گزارش

سها کرد (983212058)

سوال اول

نحوه‌ی پیاده‌سازی الگوریتم ژنتیک:

- مقداردهی اولیه رشته های باینری و تشکیل mating pool
- انتخاب والد ازmating pool به روش رولت ویل
- برای باز ترکیبی: یک نقطه ای و یونیفرم داریم که نوع از ورودی کاربر مشخص می‌شود.
- برای جهش: از احتمال و انداختن سکه استفاده شد.
- انتخاب پدر و مادر با ارزش هر والد
- فرزندان، پدر مادر های نسل بعد می‌شوند.

در این برنامه :

اگر جمعیت اولیه‌ای که کاربر وارد کرده بود فرد بود، به صورت رندوم یک فرد دیگر به آن اضافه می کنیم تا بتوانیم عمل بازترکیبی را به صورت صحیح انجام دهیم _ پایان الگوریتم هم از طریق تعداد لوپی که کاربر وارد کرده است ممکن هست و هم در صورتی که تغییری در ماکزیمم fitness افراد در نسل های اتفاق نیفتد _ احتمال رخ دادن بازترکیبی و یا جهش از ورودی گرفته می‌شود _ چگونگی انجام شدن بازترکیبی توسط کاربر وارد می‌شود _ میزان تکرار الگوریتم توسط کاربر وارد می‌شود – تعداد جمعیت اولیه توسط کاربر وارد می‌شود.

تابع fx :

تابعی است که در سوال مقدار ماکزیمم سراسری آن خواسته شده. این تابع میزان شایستگی هر فرد را مشخص می کند.

تابع selection :

این تابع به عنوان ورودی fitness هر فرد و جمعیت رندوم ساخته شده را می گیرد سپس بر اساس ارزش هر فرد یک مقداری احتمالی برای انتخاب شدن آن فرد برای نسل بعد در نظر می گیرد، سپس با رولت ویل به تعداد جمعیت مشخص شده، افراد را انتخاب می کند (که فرد تکراری هم ممکن هست در جمعیت باشد) و در نهایت لیستی از افراد به تعدادی که کاربر وارد کرده بر می گرداند.

تابع crossover :

این تابع دو فرزند و احتمال انجام باز ترکیبی و نوع آن را از کاربر دریافت می کند سپس دو فرزند جدید تولید شده را بعد از انجام دادن بازترکیبی برمی گرداند.

تابع mutation :

این تابع یک فرزند و احتمال رخ دادن جهش را از ورودی می گیرد و با در نظر گرفتن احتمال جهش برای هر بیت، تغییرات را بر روی آن اعمال می کند.

تابع genetic algorithm:

تابع سوال را گرفته و احتمال جهشی و باز ترکیبی و نوع باز ترکیبی و تعداد جمعیت اولیه را از کاربر می گیرد. چون بازه ما اعداد ۰ تا ۳۱ که در سوال مطرح شده بود، برای این تعداد عدد ۵ بیت کافی بود، هر فرد را لیستی از اعداد ۰ و ۱ در نظر گرفته شده ۵ بیتی هستند.

mating pool لیستی تصادفی از افراد است که تعداد اعضای آن را کاربر مشخص می کند.در ابتدا نفر اول mating pool را به عنوان بهترین فرد در نظر می گیریم و در ادامه یک حلقه داریم که در هر دور آن الگوریتم ژنتیک تکرار می‌شود .

بررسی خروجی :

هرچی میزان احتمال جهش و بازترکیبی و دور الگوریتم بیشتر باشه نتایج بهتر است (البته دور الگوریتم تا یک جایی تاثیر دارد و بعد از آن تفاوت چندانی ایجاد نمی کند)

سوال دوم

سوال 1 :

در این سوال، یک والد و یک فرزند داریم و در نتیجه crossover نداریم و برای (1+1) ES این ترتیب را داریم:

ابتدا یک والد رندوم در محدوده ی مسئله انتخاب می‌شود، پس از آن باید دید والد چند بعدی در نظر گرفته شده است. چون که مسئله گفته دو حالت 10 و 100 مسئله را بررسی کنیم، این دو ابعاد والد و نهایتا فرزندی که ایجاد می‌شود را مشخص می کند.

(هر یک از این حالات در واقع ابعاد یک فرد را مشخص میکند، نه تعداد افراد مسئله) با n = 10 فرقی با حالت n = 100 در اجرای الگوریتم ندارد در نتیجه یکی را بررسی می کنیم.

با رندوم یک والد ۱۰ بعدی ایجاد شد. در ادامه باید با تابع صورت سوال ، شایستگی والد را محاسبه کنیم. تابع مسئله یک خروجی صحیح می‌دهد که همان شایستگی است.

بعد از مشخص شدن شایستگی والد باید فرزند را از طریق پدر ایجاد کنیم. چون باز ترکیبی نداریم فقط با جهش فرزند را ایجاد می کنیم . برای جهش صورت مسئله گفته است که از توزیع نرمال N(0,1) استفاده کنیم. به این صورت که هربار برای هر بعد پدر، یک عدد تصادفی به کمک این توزیع ایجاد کنیم و سپس به بعد اضافه کنیم، تا نهایتا به یک بردار جدید دیگر برسیم که فرزند جدید است.(بردار فرزند هم ۱۰ بعدی است و ابعاد تغییری ندارد)

تابع f(vector):

این تابع یک ورودی به عنوان بردار می گیرد و هر بعد را به عنوان یک x درنظر می گیرد و آن را محاسبه می کند با تابعی که در صورت سوال هست و خروجی آن همان شایستگی فرد ورودی است.

تابع random_parent(n):

این تابع یک ورودی n که یک عدد صحیح است، از کاربر می گیرد که ابعاد فرد را مشخص می کند که دو حالت 10 و 100 را دارد و نهایتا یک آرایه n بعدی را بر می گرداند که فقط در مرحله اول برای ایجاد یک والد رندوم استفاده می‌شود.

تابع gaussian_mutation:

در این تابع ما یک فرد را برداری از ورودی تابع دریافت کرده و عملیات جهش را بر روی آن انجام می دهیم و خروجی تابع، فرد جدید است. جهش با اضافه کردن یک مقدار رندوم از توزیع نرمال N(0,1)اتفاق می‌افتد .

مراحل اجرای الگوریتم

تابع Evolutionary_strategy :

این تابع دو ورودی میگیرد که یکی از آنها تعداد ابعاد است (می‌توان ابعاد دیگری را نیز در ورودی داد فارغ از 10 و 100 خروجی را بررسی کرد) ، ورودی دیگر تعداد دفعات اجرا الگوریتم هست که از کاربر ورودی می گیرد.

بررسی خروجی :

خروجی های کد جهت بهینه شدن را نشان می‌دهد، همچنین مرحله را تا مشخص شود که بعد از چند دور به چه هدفی رسیده است. با تعداد لوپ مشخص شده از کاربر از جواب بهینه فاصله زیادی داریم. می‌توانیم الگوریتم را در لوپ بی نهایت اجرا کنیم تا ببینیم برای رسیدن به فیتنس مناسب چند مرحله را طی می کند. الگوریتم بعد از تعدادی دور حدودی 500 دور توانست به شایستگی خوبی برسد.

سوال دو

این سوال مثل قسمت قبل هست با این تفاوت که روش محاسبه‌ی توزیع نرمال در آن با سیگما متغیرهست که پس از یکسری دور که کاربر وارد می کند، تغییر می کند و مقدار مناسبی بر اساس قانون یک پنجم موفقیت می گیرد.

این تغییرات در evolutionary _algorithm نوشته شده و هربار سیگما جدید به عنوان متغیر ورودی به تابع mutation ارسال می‌شود. فقط همین تابع تغییر داشته و توابع دیگر تغییری ندارند با بخش قبل.

در قانون یک پنجم موفقیت تغییر کردن سیگما به طور خودکار است و مقدار مناسب نسبت به موفقیت ها و یا شکست های بدست آمده می گیرد. در این بخش هوشمند سازی پارامتر سیگما که در بخش قبلی مقدار یک داشت، را داریم، تا متناسب با فرزندان ایجاد شده و شایستگی عوض شود و به سمت جواب بهینه حرکت کند (آپدیت شدن پس از k بار تکرار الگوریتم اتفاق می‌افتد و k مقداری است که از ورودی دریافت می کنیم.)

برای پیاده سازی این سوال از کد سوال قبل استفاده می کنیم و ویژگی های زیر را به اضافه می کنیم:

– یک ورودی جدید از کاربر می گیریم که مقدار c را مشخص می کند.

– یک ورودی جدید به عنوان تعداد دور هایی که بعد از آن باید پارامتر سیگما باید آپدیت شود.

بررسی خروجی :

در خروجی مشاهده می کنیم که با زیاد شدن مقدار پارامتر c سرعت رسیدن ما به نقطه مینیمم کاهش پیدا می کند، می‌توان نتیجه گرفت که پارامتر c با سرعت پیشرفت رابطه معکوس دارد و با زیاد شدن مقدار این پارامتربه سمت 1 سرعت ما در حرکت به سمت نقاط بهینه

کاهش پیدا می کند.با بالا بودن ابعاد فاصله از جواب بهینه بسیار زیاد می‌شود، در این حالت رسیدن به جواب سخت و نیازمند تعداد حلقه های اجرای زیادی است.

سوال 3

حالت اول $ES(\mu, \lambda)$

(λتعداد جمعیت فرزندان و μ تعداد جمعیت والد)

پیاده سازي الگوريتم:

- ابتدا تابع شایستگي را پیاده سازي می‌کنیم مانند سوال قبل .

-به تعداد والد ذکر شده در صورت سوال(۲ تا) بطور رندوم والد تولید می‌کنیم.

-شایستگی هر يك از والد ها را محاسبه می‌کنیم

-تا زمانی که شرط خاتمه اتفاق بیافتد

يك جمعیت فرزندان به تعداد ذکر شده در سوال(۱۰ تا) به کمك بازترکیبی و جهش ایجاد می‌کنیم

شایستگي هر يك از فرزندان را به دست می‌آوریم

و به تعداد والد ذکر شده در صورت سوال(دو تا) بهترین افراد را از بین **جمعیت فرزندان** ایجاد شده انتخاب می‌کنیم و به عنوان جمعیت والد نسل بعد، به نسل بعد انتقال می‌دهیم.

حالت دوم $ES(\mu + \lambda)$

پیاده سازي الگوريتم:

مانند حالت قبلی است تنها با این تفاوت که در انتخاب والد برای حالت قبلی تنها از جمعیت فرزندان استفاده می کردیم ولی اینجا از **بین جمعیت والد های قبلی و فرزندان** استفاده می کنیم.

تابع f:

برای ارزیابی شایستگی هر یک از افراد هست. یک ورودی که اینجا مجموعه ای از بردار ها هست، دریافت می کند و شایستگی آن را برمی گرداند.

تابع selection :

این تابع تفاوت بین دو حالت سوال را اجرا می کند. با دریافت ورودی از کاربر مشخص می کند که کدام حالت اجرا شود.

تابع random_parent :

با این تابع به طور رندوم دو والد ایجاد می‌کنیم برای مقداردهی اولیه تا برنامه و الگوریتم را بتوانیم با آن پیش ببریم.

تابع recombination :

این تابع برای ایجاد فرزند جدید به کمک Global discrete می باشد از دو والد یک فرزند تولید می‌کند. بعد از ایجاد فرزند، به تعداد lambda آرایه آن را به تابع جهش ارسال می‌کند و جهش را نیز روی آنها انجام می‌دهد.

تابع gaussian_mutation:

این تابع یک بردار از ورودی می گیرد به عنوان بردار فرزند و روی آن بردار عمل جهش را انجام می‌دهد. چون مقدار سیگما در هر جهش مقدار تغییر می‌کند از ورودی می گیرد و آن را تغییر می‌دهد و بر اساس سیگما جدید جهش را انجام می‌دهد.

تابع evolutionary_strategy :

این تابع فراخوانی توابع و قسمت های مختلف الگوریتم هست.

بررسی خروجی :

سرعت رسیدن به ورودی بهینه بالا هست در حالت دوم یعنی وقتی انتخاب بهترین افراد از بین والد و فرزندان تولید شده باشد زیرا که کمک می‌کند در صورتی که والد از فرزندان تولید شده بهتر بود حفظ شود و دور ریخته نشود.