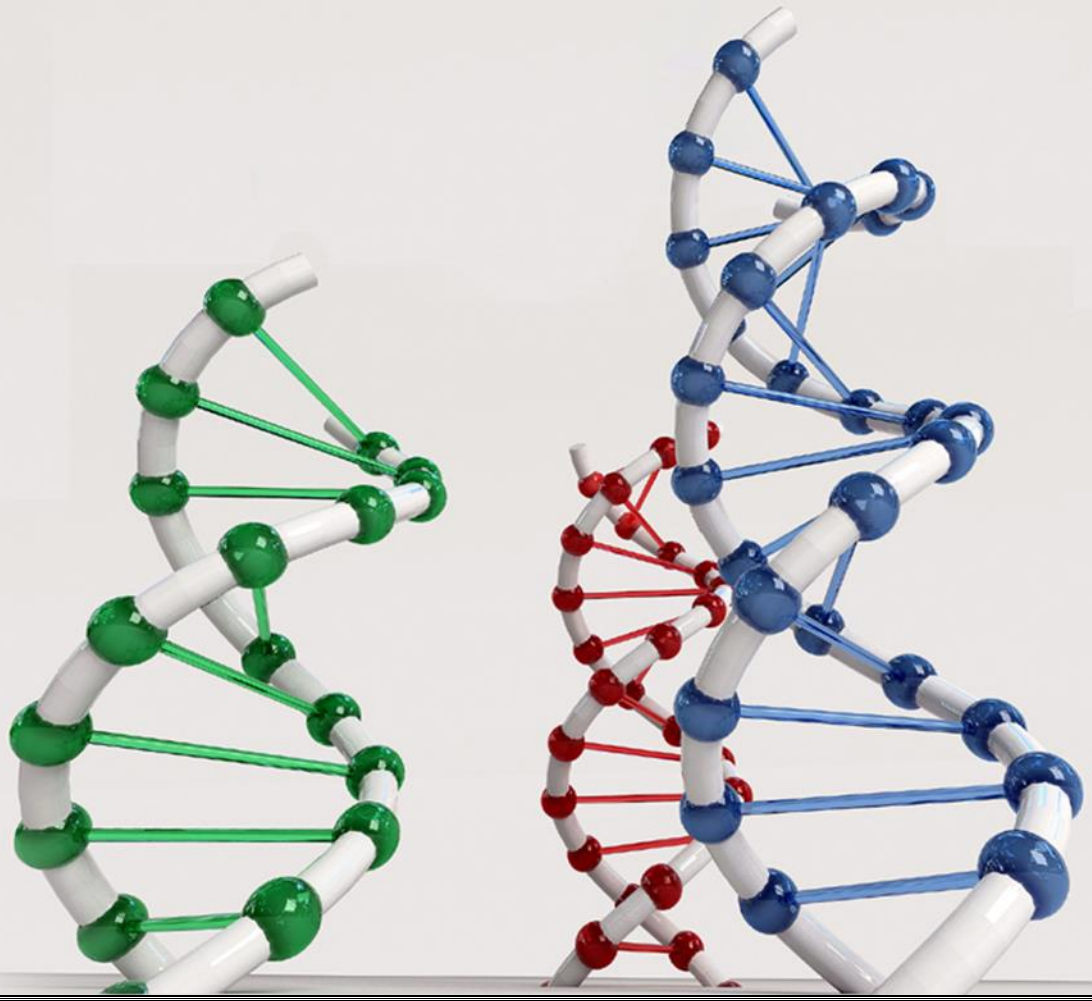
دانشگاه یزد

بررسی حل مسئله فروشنده دوره‌گرد

با استفاده از الگوریتم ژنتیک

مهدی احمدی



چکیده

مسئله فروشنده دوره‌گرد (TSP) یکی از مسائل مشهور در حوزه بهینه‌سازی ترکیبیاتی است که یافتن کوتاه‌ترین مسیر برای بازدید از مجموعه‌ای از شـهرها و بازگشـت به مبدأ را هدف قرار می‌دهد. با توجه به پیچیدگی محاسـباتی بالای این مسـئله، اسـتفاده از روش‌های فرااکتشافی به‌ویژه الگوریتم ژنتیک به عنوان راه‌حلی کارا و قابل توسعه مورد توجه قرار گرفته است.

در این پروژه، الگوریتم ژنتیک برای حل مسئله TSP پیاده‌سازی شده و عملکرد آن تحت شرایط مختلف مورد ارزیابی قرار گرفته اسـت. به‌منظور بررسـی تاثیر تنظیمات الگوریتم، انواع روش‌های انتخاب، ترکیب و جهش مورد آزمایش قرار گرفته و اثر هر یک بر کیفیت مسیر به‌دست‌آمده و مدت زمان اجرای الگوریتم تحلیل شده است.

نتایج حاصـل از آزمایش‌ها نشـان می‌دهند که انتخاب صـحیح ترکیب و جهش می‌تواند به‌طور قابل‌توجهی کیفیت مسـیر نهایی را بهبود بخشـیده و زمان دسـتیابی به جواب بهینه را کاهش دهد. این پژوهش نشـان می‌دهد که الگوریتم ژنتیک، با وجود سـادگی مفهومی، قابلیت بالایی در ارائه پاسخ‌های نزدیک به بهینه برای مسائل پیچیده مانند TSP دارد.

مقدمه

مسئله فروشـنده دوره‌گرد (Travelling Salesman Problem) یا به اختصـار (TSP) یکی از مسـائل کلاسـیک و بنیادی در حوزه بهینه‌سازی ترکیبیاتی و نظریه گراف است که جایگاه ویژه‌ای در علوم کامپیوتر، ریاضیات کاربردی و تحقیق در عملیات دارد. در این مسـئله، یک فروشـنده باید با شـروع از یک شـهر، از مجموعه‌ای از شـهرهای مشـخص بازدید کرده و در نهایت به شـهر اولیه بازگردد، به‌گونه‌ای که مجموع فاصله طی‌شده یا هزینه سفر کمینه شود. هدف اصلی در این مسئله یافتن کوتاه‌ترین مسیر ممکن برای بازدید از تمام شهرها بدون تکرار بازدید است.

کاربردهای واقعی TSP فراتر از عنوان آن هستند. این مسئله در بسیاری از حوزه‌های صنعتی و عملی از جمله برنامه‌ریزی مسیر در سـیسـتم‌های حمل‌ونقل، طراحی مدارهای الکترونیکی، لجسـتیک و زنجیره تأمین، رباتیک، و حتی بیوانفورماتیک (مانند چینش ژن‌ها) کاربرد دارد. به دلیل گسـتردگی کاربرد و اهمیت بهینه‌سـازی در این حوزه‌ها، یافتن راه‌حل‌های مؤثر برای TSP از اهمیت بالایی برخوردار است.

TSPیک مسـئله NP-Hard اسـت، به این معنا که با افزایش تعداد شـهرها، فضـای جسـتجو به‌صـورت نمایی افزایش می‌یابد و اسـتفاده از روش‌های کامل (مانند برنامه‌ریزی پویا یا جسـتجوی فراگیر) برای حل دقیق آن در مقیاس‌های بزرگ، به‌طور عملی غیرممکن می‌شـود. از این‌رو، اسـتفاده از روش‌های تقریبی و فرااکتشـافی (Metaheuristics) مانند الگوریتم‌های ژنتیک، تبرید شبیه‌سازی‌شده(Simulated Annealing) ، الگوریتم مورچگان و الگوریتم‌های تکاملی دیگر رایج شده‌اند.

الگوریتم ژنتیک Genetic Algorithm یا (GA)یکی از قدرتمندترین و پرکاربردترین روش‌های فرااکتشـافی اسـت که از اصـول تکامل طبیعی و انتخاب طبیعی داروینی الهام گرفته شـده اسـت. این الگوریتم با حفظ و ترکیب ویژگی‌های خوب نسـل‌های قبلی،

به‌تدریج به سمت جواب‌های بهینه حرکت می‌کند. دلیل اصلی استفاده از الگوریتم ژنتیک در حل مسئلهTSP ، توانایی آن در جستجوی فضاهای بزرگ و پیچیده و نیز اجتناب از گیر افتادن در کمینه‌های محلی است. با تنظیم صحیح پارامترها و طراحی مناسب توابع ترکیب و جهش، الگوریتم ژنتیک می‌تواند مسیرهایی با کیفیت بالا و در زمان مناسب ارائه دهد.

در این پروژه، با هدف بررسی تأثیر پارامترها و روش‌های مختلف انتخاب، ترکیب و جهش، الگوریتم ژنتیک برای حل مسئله فروشنده دوره‌گرد پیاده‌سازی شده و عملکرد آن در شرایط مختلف مورد بررسی قرار گرفته است.

توصیف مسئله

مسئله فروشنده دوره‌گرد (Travelling Salesman Problem – TSP) یکی از مسائل کلاسیک در حوزه بهینه‌سازی ترکیبیاتی است. در این مسئله، هدف یافتن کوتاه‌ترین مسیر ممکن برای یک فروشنده است که باید از مجموعه‌ای از شهرها دقیقاً یک بار بازدید کند و در پایان به شهر مبدأ بازگردد. این مسیر باید به گونه‌ای انتخاب شود که مجموع فاصله یا هزینه طی‌شده بین شهرها حداقل باشد.

پیچیدگی این مسئله از آن جهت است که با افزایش تعداد شهرها، تعداد حالات ممکن به‌شکل نمایی افزایش می‌یابد ($n!$برای n شهر) .این امر باعث می‌شود استفاده از روش‌های دقیق (Exact) مانند جستجوی کامل یا برنامه‌ریزی پویا، در مقیاس‌های بزرگ، از نظر زمانی بسیار پرهزینه یا حتی غیرممکن باشد. در نتیجه، استفاده از الگوریتم‌های تقریبی و فرااکتشافی مانند الگوریتم ژنتیک برای یافتن پاسخ‌های نزدیک به بهینه، یک راه‌حل مؤثر و عملی محسوب می‌شود.

روش پژوهش و پیاده‌سازی الگوریتم ژنتیک

در این پروژه، مسئله TSP با استفاده از الگوریتم ژنتیک مدل‌سازی و حل شده است. در این الگوریتم، هر مسیر ممکن برای فروشنده به عنوان یک کروموزوم در نظر گرفته شده و هر نود (شهر) در مسیر، به منزله یک نوکلئوتید در ساختار کروموزومی تعریف شده است. الگوریتم با تولید جمعیت اولیه از مسیرهای ممکن آغاز می‌شود و با تکرار عملیات‌های انتخاب، ترکیب و جهش، سعی می‌کند بهترین مسیر را با کمترین طول پیدا کند.

تولید جمعیت اولیه هر نسل

دو روش مختلف برای انتخاب جمعیت اولیه هر نسل ارائه شده است:

۱. **روش چرخ رولت:(Roulette Wheel Selection)**
در این روش، احتمال انتخاب هر کروموزوم برای ورود به جمعیت اولیه، متناسب با مقدار تابع برازش (Fitness) آن است. کروموزوم‌هایی با برازش بالاتر، شانس بیشتری برای انتخاب دارند. این روش از احتمال تصادفی برای حفظ تنوع جمعیت بهره می‌برد.
۲. **روش‌حریصانه:(Greedy Initialization)**
در این روش، جمعیت اولیه با استفاده از یک الگوریتم حریصانه تولید می‌شود؛ به این صورت که در هر گام نزدیک‌ترین شهر انتخاب می‌گردد. این روش باعث می‌شود جمعیت اولیه کیفیت نسبتاً بالاتری داشته باشد و الگوریتم سریع‌تر به جواب‌های خوب نزدیک شود، اما ممکن است تنوع ژنتیکی کاهش یابد.

عملیات ترکیب (Crossover)

برای ایجاد فرزندان جدید از والدین، سه روش مختلف برای ترکیب کروموزوم‌ها استفاده شده است:

۱. **Monoposition Crossover:**
یک نقطه تصادفی در کروموزوم انتخاب شده و قسمت‌های قبل و بعد آن بین دو والد جابجا می‌شود.
۲. **Doubleposition Crossover:**
دو نقطه برش تصادفی انتخاب می‌شود و قطعه میانی بین دو والد تعویض می‌گردد.
۳. **Simple Cross:**
بخشی از کروموزوم اول با بخشی از کروموزوم دوم به‌طور ساده جایگزین می‌شود، با رعایت ترتیب و بدون تکرار.

عملیات جهش(Mutation)

برای حفظ تنوع ژنتیکی و جلوگیری از گیر افتادن در کمینه‌های محلی، عملیات جهش اعمال می‌شود. در پیاده‌سازی انجام‌شده، دو روش متفاوت برای جهش در نظر گرفته شده‌اند:

۱. **Monopoint Mutation:**
تنها یک نقطه در کروموزوم به‌صورت تصادفی تغییر می‌کند (مثلاً دو شهر جابجا می‌شوند).
۲. **Doublepoint Mutation:**
دو نقطه به‌طور تصادفی انتخاب شده و عناصر مربوطه با یکدیگر جابجا می‌شوند یا ترتیب میان آن دو برعکس می‌شود.

شرط توقف الگوریتم

الگوریتم ژنتیک تا زمانی به تولید نسل‌های جدید ادامه می‌دهد که یکی از شرایط زیر برقرار شود:

- دستیابی به مقدار قابل قبول برای تابع خطا یا طول مسیر
- رسیدن به حد مشخصی از تعداد نسل‌ها

تنظیمات آزمایش و سناریوها

به‌منظور بررسی عملکرد الگوریتم ژنتیک در حل مسئله فروشنده دوره‌گرد(TSP) ، آزمایش‌هایی با پیکربندی‌های مختلف از پارامترها و روش‌های اصلی الگوریتم صورت گرفته است. هدف این آزمایش‌ها ارزیابی تأثیر روش‌های مختلف انتخاب، ترکیب (Crossover)و جهش (Mutation) بر کیفیت جواب نهایی و مدت زمان اجرای الگوریتم می‌باشد.

در تمام سناریوها، برای حفظ شرایط کنترل‌شده، ابعاد مسئله (تعداد شهرها)، فاصله‌ها و برخی از تنظیمات پایه ثابت نگه داشته شده‌اند تا بتوان تأثیر مستقیم تغییرات الگوریتمی را بررسی نمود.

سناریو ۱: بررسی روش‌های انتخاب جمعیت اولیه

- هدف: بررسی تأثیر انتخاب تصادفی براساس شانس (Roulette Wheel) در مقابل انتخاب حریصانه.(Greedy)
- مقایسه:
  - Greedy Initialization
  - Roulette Wheel Initialization

سناریو ۲: بررسی انواع روش‌های ترکیب(Crossover)

- هدف: بررسی تأثیر نوع Crossover بر سرعت همگرایی و کیفیت جواب‌ها.
- مقایسه بین:
  - Monoposition Crossover
  - Doubleposition Crossover
  - Simple Cross

سناریو ۳: بررسی انواع جهش(Mutation)

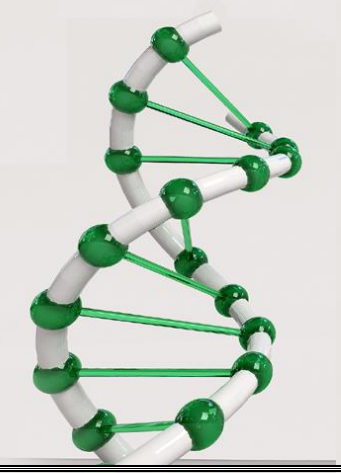
- هدف: ارزیابی اثربخشی روش‌های جهش در حفظ تنوع ژنتیکی و جلوگیری از مینیمم‌های محلی.
- مقایسه بین:
  - **Monopoint Mutation**
  - **Doublepoint Mutation**

سناریو ۴: تاثیر ترکیب تنظیمات(Best Configuration)

- هدف: ترکیب بهترین تنظیمات به‌دست‌آمده از سناریوهای پیشین برای دستیابی به بهترین عملکرد نهایی.
- تنظیمات منتخب:
  - **انتخاب Greedy :**
  - **ترکیب Doubleposition :**
  - **جهش Doublepoint :**

در پایان هر سناریو، معیارهای زیر اندازه‌گیری و ثبت شده‌اند:

- **بهینگی بهترین طول مسیر به‌دست‌آمده**
- **زمان اجرای الگوریتم**
- **تعداد نسل مورد نیاز برای رسیدن به جواب قابل قبول**

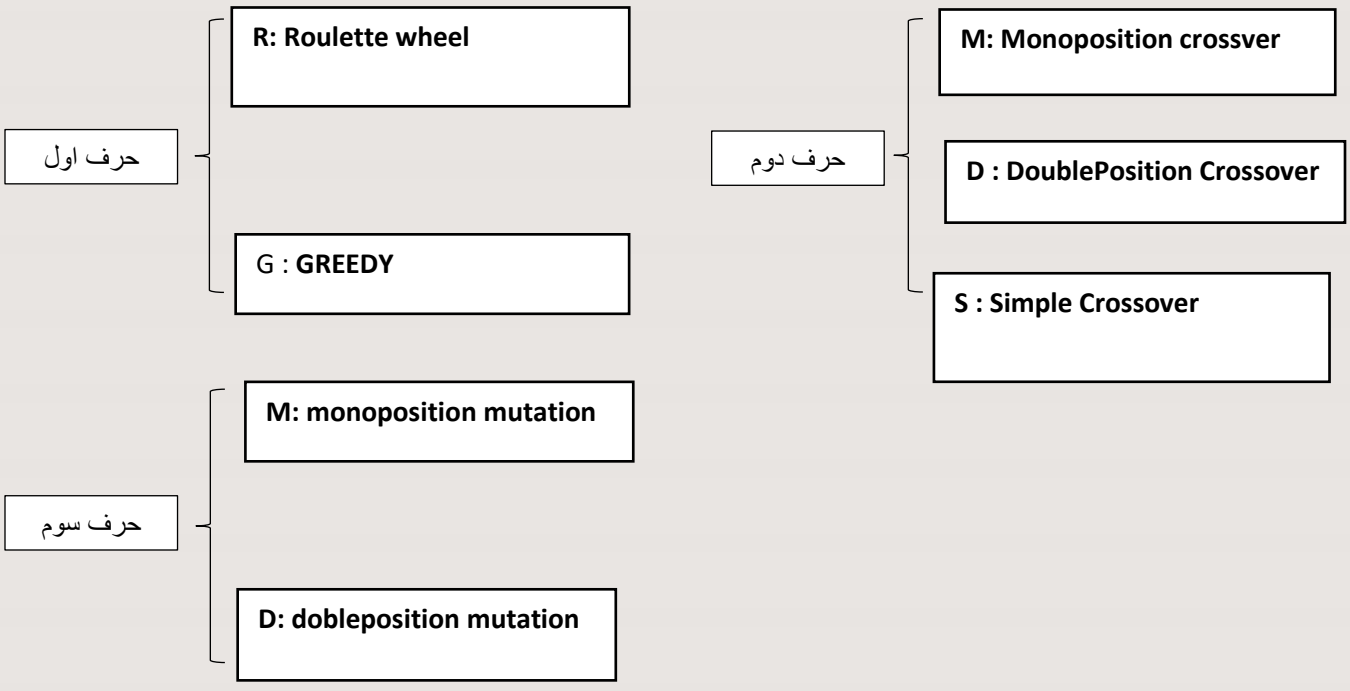


نتایج تجربی

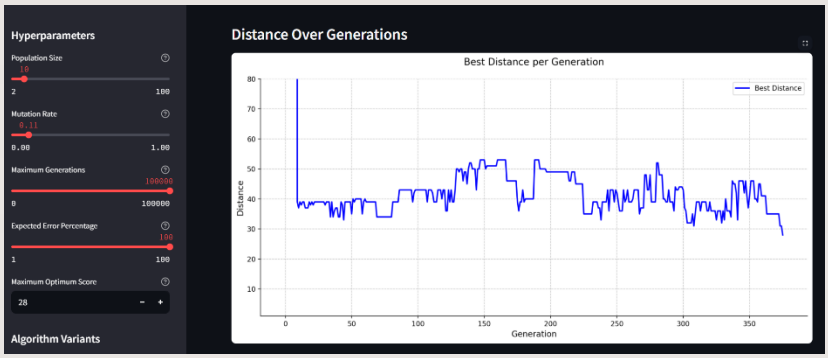
نتیجه برای چند گراف با اندازه ۱۰ نود در طی ۵ مرتبه اجرا صورت گرفته.

روش استفاده شده	زمان	نسل تولید شده	بهینگی جواب
RMM	0.19	1053	YES
RMD	0.41	5843	YES
GMM	0.63	8641	YES
GMD	0.45	6888	YES
GDM	0.28	4021	YES
GDD	0.55	7813	YES
RDM	0.27	4588	YES
RDD	0.64	8902	YES
GSM	0.23	1935	YES
GSD	013	1054	YES
RSM	0.19	2549	YES
RSD	0.18	1015	YES
میانگین	34.5	4525	YES

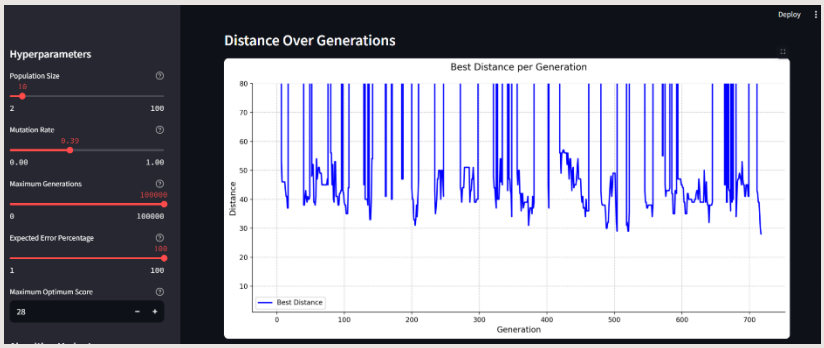
در هر مرحله به ترتیب ۳ حرف مشخص کننده سناریوی انتخابی میباشند.



همچنین مشاهده میشود که با افزایش نرخ جهش از 10درصد تا بازه 40 درصد میتواند موجب بهبود عملکرد الگوریتم در سریع تر یافتن جواب ممکن بشود ولی از جهتی پایداری نسبی نسل ها را به هم میریزد و نسل های تولید شده شباهت کمتری با والدین خود دارند و در قسمت‌های مختلف الگوریتم همیشه به سوی بهبود نتایج پیشروی نکرده است.



Mutation_rate = 0.11



Mutation_rate = 0.39

یافته‌های پژوهش و تفسیر نتایج

۱- با اســـتناد بر نتایج آزمایش نرخ جهش در بازه ۰.۱ تا ۰.۲ پلیدارترین روش و بازه ۰.۲ تا ۰.۳ بهترین روش و بازه ۰.۳ تا ۰.۴ سریع ترین جواب برای گراف های نمونه در مسئله بودند.

۲- تعداد جمعیت موجود در هر نسل ارتباط مستقیمی با پیدا کردن جواب در تعداد نسل کمتر داشت ولی هزینه زمانی هر نسل را افزایش میداد.

۳- برای گراف‌هایی با اندازه کم مانند نمونه‌های تست شده در مسئله سرعت الگوریتم پسگرد تا حد قابل قبولی بهتر از الگوریتم ژنتیک میباشد.

٤- در میان روش های تست شده روش GSD بهترین عملکرد را از خود نشان داد.

۵- در اغلب موارد در صورت محدود نبودن تعداد نسل‌های قابل تولید و غیرمنتظره نبودن نرخ جهش و سایز مناسب جمعیت ها الگوریتم تقریبا در بازه زمانی مناسب شبه کامل میباشد.

جمع بندی

در این پروژه، مسئله فروشنده دوره‌گرد (TSP) به عنوان یکی از مسائل مهم و پیچیده بهینه‌سازی ترکیبیاتی مورد بررسی قرار گرفت و الگوریتم ژنتیک به عنوان روشی فرااکتشافی جهت حل تقریبی آن پیاده‌سازی شد. ساختار الگوریتم بر پایه نمایش مسیرها به‌صورت کروموزوم، تعریف تابع برازش بر اساس طول مسیر، و به‌کارگیری عملیات‌های انتخاب، ترکیب و جهش شکل گرفت.

در بخش آزمایش‌ها، تأثیر روش‌های مختلف انتخاب شد و نتایج هر کدام به تفصیل بررسی شد. همچنین ترکیب این تنظیمات به‌صورت یک پیکربندی منتخب (best configuration) منجر به دستیابی به نتایجی قابل قبول با سرعت همگرایی بالا و زمان اجرای مناسب گردید.

در مجموع، نتایج این پروژه تأیید می‌کنند که الگوریتم ژنتیک، در صورت انتخاب و تنظیم دقیق پارامترها و عملگرها، می‌تولند به عنوان روشی کارآمد برای حل مسائل پیچیده‌ای همچون TSP مورد استفاده قرار گیرد. این الگوریتم با قابلیت توسعه، انطباق‌پذیری و سادگی مفهومی، ابزاری ارزشمند در حل مسائل بهینه‌سازی دنیای واقعی به‌شمار می‌رود.

منابع و ضمایم

کد پیاده‌سازی شده الگوریتم

داشبورد شبیه سازی شده الگوریتم

مستند readme الگوریتم

کد تولید نمونه داده ورودی مسئله

تصاویر خروجی های مسئله بر روی داشبورد