دانشگاه خواجه نصیرالدین طوسی

هوش مصنوعی و سیستم‌های خبره

استاد زینب نخعی

پیاده‌سازی مسئله N-QUEEN به کمک GA

فروردین 1401

برای اعمال الگوریتم GA نیاز است که هر حالت صفحه بازی را به یک ورودی معتبر تبدیل کنیم. در الگوریتم Genetic هر individual در population همانند دنباله‌ای از DNA ها، یک رشته از تعداد محدودی الفبا است. در این پیاده‌سازی موقعیت مکانی وزیر ها در صفحه شطرنج با رشته‌ای از اعداد مدل شده است بدین ترتیب ایندکس هر عدد در رشته شماره سطر و خود عدد شماره ستونی که وزیر در آن قرار دارد را نمایش می‌دهد. (شماره‌گذاری از سمت چپ بالا است.) در این پیاده‌سازی رشته ها به صورت لیستی از اعداد نمایش داده می‌شوند. باید به این نکته اشاره شود که در صفحه شطرنج N*N که می‌خواهیم N وزیر را قرار دهیم طول لیست N و ایندکس‌ها از 0 تا N-1 هستند. همچنین اعداد می توانند 0 تا N-1 باشند.

برای مثال صفحه شطرنج زیر را در نظر بگیرید:

x x Q x

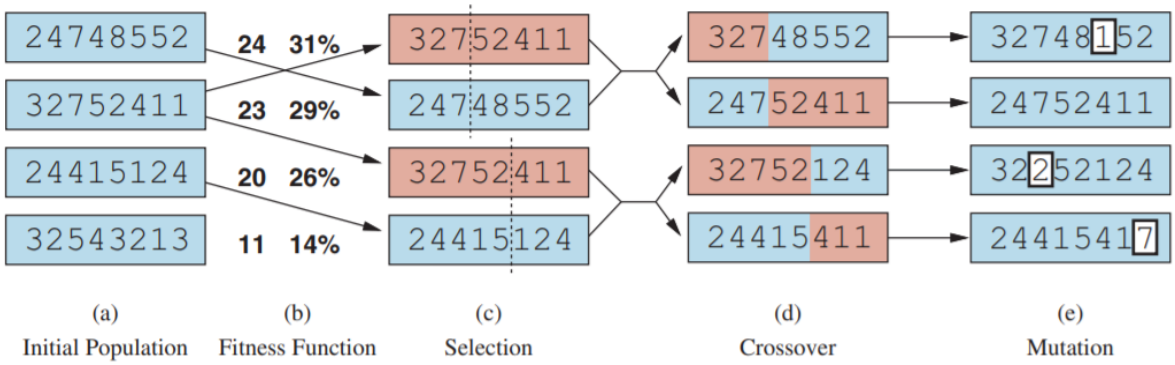
Q x x x

x x x Q

x Q x x

این صفحه به صورت [1, 3, 0, 2] مدل می‌شود.

مراحل الگوریتم Genetic به ترتیب زیر است:



Initial Population

متغیر POPULATION_SIZE سایز population را تعیین می کند و جز متغیر هایی است که با تغییر آن باید به مقدار بهینه دست یافت. در ابتدا متغیر population را با individual های رندوم مقدار دهی اولیه می‌کنیم. برای مثال برای صفحه شطرنج 8*8 و POPULATION_SIZE= 100 به متغیر population به تعداد 100 تا لیست با سایز 8 اضافه می‌کنیم که اعداد هر لیست مقداری رندوم بین 0 تا 7 دارند.

Fitness Function

برای محاسبه میزان fitness هر individual یا chromosome که نشان دهنده موقعیت مکانی وزیرها در صفحه شطرنج است، تعداد جفت وزیرانی که یکدیگر را تهدید نمی کنند می شماریم. وزیرها به صورت ستونی، سطری و یا اریب ممکن است یکدیگر را تهدید کنند. بنابراین اگر دو وزیر هیچ یک از این شرایط را نداشته باشند قابل قبول هستند و fitness score یکی زیاد می‌شود.

- **تهدید ستونی:** با توجه به شیوه مدل کردن صفحه شطرنج به صورت یک لیست که ایندکس هر عدد در لیست شماره ستون یک وزیر را نشان می‌دهد و دریک ایندکس تنها یک عدد می‌تواند قرار گیرد پس امکان ندارد دو وزیر یکدیگر را به صورت ستونی تهدید کنند.
- **تهدید سطری:** اگر دو وزیر یکدیگر را به صورت سطری تهدید کنند بدین معنا است که یک عدد در لیست دوبار تکرار شده است.
- **تهدید اریب به صورت /:** در این صورت جمع شماره سطر و ستون دو وزیر برابر است بنابراین اگر مجموع یک عدد با ایندکسش در لیست با مجموع یک عدد دیگر با ایندکسش برابر بود به صورت اریب یکدیگر را تهدید می‌کنند. برای مثال در [0, 2, 1, 3] وزیری که در سطر صفرم و ستون سوم (3=0+3) قرار دارد با وزیری که در سطر سوم و ستون صفرم (3=3+0) قرار دارد یکدیگر را تهدید می‌کنند.
- **تهدید اریب به صورت \:** در این صورت تفاضل شماره سطر و ستون دو وزیر برابر است بنابراین اگر تفاضل یک عدد با ایندکسش در لیست با تفاضل یک عدد دیگر با ایندکسش برابر بود به صورت اریب یکدیگر را تهدید می‌کنند. برای مثال در [0, 2, 1, 3] وزیری که در سطر اول و ستون اول (0=1-1) با وزیری که در سطر دوم و ستون دوم (0=2-2) قرار دارد یکدیگر را تهدید می‌کنند.

برای محاسبه میزان fitness یک chromosome وضعیت تک تک وزیرها را نسبت به سایر وزیرها بررسی می‌کنیم. در آخر از آنجایی که اگر وزیر x وزیر y را تهدید نکند، وزیر y نیز وزیر x را تهدید نمی‌کند اما ما می‌خواهیم هر دو وزیر که یکدیگر را تهدید نمی‌کنند یکبار بشماریم score را تقسیم بر 2 می‌کنیم. پیاده‌سازی این بخش در تابع fitness_score آورده شده است.

برای اینکه مسئله حل شود fintness_score باید ماکزیمم باشد که مقدار ماکزیمم آن برابر انتخاب 2 از N است.

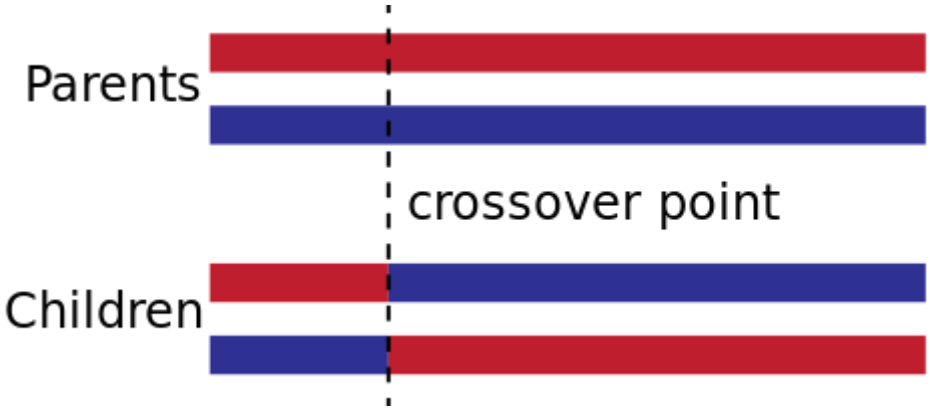
MAX_FITNESS = c(n, 2) = NUM_QUEENS * (NUM_QUEENS - 1)/2

K Tournament Selection

در این مرحله تعدادی از chromosome ها را برای نسل بعد برمی‌گزینیم. برای این کار از روش K tournament selection استفاده شده است که در آن به صورت رندوم K تا chromosome از population انتخاب می کنیم و از بین آن ها، آنکه fitness score بیشتری دارد را برمی‌گزینیم. پیاده‌سازی این مرحله در تابع K_tournament_selection آورده شده است.

Crossover

به کمک K_tournament_selection دو chromosome برمی‌گزینیم و آن دو را به مرحله crossover منتقل می‌کنیم. در این مرحله با احتمال CROSSOVER_RATE دو chromosome را ترکیب می‌کنیم در غیر این صورت خودشان به نسل بعد منتقل می‌شوند. برای اینکار یک عدد رندوم x بین 0 تا N-1 انتخاب می‌کنیم که N در اینجا تعداد وزیرها یا همان طول لیست chromosome است و به کمک آن دو offspring تولید می‌کنیم، یکی ترکیب موقعیت مکانی 0 تا x-1 وزیر در chromosome اول با موقعیت مکانی x تا N-1 وزیر در chromosome دوم است و دیگری ترکیب موقعیت مکانی 0 تا x-1 وزیر در chromosome دوم با موقعیت مکانی x تا N-1 وزیر در chromosome اول است.



Mutation

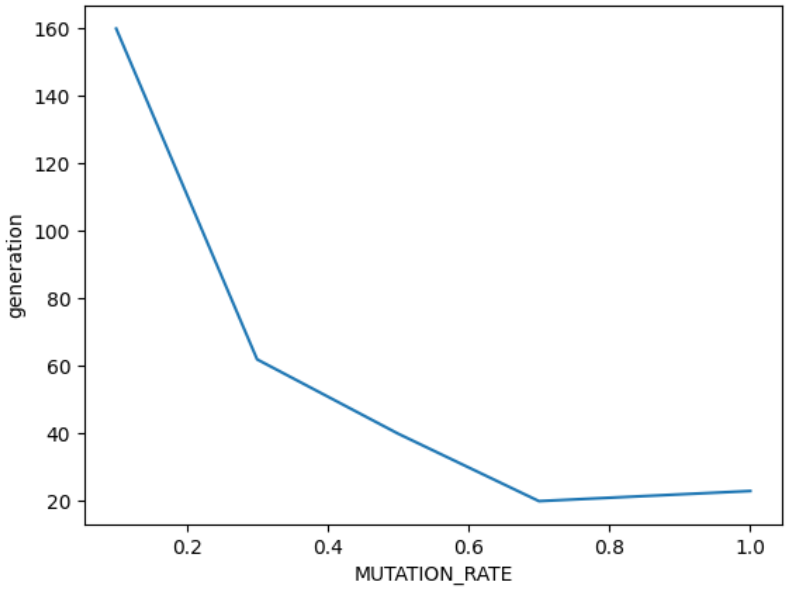
پس از مراحل بالا هر chromosome با احتمال MUTATION_RATE دچار جهش می‌شوند و سطر موقعیت مکانی یکی از وزیرها به صورت رندوم تغییر می‌کند و مقدار رندومی بین 0 تا N-1 می‌گیرد.

برای پیاده‌سازی الگوریتم ابتدا مرحله initial population انجام می‌شود و fitness score آن‌ها محاسبه می‌شود. سپس در حلقه به ترتیب، یک population جدید خالی ایجاد می‌شود و به تعداد POPULATION_SIZE مراحل زیر تکرار می‌شود:دو والد متفاوت(والد دوم انقدر عوض می‌شود تا با والد اول متفاوت باشد) از population به کمک K tournament selection انتخاب می‌شوند سپس روی آن‌ها با احتمال CROSSOVER_RATE عملیات crossover انجام می‌شود و سپس بر روی نتیجه آن عمل mutation با احتمال MUTATION_RATE انجام می‌شود. و اگر chromosome های تولید شده در population جدید

موجود نبودند به آن اضافه می‌شوند. چون در هر بار ماکزیمم دو chromosome جدید تولید می‌شود پس از POPULATION_SIZE بار تکرار این عمل ماکزیمم POPULATION_SIZE*2 تا chromosome در population جدید خواهیم داشت. بنابراین آن را بر اساس fitness score مرتب کرده و POPULATION_SIZE تا chromosome که دارای fitness score بیشتری هستند را نگه می‌داریم و با population قبلی جایگزین می‌کنیم و fitness score آن‌ها را محاسبه می‌کنیم. این کار را آنقدر تکرار می‌کنیم تا زمانی که به MAX_FITNESS برسیم و یا زمان اجرا برنامه از ۲۰ ثانیه بیشتر شود. در آخر آرایش پیدا شده را چاپ می‌کنیم.

تاثیر MUTATION RATE

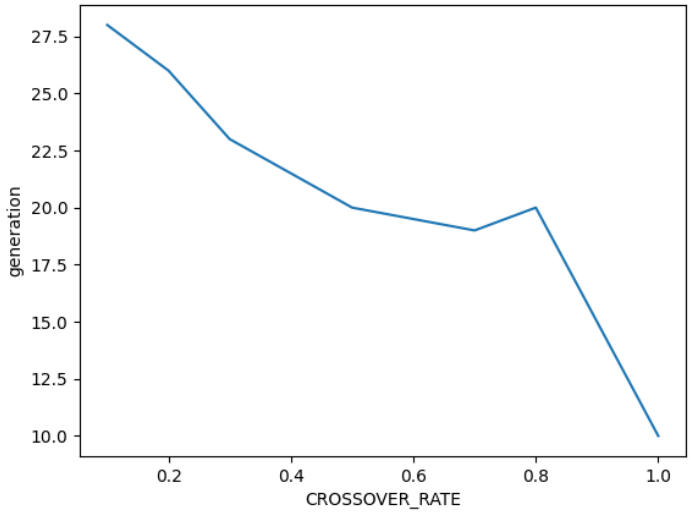
در این مدل‌سازی افزایش سرعت حل مسئله با افزایش این مقدار تا ۰.۸ مشاهده می‌شود. البته باید توجه داشت که در این مدل‌سازی این احتمال حتی اگر یک باشد تنها یک ژن در chromosome دچار تغییر می‌شود اما در مدل‌سازی هایی که این عدد احتمال تغییر هر ژن در chromosome را تعیین می‌کند با افزایش آن از یک مقداری به بعد موجب می‌شود الگوریتم به یک الگوریتم رندوم تبدیل شود و خاصیت خود را از دست دهد بنابراین در این مدل‌سازی ها باید این مقدار را کم نگه داریم.



تاثیر CROSSOVER RATE

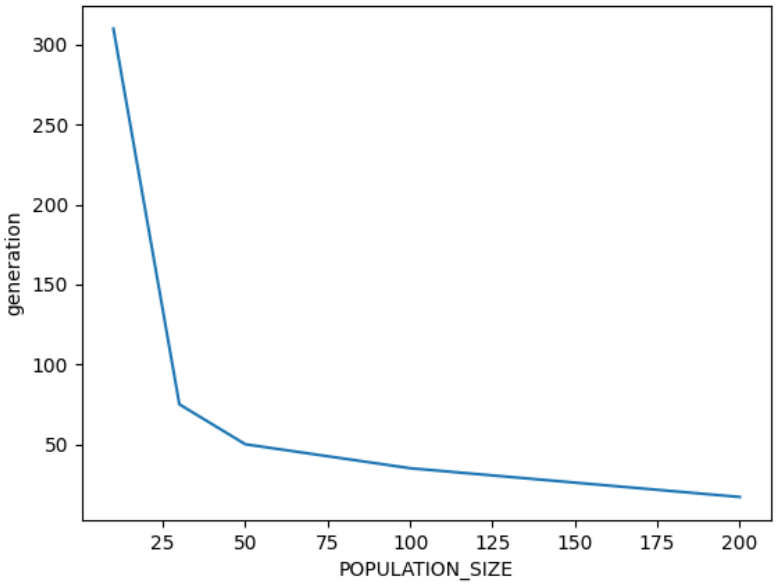
با افزایش این مقدار افزایش صعودی سرعت حل مسئله مشاهده می‌شود زیرا با افزایش مقدار chromosome های بیشتری ترکیب شده و فرزندان آن‌ها به نسل بعد منتقل می‌شوند و احتمال رسیدن به جواب صحیح را افزایش می‌دهد. در حالی که با کاهش آن تفاوت

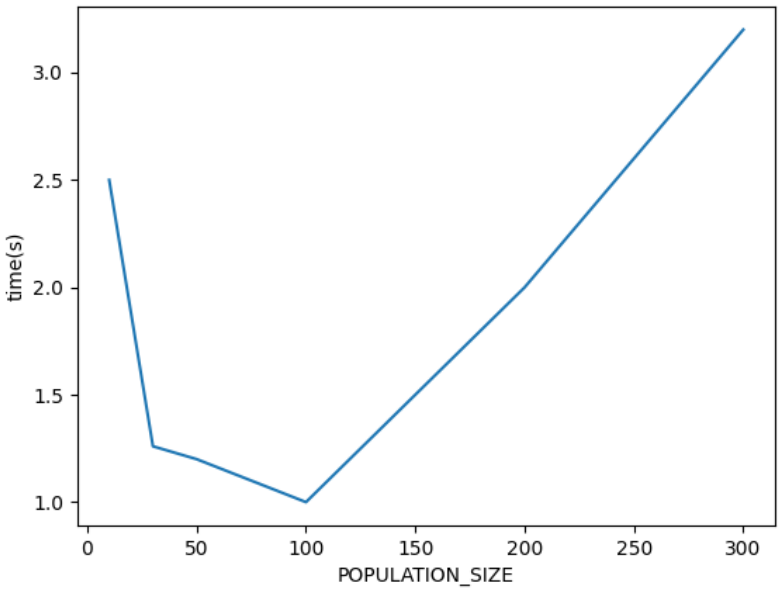
نسل قبل و بعد بسیار کم خواهد بود به طوری که با صفر شدن آن نسل بعدی همان نسل قبلی هستند مگر با mutation تعدادی از آن‌ها دچار جهش شده باشند.



تاثیر POPULATION SIZE

با افزایش این مقدار در نسل های پایین تری به جواب می‌رسیم اما از طرفی محاسباتمان بیشتر می‌شود به همین دلیل باید در این trade off به مقدار بهیه این متغیر دست پیدا کرد که هم در نسل های نسبتا پایین به جواب رسید و هم زمان اجرای برنامه خیلی زیاد نشود.


300
250
200
150
100
50
generation
25
50
75
100
125
150
175
200
POPULATION_SIZE


3.0
2.5
2.0
1.5
1.0
time(s)
0
50
100
150
200
250
300
POPULATION_SIZE