

دانشکده مهندسی

گروه مهندسی کامپیوتر

گرایش هوش مصنوعی

الگویتم ژنتیک ابتدایی

استاد:

دکتر مجتبی روحانی

دانشجو:

علی گلی

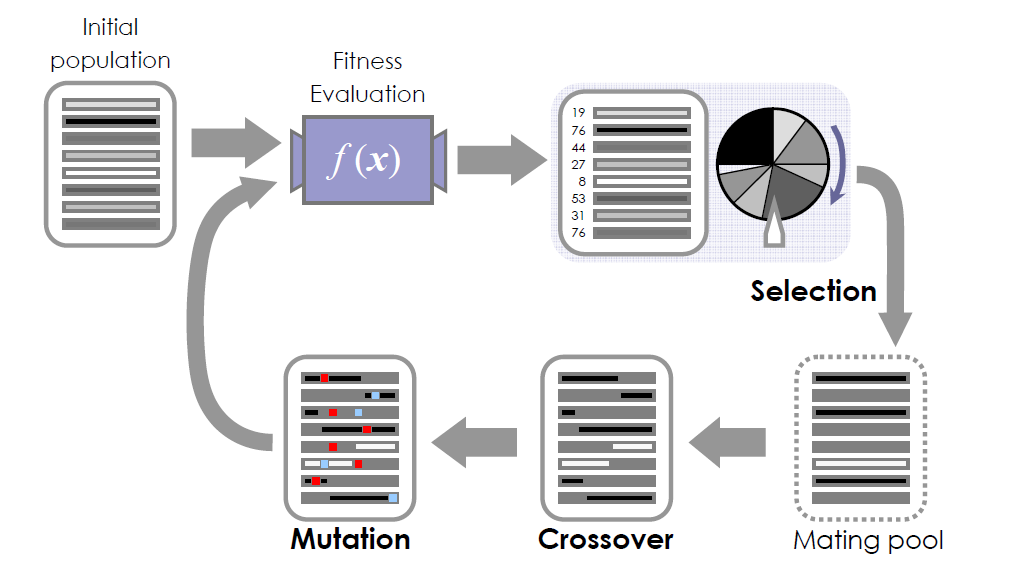
آبان ماه

سال 1402

1. **الگویتم**

**1.1 تعریف الگوریتم**

الگوریتم CGA یک الگوریتم ژنتیک ابتدایی می‌باشد که در آن یک جمعیت ابتدایی ساخته می‌شود سپس عملیات fitness روی این جمعیت صورت می‌گیرد که در آن به هر فرد یک امتیازی برحسب فنوتیپ آن ها اختصاص داده می‌شود. سپس در selection به هر فرد بابت امتیازی که دارد یک شانسی برای انتخاب شدن داده می‌شود. یعنی آنان که فنوتیپ بهتری دارند به احتمال بیشتری انتخاب می شوند. از جمعیت انتخاب شده عملیات crossover انجام می‌شود. سپس عملیات mutation انجام می‌شود. این روند یک روند تکراری است.(شکل 1)



شکل 1

**2 برنامه CGA**

**1.2 صورت کلی برنامه**

برنامه شامل 3 کلاس پایه ای می‌باشد. کلاس اول CGA می‌باشد که برای استفاده از آن باید ارث بری صورت بگیرد، و طرح کلی الگوریتم ژنتیک را در بر می‌‌گیرد. کلاس دوم functionEvaluations می باشد، که می‌توان در آن توابعی را تعریف کرد و از آنها استفاده کرد. کلاس سوم main می‌باشد، که کلاس CGA ارث بری شده است و متغیر های مورد نیاز در آن تعریف شده اند. همچنین توابع بیشتری جهت اجرای بیش از 1 بار استفاده از برنامه نیز به آن اضافه شده است که کمک می‌‌کند چندین بار الگوریتم CGA را اجرا کنید و بین نتایج مختلف میانگین گیری کنید.

**2.2 کلاس CGA**

توابع مورد نیاز این کلاس؛ به ترتیب از بالا به پایین نوشته می‌شود. این کلاس شامل تابع init هست که ابتدا تابع fitness را تنظیم می‌کند. سپس تابع run الگوریتم کلی را اجرا می‌کند. در تابع run ابتدا تابع initialization را اجرا می‌کند که ماتریس جمعیت، ماتریس والدین، تنظیم متغیر های جمع آوری کننده بدترین، بهترین و متوسط فنوتیپ هر نسل مقدار دهی اولیه می‌‌کند. ماتریس جمعیت از بدترین حالت شروع به کار می‌کند. سپس در تابع run در یک حلقه نسل افزایش پیدا می‌‌کند و در هر نسل به ترتیب از چپ به راست توابع selection, crossover, mutation اجرا می‌گردد. حال داده‌ها را جمع آوری می‌کنیم. داده های جمع آوری شده شامل: بهترین فنوتیپ نسل، بدترین فنوتیپ نسل، متوسط فنوتیپ نسل می باشد. حال که یک نسل به اتمام رسیده بعضی داده ها نیازمند متغیردهی هستند. ابتدا ماتریس والدین به 0 مقدار دهی می‌شود. سپس نسل افزایش می‌یابد.

تابع selection در ابتدا مقدار fitness تمامی افراد نسل را محاسبه و جمع می‌کند. سپس مقدار fitness هر فرد را تقسیم بر مقدار جمع شده می‌کند. حال احتمال انتخاب هر فرد به دست می‌آید. برای پیاده سازی الگوریتم roulette wheel با تابع choices از ماژول random با وزن احتمال انتخاب هر فرد اجازه می‌دهیم هر فرد با توجه به احتمال fitness انتخاب گردد. سپس به تعداد افراد جامعه با الگوریتم roulette wheel ما والدین را انتخاب می‌کنیم.

تابع crossover شروع می‌کند به انتخاب دو والد از ماتریس والدین و crossover با احتمال Pc اجازه crossover می‌دهد. در غیراینصورت همان دو نفر به عنوان دو فرزند خواهند بود.

تابع mutation برای همه افراد جامعه اجرا می‌شود. هر بیت از هر فرد جامعه با احتمال Pm می‌تواند جهش پیدا کند.

تابع function\_evaluation می‌تواند ارث بری گردد و به نحوه دلخواه نوشته شود.

کد مربوط به این کلاس در شکل های 2 و 3 انتهای گزارش قابل مشاهده است.

**3.2 کلاس functionEvaluations**

این کلاس به عنوان یک کلاس کمکی است. در آن می‌توانید توابع مختلف fitness را برنامه ریزی کنید و هر تابع را به عنوان یک آرگومان استفاده کنید. همچنین یک تابع map\_to\_range وجود دارد که در آن می‌توانید مقادیر ورودی هر تابع را به یک بازه ای از اعداد اختصاص دهید.

**4.2 کلاس main**

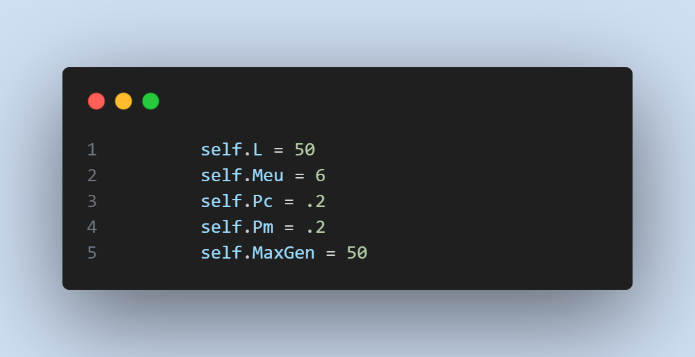
کلاس main از کلاس CGA ارث بری کرده است. متغیر های اولیه مقدار دهی شده‌اند. L اندازه هر ژن را نشان می‌دهد. Meu اندازه جمعیت را نشان می‌دهد. Pc احتمال کراس‌اوور را نشان می‌دهد. Pm احتمال جهش را نشان می‌دهد. همچنین MaxGen تعداد نسل ها را نشان می‌دهد. تابع اضافه شده runs اجازه می‌‌دهد چندین بار الگوریتم CGA اجرا شود.

**3 نتیجه**

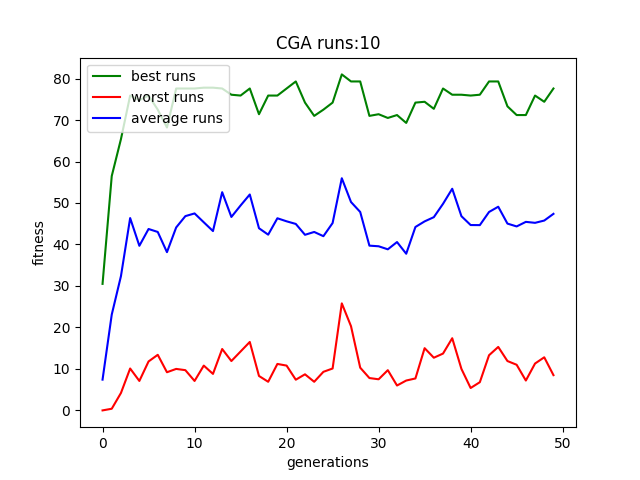
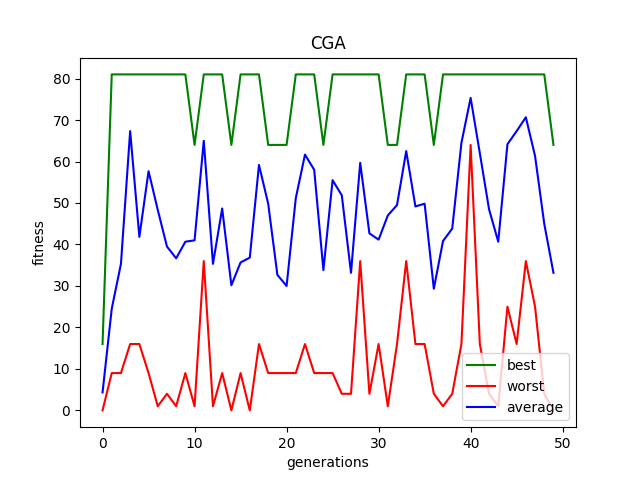
**3.1 اجرای دو تابع fitness**

دو تابع fitness با فرمول های زیر اجرا گردیده و 10 بار اجرا شده است.

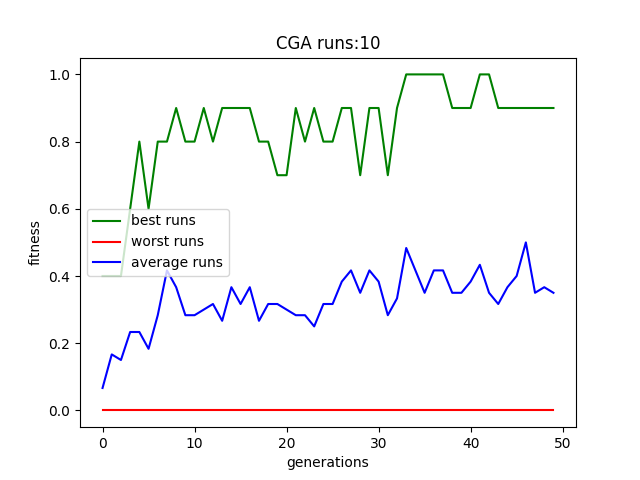
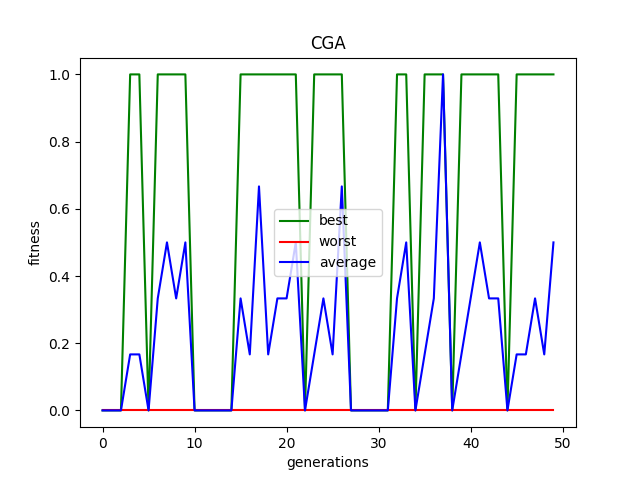
متغیر های اولیه به شرح ذیل مقدار دهی شده‌اند.



شکل 2



شکل 3 شکل 4



شکل 5 شکل 6

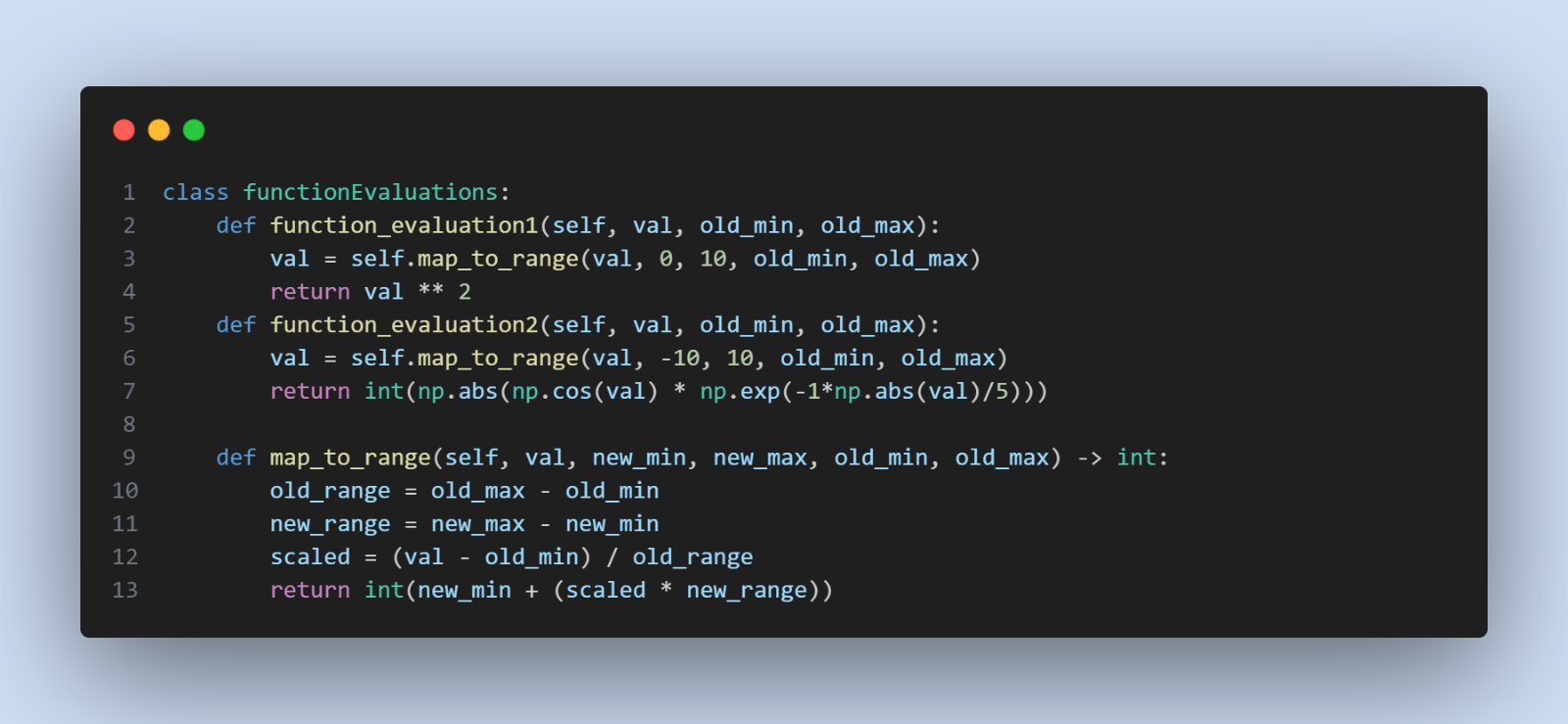
شکل 3 نشان دهنده اجرای تابع اول برای 1 بار، شکل 4 نشان دهنده اجرای تابع دوم برای 10 بار، شکل 5 و 6 نشان دهنده اجرای تابع دوم به ترتیب 1 بار و 10 بار است. جدول زیر نمودار مربوط به الگوریتم در ژن ها به طول 10، 20، 30 و 40 را نشان می دهد. (Meu=50, Pc=0.1, Pm=0.1, MaxGen=50)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| طول ژن | یک بار اجرا | | 10 بار اجرا | |
| تابع اول | تابع دوم | تابع اول | تابع دوم |
| 10 |  |  |  |  |
| 20 |  |  |  |  |
| 30 |  |  |  |  |
| 40 |  |  |  |  |

**4 کد ها**

**1.4 کد کلاس CGA**

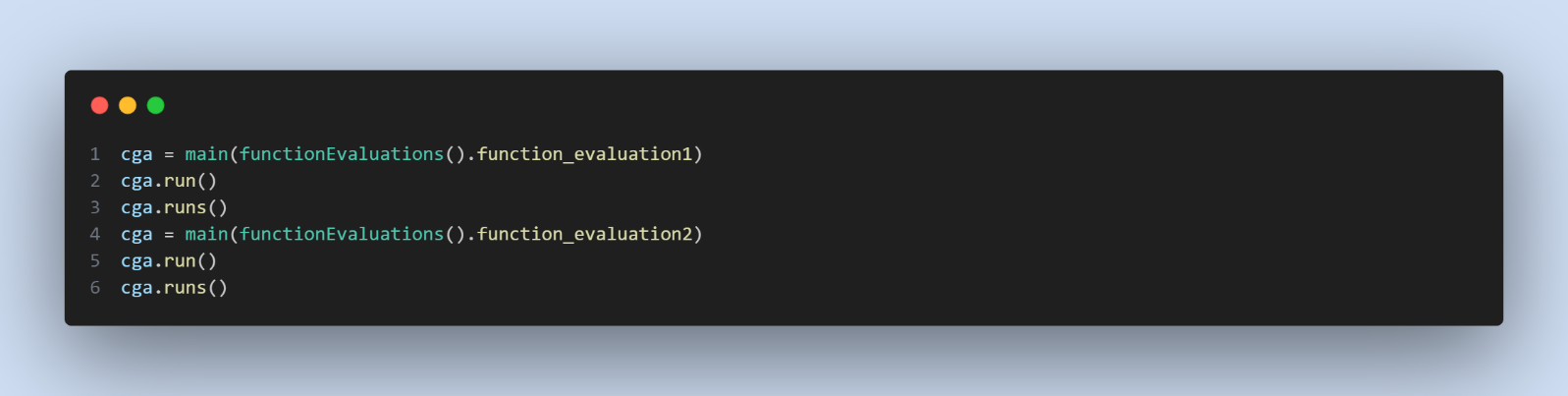
**2.4 کد کلاس functionEvaluations**



**3.4 کد کلاس main**



**4.4 نحوه اجرای کدها**



**5.4 کتابخانه ها**

