به نام خدا

پروژه اول درس هوش محاسباتی

استاد درس: دکتر حسین کارشناس

دستیاران درس:

مهندس مهدی دارونی

مهندس محمدامین دادگر

اعضای گروه:

علی مأمن پوش (993623037)

سیاوش امیرحاجلو (993613007)

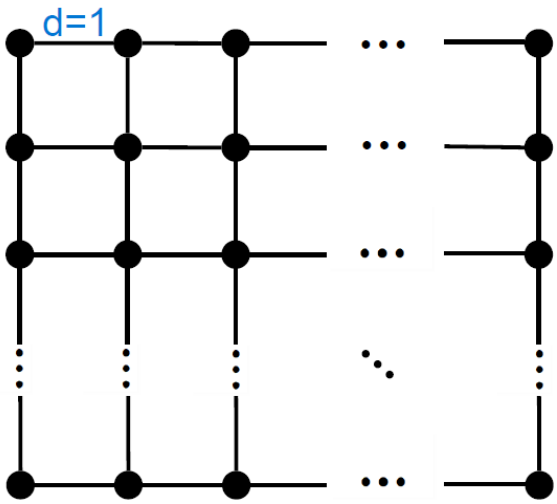
مقدمه:

الگوریتم های ژنتیک نوعی الگوریتم بهینه سازی هستند که فرآیند انتخاب طبیعی و تکامل را تقلید می‌کنند. آن‌ها با ایجاد جمعیتی از راه‌حل‌های نامزد، و سپس تکامل جمعیت در طول زمان از طریق فرآیند انتخاب، بازترکیب و جهش کار می‌کنند. در هر تکرار الگوریتم، تناسب هر راه‌حل کاندید با استفاده از تابع برازندگی ارزیابی می‌شود و به راه‌حل‌های مناسب‌تر احتمال بیشتری برای انتخاب مجدد برای بازتولید داده می‌شود. با تکرار این فرآیند در چندین تکرار (نسل)، الگوریتم‌های ژنتیک می‌توانند راه‌حل‌های تقریباً بهینه را برای مسائل پیچیده بهینه‌سازی در حوزه‌های مختلف بیابند.

الگوریتم‌های ژنتیک در زمینه‌های مختلفی از جمله مهندسی، مالی، اقتصاد و علوم کامپیوتر کاربرد دارند. در مهندسی، الگوریتم‌های ژنتیک برای بهینه‌سازی طراحی سیستم‌های پیچیده مانند بال‌های هواپیما و اجزای موتور استفاده می‌شوند. در امور مالی و اقتصاد، از آنها برای بهینه سازی استراتژی های سرمایه گذاری، انتخاب پورتفولیو و مدیریت ریسک استفاده می شود. در علوم کامپیوتر، الگوریتم‌های ژنتیک برای بهینه‌سازی پارامترهای مدل‌های یادگیری ماشین، مانند شبکه‌های عصبی و درخت‌های تصمیم‌گیری استفاده می‌شوند. به طور کلی، الگوریتم‌های ژنتیک یک ابزار بهینه‌سازی قدرتمند و انعطاف‌پذیر هستند که می‌توانند برای حل طیف وسیعی از مسائل در جایی که فضای جستجو بزرگ و پیچیده است، استفاده شود.

در این مساله، هدف پیدا کردن یک راه‌حل بهینه برای قرار دادن تعدادی دکل اینترنتی در یک شهر با ۴۰۰ محله است به طوری که هزینه ساخت و نگهداری از دکل‌ها و میزان رضایت‌مندی کاربران اینترنتی در حالت مطلوبی باشد.

هر دکل، تعدادی محله را تحت پوشش خود دارد. این بدین معنی است که پهنای باند هر دکل بین محله‌های تحت پوشش آن تقسیم می‌شود. شکل شهر به صورت یک مربع ۲۰ در ۲۰ در نظر گرفته می‌شود و آدرس محله‌ی اول یا در مختصات (0, 0) و یا در (0.5, 0.5) است. در شکل زیر می‌توانید نحوه قرارگیری محله‌ها در شهر را ببینید.



شکل 1: نحوه قرارگیری محله‌ها در شهر

در این مساله، باید تعداد بهینه دکل‌ها را بدست بیاوریم. به شرط آن که به بهینگی مطلوبی از میزان رضایت‌مندی کاربران و هزینه ساخت و نگهداری دکل‌ها دست پیدا کنیم. از الگوریتم ژنتیکی با این هدف و به دلیل گسترده بودن پارامترها و پیچیده بودن فضای حالت استفاده می‌کنیم.

بهینه‌سازی:

- **ابرپارامترها**

با توجه به اطلاعات داده دشده در فایل توضیح پروژه و آزمایشات صورت گرفته، مقدار ابرپارامترهای مساله از این قرار است:

تعداد نسل‌ها: ۲۰۰

تعداد کروموزوم‌ها: ۵۰

نرخ جهش: ۰,۱ و ۰,۹

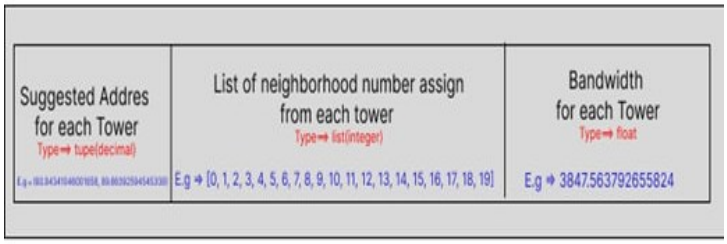
نرخ بازترکیب: ۰,۱ و ۰,۹

- **سیاست انتخاب**

تعدادی از کروموزوم‌ها انتخاب شده و با آن‌ها فرزندان تولید میشوند و از میان جمعیت موجود و فرزندان، ۵۰ کروموزوم با برازندگی بیشتر انتخاب می‌شوند. پس سیاست انتخاب به صورت $(\mu + \lambda)$ است که μ اندازه جمعیت والدین و λ اندازه جمعیت فرزندان است و روش انتخاب به صورت برشی می‌باشد.

- **ساختار کروموزوم**

در این مساله هر ژن نمایانگر یک دکل می‌باشد. این یعنی طول هر کروموزوم نشان دهنده تعداد دکل‌ها در راه حل است. تعداد دکل‌ها نمیتواند جزء پارامترهای داخل یک ژن باشد چون تغییر مقدار آن روی تغییر باقی عناصر تاثیر مستقیم دارد و تغییر باقی عناصر هم روی تغییر تعداد دکل‌ها تاثیر مستقیم دارد. در شکل زیر ساختار هر ژن از کروموزوم را مشاهده می‌کنید.



شکل 2: ساختار هر ژن از کروموزوم

ساختار کروموزوم شامل موارد زیر است.

1 - آدرس دکل که شامل یک تاپل حاوی x و y دکل است.

2 - لیست محله‌های تحت پوشش دکل که شامل لیستی از شماره محله‌ها می‌باشد.

3 - پهنای باند دکل که شامل یک عدد اعشاری است.

- **برازندگی**

برازندگی یک ژن(دکل) برابر است با امتیاز رضایت‌مندی جمعیت تحت پوشش دکل منهای هزینه کل دکل. هزینه‌ی کل دکل برابر است با هزینه‌ی اولیه ساخت دکل به علاوه‌ی هزینه‌ی نگهداری آن. طبق فایل (problem_config) هزینه‌ی اولیه دکل و هزینه‌ی نگهداری آن بر اساس هر (Mb/s) مشخص‌شده است.

برای محاسبه سرعت بر حسب (Mb/s) ابتدا باید پهنای باند اسمی (BW') را به دست آورد. سپس با تابع شبه گاوسی(COV) فاصله هر محله تا دکل را تأثیر می‌دهیم. حال پهنای باند هر محله به دست می‌آید. پهنای باند یک محله که بر تعداد افراد محله تقسیم شود، پهنای باند هر کاربر محاسبه می‌شود. حال باید طبق جدول رضایت‌مندی امتیاز آن را در تعداد کاربران ضرب نمود و محاسبه کرد. حال که امتیاز رضایت‌مندی کاربران محله به دست آمد. به همین‌گونه برای محله‌های دیگر نیز محاسبه می‌شود.

اگر مجموع امتیاز رضایت‌مندی را از هزینه کل دکل کم کنیم، برازندگی یک ژن به دست می‌آید. برای محاسبه برازندگی کروموزوم، برازندگی ژنّای آن را با هم جمع می‌کنیم.

- **الگوریتم**

الگوریتم ژنتیکی برای حل این مساله به شرح زیر است:

1) انتخاب تعداد دکل‌ها
2) ساخت ۵۰ کروموزوم به طول(با تعداد ژن) به اندازه تعداد دکل‌ها (با مقادیر تصادفی)
3) محاسبه برازندگی هر کروموزوم
4) بازترکیب دو به دوی m کروموزوم از جمعیت موجود بر اساس نرخ بازترکیب
5) جهش فرزندان به دست آمده
6) محسابه برازندگی فرزندان
7) جایگزینی والدین دارای برازندگی کمتر با فرزندان دارای برازندگی بیشتر
8) اگر تعداد نسل‌ها به ۲۰۰ رسید، بهترین کروموزوم را برگردان و در غیر این صورت به مرحله 4 برگرد.

- **مقداردهی اولیه**

در مقداردهی اولیه کروموزوم‌ها، آدرس هر دکل به صورت تصادفی و در بازه ابعاد شهر انتخاب می‌شود. محله‌های هر دکل از بین مقادیر ممکن به صورت تصادفی انتخاب می‌شوند و این اطمینان حاصل می‌شود که مقادیر تکرار نشوند. پهنای باند هر دکل هم به صورت تصادفی از بازه صفر تا حد بالای سرعت اینترنت ضرب در جمعیت تحت پوشش دکل انتخاب می‌شود.

- **تعداد دکل‌ها**

تعداد دکل‌ها میتواند به سه روش محاسبه و انتخاب شود. از طریق الگوریتم تکاملی، از طریق یک حلقه و از طریق انتخاب اعداد مناسب. در روش الگوریتم تکاملی، کند شدن روند اجرای الگوریتم و تک پارامتری بودن تعداد دکل‌ها مهم‌ترین مشکلات است. در روش استفاده از حلقه، روند اجرای برنامه سریع‌تر می‌شود اما باز هم زمان بسیار زیادی می‌برد. در روش سوم یعنی انتخاب اعداد مناسب، میتوان از یک تابع بازگشتی مانند جست و جوی باینری برای پیدا کردن بهترین تعداد دکل استفاده کرد.

برای پیاده سازی پروژه، از روش سوم استفاده کردیم. چون هم روند اجرایی سریعی دارد و هم یکی از نقاط بهینه را برمی‌گرداند. مشکل این روش لحاظ نکردن بخش‌هایی از فضای حالت برای جست و جو است که میتوانند حاوی بهترین نقطه بهینه در کل فضای حالت باشند اما برای حل این مساله الزامی به دست‌یابی به بهترین نقطه بهینه نیست.

- **بازترکیب**

از میان جمعیت موجود، با توجه به نرخ بازترکیب تعدادی کروموزوم که دارای بیشترین میزان برازندگی هستند انتخاب شده و دو به دو بازترکیب می‌شوند. برای این کار هر ژن از هر کروموزوم انتخاب شده با ژن نظیر در کروموزوم دیگر با هم بازترکیب می‌شوند و دو ژن جدید ساخته می‌شود که یکی متعلق به فرزند اول و دیگری متعلق به فرزند دوم است.

برای بازترکیب ژن‌ها باید هر پارامتر آن‌ها را با پارامتر نظیر و با روش مناسب بازترکیب کرد. به این منظور با توجه به آزمایشات صورت گرفته و نتایج مشاهده شده این روش‌ها مورد استفاده فرار گرفته اند:

الف) مختصات دکل‌ها: روش مورد استفاده --> تقطیع ریاضی تام
مقدار $\alpha=0.25$ در نظر گرفته شده.

ب) محله‌های تحت پوشش: روش مورد استفاده --> جابه‌جایی محله‌های دو دکل
به غیر از این روش، روش‌های دیگر مانند تقطیع تک نقطه ای هم استفاده شدند اما درخروجی راه‌حل تاثیری نداشتند و به دلیل این که عناصر موجود در لیست محله‌ها باید در کل کروموزوم یکتا باشند، انجام این روش‌های تقطیع موجب کند شدن روند الگوریتم می‌شود.

ج) پهنای باند دکل: روش مورد استفاده --> بازترکیب آمیختن(Blend)

مقدار $\alpha=0.25$ در نظر گرفته شده.

- **جهش**

با توجه به نرخ جهش، هر کدام از ژن‌های فرزندان تولید شده در مرحله بازترکیب، با احتمال p جهش پیدا می‌کنند. این جهش برای هر پارامتر موجود در ژن روش مخصوص به خود را دارد. به این منظور با توجه به آزمایشات صورت گرفته و نتایج مشاهده شده این روش‌ها مورد استفاده فرار گرفته اند:

الف) مختصات دکل‌ها: روش مورد استفاده --> جهش یکنواخت (uniform)

برای x و y یک عدد تصادفی در بازه‌های [x-1, x+1] و [y-1, y+1] انتخاب می‌کنیم.

ب) محله‌های تحت پوشش: روش مورد استفاده --> جهش با مقدار تصادفی (Random resetting) و shuffle کردن عناصر

در این روش یکی از عناصر موجود را به طور تصادفی انتخاب می‌کنیم و مقدار آن را تغییر می‌دهیم. چون هر عنصر موجود در ژن باید در کل کروموزوم یکتا باشد، مقدار جایگزین را از طریق انتخاب تصادفی یک عنصر از یک ژن دیگر که به صورت تصادفی انتخاب شده برمی‌داریم و جای این دو عنصر را با هم عوض می‌کنیم.

ج) پهنای باند دکل: روش مورد استفاده --> جهش یکنواخت (uniform)

در این روش ابتدا جمعیت تحت پوشش دکل را بدست می‌آوریم و سپس مقدار کف و سقف بازه انتخاب را با استفاده از ضرب جمعیت در مقدار کف و سقف سرعت اینترنت برای هر کاربر و در یک مقدار α به دست می‌آوریم.

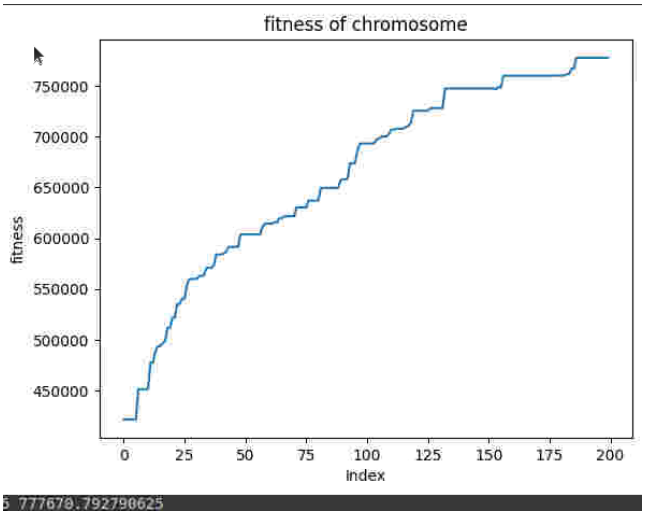
مقدار $\alpha=0.25$ در نظر گرفته شده. مقدار حد بالایی سرعت 3 Mb/s و حد پایینی 0.2 Mb/s انتخاب شده است.

آزمایش:

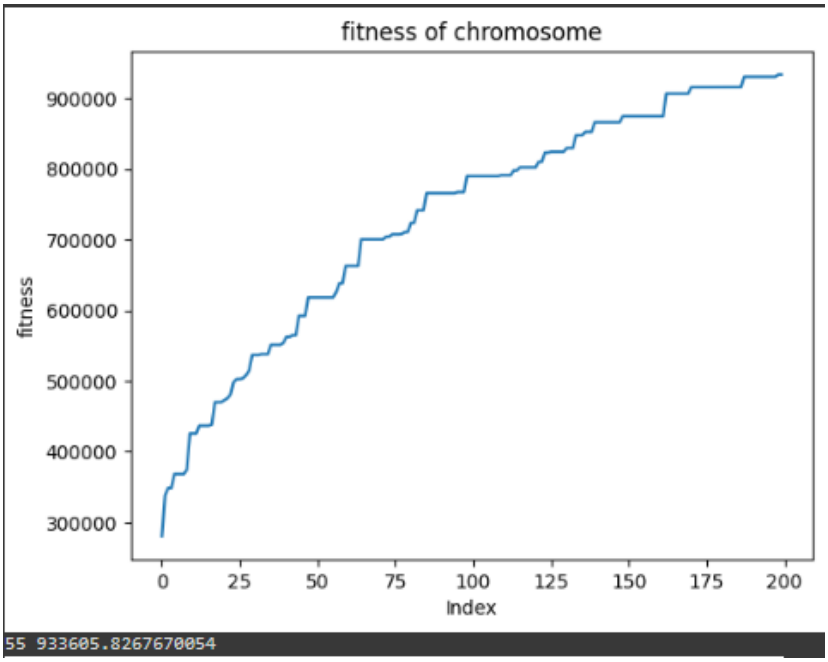
برای آزمایش برای هر تابع در کد یونیت تستی نوشته و آن را با داده‌های به‌دست‌آمده از ژن‌های کروموزوم تست نموده‌ایم. برای اینکه الگوریتم بعضی از تابع ها را بهینه کنیم با تست کردن به صورت مستقیم بر الگوریتم اصلی، استفاده کرده‌ایم.

نتایج بدست آمده:

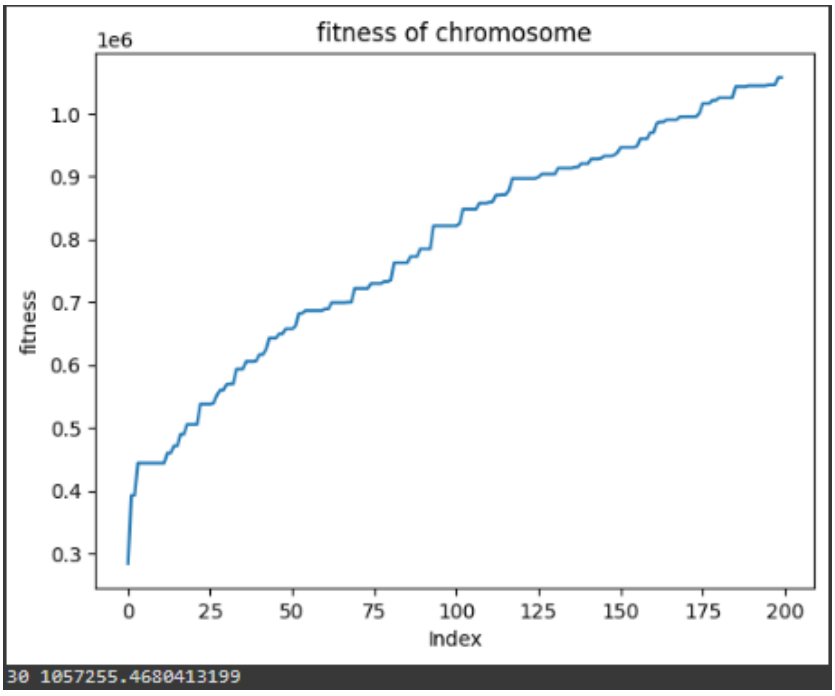
با توجه به تابع find_best برای 7 بار(بستگی به بازه جست و جو هم دارد و می‌تواند متغیر باشد) الگوریتم ژنتیکی اجرا می‌شود. می‌توان دریافت که بهترین برازندگی چگونه تغییر کرده است. برای نشان دادن روند برازندگی نسل‌ها نمودار مربوطه را با کتابخانه matplotlib رسم نموده‌ایم. برای اندازه‌گیری تایم برنامه از کتابخانه‌ی time استفاده کرده‌ایم که زمان میانگین حدودی برابر 15 دقیقه می‌باشد(15.724004741509756). انتهای خروجی‌ها ، فهرستی از کروموزوم برتر است که بیشتر برازندگی را میان بقیه دارد. تصاویر زیر مربوط به 200 نسل است.



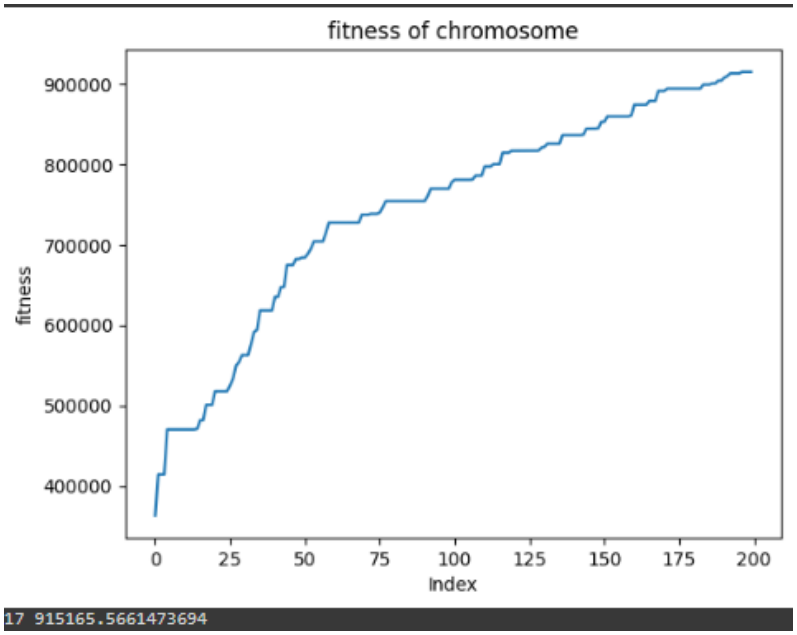
نمودار 1 : برازندگی برای ۵ دکل



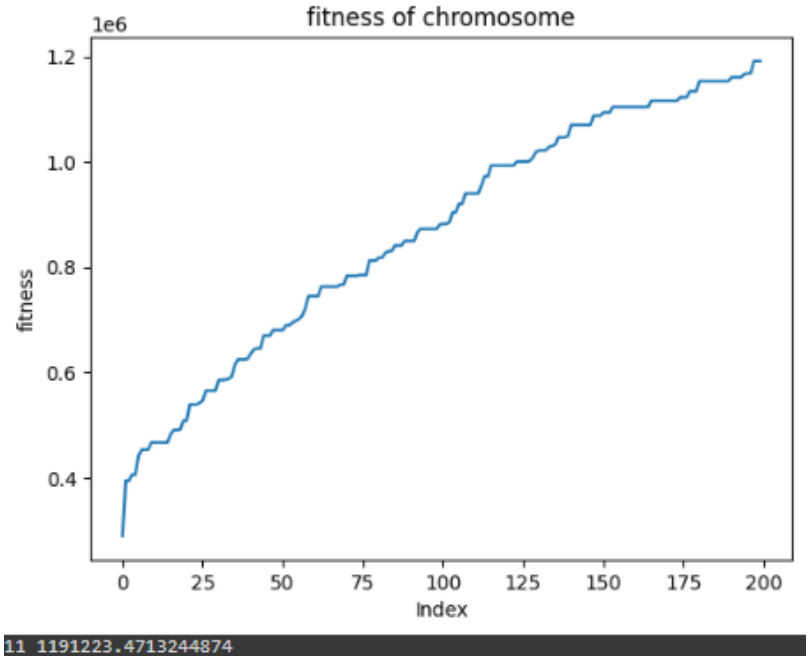
نمودار 2: برازندگی برای ۵۵ دکل



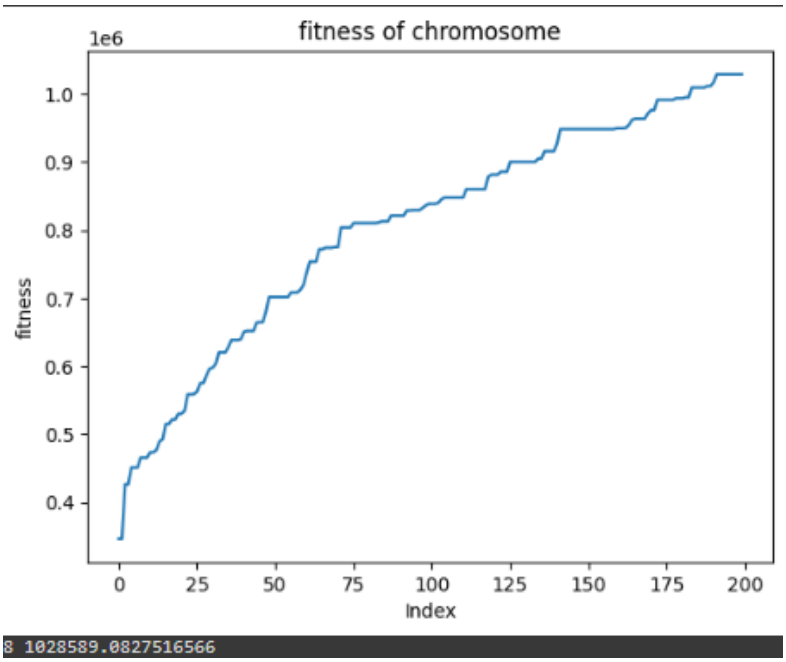
نمودار 3: برازندگی برای ۳۰ دکل



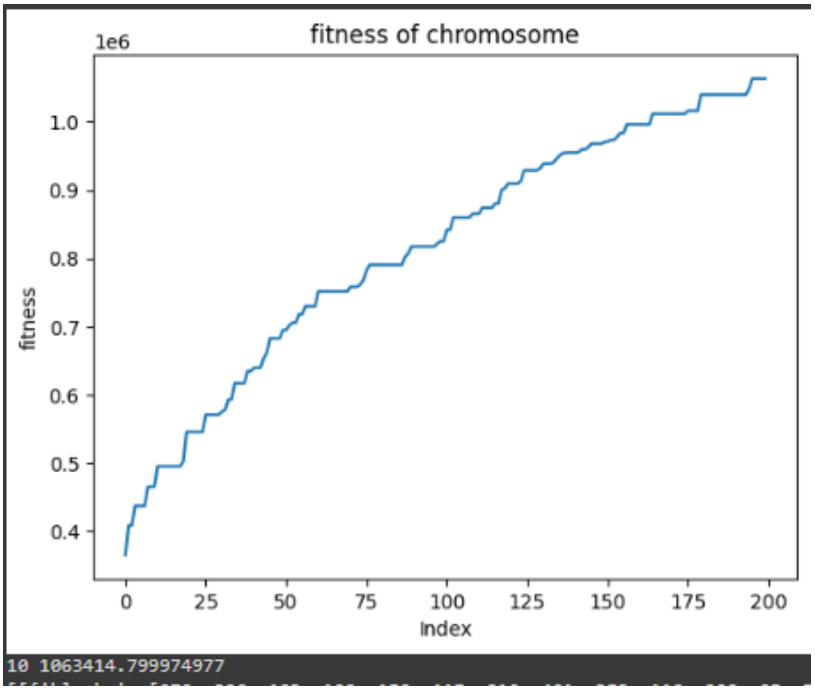
نمودار 4: برازندگی برای ۱۷ دکل



نمودار 5: برازندگی برای ۱۱ دکل

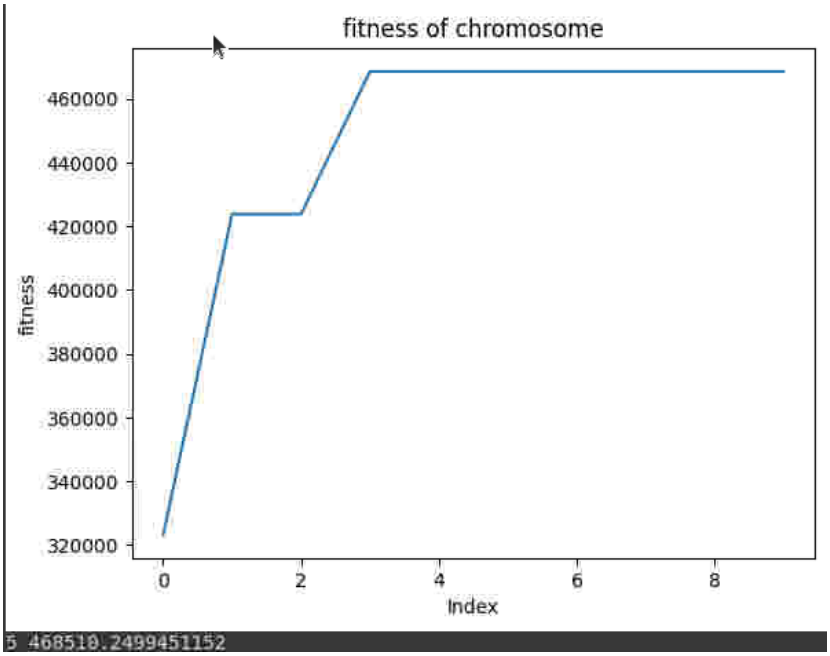


نمودار 6: برازندگی برای ۸ دکل

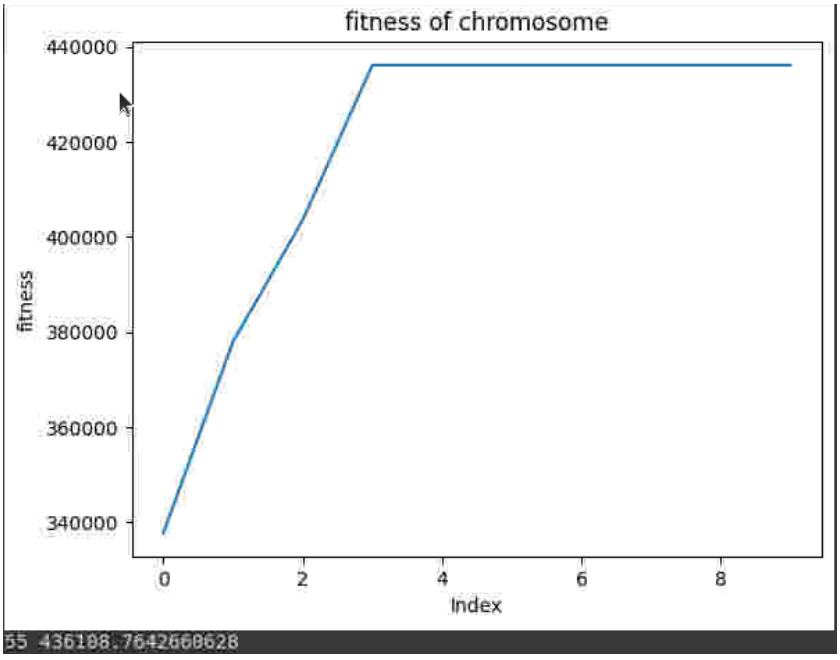


نمودار 7: برازندگی برای ۱۰ دکل

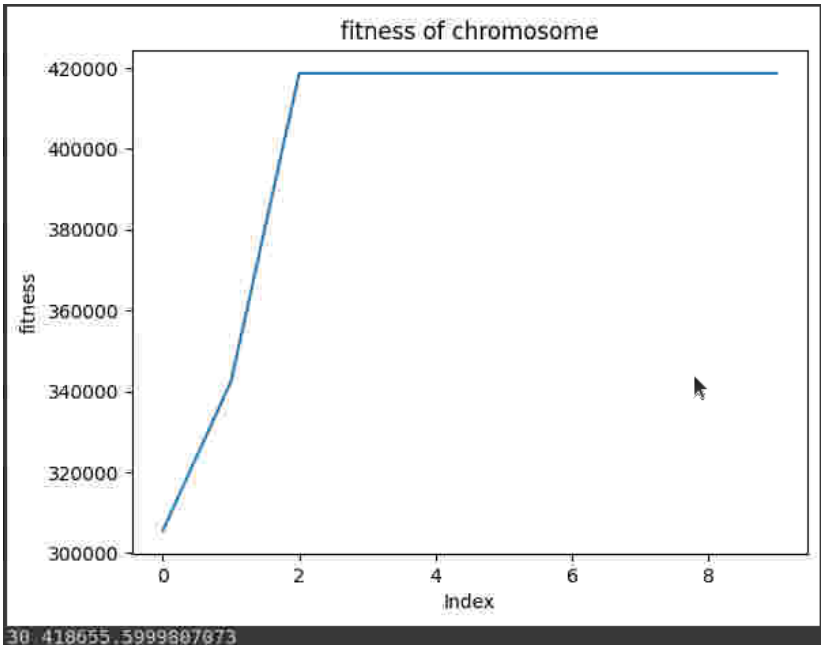
اگر تعداد جنریشن‌ها را برابر 10 نسل بکنیم نتایج به شکل زیر است:



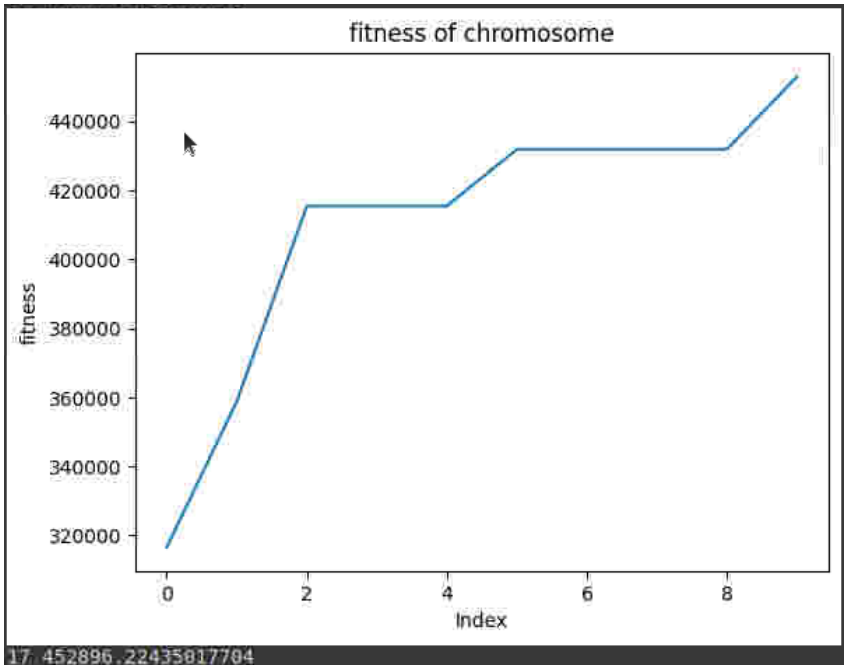
نمودار 8: برازندگی برای ۵ دکل



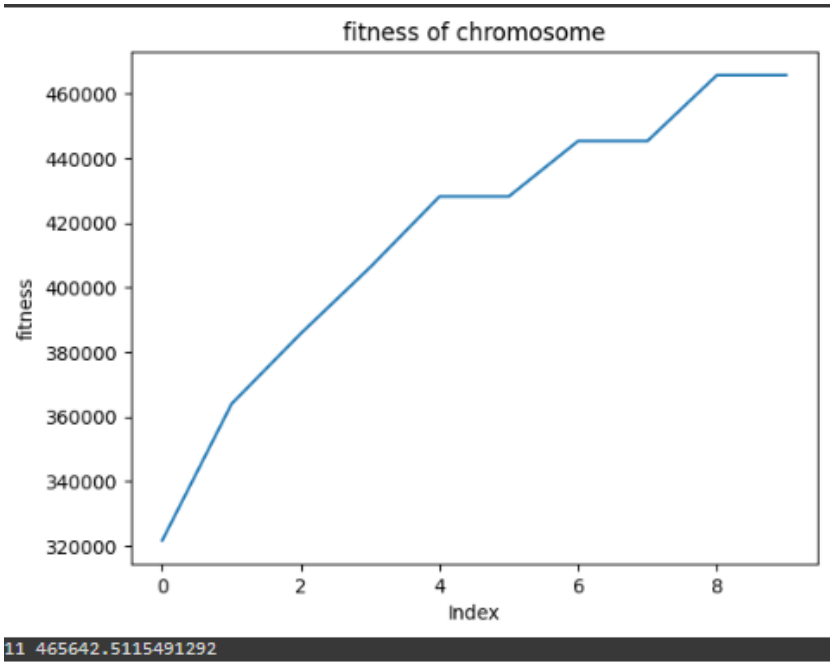
نمودار 9: برازندگی برای ۵۵ دکل



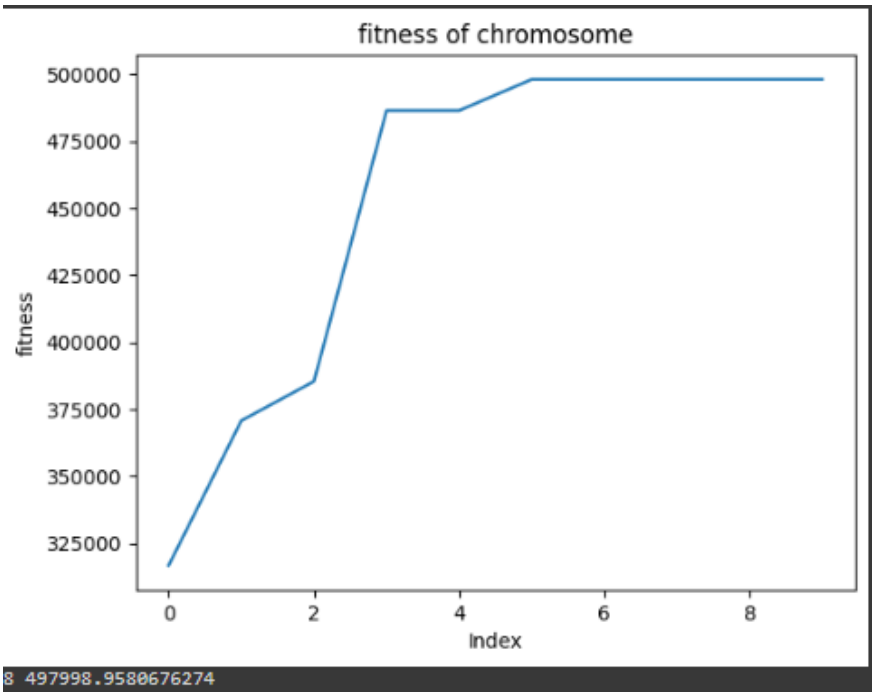
نمودار 10: برازندگی برای ۳۰ دکل



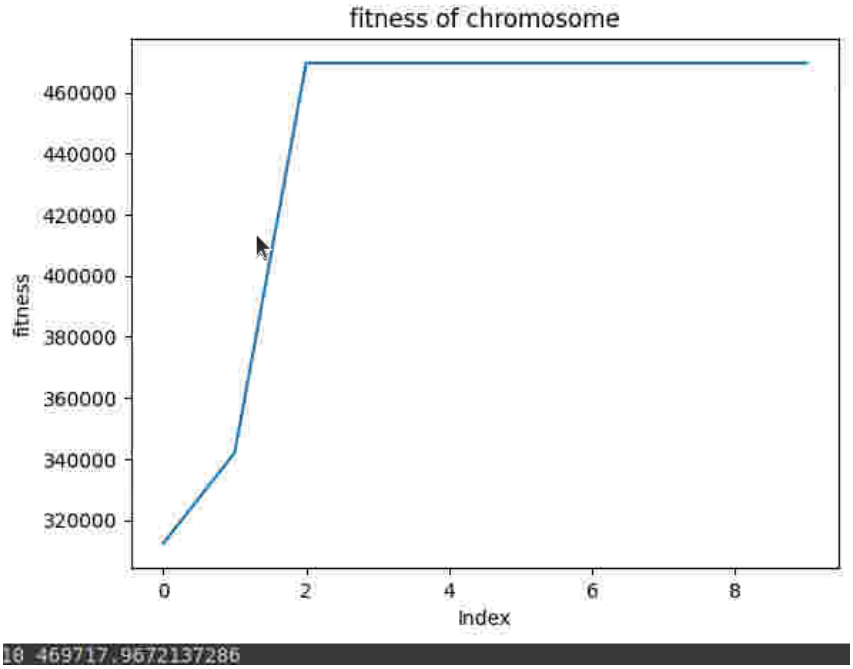
نمودار 11: برازندگی برای ۱۷ دکل



نمودار 12: برازندگی برای ۱۱ دکل



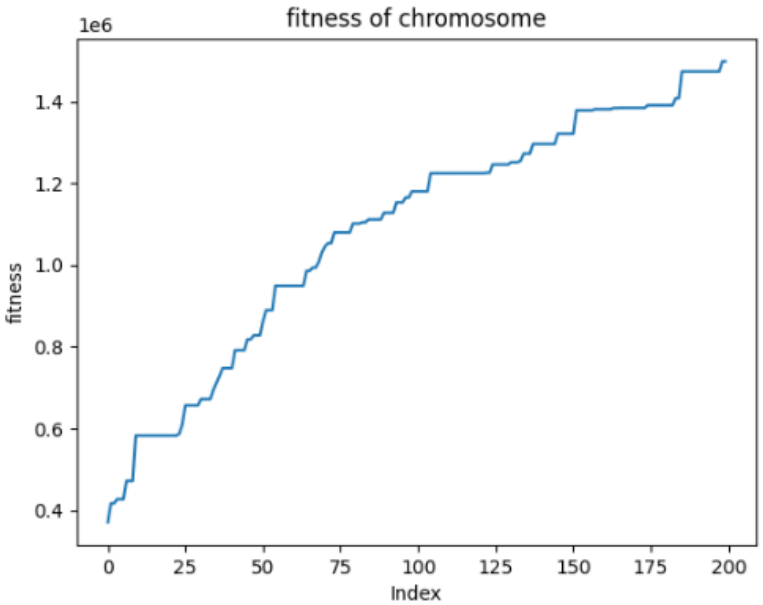
نمودار 13: برازندگی برای ۸ دکل



نمودار 14: برازندگی برای ۱۰ دکل

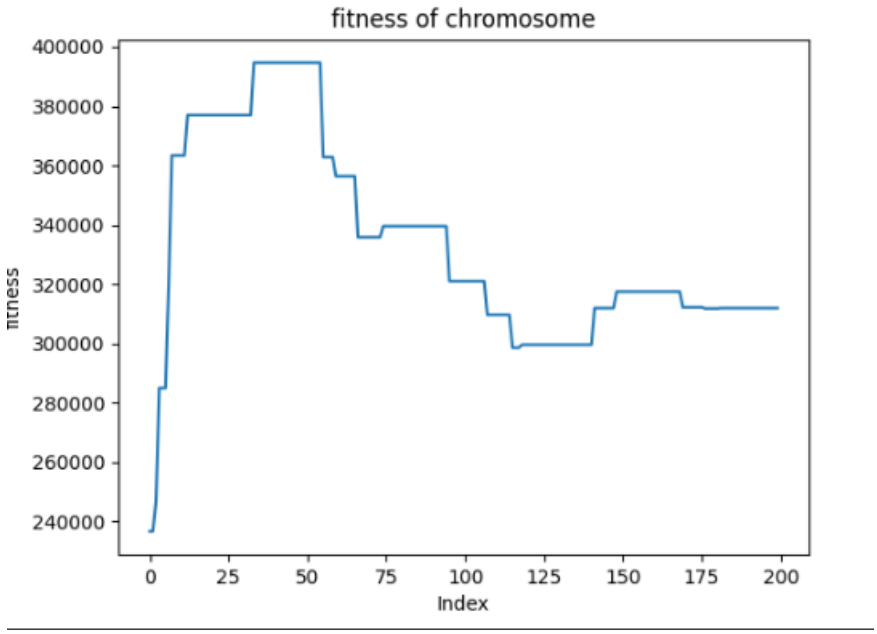
نتایج تست الگوریتم برای نرخ‌های مختلف بازترکیب و جهش برای ۱۰ دکل:

الف) با نرخ بازترکیب = 0.9 و نرخ جهش = 0.1 را می‌توانید در "نمودار ۷" مشاهده کنید.
ب) با نرخ بازترکیب = 0.9 و نرخ جهش = 0.9



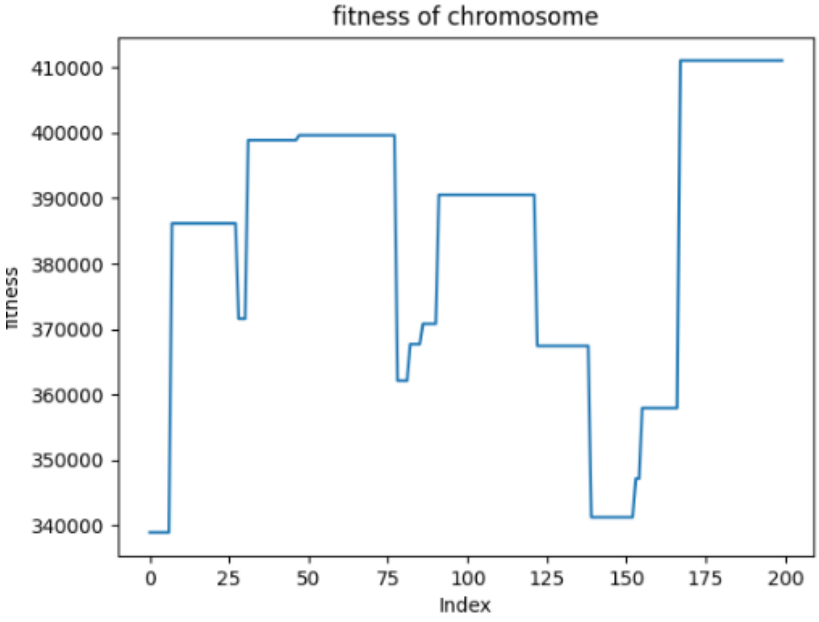
نمودار 15: برازندگی با نرخ‌های جهش و بازترکیب مختلف

ج) با نرخ بازترکیب = 0.1 و نرخ جهش = 0.1



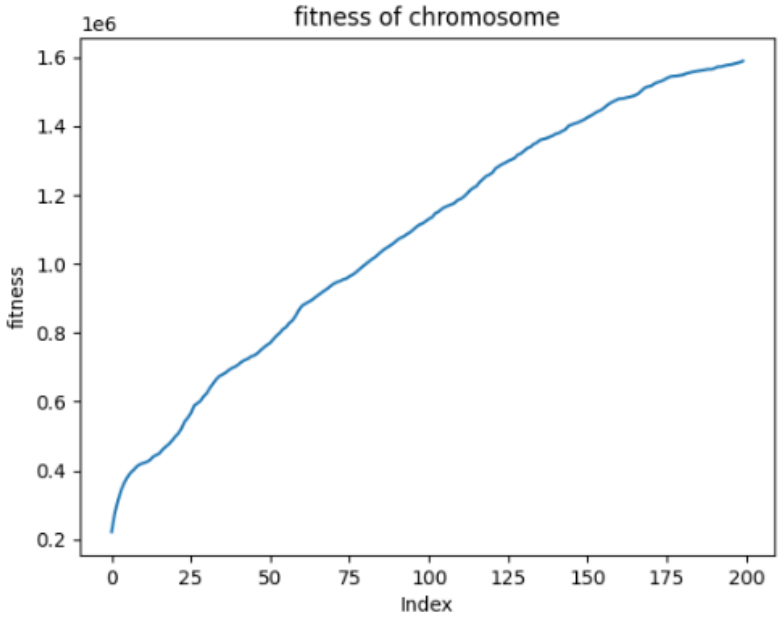
نمودار 16: برازندگی با نرخ‌های جهش و بازترکیب مختلف

د) با نرخ بازترکیب = 0.1 و نرخ جهش = 0.9



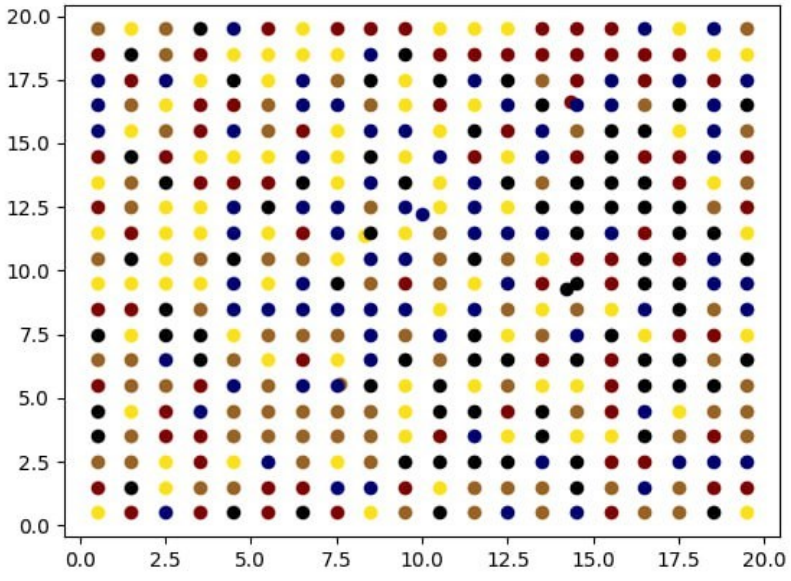
نمودار 17: برازندگی با نرخ‌های جهش و بازترکیب مختلف

نمودار ۱۸ تغییرات میانگین برازندگی کروموزوم‌ها برای ده دکا و با نرخ بازترکیب و جهش 0.9 را نمایش می‌دهد:



نمودار 18: تغییرات میانگین برازندگی

همچنین در نمودار ۱۹ میتوانید نقشه پراکندگی محله‌های هر دکل را پس از اجرای برنامه و برای تعداد ۱۰ دکل مشاهده کنید:



تحلیل نتایج:

طبق نمودارهای بالا در بخش نتایج بدست آمده می‌توان برداشت کرد که:

- با توجه به "نمودار ۱" نمودار برازندگی از 450000 تا 750000 تغییر کرده است و پیشرفت 300000 داشته و تقریبا بعد از 130 نسل شیب افزایش مقدار برازندگی رو به صفر رفته است.
- با توجه به "نمودار ۲" نمودار برازندگی از 300000 تا 900000 تغییر کرده است و پیشرفت 600000 داشته و تقریبا بعد از 160 نسل شیب افزایش مقدار برازندگی رو به صفر رفته است.
- با توجه به "نمودار ۳" نمودار برازندگی از 0.3 میلیون تا 1.0 میلیون تغییر کرده است و پیشرفت 0.7 میلیون داشته و تقریبا بعد از 180 نسل شیب افزایش مقدار برازندگی رو به صفر رفته است.
- با توجه به "نمودار ۴" نمودار برازندگی از 400000 تا 900000 تغییر کرده است و پیشرفت 500000 داشته و تقریبا بعد از 160 نسل شیب افزایش مقدار برازندگی رو به صفر رفته است.
- با توجه به "نمودار ۵" نمودار برازندگی از 0.4 تا 1.2 تغییر کرده است و پیشرفت 0.8 داشته و تقریبا بعد از 175 نسل شیب افزایش مقدار برازندگی رو به صفر رفته است.
- با توجه به "نمودار ۶" نمودار برازندگی از 0.4 تا 1 تغییر کرده است و پیشرفت 0.6 داشته و تقریبا بعد از 190 نسل شیب افزایش مقدار برازندگی رو به صفر رفته است.
- با توجه به "نمودار ۷" نمودار برازندگی از 0.3 تا 1 تغییر کرده است و پیشرفت 0.7 داشته و تقریبا بعد از 180 نسل شیب افزایش مقدار برازندگی رو به صفر رفته است.

در رابطه با نرخ جهش و نرخ بازترکیب با توجه به نمودارهای "۷و ۱۵و ۱۶و ۱۷" می‌توان این نتیجه را گرفت که با افزایش نرخ جهش، فضای جست و جو افزایش میابد. به طوری که نمودارهای موبوط به نرخ جهش 0.9 دارای تغییرات بیشتری هستند. اما با افزایش مقدار نرخ بازترکیب، همگرایی نمودارها و نتایج نهایی بیشتر می‌شود که نشان دهنده بهبود راه‌حل‌های موجود است. با توجه به نمودارها و نتایج بدست آمده بهترین نرخ بازترکیب و جهش بین این چهار مورد، به ترتیب 0.9 و 0.9 است. چون فضای جست و جو افزایش میابد و راه‌حل‌های بهتری ساخته می‌شوند و همگرایی نمودار برازندگی به وجود می‌آید. هرچند سرعت رسیدن به نقطه بهینه و ثابت کاهش میابد.

پیشنهادها:

می‌توانستیم برای بدست آوردن تعداد دکل‌ها به جای اینکه از تابع find_best به صورت بازگشتی استفاده نماییم، از الگوریتم ژنتیکی برای بدست آوردن تعداد دکل‌ها استفاده نماییم. در راه‌حل دیگر می‌توانستیم که با چند حلقه‌ی تو در تو مقدار بهینه را محاسبه نماییم و سپس هر حلقه را به یک پارامتر اختصاص بدهیم.

برای بازترکیب آدرس محله‌ها میتوان از تقطیع چند نقطه‌ای و برای آدرس هردکل می توان از بازترکیب آمیختن نیز استفاده نمود.

برای مقدار دهی اولیه هم می‌توان محله‌ها را به صورت ترتیبی به دکل‌ها اختصاص داد.

مراجع:

- اسلایدهای درسی دکتر کارشناس

- Kruse, Borgelt, Braune, Mostaghim & Steinbrecher, "Computational intelligence: A methodological approach," 2nd edition, Springer, 2016.
- Eiben and Smith, "Introduction to evolutionary computing," 2nd edition, Springer, 2015.

- Vijini Mallawaarachchi, "Introduction to Genetic Algorithms", 2017, available on:

  https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-

  code-e396e98d8bf3

- Tutorialspoint team, "Genetic Algorithms - Further Readings" available on: https://www.tutorialspoint.com/genetic_algorithms/genetic_algorithms_further_readings.htm