابطه(۴-۷) است ، استفاده می‌کنیم.

$$z = L(x.y) \quad (۶-۰)$$

$$\hat{z} = z(1 + \alpha V_c) \quad (۷-۰)$$

در روابط (۴-۶) و (۴-۷)، $z$ تابع هدف بدون قید وجود موانع است و $\hat{z}$ ، تابع هدف با اعمال قید وجود موانع است. $\alpha$ نیز ضریب اعمال جریمه است که خودمان به صورت دستی تنظیم می‌کنیم. در نهایت ما از $\hat{z}$ به عنوان تابع هزینه در الگوریتم‌های خود استفاده می‌کنیم.

اگر بخواهیم یک جمع بندی کلی روی مطالبی که بیان شد، داشته باشیم، می‌توانیم بگوییم که ما برای طول هر مسیر ، یک تعداد ثابت نقطه به عنوان نقاط اصلی در محیط در نظر می‌گیریم. سپس به کمک روش اسپلاین، مسیر کلی و نقاط تشکیل دهنده آن را به‌دست می آوریم. سپس برای مسیر تولید شده $\hat{z}$ را به‌دست می آوریم و سعی می‌کنیم به کمک الگوریتم های بهینه سازی مثل الگوریم FA ، مسیری با کمترین $\hat{z}$ را به‌دست آوریم. الگوریتم های بهینه سازی در واقع ، نقاطی را به عنوان نقاط اصلی تولید می‌کنند و با توجه به روندشان ، در هر تکرار سعی می‌کنند که نقاطی را تولید کنند که $\hat{z}$ به‌دست آمده توسط مسیر تولید شده از آن نقاط، کمترین مقدار را داشته باشد.

در فصل آینده به شرح دقیق الگوریتم‌های برگرفته شده از طبیعت که از آنها در روند پروژه خود استفاده کرده‌ایم، خواهیم پرداخت.

# فصل ششم: معرفی الگوریتم های برگرفته شده از طبیعت

در سالیان گذشته، توجه به الگوریتم های برگرفته‌شده از طبیعت بسیار زیاد شده‌است و در واقع به یکی از مورد توجه ترین الگوریتم‌ها و تکنیک ها برای مسائل بهینه سازی تبدیل شده‌اند. به کمک این الگوریتم ها توانسته‌اند، در مسائل NP-Hard مثل مسئله فروشنده دوره گرد[1] به شکل بهتر و مدت زمان معقول‌تری به جواب بهینه برسند.

الگوریتم هایی مثل الگوریتم حشرات شب‌تاب[2] یا الگوریتم کلونی زنبورها[3] یا الگوریتم ازدحام ذرات[4] مبتنی بر هوش ازدحامی هستند. در واقع در این الگوریتم‌ها ، یک سری عامل وجود دارد که توانایی بالایی ندارند و کار ساده‌ای انجام می‌دهند ولی این عوامل هنگامی که با یک‌دیگر و درکنار هم کار می‌کنند نتایج بسیار خوبی را به‌دست می آورند.

الگوریتم ژنتیک، از الگوریتم‌های بسیاری معروفی‌است که با آن آشنا هستیم و البته در فصل آینده به صورت مختصر به آن خواهیم‌پرداخت. الگوریتم ژنتیک به کمک مکانیزم های تکاملی نظیر جهش[5]، ترکیب[6] و تولید مثل[7] انتخاب طبیعی[8] باقی ماندن بهترین‌ها[9]، جواب‌هایی را به عنوان جواب کاندید برای حل مسئله بهینه سازی ارائه می‌دهند. الگوریتم های مبتنی بر هوش ازدحامی شباهت زیادی به الگوریتم ژنتیک دارند و از لحاظ ساختاری ساده‌تر از الگوریتم ژنتیک هستند زیرا در آنها مکانیزم‌های تکاملی مثل جهش و ترکیب وجود ندارد. مکانیزم این الگوریتم ها مبتنی بر تصادفی بودن[10] ارتباط سراسری[11] میان عامل ها برای رسیدن به تکامل و جواب بهینه سراسری می باشد. از طرفی چون که عمدتا روی اعداد حقیقی کار می‌کنند، پیاده سازی الگوریتم آنها ساده‌تر می باشد.

در ادامه این فصل به شرح سه الگوریتم حشرات شب‌تاب، کلونی زنبورها و ازدحام ذرات خواهیم‌پرداخت و به طور مفصل طرز کار این الگوریتم‌ها را شرح خواهیم داد.

---

[1] TSP
[2] Firefly algorithm
[3] artificial bee colony
[4] particle swarm
[5] mutation
[6] recombination
[7] reproduction
[8] natural selection
[9] survival of fittest
[10] randomness
[11] global communication

## الگوریتم حشرات شب تاب

الگوریتم حشرات شب‌تاب یا به اختصار FA در سال ۲۰۰۷ توسط جناب شی یانگ[1] معرفی شد. به طور خلاصه نحوه عملکرد الگوریتم FA بدین صورت است که ابتدا تعدادی حشره شب‌تاب مصنوعی به طور تصادفی در دامنه مساله توزیع می‌شوند و سپس هر حشره شب‌تاب از خود نوری ساطع می‌کند که شدت آن متناسب با میزان بهینگی نقطه‌ای است که آن حشره در آن واقع شده‌است. سپس شدت نور هر حشره مرتبا با شدت نور سایر حشرات شب‌تاب مقایسه شده و حشره کم نورتر به سوی حشره پرنورتر جذب می‌شود. در عین حال پر نور ترین حشره نیز با هدف افزایش شانس یافتن جواب بهینه سراسری به طور تصادفی در دامنه مساله حرکت می‌کند. همانطور که می‌بینیم، در الگوریتم FA حشرات شب‌تاب از طریق انتشار نور به تبادل اطلاعات با یکدیگر می‌پردازند. ترکیب این عملیات دسته جمعی باعث می‌شود که گرایش کلی حشرات شب‌تاب به سوی نقاط بهینه‌تر باشد.

الگوریتم حشرات شب‌تاب بر مبنای رفتار چشمک زدن حشرات شب‌تاب در طبیعت، پایه‌ریزی شده است. چشمک زدن این حشرات بر اساس یک فرآیند نورافشانی زیستی بوده و تعیین تابع صحیح سیستم تولید سیگنال آن هنوز مورد مناقشه‌است. کارکرد و مبنای این چشمک زدن حشرات را میتوان به دو مورد زیر خلاصه کرد.

۱-جذب جنس مخالف برای جفت گیری

۲- جذب طعمه و شکار آن (به این صورت که نورهایی برای جذب جنس مخالف گونه‌های دیگر از حشرات شب‌تاب از خود ساطع می‌کنند و پس از نزدیک شدن طعمه ، آن را شکار می‌کنند.)

علاوه بر موارد بالا، محققان در تحقیقات خود دریافته‌اند که این حشرات به عنوان یک مکانیزم محافظتی برای ارسال هشدار، از خود نور ساطع می‌کنند.

یکی از نکات مهمی که در مورد چشمک زدن حشرات شب تاب باید در نظر گرفت مسئله میزان شدت نور دریافتی از یک منبع نور در یک فاصله مشخص است. رابطه بین این میزان شدت نور و فاصله، از قانون مربع معکوس پیروی می‌کند. در واقع اگر $I$ میزان شدت نور دریافتی و $r$ میزان فاصله از منبع نور باشد، نسبت $I \propto \frac{1}{r^2}$ برقرار است. علاوه بر آن محیطی که حشرات در آن قرار دارند، بر میزان شدت نور دریافتی تاثیر می‌گذارد. در واقع محیط تا حدی نور را به خود جذب می‌کند که این خود باعث می‌شود که رابطه بین میزان شدت نور دریافتی و میزان فاصله کاملا متناسب با قانون مربع معکوس نباشد. این نکته باعث می‌شودکه حشرات شب‌تاب تنها در محدوده‌ای معین قابل تشخیص

[1] Xin-she Yang

باشند و اگر فاصله‌ها تغییر کند، میزان نوری که از سایر حشرات دریافت می‌کنند، تغییر کند و یا برخی حشرات دیگر دیده نشوند و یا برخی به تازگی قابل مشاهده باشند.

در الگوریتم حشرات شب‌تاب، ابتدا تعدادی حشره شب‌تاب مصنوعی به طور تصادفی در دامنه مساله پخش می‌شود. سپس به هر یک از این حشرات شب تاب مصنوعی با استفاده از مقدار به دست آمده برای تابع هدف در آن نقطه یک شدت نور مناسب نسبت داده می‌شود. نحوه انتساب «شدت نور» به هر یک از حشرات به گونه‌ای است که با افزایش میزان بهینگی نقطه‌ای که هر حشره در آن واقع شده، شدت نور ساطع شده از آن افزایش می‌یابد. بدیهی‌است که بسته به میزان فاصله، این حشرات شب‌تاب یکدیگر را با نورهایی با شدت های مختلف مشاهده خواهندکرد. در مرحله بعد حشرات شب‌تاب کم نور تر مرتبا به سوی حشرات پرنورتر جذب می‌شوند و این فرایند تکراری تا آنجا ادامه می‌یابد که همه حشرات شب‌تاب در یک نقطه، که احتمالا نقطه بهینه سراسری است، تجمع یابند و یا اینکه تعداد از پیش تعیین‌شده‌ای از تکرارها انجام شده باشد.نور تولید شده توسط حشرات شب تاب را میتوان به نحوی فرمول بندی و پیاده کرد که متناظر با یک تابع هدف باشد. تابع هدفی که می‌خواهیم توسط الگوریتم بهینه سازی شود. در ادامه یک سری مفاهیم مقدماتی و جزییات الگوریتم حشرات شب‌تاب و نحوه فرمول بندی آن را بیان می‌کنیم.

## مفاهیم مرتبط

برای این که بتوان الگوریتم حشرات شب‌تاب را ساده‌تر فرموله کرد، سه فرض زیر در نظر گرفته می‌شود:

۱- تمام حشرات شب‌تاب تک جنسی هستند. در واقع با این فرض، این حالت را بیان می‌کنیم که هر حشره‌ای مستقل از نوع جنسیت خود، میتواند جذب حشره‌ی دیگر شده و در واقع به سمت آن حرکت کند.

۲- میزان جذابیت هر حشره شب‌تاب متناسب با شدت نوری است که سایر حشرات شب‌تاب از اون دریافت می‌کنند. در واقع یعنی با افزایش شدت نور هر حشره شب‌تاب ( یا همان میزان بهینگی نقطه ای که در آن قرار داریم)، حشرات شب‌تاب بیشتری به سمت آن جذب می‌شود. همچنین بدیهی‌است که با کاهش فاصله بین دو حشره، آن دو یکدیگر را با شدت نور بیشتری مشاهده کرده و جذابیت‌شان برای هم افزایش می‌یابد. به عنوان یک قاعده در الگوریتم FA، در مورد هر دو حشره شب‌تاب، حشره شب تاب کم نورتر به سوی حشره شب‌تاب پرنورتر حرکت می‌کند. همچنین، حشره‌ای که از همه پر نورتر باشد به سوی هیچ یک از دیگر حشرات شب‌تاب جذب نمی شود و به طور تصادفی در دامنه مساله به حرکت در می‌آید.

۳- شدت نور هر یک از حشرات شب‌تاب توسط مقدار به دست آمده برای تابع هدف در آن نقطه تعیین می‌شود. برای این منظور در مسائل بیشینه‌سازی می توان شدت نور را متناسب با مقدار تابع هدف در آن نقطه در نظر گرفت و در مسائل کمینه‌سازی از تناسب و معکوس کردن استفاده کرد. در هر صورت نحوه تعریف تابع شدت نور باید به گونه‌ای باشد که میزان درخشندگی هر حشره شب‌تاب به طور یکنواخت با افزایش میزان بهینگی آن افزایش یابد.

بر اساس این سه فرض مسئله بهینه سازی به کمک الگوریتم حشرات شب تاب به صورت زیر تعریف میگردد.

**۱-** مقدار دهی اولیه به پارامتر ها: مقادیری مناسب برای پارامترهای مسئله که در زیر بیان می کنیم، نسبت می‌دهیم.

$\gamma$ : ضریب جذب نور توسط محیط

MaxIt: حداکثر تعداد تکرار الگوریتم (شرط خاتمه الگوریتم)

$\beta_0$ : حداکثر ضریب جذابیت بین دو حشره شب تاب

$\alpha$: ضریب بردار جابجایی تصادفی

Npop: تعداد حشرات شب تاب

**۲-** مقدار دهی اولیه به حشرات شب‌تاب: یک جمعیت اولیه از حشرات شبتاب به صورت تصادفی در دامنه مسئله تولید می‌شود.

۳- تابع هدف $f$ را مشخص می کنیم.

۴- شدت نور $i$ امین حشره شب‌تاب را به کمک تابع هدف $f$ مشخص می کنیم.

۵- در یک حلقه بررسی می کنیم که مرتبه تکرار الگوریتم از ماکزیمم مقداری که تعیین کردیم بیشتر نباشد. اگر بیشتر بود از حلقه خارج می‌شویم و به مرحله ۱۰ می‌رویم. در غیر این صورت وارد حلقه می‌شویم.

۶- یک حلقه ***for*** که به کمک شمارنده $i$ به تعداد حشرات شب تاب جلو میرود را ایجاد می کنیم.

۷- درون حلقه قبلی، یک حلقه ***for*** که به کمک شمارنده $j$ به تعداد حشرات شب تاب جلو میرود را ایجاد می کنیم.

۸- اگر میزان شدت نور حشره $i$ از حشره $j$ بیشتر بود ، حشره $j$ به کمک رابطه (۶-۱) به سمت حشره $i$ حرکت می کند.

$$x_{i\,new} = x_i + \beta_0 e^{-\gamma r^2 ij}\left(x_{ik} - x_{jk}\right) + \alpha\epsilon_i \qquad (۱-۶)$$

۹- حشرات شب‌تاب بر اساس تابع هزینه به روز رسانی شده و مرتب می‌گردند و بهترین جواب تعیین می‌گردد. سپس به مرحله ۵ برمی‌گردد.

۱۰- بهترین جواب به‌دست آمده در طول تمام تکرار ها به عنوان جواب بهینه معرفی می‌گردد.

حال نحوه رسیدن به رابطه (۶-۱) را بیان می‌کنیم. همان طور قبلا بیان کردیم ، در ساده‌ترین حالت ممکن رابطه بین میزان شدت نور و فاصله، مشابه رابطه (۶-۲) می‌باشد.

$$I(r) = \frac{I_s}{r^2} \qquad (۲-۰)$$

از طرفی اگر بخواهیم تاثیر محیط و جذب نور توسط محیط را بر روی این رابطه اثر دهیم و علاوه بر آن مانع از بینهایت شدن میزان شدت نور در فاصله ۰ شویم ، به رابطه (۶-۳) می‌رسیم که در آن $\gamma$ ضریب جذب نور توسط محیط است.

$$I(r) = I_O e^{-\gamma r2} \qquad (۳-۰)$$

از آن جا که میزان جذابیت حشرات شب‌تاب متناسب با شدت نوری است که توسط حشرات دیگر مشاهده می‌شود، پارامتر جذابیت برای یک حشره شب‌تاب را میتوان از طریق رابطه (۶-۴) بیان کرد.

$$\beta(r) = \beta_0 e^{-\gamma r^2} \qquad (۴-۰)$$

در رابطه (۶-۴)، $\beta_0$ معادل میزان جذابیت هر حشره شب‌تاب در $r = ۰$ می‌باشد.در واقع همان حداکثر ضریب جذابیت بین دو حشره شب‌تاب می باشد. البته میتوان برای آن که با سرعت کمتری همگرایی صورت بگیرد از رابطه(۶-۵) به جای رابطه(۶-۴) استفاده نمود.

$$\beta(r) = \frac{\beta_0}{1 + \gamma r^2} \qquad (۵-۰)$$

از طرفی فاصله میان دو حشره شب‌تاب در یک فضای جست و جوی دوبعدی را میتوان به کمک رابطه (۶-۶) به‌دست آورد.

$$r_{ij} = ||x_i - x_j|| = \sqrt{(x_i - x_j)^2 - (y_i - y_j)^2} \qquad (۶-۰)$$

برای آن که در میبنیمم و ماکزیمم‌های محلی گیر نکنیم و احتمال یافتن جواب‌های بهتر را نیز بالا ببریم، یک حرکت تصادفی به حرکت حشره $i$ به سمت $j$ اضافه می‌کنیم که از یک پارامتر تصادفی کننده به نام $\alpha$ و یک بردار جابه جایی $\epsilon_i$ که از یک توزیعِ یکنواخت در بازه‌ای مشخص می‌باشد تشکیل می‌شود. در نهایت به فرمول (۶-۷) میرسیم[۹].

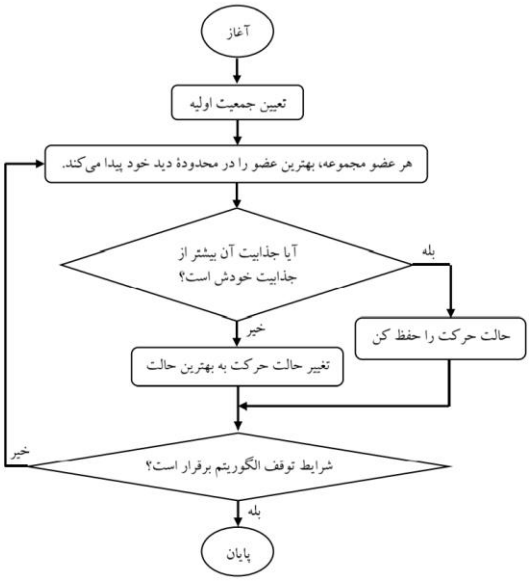
$$x_{i\,new} = x_i + \beta_0 e^{-\gamma r^2 ij}\left(x_{ik} - x_{jk}\right) + \alpha\epsilon_i$$

در نهایت اگر بخواهیم پارامتر های اختصاصی این الگوریتم را دوباره بیان کنیم، به جدول شکل ۶-۱ میرسیم.

جدول ۶-۱ پارامتر های الگوریتم حشرات شب‌تاب

| | |
|---|---|
| حداکثر تعداد تکرار | maxIt |
| تعداد جمعیت حشرات | npop |
| ضریب جذب نور | $\gamma$ |
| میزان جذابیت هر حشره شب تاب در $r$ =۰ | $\beta_0$ |
| یک پارامتر تصادفی کننده | $\alpha$ |
| محدوده تغییرات | $\delta_{ik}$ |

فلوچارت کلی الگوریتم حشرات شب‌تاب را نیز میتوان در شکل ۶-۱ مشاهده کرد.




شکل ۶-۱ فلوچارت الگوریتم حشرات شب‌تاب

## الگوریتم تراکم ذرات

بهینه سازی تراکم ذرات یکی از الگوریتم های بهینه سازی برگرفته شده از طبیعت است. این الگوریتم نخستین بار توسط راسل ابرهات[1] و جیمز کندی[2] با الهام گرفتن از حرکت جمعی پرندگان و ماهی‌ها در جست‌وجوی و یافتن غذا معرفی شد. بر اساس این الگو، اگر گروهی از پرندگان در حال چرخیدن به دور ناحیه ای که بوی غذا را استشمام کرده‌اند باشند، پرنده ای که به منبع غذا نزدیک‌تر است بلندتر جیغ می‌کشد و دیگر پرندگان در جهت حرکت آن پرنده خواهند چرخید. اگر پرنده‌ای به منبع غذا نزدیک‌تر شد جیغ بلندتری می‌کشد و دیگر پرندگان به سمت آن پرنده حرکت می‌کنند. اگر هر حلقه از پرندگان به هدف نزدیک تر شود، پرنده اول جیغ بلندتری می‌کشد و حلقه های دیگر حول آن حلقه خواهند چرخید و این روند تا جایی که یکی از پرندگان به منبع غذا برسد، ادامه خواهد یافت. این الگوریتم در آغاز برای شبیه سازی پرواز دسته جمعی پرندگان مورد استفاده قرار می‌گرفت ولی پس از ساده سازی و ایجاد تغییرات کوچکی در آن مشاهده شد که این الگوریتم در واقع یک نوع عمل بهینه سازی را انجام می‌دهد و به همین دلیل می‌تواند برای حل سایر مسائل بهینه سازی نیز مورد استفاده قرار گیرد. در این الگوریتم برای حل یک مسئله بهینه سازی جمعیتی از جواب‌های کاندید با استفاده از یک فرمول ساده به طور تصادفی در دامنه مسئله به حرکت در می‌آیند و آن را با هدف یافتن جواب بهینه سراسری مورد جست و جو قرار میدهند. در الگوریتم تراکم ذرات، فضای مسئله با جمعیت اولیه ای از پاسخ ها پوشش داده می‌شود که این جمعیت در تکرار روند الگوریتم ، به سوی پاسخ بهینه هدایت می‌گردد. قاعده کار

[1] Russell C. Eberhart
[2] James Kennedy

به این شکل است که هر عضو[1] از جمعیت نماینده یک پاسخ است و با استفاده از حافظه قبلی خودش و اطلاعات جمعی دیگر اعضای جمعیت، خود را به روزرسانی می‌نماید. مزیت ویژه این الگوریتم این است که پیاده سازی آن آسان و بسیار مؤثر است و اگر پاسخی وجود داشته باشد، الگوریتم آن را پیدا خواهد کرد.

مهم ترین نقاط قوت الگوریتم تراکم ذرات عبارتند از:

- عدم حساسیت به مقیاس متغییر های مسئله
- کم بودن تعداد پارامتر ها
- پیاده سازی ساده
- قابلیت پیاده سازی به صورت موازی برای انجام پردازش های هزمان

مهم ترین نقطه ضعف این الگوریتم (نسخه استاندارد)، جست‌وجوی محلی ضعیف آن است. در واقع در این الگوریتم معمولا ذرات به سرعت به سمت پاسخ بهینه سراسری کشیده شده و همگرا می‌شوند ولی یافتن موقعیت دقیق نقطه بهینه سراسری که مربوط به جست‌وجوی محلی است، گاهی ممکن است زمان بر باشد. البته این مشکل در نسخه بهبود یافته این الگوریتم تاحد زیادی رفع گردیده است.

## قوانین حاکم بر تراکم ذرات در طبیعت

جانورانی مثل پرندگان و ماهی ها در حرکت جمعی خود از یک سری قوانین پیروی می‌کنند که پیروی از این قوانین باعث ایجاد نظم فوق العاده در حرکت این گونه جانوران شده است. این قوانین بسیار ساده هستند و جانوران با هوش کمی که دارند نیز از قوانین پیروی می‌کنند. این قوانین عبارتند از:

۱- سعی می‌کنند بیش از حد به سایر موجودات دیگر در حال حرکت نزدیک نشوند.

۲- حرکت خود را در جهت حرکت سایر جانوران به خصوص آنهایی که در نزدیکی خود قرار دارند، تنظیم می‌کنند.

۳- سعی می‌کنند متوسط فاصله خود را با سایر موجودات حفظ کنند تا فضای خالی در مجموعه ایجاد نشود.

[1] particle

در الگوریتم تراکم ذرات یا PSO نیز از ترکیب سه قانون فوق برای به حرکت در آوردن مجموعه‌ای از ذرات(جواب های کاندید) به طور موثر در دامنه مسئله استفاده می‌شود. در این الگوریتم، ذرات با پیروی از قوانین زیر در دامنه مسئله حرکت می کنند.

۱- هر ذره به محض پیدا کردن موقعیت هدف (پاسخ بهینه)آن را به نحو مناسبی به اطلاع سایرذرات گروه می‌رساند.

۲- سایر ذرات به طور غیر مستقیم و تا حدی تصادفی به سوی این هدف (یا همان نقطه بهینه تابع هزینه)کشیده می‌شوند. این حرکت جمعی به سوی نقطه بهینه به گونه‌ای است که اولا ذرات بیش از حد به یکدیگر نزدیک نمی‌شوند و ثانیا ارتباطشان را با یکدیگر حفظ می کنند.

۳- حرکت هر یک از ذرات متاثر از بهترین موقعیت یافته شده برای تابع هدف توسط خود آن ذره و نیز سایر ذرات گروه از آغاز اجرای الگوریتم تا تکرار فعلی است.

## روند کلی الگوریتم pso

با این فرض که D، بعد فضای جست‌وجو و nPop تعداد اعضای جمعیت است، موقعیت و سرعت، مشخصه های اصلی هر یک از این اعضا هستند که برای عضو i ام به صورت $X_i = \{x_{i_1}.x_{i_2}.x_{i_3}\cdots.x_{i_D}\}$ و $V_i = \{v_{i_1}.v_{i_2}.v_{i_3}\cdots.v_{i_D}\}$ تعریف می‌شوند. هر عضو از سویی حاوی اطلاعات بهترین موقعیت قبلی خودش می‌باشد که به صورت $P_i = \{p_{i_1}.p_{i_2}.p_{i_3}\cdots.p_{i_D}\}$ تعریف می‌شود. همچنین اطلاعات بهترین موقعیت کل جمعیت که به صورت $P_g = \{p_{g_1}.p_{g_2}.p_{g_3}\cdots.p_{g_D}\}$ است را نزد خود ذخیره دارد. سپس به کمک محدوده‌های $[x_{min}.x_{max}]$ و $[v_{min}.v_{max}]$ که محدوده‌ی موقعیت و سرعت است، انتخاب های رندمی صورت گرفته و جمعیت اولیه ای ساخته می‌شود. پس از تعیین جمعیت اولیه، کیفیت هر عضو از جمعیت، با استفاده از تابع هزینه ارزیابی می‌شود و بهترین عضو، تحت عنوان $p_{gbest}$ ذخیره می گردد. در ابتدا بهترین وضعیت برای هر عضو بینهایت (یک عدد بسیار بزرگ) در نظرگرفته می‌شود و سپس وضعیت عضو i ام در تکرار اول الگوریتم به عنوان بهترین وضعیت آن عضو تحت عنوان $p_{ibest}$ در حافظه آن ذخیره می‌شود. در هر تکرار الگوریتم سرعت و موقعیت هر عضو به کمک روابط (۶-۸) و (۶-۹) بروزرسانی می‌شود.

$$V_i^{k+1} = wV_i^k + c_1r_1\left(p_i - X_i^k\right) + c_2r_2(p_g - X_i^k) \qquad (۸-۰)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \qquad (۹-۰)$$

در رابطه (۶-۸)، $w$ وزن اینرسی که معمولا بین ۰/۴ و ۰/۹ انتخاب می‌شود. $c_1$ ضریب یادگیری خصوصی و $c_2$ ضریب یادگیری سراسری است. $r_1$و $r_2$ نیز دو عدد تصادفی است که از بازه ۰ تا ۱ انتخاب می‌شود. مقدار n برابر جمعیت و k مرتبه تکرار است.

همان گونه که از رابطه (۶-۸) مشخص است ، بردار سرعت جدید عضو i ام، از جمع سه بردار دیگر به دست می آید. بردار نخست، ضریبی از بردار سرعت تکرار قبلی الگوریتم است. بردار دوم ، ضریبی از بردار واصل موقعیت فعلی ذره و بهترین موقعیتی که تا کنون آن ذره داشته است و بردار سوم نیز ضریبی از بردار واصل موقعیت فعلی ذره و بهترین موقعیت کل مجموعه تا آن لحظه می‌باشد.

به کمک روابط (۶-۸) و (۶-۹) و در هر تکرار ، پس از به روزرسانی وضعیت همه اعضای جمعیت، کیفیت(مقدار تابع هزینه) هر عضو به عنوان نماینده یک پاسخ مسئله، دوباره با استفاده از تابع هزینه ارزیابی می گردد. اگر به روزرسانی وضعیت عضو i ام، منجر به بهبود کیفیت آن شد، وضعیت آن جایگزین $p_{ibest}$ و اگر به روزرسانی، منجر به بهبود وضعیت نشده باشد، $p_{ibest}$ تغییر نخواهد کرد. همچنین اگر بهترین عضو جمعیت جدید، بهتر از بهترین عضو جمعیت قبلی باشد، $p_{gbest}$ نیز جایگزین می‌شود. در پایان، زمانی که همه اعضا حول یک پاسخ واحد همگرا شدند، پاسخ بهینه مسئله به دست آمده است[۱۰].

به منظور همگرایی دقیق‌تر و مؤثرتر اعضای جمعیت به پاسخ بهینه، سرعت آنها در هر تکرار الگوریتم، با تنظیم مقدارw کاهش می‌یابد. اگر w بزرگ باشد، جست‌وجو سراسری و اگر w کوچک باشد، جست‌وجوی محلی کیفیتش بالا می‌رود. لذا این پارامتر نیز باید در هر تکرار با یک نرخ کاهشی، بروزرسانی گردد. این پارامتر در محدود ۰/۴ و ۰/۹ انتخاب می‌شود و در هر تکرار به کمک رابطه (۶-۱۰) بروز رسانی می گردد.

$$w_{it} = w_{max} - \left(\frac{w_{max} - w_{min}}{MaxIt}\right) it \qquad (۱۰-۰)$$

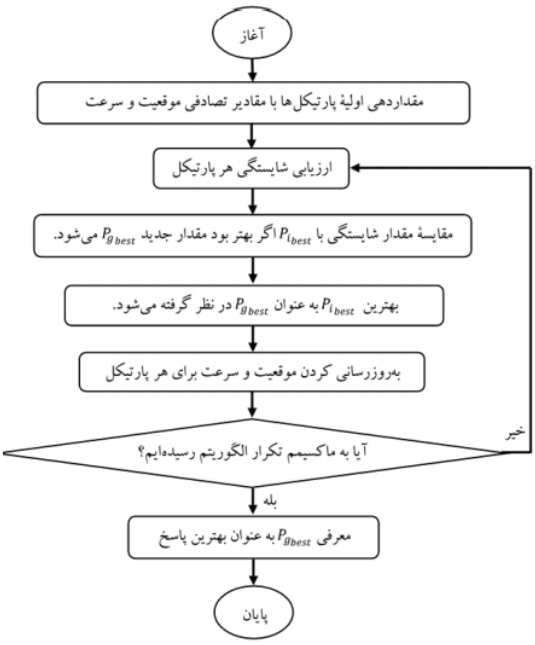
در رابطه بالا $w_{max}$ برابر ۰/۴ و $w_{max}$ برابر ۰/۹ است و $MaxIt$ تعداد کل تکرار های الگوریتم و مرتبه تکرار است.

اگر بخواهیم تمام پارامتر های الگوریتم را در یک جدول نمایش دهیم، به جدول ۶-۲ می‌رسیم.

جدول ۶-۲ پارامتر های الگوریتم تراکم ذرات

| | |
|---|---|
| حداکثر تعداد تکرار | maxIt |
| تعداد جمعیت حشرات | npop |
| وزن اینرسی | $w$ |
| ضریب یادگیری فردی | $c_1$ |
| ضریب یادگیری سراسری | $c_2$ |

در نهایت فلوچارت کلی الگوریتم به صورت شکل ۶-۲ است




شکل ۶-۲ فلوچارت الگوریتم تراکم ذرات

## الگوریتم کلونی زنبورها

در دهای گذشته، الگوریتم‌های متفاوتی با الهام از رفتار زنبورها برای حل انواع مسائل بهینه‌سازی‌معرفی شده است. در واقع اگر فردی بخواهد در این زمینه مطالعه ای داشته باشد، به دلیل وجود الگوریتم‌های مختلف در رابطه با زنبورها، دچار سردرگمی می‌شود. علت این که بر خلاف دو الگوریتم دیگر، یک الگوریتم پایه وجود ندارد این است که الگوریتم های مبتنی بر رفتار زنبورها، توسط افراد و تیم های مختلف، در یک بازه زمانی نزدیک بهم و مستقل از هم معرفی شده‌است و در نتیجه این الگوریتم‌ها در مسیر های متفاوت و توسط توسعه‌دهندگان مختلف ارائه شده و همین باعث شده‌است که نتوان آن ها را در یک قالب کلی بیان کرد. با این حال دو الگوریتم بهینه سازی کلونی زنبورها[1]یا به مخفف BCO و الگوریتم کلونی زنبور عسل مصنوعی[2] دو مورد از الگوریتم های نسبتا کلاسیک و جا افتاده در رابطه با زنبورها هستند. از آنجا که الگوریتم ABC در مسائل بهینه سازی پیوسته مورد استفاده قرار می‌گیرد، ما نیز در ادامه به معرفی و شرح این الگوریتم می‌پردازیم.

الگوریتم کلونی زنبورعسل مصنوعی در سال ۲۰۰۵ و توسط بس تورک[3]و کارابوگا[4]معرفی گردید. این الگوریتم با الهام از رفتار زنبورهای عسل برای یافتن منابع غذا طراحی شده‌است. در این الگوریتم، جمعیتی از منابع غذا وجود دارد که کلونی زنبورها این منابع را در طول زمان بهبود می بخشند. در این الگوریتم زنبورهای مصنوعی موجود در کلونی به سه دسته تقسیم می شوند: زنبورهای مشغول به کار که یک منبع غذایی را کشف کرده و از آن غذا می آورند، زنبورهای تماشاگر که درون کندو بوده و به رقص زنبورهای مشغول به کار نگاه می‌کند تا با استفاده از آن از موقعیت منبع غذا آگاه شوند و زنبور های پیشاهنگ که به طور تصادفی در محیط پیرامون کندو به دنبال غذا می‌گردند. روند طبیعی کار به این صورت است که ابتدا تعدادی زنبور به عنوان زنبور پیشاهنگ تعدادی منابع غذایی را در اطراف کندو پیدا می‌کنند و هنگامی که به کندو بازمی‌گردند، با یک مدل رقص خاص ، اطلاعاتی از آن منبع غذایی را در اختیار زنبورهایی با نام زنبور های ناظر قرار می‌دهند. بسته به این اطلاعات و این که هر منبع غذایی تا چه میزان خوب هست، تعدادی زنبور به عنوان زنبور کارگر به آن منابع غذایی می‌روند و شهد جمع می‌کنند و به کندو بازمی‌گردند. هر چه منبع غذایی با کیفیت‌تر باشد، تعداد زنبور بیشتری به عنوان زنبور کارگر به آن منبع ارسال می‌گردد.

اگر بخواهیم الگوریتم ABC را مرحله به مرحله بیان کنیم و مواردی که باعث میشود الگوریتم ABC ، جزو الگوریتم‌های خوب برای حل مسائل بهینه سازی باشد را شرح دهیم، به این صورت بیان خواهیم کرد:

[1] bee colony optimization
[2] artificial bee colony
[3] Basturk
[4] Karaboga

**مرحله مقدار دهی اولیه**

در ابتدای کار، الگوریتم بردار هایی را به عنوان پاسخ مسئله ( یا به طور معادل، موقعیتی که زنبور آن را به عنوان منبع غذا گزارش داده است) به صورت تصادفی تولید می‌کند. به عنوان مثال برای زنبور m ام ، یک بردار به صورت زیر تشکیل می‌شود.

$$x_m = [x_{m1}. x_{m2} \dots . x_{mn}]$$

که در این جا m ، شماره زنبور است که این m به تعداد Npop است. همچنین n نشان دهنده ی پارامتر های مسئله ای است که قرار است آن را بهینه کنیم. برای تولید هر $x_{ml}$ از رابطه (۶-۱۱)استفاده می‌شود.

$$x_{ml} = l_j + rand(0.1) * (u_j - l_j) \qquad (۱۱-۰)$$

که در رابطه (۶-۱۱) $l_j$ و $u_j$ کران های پایین و بالای پارامتر های مسئله هستند و $rand$ نیز یک مولد عدد تصادفی با توزیع یکنواخت می‌باشد.

**مرحله زنبور های کارگر**

پس از تولید پاسخ های اولیه، زنبور های کارگر به کمک رابطه ای که در ادامه بیان می‌کنیم، شروع به جست‌وجوی محلی در اطراف منبع های غذا (پاسخ های تولید شده در مرحله قبل) می‌کنند. رابطه ای که هر کدام از زنبورها در واقع به کمک آن ، به جست‌وجوی محلی و یافتن نقاط جدید نزدیک به منبع غذای قبلی میکنند، رابطه(۶-۱۲) است.

$$xNew_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \quad i \neq k \qquad (۱۲-۰)$$

که در رابطه (۶-۱۲)، j یکی از بعد ها (پارامترهای ) پاسخ است. $xNew_{ij}$ موقعیت جدید تولید شده (پاسخ جدید) است و $x_{ij}$ ، $\varphi_{ij}$ یک عدد تصادفی بین ۱- و ۱ و k نیز یک عدد تصادفی بین ۱ و تعداد کل پاسخ هاست. در واقع به کمک رابطه بالا، زنبورهایی که یک منبع غذایی را در نظر گرفتند، به صورت تصادفی و با گامی کوچک، در جهت یکی دیگر از منابع غذایی، به امید یافتن منبع بهتر حرکت می‌کنند . اگر کیفیت پاسخ جدید از کیفیت پاسخ قبلی بهتر باشد، پاسخ جدید جایگزین پاسخ قبلی می‌گردد و اگر پاسخ بهتر نبود، یک واحد به تعداد جست و جو های بی فایده در اطراف آن منبع غذا اضافه می‌گردد.

**مرحله زنبور های تماشاگر**

در این مرحله، بعد از آن که زنبور های کارگر به کندو بازگشتند، به زنبور های تماشگر تبدیل می‌شوند. زنبور های تماشگر، با یک احتمالی که در ادامه نحوه به دست آوردن آن را توضیح خواهیم داد، به سمت منابع غذایی پیدا شده می‌روند (هر چه کیفیت منبع غذا بیشتر باشد، احتمال رفتن زنبورهای بیشتر به آن منبع ، بیشتر است) و دوباره به جست‌وجوی محلی می‌پردازند. به کمک همان رابطه ای که در مرحله قبل توضیح داده شد یعنی رابطه (۶-۱۲)، این جست‌وجو را انجام می‌دهند. برای به دست آوردن احتمال بیان شده نیز ، به این صورت عمل می کنیم که به کمک رابطه (۶-۱۳)، میزان شایستگی هر یک منابع غذایی پیدا شده را به‌دست می‌آوریم

$$fit_i = \begin{cases} \frac{1}{1+f(x_i)} & f(x_i) \geq 0 \\ 1+|f(x_i)| & f(x_i) < 0 \end{cases} \qquad (۱۳-۰)$$

در رابطه (۶-۱۳) ، $f(x_i)$ میزان هزینه پاسخ $x_i$ می‌باشد. به این صورت میزان شایستگی هر کدام از منابع غذایی(پاسخ های مسئله) به‌دست می‌آید . سپس به کمک رابطه (۶-۱۴)، احتمال رفتن زنبورهای تماشگر به سمت آن منبع غذایی را به‌دست می‌آوریم.

$$p_i = \frac{fit_i}{\sum fit_i} \qquad (۱۴-۰)$$

در نتیجه به کمک روابط بیان شده ، جست و جوی محلی بیشتری در اطراف پاسخ های بهتر، انجام می‌شود که احتمال پیدا کردن نقاط بهینه تر را بالا می‌برد.

**مرحله زنبور های پیشاهنگ**

اگر یک منبع غذایی به عنوان پاسخ در تعدادی مشخصی از دفعات جست‌وجو، بهبود نیافت، یعنی زنبورها در جست‌وجوی محلی خود ، منبع غذایی بهتری در اطراف یکی از منابع غذایی پیدا نکردند، دیگر عملیات جست‌وجو متوقف می‌شود و آن منبع با یک منبع غذایی جدید که به کمک رابطه ای که در اول الگوریتم بیان شد، جایگزین می‌شود. به این صورت میزان جست‌وجوی سراسری الگوریتم را نیز در سطح خوبی نگه می‌داریم و مانع از گیرکردن در نقاط اکسترمم محلی می‌شویم. این تعداد دفعات که برای متوقف و جایگزین کردن منبع غذای جدید( پاسخ جدید) در نظر می گیریم را معمولا برابر با حاصل ضرب جمعیت اولیه در تعداد پارامتر های مسئله قرار می‌دهیم. همچنین این را در نظر داشته باشیم که این شمارش تعداد جست‌وجوی های بی فایده را در مرحله زنبور های کارگر انجام می‌دهیم. در

نهایت اگر به شرایط خاتمه الگوریتم رسیدیم ، با کیفیت ترین منبع به عنوان پاسخ مسئله و بهینه سراسری بیان می‌شود و اگر همچنان شرایط خاتمه مسئله فراهم نشده بود، دوباره به مرحله زنبور های کارگر باز می گردیم.

در نهایت الگوریتم ABC را میتوان در سه مرحله به صورت خلاصه توضیح داد.

۱- فرستادن زنبورهای کارگر به سمت منابع غذایی و ارزیابی کیفیت پاسخ آنها(جست و جوی محلی).

۲- انتخاب منابع غذایی بهتر و جایگزینی آن با منابع قبلی به کمک زنبور های ناظر و با استفاده از اطلاعاتی که زنبور های کارگر می گیرند.

۳- انتخاب زنبورهای پیشاهنگ و فرستادن آنها برای یافتن منبع غذایی جدید[۱۱].

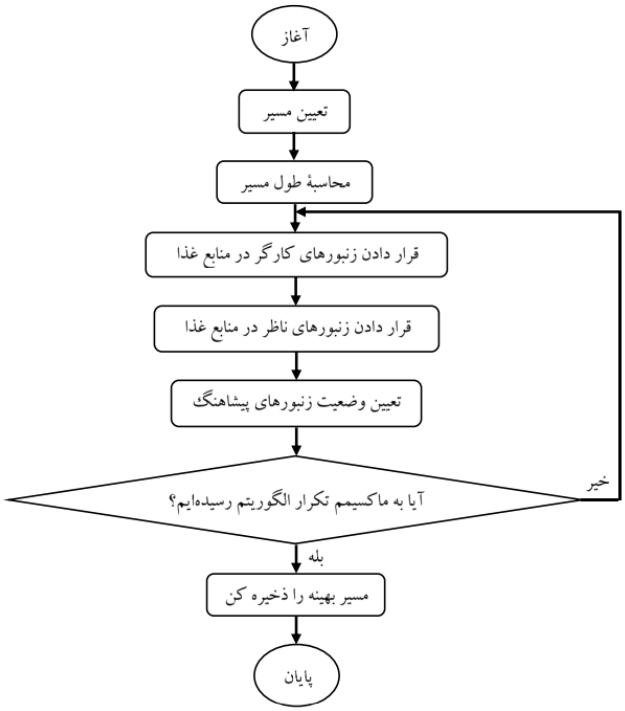
این نکته را باید در نظر داشته باشیم که پارامتر های این الگوریتم بسیار کم است و اکثر مقادیر به صورت تصادفی و یا محاسبه یک رابطه به‌دست می آیند. با این حال همان تعداد کم پارامتر ها نیز، بر خلاف الگوریتم های دیگر، در مقالات ، مقادیر خاصی برای آن ها در نظر گرفته نشده و ما با آزمون و خطا به مقدار مناسب آن ها می‌رسیم.

جدول پارامتر های این الگوریتم در جدول ۶-۳ آمده است.

جدول ۶-۳ پارامتر های الگوریتم کلونی زنبورها

| | |
|---|---|
| حداکثر تعداد تکرار | maxIt |
| تعداد جمعیت زنبور ها | npop |
| حداکثر تعداد جست و جوی بی فایده برای هر منبع غذایی | *limit* |
| ضریبی برای میزان جابه جایی در اطراف منبع غذایی برای جست و جوی محلی(که البته معمولا بین ۱- و ۱ در نظر گرفته میشود) | $\varphi_{ij}$ |

فلوچارت کلی این الگوریتم متناسب با مسئله مسیر یابی ما به صورت شکل ۶-۳ خواهد بود.




شکل ۶-۳ فلوچارت الگوریتم کلونی زنبورها

# فصل هفتم: تنظیم پارامتر

در این فصل ابتدا در مورد پارامتر های سه الگوریتم مطرح شده در فصل قبل توضیحاتی داده و مقادیر مناسبی که برای هرکدام در مقالات معتبر بیان شده است را مطرح می‌کنیم. پس از بیان این مقادیر، آزمایش ها و مقادیری را که خودمان در این پروژه ، برای مسئله مسیریابی انجام داده‌ایم و به‌دست آورده‌ایم را شرح می‌دهیم.

## پارامتر های الگوریتم PSO و تنظیم آن

همان طور که در فصل گذشته گفته شد، رابطه اصلی الگوریتم PSO که در واقع بر اساس آن ، عضوها در فضای مسئله ، جست‌وجو می‌کنند، رابطه (۷-۱) و (۷-۲) می‌باشد.

$$V_i^{k+1} = wV_i^k + c_1 r_1 (p_i - X_i^k) + c_2 r_2 (p_g - X_i^k) \quad (۱-۰)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (۲-۰)$$

پارامتر های تخصصی این الگوریتم و این رابطه ها $W$ و $c_1$ و $c_2$ است. $W$ وزن اینرسی و $c_1$ و $c_2$ ضرایب یادگیری فردی و اجتماعی می‌باشد. $r_1$ و $r_2$ نیز دو عدد تصادفی بین ۰ و ۱ هستند که در هر بار تکرار الگوریتم، یک عدد تصادفی جدید جایگزین می‌شود .

پارامتر $W$ که وزن اینرسی نام دارد، در واقع پارامتری است که سرعت حرکت ذرات در فضای جست‌وجو را تنظیم می‌کنیم. هر چقدر که مقدار $W$ بزرگ باشد، جست‌وجو در فضای مسئله، به صورت جست‌وجوی سراسری با گام های بزرگ تغییر حالت پیدا می‌کند و هرچقدر که این مقدار $W$ کوچک‌تر باشد، جست‌وجو به سمت یک جست‌وجوی محلی با گام های کوچک‌تر پیش خواهد رفت. ما برای حل مسائل بهینه سازی به کمک الگوریتم PSO نیاز داریم که در ابتدای کار، جست‌وجوی ما بیشتر به صورت جست و جوی سراسری باشد. در واقع در ابتدای کار می‌خواهیم محدوده‌ی بهترین جواب ها را پیدا کنیم. رفته رفته که محدوده ی بهترین جواب یا بهترین جواب ها را پیدا کردیم، تمایل داریم مقدار دقیق جواب بهینه را پیدا کنیم. در واقع با این کار دیگر مشکل پیدا کردن بهترین مقدار اولیه برای شروع الگوریتم را نداریم. چرا که الگوریتم مستقل از جواب های اولیه، در ابتدا، شروع به جست‌وجوی سراسری می‌کند و بعد شروع به جست‌وجوی محلی می‌کند. بنابراین ما نیز داریم که رفته رفته مقدار $W$ به سمت اعداد کوچک کاهش پیدا کند. مطابق با مقاله[۱۲] که توسط شی و ابرهات در سال ۱۹۹۸ ارائه شد ، بهترین محدوده ای که $W$ میتواند داشته باشد، محدوده ای بین ۱ و ۰/۵ می‌باشد. در واقع در ابتدای شروع الگوریتم مقدار $W$ را برابر عددی نزدیک به ۱، مثلا ۱/۲ قرار

بدهیم و مقدار آن در هر با تکرار حلقه الگوریتم مقداری کاهش دهیم، به نحوه که در آخرین تکرارهای حلقه الگوریم، مقدار آن در نزدیکی عدد ۰/۵ باشد. این دو فرد این محدود اعداد را بر اساس انجام تعداد زیادی آزمایش بر روی یک تابع برای یافتن مقدار بهینه آن تابع به ازای $w$ مختلف به‌دست آورده‌اند.

پارامتر های $c_1$که ضریب یادگیری فردی و $c_2$که ضریب یادگیری اجتماعی نام دارند به ترتیب اهمیت نسبی موقعیت خود ذره نسبت به خود و موقعیت کل ذرات جمعیت می باشد. در واقع ضرایب $c_1$ و $c_2$ بیانگر شدت هل دادن ذره به سمت بهترین موقعیت تجربه شده و بهترین موقعیت کل فضای جست و جو می‌باشد. این ضرایب طبق مقاله[۱۱]که در واقع مقاله اصلی در مورد الگوریتم pso است، بسته مسئله و شرایط آن میتواند مقادیر مختلفی داشته باشد.البته طبق گفته نویسندگان مقاله، بر اساس آزمایش های صورت گرفته، مقدار ۲ برای اکثر کاربرد ها، مقدار مناسبی برای $c_1$ و $c_2$ می‌باشد.

## پارامتر های الگوریتم FA و تنظیم آن ها

در فصل قبل که رابطه کلی الگوریتم fa که حشرات شب‌تاب ( پاسخ های به‌دست آمده برای مسئله) بر اساس آن در فضای جست و جوی مسئله ، جابه جا میشوند به فرم رابطه (۷-۳) می‌باشد.

$$x_{i\,new} = x_i + \beta_0 e^{-\gamma r^2 ij}\left(x_{ik} - x_{jk}\right) + \alpha\epsilon_i \qquad (۳-۰)$$

پارامتر های تخصصی و مربوط به این الگوریتم $\beta_0$ و $\gamma$ و $\alpha$ می‌باشد. $\gamma$ ضریب جذب نور توسط محیط است. این پارامتر نحوه کاهش کشش حشرات شب‌تاب به سمت یکدیگر را با افزایش فاصله بین آنها، تعیین می کند. به‌ازای $\gamma = 0$ جذابیت حشرات شب‌تاب برای یکدیگر مستقل از فاصله بین آنها می‌باشد و برابر با عددی ثابت خواهد بود که البته این رفتار در تناقض با رفتار واقعی حشرات شب‌تاب در طبیعت می‌باشد. همچنین مقدار خیلی بزرگ $\gamma$ ، باعث می‌شود که جذابیت حشرات شب‌تاب به صفر میل کند که این خود باعث می‌شود مسئله به یک مسئله جست‌وجوی تصادفی و بدون همکاری با سایر اعضای جمعیت برای یافتن جواب بهینه در فضای مسئله، تبدیل شود. مطابق با مرجع شماره [۱۳] استفاده از مقادیر بین ۰ تا ۱۰ برای $\gamma$ پیشنهاد شده‌است. همچنین مرجع [۱۴] نیز توصیه کرده است برای $\gamma$ از رابطه (۷-۴) استفاده کنید.

$$\gamma = \frac{\gamma_0}{r_{max}^2} \quad (۴-۷)$$

که $r_{max}$ برابر با حداکثر فاصله بین دو حشره شب‌تاب می‌باشد. همچنین برای مقدار $\gamma_0$ انتخاب یک عدد بین ۰ و ۱ پیشنهاد شده است.

پارامتر $\beta_0$ بیانگرحداکثر ضریب جذابیت بین دو حشره شب‌تاب را بیان می‌کند. در حالت کلی $\beta_0$ باید عددی بین ۰ و ۱ انتخاب شود. اگر $\beta_0$ برابر ۰ قرار گیرد، عملا یک جست‌وجوی تصادفی بدون حافظه که در آن هر یک از حشرات شب‌تاب بدون همکاری با سایر حشرات شب‌تاب ، به تنهایی به دنبال جواب می‌گردد، انجام خواهد شد. همچنین اگر $\beta_0$ برابر ۱ قرار گیرد، در واقع درخشان ترین حشره شب‌تاب ، با تمام قدرت سایر حشرات را به سمت خود جذب می‌کند. مطابق با مقاله [۱۳] و[۱۴] توصیه شده است که از مقدار ۱ برای $\beta_0$ استفاده شود تا در واقع الگوریتم با تمام قدرت به دنبال بهترین جواب باشد.

$\alpha$ ضریب برداری جابه جایی تصادفی می‌باشد که در واقع این پارامتری است که کمک می‌کند، در مینیمم های محلی گرفتار نشویم و در صورت بودن جواب بهتر، آن را پیدا کنیم. مطابق با مقاله [۱۴] بهترین مقدار برای آن برابر ۰/۲ است. این بهترین مقدار را از روی اجرای الگوریتم روی توابع مختلف و با مقادیر مختلف $\alpha$ به‌دست آورده است.

## پارامتر های الگوریتم ABC و تنظیم آن ها

همان طور که در فصل قبل هم بیان شد، در الگوریتم کلونی زنبور ها ، رابطه ای که بر اساس آن زنبور ها در فضای جست‌وجو ، جابه‌جا میشوند و به دنبال جواب بهینه میگردند رابطه (۷-۵) می‌باشد.

$$xNew_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \quad i \neq k \quad (۵-۰)$$

البته این الگوریتم ، از بخش ها و مراحل مختلفی تشکیل شده است و مانند الگوریتم های قبلی فقط بر اساس یک رابطه، کار خودش را انجام نمی‌دهد. همان طور که در گذشته نیز بیان شد، در این الگوریتم اگر زنبور های کارگر پس از تعدادی مشخص تلاش و جست‌وجو در اطراف یکی از پاسخ های به‌دست آمده، نتوانستند جواب بهتری پیدا کنند، به زنبور پیشاهنگ تبدیل می‌شوند و به صورت تصادفی، مکانی دیگر در فضای جست و جو را می‌پیمایند.این تعداد مشخص تلاش را $limit$ مسئله نام گذاری می کنیم. پس در مجموع، این الگوریتم غیر از پارامتر های عمومی که تعداد اعضای جمعیت و تعداد تکرار الگوریتم می باشد، دو پارامتر تخصصی $limit$ و $\varphi_{ij}$ دارد.

$limit$ یا نام دیگر آن که معیار کنارگذاری هست، در واقع حداکثر فرصتی است که برای یافتن جواب بهتر در اطراف جواب های یافت شده، به الگوریتم می‌دهیم و $\varphi_{ij}$ نیز ، ضریب میزان گامی است که زنبور می‌تواند در اطراف جواب یافت شده، جست‌وجوی محلی انجام دهد تا پاسخ بهتری پیدا کند.

بر اساس کتاب الگوریتم‌های بهینه سازی الهام گرفته شده از طبیعت دکتر فرشاد مریخ بیات که در واقع یک گردآوری از یک سری مقالات معتبر در این حوزه می‌باشد، برای این الگوریتم برخلاف دو الگوریتم دیگر، هیچ روش نظام مندی برای تعیین پارامتر های آن وجود ندارد و بسته به مسئله ای که می‌خواهیم آن را حل کنیم، پارامترهای مسئله متفاوت خواهد بود . با این حال، طبق بررسی های صورت گرفته بر روی مقاله اصلی مرتبط به معرفی $ABC$[۱۱] و همچنین مقاله [۱۵]، نکاتی را در مورد مقداردهی الگوریتم‌ها این مسئله بیان شده‌است ولی با این حال مقدار عددی ثابتی برای هر کدام بیان ذکر نشده‌است. بر اساس مقاله [۱۵] ، تغییر ابعاد مسئله در الگوریتم $ABC$ نسبت به الگوریتمی مثل $PSO$، تغییر چندانی در جواب به‌دست آمده ایجاد نمی‌کند. درواقع در الگوریتمی مثل $PSO$، اگر ابعاد مسئله را زیاد کنیم ، به احتمال خیلی زیاد جواب مناسبی را نسبت به حالت قبلی به‌دست نخواهیم آورد ولی الگوریتم $ABC$ حتی در حالت‌هایی که ابعاد مسئله را چند برابر می‌کنیم، مقاومت خوبی دارد و همچنان پاسخ بهینه را پیدا می‌کند. مقاله[۱۵] نتایج و صحبت های خود را بر اساس اجرا الگوریتم ها بر روی یک سری توابع معروف و پیچیده و با تعداد بالا بیان می‌کند. همچنین در مورد تعداد اعضای جمعیت زنبورها نیز بیان کرده است که بر خلاف الگوریتم $PSO$ ، الگوریتم $ABC$ با تعداد کم جمعیت، همچنان می‌تواند به پاسخ بهینه برسد. در ضمن در مورد مقدار دهی اولیه برای زنبور ها در ابتدا الگوریتم نیز بیان کرده است که چون که این الگوریتم هم در فاز جست‌وجو سراسری و هم در فاز جست‌وجو محلی ، از تعادل خوبی برخوردار است، در نتیجه مقدار دهی اولیه، حتی اگر خیلی بد باشد، الگوریتم باز هم می‌تواند جواب بهینه را پیدا کند زیرا این الگوریتم تقریب خوبی از کل فضای مسئله را جست‌وجو می‌کند. همچنین در مورد پارامتر آستانه کنارگذاری یا همان $limit$، هم مقاله [۱۱] و هم[۱۵]، مقدار مناسب برای این پارامتر را برابر حاصل ضرب تعداد زنبورهای جست‌وجو گر در ابعاد مسئله قرار داده اند. دلیل قرار دادن این مقدار را هم به این صورت بیان کرده است که در هر بار

تکرار الگوریتم و در واقع در هر بار جست‌وجو زنبورها، چون که نهایتا یک بعد از ابعاد پاسخ به روز می‌شود، پس در حداقل‌ترین حالت ممکن نیاز است به اندازه تعداد زنبور ها در ابعاد مسئله، فرصت جست‌وجو را به الگوریتم بدهیم. البته در مقاله*[15]*و بر اساس آزمایش های متعددی که نویسندگان آن انجام داده اند، بیان شده‌است که الگوریتم وابستگی زیادی به مقدار این پارامتر ندارد و به ازای مقادیر مختلف، باز هم به جواب میرسد. تنها نکته ای که باید به آن توجه کرد این است که اگر جمعیت اولیه زنبور ها را مقدار کمی انتخاب کردیم، باید حواسمان باشد که مقدار *limit* خیلی کم نباشد زیرا در این صورت الگوریتم بیشتر در فاز جست‌وجو تصادفی خواهد رفت. در مورد پارامتر ضریب گام جابه‌جایی نیز تنها به همین نکته بسنده کرده است که آن را به صورت یک عدد تصادفی بین ۱- تا ۱ قرار می‌دهیم. البته در آخر نیز بیان کرده است که میتوان از برخی مقادیر آماری مثل واریانس استفاده کرد.

درنهایت همان طور که در اول بیان شد، در این الگوریتم، مقدار ثابت و خاصی برای پارامتر های تخصصی آن ، در مقاله های مرتبط با آن بیان نشده است و تنها نکاتی در مورد نحوه مقدار دهی بیان شده است که بسته به مسئله خودمان باید به آن نکات نیز توجه داشته باشیم.

در ادامه، کار تنظیم پارامتر را به صورت دستی و با انجام آزمایش های متعدد و اجرای چندین باره الگوریتم ها روی محیط های مختلف انجام می‌دهیم. در اینجا معیار خود را الگوریتم ژنتیک که یکی از معروف ترین الگوریتم‌های فرا ابتکاری است، قرار می‌دهیم. پیش از آن که وارد بخش تنظیم پارامتر بشویم، ابتدا توضیحات مختصری در مورد الگوریتم ژنتیک در ادامه شرح می‌دهیم.

## الگوریتم ژنتیک

الگوریتم ژنتیک که احتمالا معروف ترین الگوریتم بهینه سازی مبتنی بر هوش دسته جمعی است، اولین بار در دهه هفتاد میلادی توسط فردی به نام جان هلند[1]ابداع شد. الگوریتم ژنتیک در واقع یک روش جست‌وجو برای یافتن جواب تقریبی مسائل بهینه سازی با استفاده از مفاهیم علم زیست شناسی مانند وراثت و جهش است. در این الگوریتم که بر اساس نظریه تکامل داروین بنا شده است، ابتدا متغییر های مسئله با استفاده از رشته‌های دودویی مناسب کد گذاری می شوند وسپس با شبیه سازی کامپیوتری قوانین تنازع برای بقا، دائما رشته‌های مناسب‌تری به دست می‌آید. منظوراز مناسب‌تر در واقع جواب‌های بهینه‌تر می‌باشد. احتمال یافتن جواب بهینه سراسری توسط الگوریتم ژنتیک در صورت استفاده از مقادیر مناسب برای پارامترهای الگوریتم بسیار بالا می‌باشد.

[1] john holland

حل هر مسئله بهینه‌سازی پیوسته با استفاده از الگوریتم ژنتیک با انتخاب جمعیت اولیه ای از بردارها یا همان رشته های دودویی به صورت تصادفی آغاز می‌شود. در اکثر مواقع تعداد این رشته‌های دودویی در طی اجرا الگوریتم ثابت می‌ماند. پس از ایجاد جمعیت اولیه از رشته ها و با انجام یک سری عملیات بر روی آنها، دائما مجموعه جدیدی از رشته ها با برازندگی بهتر تولید می‌شود. در الگوریتم ژنتیک برای تولید یک مجموعه از رشته‌های جدید از روی رشته‌های قبلی ، سه نوع عملیات روی رشته ها صورت میگیرد. تقاطع، جهش و گزینش. معمولا سهم هر یک از این سه عملیات پیش از اجرای برنامه مشخص می‌شود. پس از تولید هر مجموعه جدید از رشته های دودویی، مقدار متناسب با هر رشته توسط تابعی مشخص و به آنها اختصاص داده می‌شود و شرط خاتمه الگوریتم بررسی می‌شود. در واقع درهراجرای الگوریتم مراحل گزینش، تقاطع و جهش به ترتیب انجام می‌شوند و سپس ارزیابی هر یک از آنها صورت می‌گیرد. در صورت برقرار نبودن شرط توقف این مراحل دوباره تکرار خواهند شد. پس از تولید نسل های جدیدی در تکرار های بعدی و برقرار شدن شرط توقف الگوریتم، الگوریتم متوقف می‌شود و بهترین رشته به‌دست آمده در آخرین نسل، به عنوان جواب مسئله ارائه می‌شود. با توجه به ماهیت تصادفی الگوریتم ژنتیک باید این مورد را بیان کرد که ممکن است که جواب به‌دست آمده جواب بهینه سراسری نباشد و به همین دلیل لازم است تا این الگوریتم چندین بار اجرا شود و نتایج مقایسه گردد تا بهترین جواب به‌دست آید.

در واقع اگر بخواهیم روند کلی الگوریتم ژنتیک را بیان کنیم ، به مراحل زیر خواهیم رسید.

۱- ایجاد جمعیت تصادفی و ارزیابی آنها

۲- انتخاب والدین و انجام عملگر ترکیب برای ایجاد فرزندان جدید

۳ - انتخاب از اعضای جمعیت برای انجام عمل جهش و ایجاد جمعیت جدید

۴- ادغام جمعیت اصلی ، فرزندان و جهش یافتگان و ایجاد یک جمعیت جدید

۵- بررسی شرط توقف الگوریتم.در صورت عدم توقف از مرحله ۲ دوباره شروع می‌کنیم[۱۶] .

## تنظیم پارامتر به روش دسته بندی

برای این کار ما سه محدوده ی اعداد بزرگ، متوسط و کوچک را برای پارامتر ها در نظر می‌گیریم. برای مقدار بزرگ ، اعدادی در نزدیکی ۱۰۰۰۰ ، برای مقدار متوسط، اعدادی در حدود ۱۰۰ و برای مقدار کوچک ، اعدادی در حدود ۱ در نظر گرفته می‌شود. هم چنین ضریب خطا به قدری بزرگ در نظر گرفته شده است که تا اگر برخوردی صورت گرفت ، مقدار هزینه مسیر عدد بزرگی به‌دست آید. برای این کار یکی از محیط هایی که در فصل های گذشته

در مورد آن توضیحاتی دادیم، به عنوان محیط آزمایش خود قرار می‌دهیم. همچنین معیار ما ، مسیری است که الگوریتم ژنتیک با هزینه ۴/۶ برای ما تولید می کند. برای هر حالتی ، الگوریتم ها را ۵ مرتبه با تعداد تکرار ۳۰۰ بار اجرا می کنیم. از آنجا که الگوریتم PSO و FA هر کدام سه پارامتر مخصوص به خود و الگوریتم ABC نیز ۲ پارامتر مخصوص به خود را دارا می‌باشد، در مجموع ۶۳ حالت مختلف وجود دارد که برای هر کدام ۵ مرتبه الگوریتم مربوطه اجرا شده است . برای طولانی نشدن این فصل از نوشتن تمام نتایج به دست آمده در این فصل خودداری می کنیم و فقط بهترین نتیجه را برای هر کدام از الگوریتم ها بیان می کنیم. با این حال تمام آزمایش ها و نتایج به‌دست آمده در پیوست شماره ۱ گزارش قابل مشاهده می‌باشد.

برای الگوریتم PSO ، هنگامی که هرسه مقادیر $w$ و $c_1$ و $c_2$ در بازه اعداد کوچک در نظر بگیرم، بهترین نتایج را به‌دست می آوریم.برای الگوریتم ABC، زمانی که limit بزرگ و $\varphi$ کوچک در نظر بگیریم، نتایج به‌دست آمده قابل قبول است و به نتیجه الگوریتم ژنتیک بسیار نزدیک می‌باشد.برای الگوریتم FA نیز هنگامی که مقادیر $\beta_0$ و $\gamma$ و $\alpha$ را در بازه اعداد کوچک در نظر می گیریم، بهترین نتایج را به‌دست می‌آوریم.

در نهایت، برای انجام آزمایش و بررسی عملکرد الگوریتم ها مقادیر زیر برای پارامترهای هر الگوریتم در نظر گرفته ایم.

۱- برای الگوریتم PSO، مقدار $w$ را برابر ۱ و مقدار $c_1$ و $c_2$ برابر ۲ قرار می‌دهیم.

۲- برای الگوریتم ABC، مقدار limit را برابر حاصل ضرب تعداد زنبور ها در تعداد متغییر های مسئله و مقدار $\varphi$ را برابر ۱ قرار می‌دهیم.

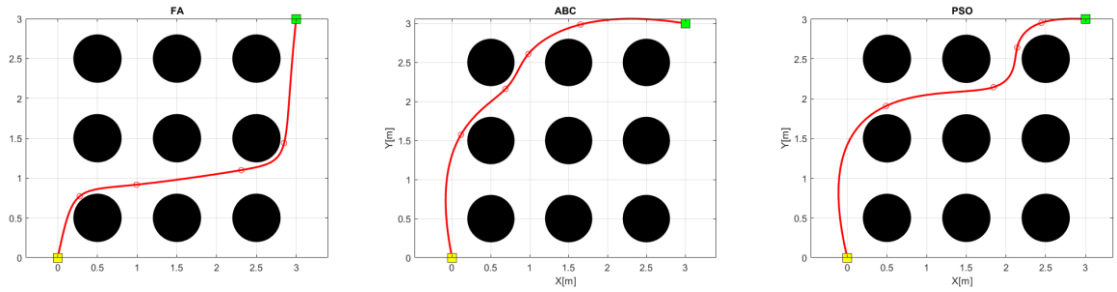
۳- برای الگوریتم FA، مقدار $\beta_0$ را برابر ۲ و مقدار $\gamma$ و $\alpha$ را برابر ۱ قرار می‌دهیم.

# فصل هشتم: مقایسه سه الگوریتم PSO، ABC و FA

در فصل چهارم، محیط هایی که قرار است در آن آزمایش‌های خود را انجام دهیم، معرفی کردیم. همچنین معیارهای مهمی که میتوان به کمک آنها الگوریتم ها را با یک‌دیگر مقایسه کرد را بیان کردیم. حال در این فصل قصد داریم سه الگوریتم PSO و FA و ABC در شرایط یکسان با یک‌دیگر مقایسه کنیم. برای این کار هر الگوریتم را بر روی هر یک از محیط ها معرفی شده ۴۰ مرتبه اجرا می کنیم. شرط توقف تمام الگوریتم ها را نیز اجرا ۳۰۰ بار حلقه اصلی الگوریتم قرار داده‌ایم. مقادیر پارامتر های هر الگوریتم را نیز در فصل قبل مشخص کردیم. پس از اجرا و به‌دست آمدن نتایج، برای هر محیط، سه الگوریتم را در قالب یک جدول و دو نمودار با یک‌دیگر مقایسه می کنیم.

## نتایج به‌دست آمده برای محیط ۱

در تصویرشکل ۸-۱ کوتاه ترین مسیر های تولید شده توسط هر کدام از الگوریتم ها پس از ۴۰ مرتبه اجرا به دست آمده است. نتایج عددی و آماری به‌دست آمده از آزمایش الگوریتم ها درمحیط ۱ در جدول۸-۱ آمده است.



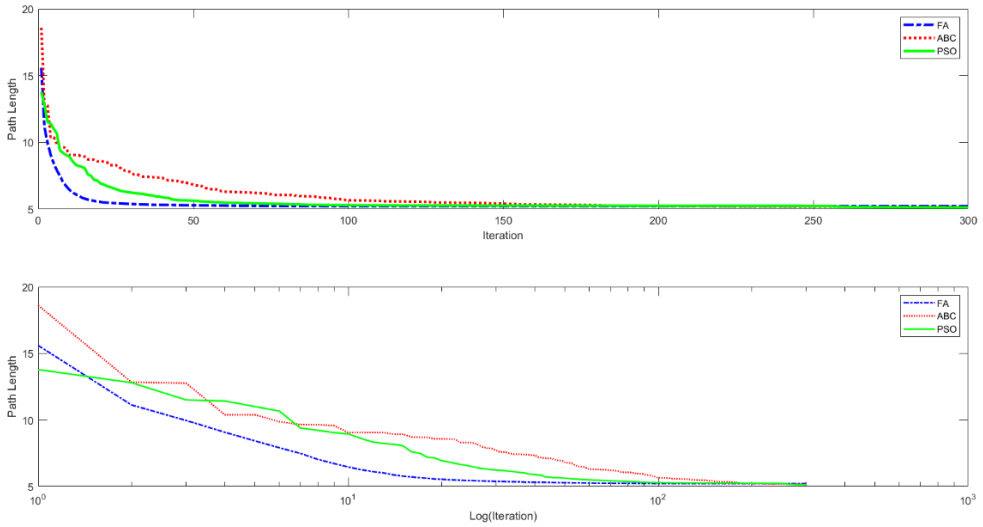

شکل ۸-۱ کوتاه ترین مسیر های طراحی شده توسط سه الگوریتم در محیط۱

جدول ۸-۱ نتایج به‌دست آمده از آزمایش سه الگوریتم روی محیط ۱

| معیار مقایسه / الگوریتم | طول کوتاه ترین مسیر | میانگین طول مسیر های به دست آمده | میانگین زمان محاسبات برای ۳۰۰ تکرار (ثانیه) | نرخ موفقیت | میانگین طول مسیر به دست آمده در ۲۰ ثانیه |
|---|---|---|---|---|---|
| PSO | 4.765 | 5.134 | 28.5 | 80 | 5.248 |
| ABC | 4.854 | 5.198 | 41.4 | 40 | 5.444 |
| FA | 4.450 | 5.220 | 1082.5 | 95 | 7.9096 |

از نتایج به‌دست آمده از این آزمایش می‌توان دریافت که کوتاه‌ترین مسیر به‌دست‌آمده توسط الگوریتم FA می‌باشد. البته به این نکته نیز باید توجه داشت که میانگین طول مسیر به دست آمده در الگوریتم ABC و PSO بهترمی باشد. نقطه ضعف الگوریتم FA مدت زمان محاسبات آن است که بسیار بالاتر از دو الگوریتم دیگر می‌باشد. با این حال نرخ موفقیت آن در دستیابی به مسیرهایی بدون برخورد بسیار بالا است. در حالی که دو الگوریتم دیگر به خصوص ABC عملکرد خوبی ندارد و این نکته بسیار مهم است. همچنین در بررسی سه الگوریتم در مدت زمان ثابت ۲۰ ثانیه میتوان دید که عملکرد PSO از سایر الگوریتم‌ها بهتر بوده‌است. در واقع با این که الگوریتم FA با تعداد تکرار کمتری به نتیجه بهینه رسیده است، ولی چون زمان هر تکرار این الگوریتم زیاد است، در زمان ثابت ۲۰ ثانیه الگوریتم PSO بهتر عمل کرده است.

همچنین میانگین روند کمینه سازی الگوریتم ها برای ۳۰۰ تکرار در ۴۰ آزمایش انجام شده به صورت نمودار شکل (۹-۲) به نمایش در آمده است.(در نمودار اول محور افقی به صورت عدد حقیقی و در نمودار دوم محور افقی به شکل لگاریتمی است تا روند بهینه سازی بهتر نمایش داده شود.)

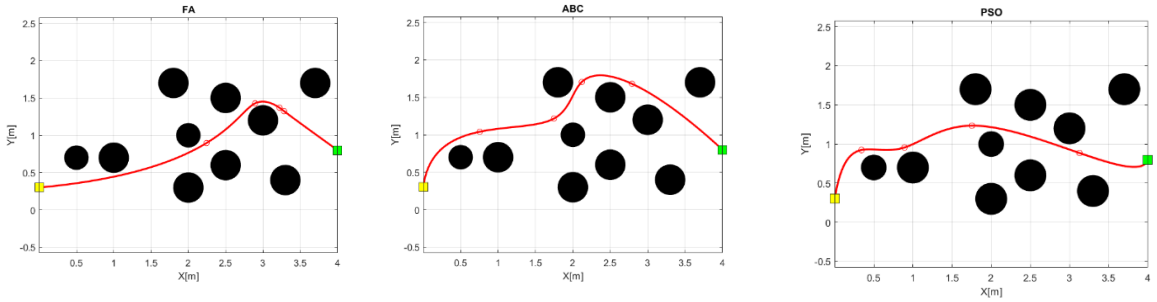



شکل ۸-۲ نمودار روند بهینه سازی سه الگوریتم در محیط ۱

همان طور که از نمودار های شکل۸-۲ قابل مشاهده است، الگوریتم FA پس از گذشت حدود ۳۰ تکرار به پاسخ بهینه خود می‌رسد و پس از آن جواب هایی که به‌دست می آورد، تغییر زیادی نمی‌کنند. الگوریتم PSO نیز پس از گذشت حدود ۱۰۰ تکرار به پاسخ بهینه خود نزدیک شده‌است. الگوریتم ABC نیز همان طور که از نمودارها قابل مشاهده است پس از گذشت حدود ۱۵۰ تکرار به پاسخ بهینه رسیده است و پس از آن تغییر چندانی نمی‌کند.

## نتایج به‌دست آمده برای محیط ۲

در تصویر شکل۸-۳ کوتاه ترین مسیر های تولید شده توسط هر کدام از الگوریتم ها پس از ۴۰ مرتبه اجرا به دست آمده است. نتایج عددی و آماری به‌دست آمده از آزمایش الگوریتم ها در محیط ۲ در جدول ۸-۲ قابل مشاهده‌است.



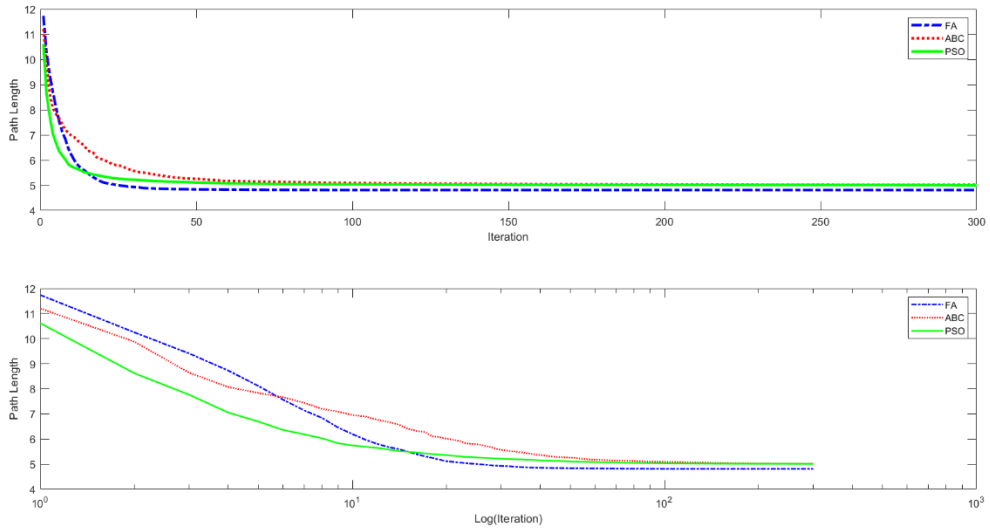

شکل ۸-۳ کوتاه ترین مسیر های طراحی شده توسط سه الگوریتم در محیط۲

جدول ۸-۲ نتایج به‌دست آمده از آزمایش سه الگوریتم روی محیط۲

| معیار مقایسه / الگوریتم | طول کوتاه ترین مسیر | میانگین طول مسیر های به دست آمده | میانگین زمان محاسبات برای ۳۰۰ تکرار (ثانیه) | نرخ موفقیت | میانگین طول مسیر به دست آمده در ۲۰ ثانیه |
|---|---|---|---|---|---|
| PSO | 4.552 | 4.981 | 26.3 | 100 | 5.012 |
| ABC | 4.471 | 5.012 | 42.1 | 80 | 5.053 |
| FA | 4.746 | 4.887 | 1057.5 | 100 | 6.360 |

در این محیط کوتاه ترین مسیر به دست آمده توسط الگوریتم ABC به‌دست آمده است. از لحاظ میانگین طول مسیر، الگوریتم FA بهتر عمل کرده و میانگین طول مسیر هایی که توسط این الگوریتم به دست می‌آید، از سایر الگوریتم‌ها کمتراست. الگوریتم‌های FA و PSO با این که عملکردی به خوبی ABC نداشته‌اند با این حل نتایج مطلوبی را به دست آورده اند. همچنین این دو الگوریتم نرخ موفقیت ۱۰۰ درصدی داشته‌اند که یعنی تمام مسیرهایی که در تکرار های مختلف به‌دست آورده اند، مسیرهای بدون برخوردی بوده است. در مقایسه سه الگوریتم در زمان ثابت ۲۰ ثانیه مشاهده می کنیم که دو الگوریتم PSO و ABC عملکرد نزدیک بهم داشته اند و PSO کمی بهتر از ABC است. برای همین مدت زمان، الگوریتم FA چون دارای محاسبات سنگین‌تری نسبت به دو الگوریتم دیگر است، میانگین طول مسیر به‌دست آمده توسط آن در مدت ۲۰ ثانیه از بقیه بالاتر و در واقع بدتر است.

در ادامه مانند شبیه سازی در محیط ۱، میانگین روند کمینه سازی الگوریتم ها در ۳۰۰ تکرار در ۴۰ آزمایش انجام گرفته را به صورت دو نمودار با محور افقی اعداد حقیقی و لگاریتمی برای مشاهده بهتر تغییرات ،در شکل (۹-۵) به نمایش می گذاریم.

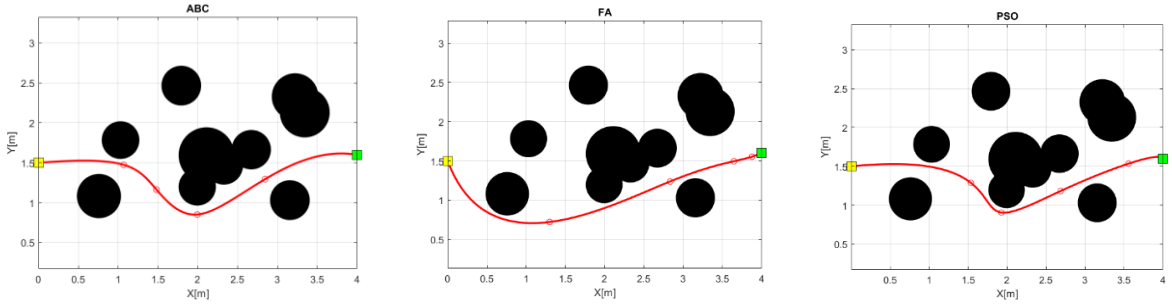



شکل ۸-۴ نمودار روند بهینه سازی سه الگوریتم در محیط۲

همان طور که از نمودار های شکل ۸-۴ قابل مشاهده است الگوریتم FA توانسته پس از تقریبا ۳۵ تکرار به مقدار بهینه خود برسد و پس از آن تغییر محسوسی نکرده‌است. همچنین الگوریتم PSO نیز توانسته پس از تقریبا ۵۰ تکرار به مقدار بهینه خود برسد. همچنین الگوریتم ABC پس از تقریبا ۱۰۰ تکرار به مقدار بهینه خود رسیده است.

## نتایج به‌دست آمده برای محیط ۳

در تصویر شکل ۸-۵، کوتاه ترین مسیر های تولید شده توسط هر کدام از الگوریتم ها پس از ۴۰ مرتبه اجرا به دست آمده است. نتایج عددی و آماری به دست آمده از آزمایش الگوریتم ها در محیط ۳ در جدول۸-۳ در ادامه آمده است.




شکل ۸-۵ کوتاه ترین مسیر های طراحی شده توسط سه الگوریتم در محیط۳

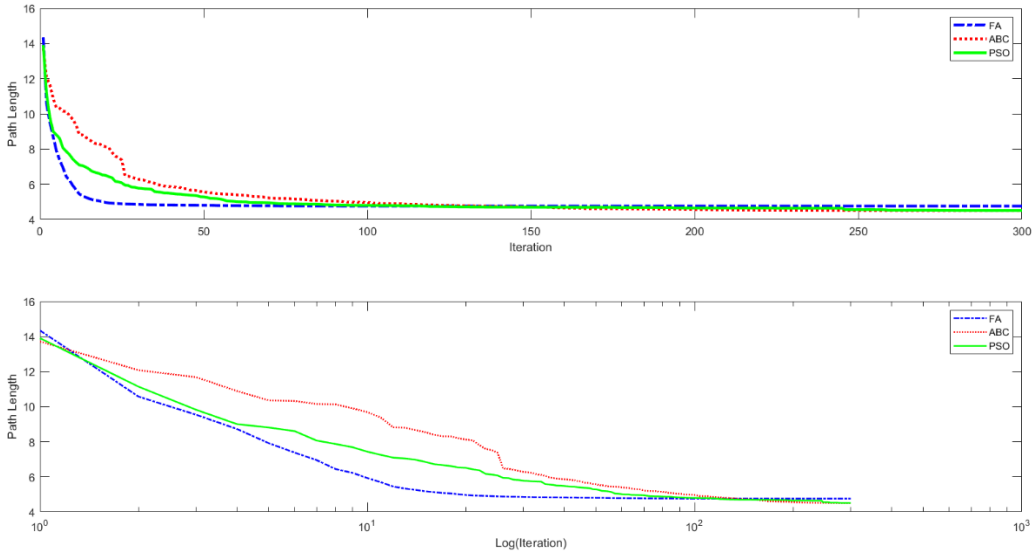
جدول ۸-۳ نتایج به‌دست آمده از آزمایش سه الگوریتم روی محیط۳

| معیار مقایسه / الگوریتم | طول کوتاه ترین مسیر | میانگین طول مسیر های به دست آمده | میانگین زمان محاسبات برای ۳۰۰ تکرار (ثانیه) | نرخ موفقیت | میانگین طول مسیر به دست آمده در ۲۰ ثانیه |
|---|---|---|---|---|---|
| PSO | 4.235 | 4.512 | 27 | 97.5 | 4.652 |
| ABC | 4.250 | 4.675 | 40.23 | 60 | 4.705 |
| FA | 4.232 | 4.752 | 1075 | 100 | 8.801 |

همان طور که از نتایج مشخص است هر سر الگوریتم برای این محیط عملکردی نسبتا مشابه داشته‌اند و نتایج نسبتا خوبی را به‌دست آورده اند. کوتاه ترین مسیر توسط الگوریتم FA به دست آمده است.از لحاظ میانگین طول مسیر، الگوریتم PSO از سایرین کمی بهتر عمل کرده است. همچنین الگوریتم FA در تمام آزمایش ها مسیر بدون برخوردی را پیدا کرده است. در این آزمایش نرخ موفقیت الگوریتم ABC نسبت به دو الگوریتم دیگر عدد کمتری است که این نکته بسیار مهم است. در ضمن همان طور که مشخص است، مدت زمان اجرا الگوریتم PSO از همه کمتر و FA از بقیه بیشتر

است. در مقایسه سه الگوریتم در زمان ثابت ۲۰ ثانیه همان طور که از جدول قابل مشاهده است ، PSO بهترین نتیجه را از سایر الگوریتم ها به‌دست آورده است.

در ادامه نیز مانند دو مورد قبل ، میانگین روند کمینه سازی الگوریتم ها را در ۳۰۰ تکرار در ۴۰ آزمایش انجام شده را در قالب دو نمودار در شکل (۹-۶) به نمایش می گذاریم.

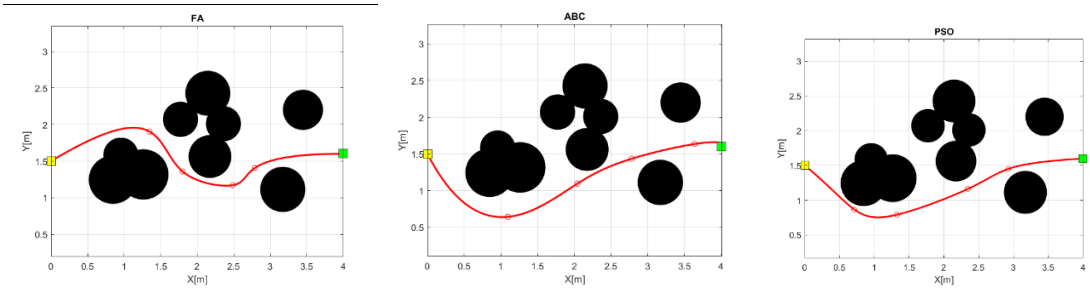



شکل ۸-۶ نمودار روند بهینه سازی سه الگوریتم در محیط۳

همان طور که از نمودار شکل ۸-۶ قابل مشاهده است ، الگوریتم FA سریعا به مقدار بهینه خود رسیده است. این الگوریتم پس از حدودا ۲۵ تکرار به این مقدار رسیده است. در حالی که PSO حدود پس از ۱۰۰ تا ۱۲۰ تکرار به مقدار بهینه خود رسیده است. الگوریتم ABC نیز کمی بد تر از PSO عمل کرده و حدود پس از ۱۴۰ تکرار به مقدار بهینه رسیده و پس از آن دچار تغییرات محسوس نشده است.

## نتایج به‌دست آمده برای محیط ۴

همانند محیط های قبل، در این محیط نیز هر سه الگوریتم را ۴۰ مرتبه اجرا کرده ایم. در ادامه کوتاه ترین مسیر به‌دست آمده توسط هر کدام از الگوریتم ها در تصویر شکل ۸-۷ به نمایش در آمده و بعد از آن نتایج عددی آزمایش های صورت گرفته را در جدول ۸-۴ به نمایش می گذاریم.



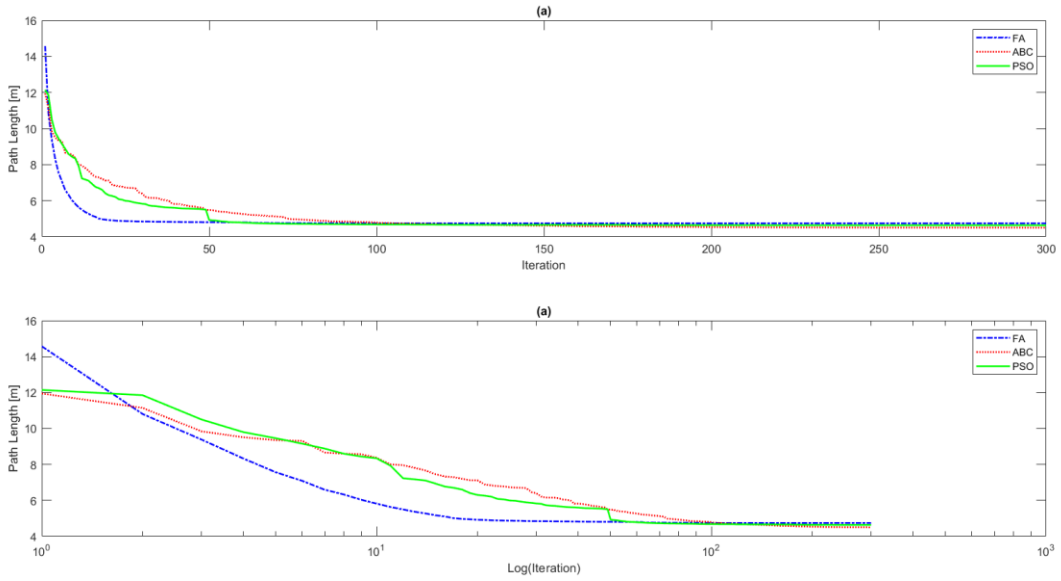

شکل ۸-۷ کوتاه ترین مسیر های طراحی شده توسط سه الگوریتم در محیط۴

جدول ۸-۴ نتایج به‌دست آمده از آزمایش سه الگوریتم روی محیط ۴

| معیار مقایسه / الگوریتم | طول کوتاه ترین مسیر | میانگین طول مسیر های به دست آمده | میانگین زمان محاسبات برای ۳۰۰ تکرار (ثانیه) | نرخ موفقیت | میانگین طول مسیر به دست آمده در ۲۰ ثانیه |
|---|---|---|---|---|---|
| PSO | 4.302 | 4.632 | 26.63 | 97.5 | 4.744 |
| ABC | 4.321 | 4.611 | 40.5 | 47.5 | 4.612 |
| FA | 4.431 | 4.746 | 1051.7 | 95 | 7.565 |

همان طور که از نتایج پیداست ، الگوریتم PSO در یافتن بهترین پاسخ عملکرد بهتری از دو الگوریتم دیگر دارد. با این حال دو الگوریتم دیگر یعنی FA و ABC نتیجه ای نزدیک به نتیجه PSO داشته‌اند و در کل هر سه الگوریتم از لحاظ پیدا کردن کوتاه ترین و میانگین طول مسیر تقریبا مشابه یک‌دیگر بوده‌اند. دو الگوریتم PSO و FA نرخ موفقیت بسیاری خوبی داشته‌اند و الگوریتم ABC همانند سایر محیط ها ، نرخ موفقیت ناامید کننده ای داشته است. همچنین همانند محیط های دیگر الگوریتم PSO کمترین میانگین زمان محاسبات و FA بیشتر میانگین زمان را دارد. در مقایسه سه

الگوریتم در مدت زمان ثابت ۲۰ ثانیه مشاهده می‌کنیم که بر خلاف موارد قبل ABC عملکرد بهتری از PSO دارد و طبق انتظار ،FA بدترین عملکرد در این زمان را داراست.

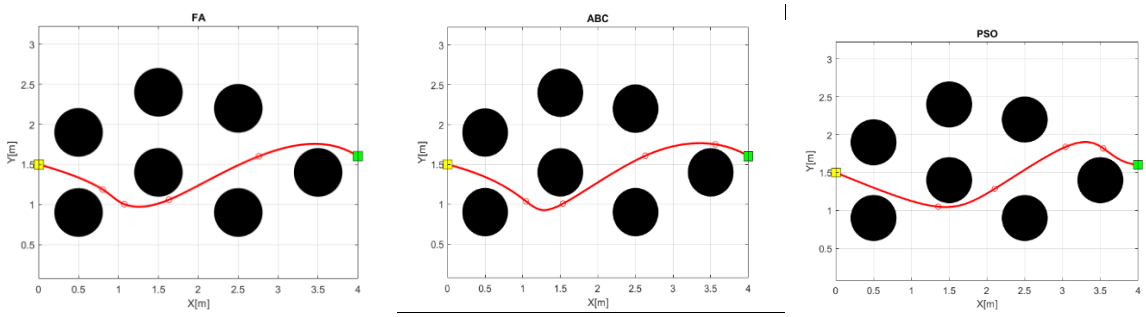



شکل ۸-۸ نمودار روند بهینه سازی سه الگوریتم در محیط۴

همان طور که از نمودارهای شکل ۸-۸ قابل مشاهده است، الگوریتم FA تقریبا بعد از تکرار مرتبه ۲۰، الگوریتم PSO تقریبا بعد از تکرار ۶۰ و الگوریتم ABC بعد از تکرار ۱۰۰ به مقدار بهینه خود می‌رسند و از آن به بعد تغییرات کمی دارند.

## نتایج به‌دست آمده برای محیط ۵

در این محیط نیز ، مانند سایر محیط ها الگوریتم ها را ۴۰ مرتبه اجرا میکنیم و در ادامه بهترین مسیر ها به‌دست آمده توسط هر کدام از الگوریتم ها در شکل ۸-۹ به نمایش گذاشته و نتایج عددی و آماری در جدول ۵-۸ نمایش می‌دهیم.

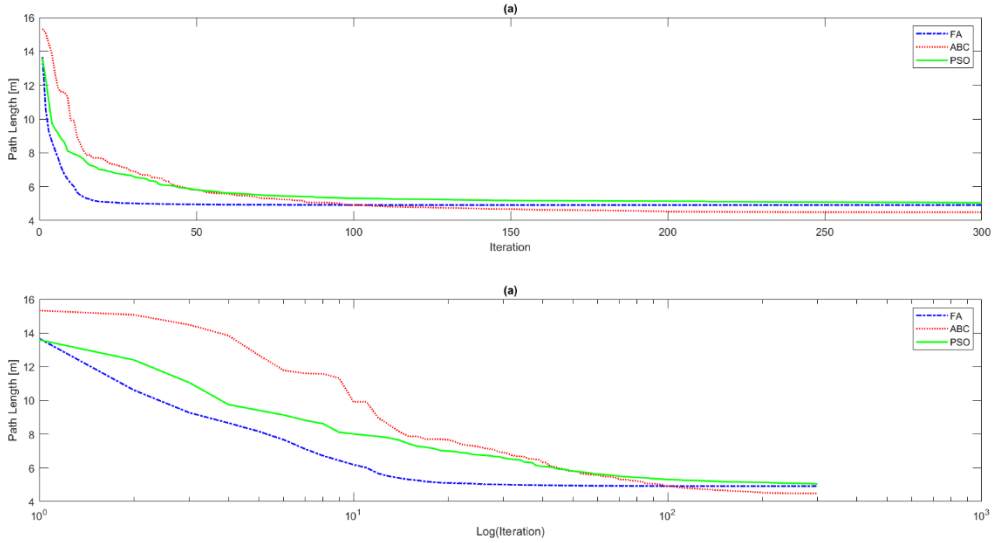



شکل ۸-۹ کوتاه ترین مسیر های طراحی شده توسط سه الگوریتم در محیط۵

جدول ۸-۵ نتایج به‌دست آمده از آزمایش سه الگوریتم روی محیط ۵

| معیار مقایسه / الگوریتم | طول کوتاه ترین مسیر | میانگین طول مسیر های به دست آمده | میانگین زمان محاسبات برای ۳۰۰ تکرار (ثانیه) | نرخ موفقیت | میانگین طول مسیر به دست آمده در ۲۰ ثانیه |
|---|---|---|---|---|---|
| PSO | 4.311 | 5.03 | 25.77 | 95 | 5.096 |
| ABC | 4.313 | 4.49 | 40.7 | 50 | 4.673 |
| FA | 4.310 | 4.91 | 1043.1 | 100 | 8.16 |

همان طور که از نتایج پیداست طول کوتاه ترین مسیر در هر سه الگوریتم تقریبا یکسان است. از لحاظ میانگین طول مسیر های به‌دست آمده ، الگوریتم ABC از همه بهتر عمل کرده است. پس از آن الگوریتم FA و پس از آن PSO. از لحاظ نرخ موفقیت FA از همه بهتر عمل کرده و PSO نیز تقریبا به همان اندازه خوب عمل کرده است. الگوریتم ABC نیز نرخ موفقیت خوبی نداشته است. در مورد میانگین زمان محاسبات نیز مانند سایر محیط ها ، PSO از همه سریع تر و FA از همه کندتر عمل کرده است. در این محیط نیز با انجام ازمایش در مدت زمان ثابت به این نتیجه رسیدیم که الگوریتم ABC بهترین عملکرد را دارد .




*شکل ۸-۱۰ نمودار روند بهینه سازی سه الگوریتم در محیط۵*

همان طور که از نمودار های شکل ۸-۱۰ مشخص است الگوریتم FA پس از حدود ۲۵ تکرار ، الگوریتم PSO پس از حدود ۱۵۰ تکرار و الگوریتم ABC نیز پس از ۱۶۰ تکرار به مقدار بهینه خود رسیده و پس از آن تغییرات نامحسوس کرده اند.

## بررسی و نتیجه‌گیری کلی

در مجموع ۲۰۰ بار اجرا الگوریتم ها بر روی ۵ محیط مختلف تحت شرایط یکسان نتایج بالا به‌دست آمده است که با بررسی کلی آنها به موارد زیر میرسیم.

۱- هر سه الگوریتم از عملکرد خوب و قابل قبولی در دستیابی به مسیر بهینه برخوردار بودند. الگوریتم FA در سه محیط و دو الگوریتم دیگر در هر کدام در یک محیط، از سایر الگوریتم ها نتایج بهتری داشته اند. همچنین در یک محیط سه مسیر به‌دست آمده توسط الگوریتم ها تقریبا یکی می‌باشد.

۲- در مورد زمان محاسبات ، الگوریتم PSO در همه محیط ها از بقیه الگوریتم ها بهتر عمل کرده است و سرعت محاسبات آن بسیار بالاست. همچنین الگوریتم ABC عملکرد قابل قبولی داشته است. اما الگوریتم FA نسبت به دو الگوریتم دیگر از زمان محاسبات بسیار بالاتری برخوردار بوده است که این یک عیب برای الگوریتم محسوب می‌شود.

۳- به طور میانگین در ۵ محیطی که آزمایش صورت گرفته است، الگوریتم FA پس از ۳۰ بار تکرار، الگوریتم PSO پس از ۱۴۰ بار و الگوریتم ABC پس از ۱۵۰ بار تکرار به پاسخ نهایی خود همگرا شده اند که این نشان دهنده سرعت بالا همگرایی در الگوریتم FA می‌باشد.

۴- با توجه به نتایج به‌دست آمد ، نرخ موفقیت در دستیابی به مسیر بدون برخورد در الگوریم FA بیشترین مقدار و در الگوریتم PSO کمتر مقدار را دارا می باشد. در واقع در الگوریتم FA نرخ موفقیت برابر ۹۸ درصد و در الگوریتم ABC برابر با ۵۵/۵ درصد می‌باشد.همچنین در الگوریتم PSO دارای مقدار ۹۴ درصد می‌باشد.

۵- در مورد میانگین طول مسیر به دست آمده در مدت زمان ۲۰ ثانیه برای سه الگوریتم ، الگوریتم PSO در ۳ محیط و الگوریتم ABC نیز در دو محیط بهترین جواب را نسبت به سایر الگوریتم‌ها به‌دست آورده اند.الگوریتم FA به دلیل سنگین بودن محاسباتش و این هر بار تکرار حلقه آن زمان زیادی صرف می‌شود، با این که زود همگرا می‌شود ولی در مدت زمان ۲۰ ثانیه نتوانسته است به خوبی سایر الگوریتم‌ها کار کند.

در مجموع نتایج به‌دست آمده نشان داد که هر سه الگوریتم نتایج مطلوبی را در به‌دست آوردن مسیر مناسب برای ربات ، به ما می‌دهند و در واقع میتوان با کار بیشتر بر روی آنها و انجام اصلاحات بر روی آنها و دستکاری پارامتر ها ، حتی نتایج مناسب تری نیز به‌دست آورد.

# فصل نهم: جمع بندی و نتیجه گیری

ما در این پروژه قصد داشتیم که در مورد مسیریابی ربات‌ها مطالبی را ارائه‌بدهیم و یک دسته از الگوریتم‌های مناسب برای این کار یعنی الگوریتم‌های برگرفته شده از طبیعیت را شرح داده و عملکرد آنها را نشان داده و یک سری از این الگوریتم‌ها را با یک دیگر مقایسه کنیم. برای این کار در ابتدا بحث مسیریابی برای ربات را شروع کردیم و انواع مسائل مسیریابی و الگوریتم‌های مسیریابی برای ربات ها را مطرح کردیم. در ادامه توجه و تمرکزمان را روی الگوریتم‌های برگرفته شده از طبیعیت قرار دادیم و به شرح کامل سه نمونه از این الگوریتم‌ها پرداختیم.سپس با طراحی محیط‌هایی، قصد داشتیم که عملکرد این الگوریتم‌ها را بررسی کنیم. قبل از اجرا الگوریتم ها و مقایسه آنها، کار تنظیم پارامتر را انجام دادیم و با انجام آزمایش های متعدد و استفاده از گفته‌های مقالات معتبر، مقادیر مناسب برای پارامترهای الگوریتم‌ها را به‌دست آوردیم. سپس کار آزمایش و مقایسه الگوریتم‌ها را بر روی محیط‌هایی که از قبل طراحی کردیم، انجام دادیم. با توجه به نتایج به‌دست آمده و ارزیابی‌های صورت گرفته، تا حد زیادی اهداف ما در این پروژه که معرفی الگوریتم‌های برگرفته شده از طبیعیت به عنوان الگوریتم‌های ساده و در عین حال قدرتمند، بود تحقق پیدا کرد.

برای ادامه کار این پروژه مسیرهای مختلف وجود دارد. الگوریتم‌های برگرفته شده از طبیعت بسیار زیاد هستند که میتوان آن‌ها را نیز بررسی کرد. همچنین می‌تواند محیط آزمایش را به حالت پویا در آورد. مسئله دیگری که میتوان در این زمینه به آن پرداخت ، استفاده از چند ربات به جای یک ربات در مسیر است که هرکدام باید مسیر مناسب خود را پیدا کنند و در عین حال، نباید به یک‌دیگر برخورد داشته باشند و از این دست مسائل که در این پروژه فرصت نشد که آنها نیز مورد بررسی قرار بگیرند.

# پیوست‌ها

## پیوست شماره ۱:

## نتایج آزمایش‌های تنظیم پارامتر دستی PSO

حالت ۱ ) مقدار w بزرگ ، مقدار c1 بزرگ ، مقدار c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.115 |
| 2 | 4.6851 |
| 3 | 5.1819 |
| 4 | 5.135 |
| 5 | 5.3755 |

حالت سوم) مقدار w بزرگ ، مقدار c1 بزرگ ، مقدار c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.3305 |
| 2 | 5.9333 |
| 3 | 5.7848 |
| 4 | 5.6969 |
| 5 | 6.0794 |

حالت چهارم) w بزرگ ، c1 متوسط ، c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.3057 |
| 2 | 5.3431 |
| 3 | 5.4034 |
| 4 | 5.2859 |
| 5 | 5.1608 |

حالت پنجم) w بزرگ c1 متوسط، c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 8.2517 |
| 2 | 6.0215 |
| 3 | 6.9394 |
| 4 | 6.3293 |
| 5 | 9.238 |

حالت ششم)w بزرگ ،c1 متوسط ، c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.6454 |
| 2 | 7.1096 |
| 3 | 6.3064 |
| 4 | 6.9698 |
| 5 | 6.6589 |

حالت هفتم) w بزرگ، c1 کوچک، c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.24 |
| 2 | 5.14 |
| 3 | 5.1938 |
| 4 | 4.6554 |
| 5 | 5.408 |

حالت هشتم) w بزرگ، c1 کوچک ، c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.24 |
| 2 | 5.14 |
| 3 | 5.1938 |
| 4 | 7.7344 |
| 5 | 8.1923 |

حالت نهم)w1 بزرگ،c1کوچک ، c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 8.1073 |
| 2 | 5.8146 |
| 3 | 6.0256 |
| 4 | 7.2956 |
| 5 | 6.7781 |

حالت ۱۰ ) مقدار w متوسط ، مقدار c1 بزرگ ، مقدار c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.3144 |
| 2 | 5.415 |
| 3 | 4.991 |
| 4 | 5.265 |
| 5 | 4.6878 |

حالت ۱۱) مقدار w متوسط ، مقدار c1 بزرگ ، مقدار c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.5547 |
| 2 | 5.7694 |
| 3 | 5.0658 |
| 4 | 5.2043 |
| 5 | 5.2625 |

حالت ۱۲) مقدار w متوسط ، مقدار c1 بزرگ ، مقدار c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.510 |
| 2 | 6.0735 |
| 3 | 5.0114 |
| 4 | 5.4582 |
| 5 | 6.1535 |

حالت ۱۳) w متوسط ، c1 متوسط ، c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.9709 |
| 2 | 5.1726 |
| 3 | 4.9252 |
| 4 | 5.0986 |
| 5 | 4.9891 |

حالت ۱۴) w متوسط c1 متوسط، c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.9737 |
| 2 | 4.9382 |
| 3 | 4.9641 |
| 4 | 5.3568 |
| 5 | 5.1504 |

حالت ۱۵)w متوسط ،c1 متوسط ، c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.4886 |
| 2 | 5.8343 |
| 3 | 6.0525 |
| 4 | 5.4259 |
| 5 | 5.3279 |

حالت ۱۶) w متوسط ، c1 کوچک، c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.9826 |
| 2 | 5.3604 |
| 3 | 4.9291 |
| 4 | 4.5867 |
| 5 | 4.9159 |

حالت ۱۷) w متوسط ، c1 کوچک ، c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.8595 |
| 2 | 4.924 |
| 3 | 4.7428 |

| | |
|---|---|
| 4 | 5.268 |
| 5 | 4.9541 |

حالت ۱۸)w1 متوسط ،c1کوچک ، c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.2067 |
| 2 | 7.6772 |
| 3 | 9.808 |
| 4 | 14.3041 |
| 5 | 8.1076 |

حالت ۱۹ ) مقدار w کوچک ، مقدار c1 بزرگ ، مقدار c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.0487 |
| 2 | 5.221 |
| 3 | 5.3043 |
| 4 | 5.0971 |
| 5 | 5.1914 |

حالت ۲۰) مقدار w کوچک ، مقدار c1 بزرگ ، مقدار c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.3899 |
| 2 | 5.5131 |
| 3 | 4.9746 |
| 4 | 5.1248 |
| 5 | 5.3371 |

حالت ۲۱) مقدار w کوچک ، مقدار c1 بزرگ ، مقدار c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.7902 |
| 2 | 6.9201 |
| 3 | 5.4254 |
| 4 | 8.5047 |
| 5 | ۶,۵۵۱۲ |

حالت ۲۲) w کوچک ، c1 متوسط ، c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.4655 |
| 2 | 4.7749 |
| 3 | 8.424 |
| 4 | 9.6721 |

| | |
|---|---|
| 5 | 4.8276 |

حالت ۲۳) w کوچک c1 متوسط، c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.4756 |
| 2 | 5.2193 |
| 3 | 4.7328 |
| 4 | 5.0124 |
| 5 | 4.9201 |

حالت ۲۴)w کوچک ،c1 متوسط ، c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.1276 |
| 2 | 5.5625 |
| 3 | 5.08 |
| 4 | 4.5584 |
| 5 | 4.8619 |

حالت ۲۵) w کوچک ، c1 کوچک، c2 بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.0416 |
| 2 | 4.9732 |
| 3 | 4.9856 |
| 4 | 4.9936 |
| 5 | 4.9344 |

حالت ۲۶) w کوچک ، c1 کوچک ، c2 متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.8479 |
| 2 | 4.8884 |
| 3 | 4.8456 |
| 4 | 5.1736 |
| 5 | 4.8515 |

حالت ۲۷)w1 کوچک ،c1کوچک ، c2 کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.7812 |
| 2 | 4.4245 |
| 3 | 4.8728 |
| 4 | 4.8479 |
| 5 | 4.8495 |

## نتایج آزمایش‌های تنظیم پارامتر دستی الگوریتم FA

حالت ۱)گاما بزرگ ، بتا بزرگ ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.7271 |
| 2 | 7.0658 |
| 3 | 6.83906 |
| 4 | 7.71031 |
| 5 | 7.7009 |

حالت ۲) گاما بزرگ ، بتا بزرگ ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.38095 |
| 2 | 6.28765 |
| 3 | 6.23588 |
| 4 | 6.36661 |
| 5 | 6.46356 |

حالت سوم) گاما بزرگ ، بتا بزرگ ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.03330 |
| 2 | 4.83220 |
| 3 | 4.89461 |
| 4 | 5.03820 |
| 5 | 5.06229 |

حالت چهارم) گاما بزرگ ، بتا متوسط ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.6073 |
| 2 | 6.5751 |
| 3 | 6.7225 |
| 4 | 6.87291 |
| 5 | 7.8726 |

حالت پنجم) گاما بزرگ ، بتا متوسط ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.0000 |
| 2 | 6.2770 |
| 3 | 6.1660 |

| | |
|---|---|
| 4 | 6.3293 |
| 5 | 6.25450 |

حالت ششم) گاما بزرگ ، بتا متوسط ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.0379 |
| 2 | 4.6680 |
| 3 | 4.66105 |
| 4 | 7.6577 |
| 5 | 4.76872 |

حالت هفتم) گاما بزرگ ، بتا کوچک ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 8.1912 |
| 2 | 7.69736 |
| 3 | 7.87432 |
| 4 | 6.6816 |
| 5 | 6.5190 |

حالت هشتم) گاما بزرگ ، بتا کوچک ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.3101 |
| 2 | 6.4694 |
| 3 | 6.2848 |
| 4 | 5.0743 |
| 5 | 6.3549 |

حالت نهم) گاما بزرگ ، بتا کوچک ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.4101 |
| 2 | 5.0992 |
| 3 | 4.7658 |
| 4 | 5.1132 |
| 5 | 4.3496 |

حالت ۱۰ ) گاما متوسط ، بتا بزرگ ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.7080 |
| 2 | 6.3858 |
| 3 | 6.4772 |
| 4 | 7.8078 |
| 5 | 7.5182 |

حالت ۱۱) گاما متوسط ، بتا بزرگ ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.3452 |
| 2 | 6.2122 |
| 3 | 6.2875 |
| 4 | 6.2146 |
| 5 | 6.3668 |

حالت ۱۲) گاما متوسط ، بتا بزرگ ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.6788 |
| 2 | 4.6790 |
| 3 | 4.7772 |
| 4 | 5.0211 |
| 5 | 4.3735 |

حالت ۱۳) گاما متوسط ، بتا متوسط ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.6975 |
| 2 | 7.3104 |
| 3 | 7.6943 |
| 4 | 7.0862 |
| 5 | 8.3192 |

حالت ۱۴) گاما متوسط ، بتا کوچک ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.3579 |
| 2 | 6.1831 |
| 3 | 6.3228 |
| 4 | 6.3299 |
| 5 | 6.2766 |

حالت ۱۵) گاما متوسط ، بتا متوسط ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.7648 |
| 2 | 4.7736 |
| 3 | 4.7920 |
| 4 | 4.7600 |
| 5 | 4.3455 |

حالت ۱۶) گاما متوسط ، بتا کوچک ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.8418 |
| 2 | 7.7500 |
| 3 | 8.5979 |

| | |
|---|---|
| 4 | 7.3657 |
| 5 | 7.8756 |

حالت ۱۷) گاما متوسط ، بتا کوچک ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.2932 |
| 2 | 6.3342 |
| 3 | 6.2848 |
| 4 | 6.4804 |
| 5 | 6.1110 |

حالت ۱۸) گاما متوسط ، بتا کوچک ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.0601 |
| 2 | 5.0772 |
| 3 | 4.7881 |
| 4 | 5.1169 |
| 5 | 4.7741 |

حالت ۱۹ ) گاما کوچک ، بتا بزرگ ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.9958 |
| 2 | 9.3055 |
| 3 | 7.6971 |
| 4 | 5.0971 |
| 5 | 7.7154 |

حالت ۲۰) گاما کوچک ، بتا بزرگ ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.3894 |
| 2 | 6.3484 |
| 3 | 6.3370 |
| 4 | 6.4206 |
| 5 | 6.4115 |

حالت ۲۱) گاما کوچک ، بتا بزرگ ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.6810 |
| 2 | 5.0832 |
| 3 | 4.7042 |
| 4 | 5.0411 |
| 5 | 4.6869 |

حالت ۲۲) گاما کوچک ، بتا متوسط ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|

| | |
|---|---|
| 1 | 7.8555 |
| 2 | 6.9375 |
| 3 | 8.8010 |
| 4 | 7.1502 |
| 5 | 7.7011 |

حالت ۲۳) گاما کوچک ، بتا متوسط ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.5051 |
| 2 | 6.3915 |
| 3 | 6.1819 |
| 4 | 6.0719 |
| 5 | 6.4951 |

حالت ۲۴) گاما کوچک ، بتا متوسط ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.6560 |
| 2 | 4.7057 |
| 3 | 4.6701 |
| 4 | 4.7161 |
| 5 | 4.3649 |

حالت ۲۵) گاما کوچک ، بتا کوچک ، آلفا بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.7179 |
| 2 | 7.1907 |
| 3 | 7.3746 |
| 4 | 7.8366 |
| 5 | 7.3201 |

حالت ۲۶) گاما کوچک ، بتا کوچک ، آلفا متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.3522 |
| 2 | 6.3013 |
| 3 | 6.2778 |
| 4 | 6.3932 |
| 5 | 6.3535 |

حالت ۲۷) گاما کوچک ، بتا کوچک ، آلفا کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.3471 |
| 2 | 4.3322 |
| 3 | 4.3484 |
| 4 | 4.8322 |
| 5 | 5.0337 |

## نتایج آزمایش‌های تنظیم پارامتر دستی الگوریتم ABC

حالت ۱) limit بزرگ و $\varphi$ بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.5446 |
| 2 | 8.9465 |
| 3 | 12504.1953 |
| 4 | 7.4845 |
| 5 | 19703.8831 |

حالت ۲) limit بزرگ و $\varphi$ متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 8.708 |
| 2 | 8.0346 |
| 3 | 60625.2982 |
| 4 | 7.7278 |
| 5 | 7.8928 |

حالت سوم) limit بزرگ و $\varphi$ کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 4.3806 |
| 2 | 4.3935 |
| 3 | 4.5217 |
| 4 | 4.4465 |
| 5 | 4.4115 |

حالت چهارم) limit متوسط و $\varphi$ بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 7.8423 |
| 2 | 5.7936 |
| 3 | 18734.157 |
| 4 | 6.2915 |
| 5 | 6.9227 |

حالت پنجم) limit متوسط و $\varphi$ متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.7358 |
| 2 | 7.877 |
| 3 | 8.7902 |
| 4 | 57279.1912 |
| 5 | 8.259 |

حالت ششم)limit متوسط و $\varphi$ کوچک

| شماره آزمایش | هزینه مسیر |
|---|---|

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.3248 |
| 2 | 5.6473 |
| 3 | 5.4559 |
| 4 | 5.3371 |
| 5 | 5.9241 |

| 1 | 5.6826 |
|---|---|
| 2 | 5.6815 |
| 3 | 5.0799 |
| 4 | 5.5511 |
| 5 | 6.3277 |

حالت هفتم) limit کوچک و $\varphi$ بزرگ

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 6.2118 |
| 2 | 6.1916 |
| 3 | 6.5504 |
| 4 | 5.9022 |
| 5 | 6.3048 |

حالت هشتم)limit کوچک و $\varphi$ متوسط

| شماره آزمایش | هزینه مسیر |
|---|---|
| 1 | 5.7868 |
| 2 | 5.959 |
| 3 | 5.4827 |
| 4 | 6.3152 |
| 5 | 5.9371 |

حالت نهم)limit کوچک و $\varphi$ کوچک

# فهرست مراجع و منابع:


۱. Gasparetto, A., et al., *Path planning and trajectory planning algorithms: A general overview.* Motion and operation planning of robotic systems, 2015: p. 3-27.

۲. Radmanesh, M., et al., *Overview of path-planning and obstacle avoidance algorithms for UAVs: a comparative study.* Unmanned systems, 2018. **6**(02): p. 95-118.

۳. Zhao, Y., Z. Zheng, and Y. Liu, *Survey on computational-intelligence-based UAV path planning.* Knowledge-Based Systems, 2018. **158**: p. 54-64.

۴. Mac, T.T., et al., *Heuristic approaches in robot path planning: A survey.* Robotics and Autonomous Systems, 2016. **86**: p. 13-28.

۵. Asano, T., et al. *Visibility-polygon search and Euclidean shortest paths*. in *26th annual symposium on foundations of computer science (SFCS 1985)*. 1985. IEEE.

۶. Canny, J. *A Voronoi method for the piano-movers problem*. in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. 1985. IEEE.

۷. Siméon, T., J.-P. Laumond, and C. Nissoux, *Visibility-based probabilistic roadmaps for motion planning.* Advanced Robotics, 2000. **14**(6): p. 477-493.

۸. Karaman, S. and E. Frazzoli, *Sampling-based algorithms for optimal motion planning.* The international journal of robotics research, 2011. **30**(7): p. 846-894.

۹. Yang, X.-S. *Firefly algorithms for multimodal optimization*. in *International symposium on stochastic algorithms*. 2009. Springer.

۱۰. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Proceedings of ICNN'95-international conference on neural networks*. 1995. IEEE.

۱۱. Karaboga, D., *An idea based on honey bee swarm for numerical optimization*. 2005, Citeseer.

۱۲. Shi, Y. and R. Eberhart. *A modified particle swarm optimizer*. in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. 1998. IEEE.

۱۳. Yang, X.-S., *Nature-inspired metaheuristic algorithms*. 2010: Luniver press.

۱۴. Rathore, K., *Parameter tuning in firefly algorithm.* 2018.

١٥. Akay, B. and D. Karaboga. *Parameter tuning for the artificial bee colony algorithm*. in *International conference on computational collective intelligence*. 2009. Springer.
١٦. Whitley, D., *A genetic algorithm tutorial.* Statistics and computing, 1994. **4**(2): p. 65-85.