الگوریتم ژنتیک

ملیکا مرافق ۸۱۰۱۹۷۵۸۱

هدف :

پیدا کردن دنباله ای از گیت ها(که در جدول درستی داده شده صدق کنند) به وسیله ی الگوریتم ژنتیک است .این الگوریتم مناسب فضاهای گسسته بزرگ می باشد .

توضیح پروژه :

باید دنباله ای از گیت های منطقی(and ,or,xor,nand,nor,xnor) را بیابیم که به صورت زیر است:

۲ ستون ابتدایی جدول درستی ورودی اولین گیت هستند و ورودی باقی گیت ها خروجی گیت قبل و اولین ستون استفاده نشده از جدول درستی است.

هدف این است که دنباله ای تولید شود به طوری که درجدول درستی داده شده صدق کنند.

فاز اول:

ژن : هر ژن یک عدد بین 0 ,5 است که نشان دهنده ی یکی از گیت های and ,or,xor,nand,nor,xnor.است

کروموزوم (Individual): هر کروموزوم به تعداد گیت های مدار ژن دارد و ژن nم نمایش دهنده ی گیت nم است.

فاز دوم:

به تعداد افراد لیستی N تایی از اعداد رندوم بین ۰ تا ۵ تولید می کنیم که N بیانگر تعداد ژن هاست.

فاز سوم:

معیار سازگاری را تعداد سطر هایی از جدول درستی در نظر گرفتیم که در مدار صدق می کنند.

فاز پنجم:

۱)معیار تناسب را تعداد سطر هایی از جدول درستی در نظر گرفتیم که در مدار صدق می کنند. هدف رسیدن به مداری است که تمام خانه های جدول در آن صدق کنند.(fitness = تعداد سطر های جدول درستی)

۲) روش انتخاب FPS چون افراد با fitness بیشتر با احتمال بیشتری انتخاب می شوند و احتمال به وجود آمدن فرزندانی با fitness بهتر بیشتر است.

Tournament : که زود تر به جواب بهتری(نسبت به روش قبل) می رسیم اما با گذشت زمان diversity کاهش می یابد

۳) در یک رویکرد mutation را بر روی تمام فرزندان و با احتمال pm بر روی هر ژن انجام دادیم .

در رویکردی دیگر با احتمال pm یک فرزند برای mutation انتخاب کردیم و یک بیت آن را به صورت رندوم عوض کردیم

در هر دو روش ژن انتخاب شده اگر and بود nand و اگر nor بود or ,...)

که درنهایت روش دوم به دلیل رسیدن به راه حل های بهتر انتخاب شد.(هر چند راه حل اول گوناگونی بیشتری داشت)

$\frac{1}{population\ size} < pm < \frac{1}{chromosome\ lenght}$ در ابتدا pm را نزدیک $\frac{1}{population\ size}$ قرار می دهیم و در طی نسل ها این مقدار کاهش می یابد تا به حدود $\frac{1}{chromosome\ lenght}$ برسد .(با گذشت زمان که کیفیت کلی بهتر شده mutation را کم می کنیم تا تغییر زیادی در ژن های کروموزوم ها ایجاد نشود).

Crossover:

1-point crossover و uniform crossover که از بین این دو در مجموع 1-point به دلیل دستیابی به جواب های بهتر انتخاب شد.

در صورت انتخاب Pc= 0.7 با احتمال 0.3 هر فرد عینا در نسل بعدی کپی می شود که باعث شده diversity با مرور زمان کاهش یابد.در نهایت Pc = 0.8 انتخاب شد.

۴) اگر راه حلی با fitness زیاد داشته باشیم احتمال انتخاب آن به شدت افزایش می یابد و راه حل هایی تولید می شوند که شباهت زیادی به این راه حل دارند بنابراین diversityرا کم می کند و ممکن است که در local maximum گیر کند و به جواب بهینه نرسیم . برای رفع این مشکل و افزایش diversity میتوانیم :

در هر step افراد رندوم را اضافه کنیم .

جمعیت اولیه را افزایش دهیم.(در تعداد بیشتر گوناگونی بیشتری داریم)

از روش انتخاب دیگری استفاده کنیم.(Ranked-based selection)

در اینجا از رندوم استفاده شده.

Hyper parameter:

نکته :در تمام حالات برای تعیین یک پارامتر ، باقی پارامتر ها و الگوریتم ها را ثابت در نظر گرفتیم.

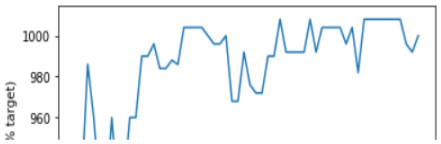
اندازه جمعیت :

ابتدا با جمعیت 50 شروع کردیم که پس از حدود ۱۰۰ سیکل local maximum مقدار زیادی از جمعیت را دربر گرفته بود .و گوناگونی به شدت کاهش پیدا کرده بود.سپس جمعیت را برابر ۱۰۰ قرار دادیم (.با افزایش جمعیت به ۲۰۰ با توجه به افزایش بیشتر جمعیت را به ۱۰۰ کاهش دادیم)

جمعیت بیشتر diversity را افزایش می دهد و همگرایی به جواب های بهتر زود تر صورت می گیرد(جمعیت کم diversity را خیلی زود و قبل از پیدا کردن جواب مناسب از دست می دهد.)

```
gen [5, 2, 0, 5, 4, 4, 3, 2, 4] fit 704
gen [1, 5, 0, 5, 3, 1, 0, 5, 4] fit 608
gen [5, 3, 2, 1, 1, 1, 0, 2, 4] fit 944
[3, 5, 4, 2, 0, 1, 0, 2, 4]
202.84081625938416
```

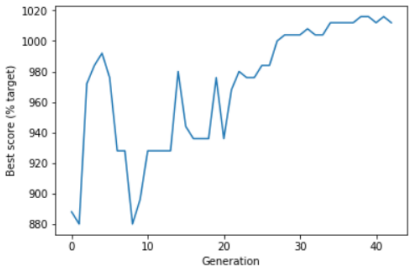
```
In [14]: plt.plot(best_score)
         plt.xlabel('Generation')
         plt.ylabel('Best score (% target)')
         plt.show()
```



pop_size = 100

```
gen [1, 0, 4, 2, 0, 1, 3, 5, 4] fit 1012
gen [5, 4, 1, 1, 3, 5, 2, 0, 4] fit 840
gen [1, 4, 5, 3, 0, 1, 1, 5, 4] fit 784
gen [2, 4, 5, 0, 2, 5, 4, 2, 4] fit 736
gen [0, 3, 4, 5, 5, 3, 0, 1, 4] fit 864
[0, 2, 4, 2, 0, 1, 3, 5, 4]
100.62244081497192
```

```
In [14]: plt.plot(best_score)
         plt.xlabel('Generation')
         plt.ylabel('Best score (% target)')
         plt.show()
```

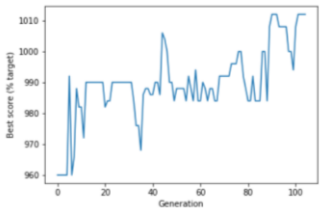


0: and - 1: or - 2: xor - 3: nand - 4: nor - 5: xnor

```
gen [3, 2, 4, 0, 0, 5, 3, 5, 4] fit 800
gen [1, 2, 2, 3, 4, 2, 3, 5, 4] fit 960
gen [3, 2, 4, 0, 2, 1, 3, 5, 4] fit 976

[0, 2, 4, 2, 0, 1, 3, 5, 4]
296.7786829471588
```

```
In [14]: plt.plot(best_score)
         plt.xlabel('Generation')
         plt.ylabel('Best score (% target)')
         plt.show()
```



pop_size = 120