بسم الله الرحمن الرحیم

گزارش پروژه اول درس هوش مصنوعی

پیاده سازی مسئله n وزیر به کمک الگوریتم ژنتیک

ترم دوم سال تحصیلی 1399 – 1400

اعضای گروه: محمدحسین اصغری(9623003)، عرفان بهرامی(9624513)، محمدجواد طاهری کیا(9631063)

مقدمه

کد cpp شامل چندین فایل و توابع مهم است که در بخش "تشریح جز به جز فایل های کد استفاده شده در پروژه" قرار گرفته است. در ابتدا به تشریح کدهایی که برای پیاده سازی حل مسئله ی n وزیر با استفاده از الگوریتم ژنتیک، می پردازیم.

ورودی

در ورودی عدد n به عنوان تعداد سطر و ستون های صفحه ی شطرنج از کاربر گرفته می شود.

خروجی

در صورت پیدا شدن جواب، اعداد ستون قرار گیری وزیر ها در صفحه ی شطرنج به عنوان جواب نمایش داده می شود.

تابع fitness

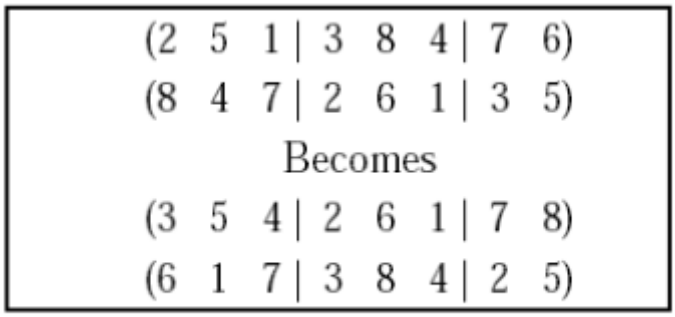
الگوریتم ژنتیک قصد در maximum سازی مقدار خروجی این تابع دارد. در ابتدا تعداد تهدید های ستونی وزیر ها را بد ست می آورد و سپس به سراغ تهدیدات قطری می رود. در نظر داشته با شیم که در بدترین حالت که تمامی وزیر ها یکدیگر را تهدید می کنند ما n(n-1)/2 تهدید خواهیم داشت. تابع fitness مقدار حاصل جمع تهدیدات ستونی و قطری را از کل تهدیدات ممکنه در بدترین حالت (max_fitness) کم می کند و به عنوان خروجی بر میگرداند.

تابع mutation و Cross over

با توجه به اینکه در هر خانه از کروموزم عدد قرار گیری ستون وزیر قرار گرفته است، برای mutation از swap استفاده کردیم تا حتما عدد جهش یافته از عدد قبلی متفاوت باشد. فرایند mutation بعد از cross over انجام می شود. در Corss over دو بار عمل برش انجام میشود و جابجایی ها برای ژن های فرزند انجام میگیرد و سپس نرمال سازی ژن ها صورت میگیرد. به عبارتی از PMX crossover استفاده کردیم.

(2 5 1 | 3 8 4 | 7 6)
(8 4 7 | 2 6 1 | 3 5)
Becomes
(2 5 1 | 2 6 1 | 7 6)
(8 4 7 | 3 8 4 | 3 5)

PMX crossover – first step



PMX crossover – second step

تابع main

برای بهتر یافتن جواب ها و تست مقادیر مختلف برای pMut (احتمال رخداد جهش) و pCross (احتمال رخ داد عمل cross over)، کل فرایند تولید نسل و آمار گیری از آن به تعداد مشخصی با مقادیر احتمالات تصادفی انجام می شود. قاعدتا با توجه به حجم هر نسل که برابر 100 عدد می باشد پس از چندین بار تولید نسل میبایست حافظه برنامه را آزاد کنیم. لذا برای performance بهتر تابعی تحت عنوان free_pop که وظیفه آزاد سازی حافظه ی نسل های قبلی را بر عهده دارد پیاده سازی شده است.

جواب برای n =15 را در تصویر زیر به همراه زمان محاسبه آن مشاهده می کنید.

```
p_cross: 0.092049, p_mut:0.119823

[+] Solution founded!
(one of them is: )
[6 8 5 7 15 10 14 1 3 9 12 2 4 11 13 ]

--------------------------------
Process exited after 21.2 seconds with return value 0
Press any key to continue . . .
```

تشریح جزء به جزء فایل های کد استفاده شده در پروژه

بخش اول: فایل eval

- تابع eval
  - این تابع جمعیت مورد بررسی را به همراه بیشینه fitness به عنوان مهم ترین ورودی های خود میگیرد و با احتساب تفاضل حاصل جمع تهدیدهای قطری و سطری و بیشینه fitness، مقدار ارزیابی شده را خروجی میدهد.

- سایر توابع این بخش، پردازش داده ها را تسهیل کرده و اطلاعات را برای بقیه قسمت ها آماده میکنند.

بخش دوم: فایل gen

- در این فایل تابع generation پیاده سازی شده است.
  - از آنجا که این قسمت توسط خود استاد پیاده سازی شده است؛ گزارشی مختصر در رابطه با آن، صرفا جهت اثبات فهم پذیری این فایل توسط ارائه دهندگان تشریح میشود. این قسمت وظیفه تولید جمعیت برای هر بار تکرار شدن الگوریتم را دارد؛ چرا که ما باید بدانیم چه نسلی از جمعیت آماده تاثیر پذیرفتن از عملگر های cross over و mutation هستند.

بخش سوم: فایل random

- اگرچه این قسمت نیز توسط مدرس پیاده سازی شده است؛ ولی برای عدم ایجاد اخلال در فرآیند گزارش توضیح مختصری داده میشود. کارکرد این فایل، تولید اعداد تصادفی برای استفاده در فایل های دیگر است. در واقع شانس تصادفی حضور اعضای جمعیت توسط این فایل باید ایجاد و بررسی شود. کارکرد دیگر این فایل آن است که ضمن تاثیر از ورودی seed random number، یک عدد تصادفی را برای تولید جمعیت تصادفی کروموزم ها به کار میگیرد.

بخش سوم: فایل rep

- عمده وظیفه این فایل، گزارش نتایج حاصل از اجرای الگوریتم به کاربر است. در واقع توسط توابع این فایل، خروجی و نیز تجربیات سیستم به صورت جزء به جزء در اختیار اجراکننده قرار میگیرد.

بخش چهارم: فایل select

- در این فایل، چرخ رولت پیاده سازی شده است. از آنجا که از ما انتظار نخواهد رفت فایل های آماده توسط استاد را بررسی کنیم؛ به توضیحی مجمل اکتفا خواهیم نمود.
  - منطق چرخ رولت: در مکانیزم چرخ رولت، هر یک از کروموزوم ها بسته به میزان مناسب بودنشان (بر اساس تابع برازش) احتمال انتخاب شدن را دارند. به عبارت دیگر هر چه یک کروموزوم بهتر باشد؛ احتمال انتخاب شدنش برای تولید نسل بعدی بیشتر بوده و برعکس هر چه کروموزوم بدتر باشد، احتمال انتخاب شدن آن برای تولید نسل بعدی، کمتر خواهد بود.
  - شیوه پیاده سازی چرخ رولت: همانطور که در چرخه الگوریتم ژنتیک مشخص شده است؛ در گام دوم، همه کروموزوم ها ارزیابی میشوند؛ یعنی میدانیم که هر کروموزوم بر اساس

تابع برازش چه ارزشـی بدسـت آورده اسـت. با اسـتفاده از این مقادیر، ما میتوانیم احتمال انتخاب شدن هر کروموزوم را مشخص کنیم. این احتمال از فرمول زیر حاصل میشود:

Probability (chromosomes C) = Fitness (chromosomes C) / Sum Fitness (All chromosomes)

بخش پنجم: فایل utils

- در این فایل، گزارش خطا به صورت نمایش خروجی خطا در خروجی استاندارد انجام میشود.

بخش ششم: فایل xover

- در این فایل، کراس اور صـــورت خواهد گرفت. به کمک این فایل، فرزندان قادرند ژن های متفاوتی نسبت به والدین خود تولید کنند. در اینجا پارامتر احتمال ترکیب نیز مطرح خواهد بود. به این معنا که تنها p% از رشته ها یا کروموزوم های موجود توسط عملگر xover دستکاری خواهند شد.