«به نام خدا»

صحبتی نوشاد ۹۷۱۴۳۰۴۸ توضیح در مورد الگوریتم ژنتیک برای

مسئله‌ی ۹ وزیر:

برای حل با استفاده از الگوریتم ژنتیک مراحل زیر را انجام می‌دهیم:

۱) می‌آییم شروع می‌کنیم حالت‌ها را نمایش دادن، حالت‌ها را در قالب ژن‌ها کدشان می‌کنیم و این ژن‌ها را در کنار هم می‌چینیم و کروموزم‌ها را می‌سازیم. (Choromosome)

۲) یک جمعیت اولیه تولید می‌کنم (population) بعد آن جمعیت اولیه به صورت تصادفی تولید شده‌اند که K تا حالت هست. (K تا حالت اولیه تولید می‌کنم)

۳) بعد به ازای تک تک این حالت‌ها می‌آییم مقدار تابع fitness function را محاسبه می‌کنیم.

هرچه مقدار این بیشتر باشد حالت من بهتر خواهد بود و این عملاً ارزیابی حالت هست.

۴) عملیات Cross-over: می‌آییم دو تا حالت را در نظر می‌گیرد و از ترکیب این دو تا یک حالت جدید می‌سازد.

۵) عملیات Mutation: عملاً جهش ژنتیکی انجام می‌دهد یعنی اینکه این نمادهای داخل این رشته تغییر کند و برای این هست که تنوع را حفظ کنیم.

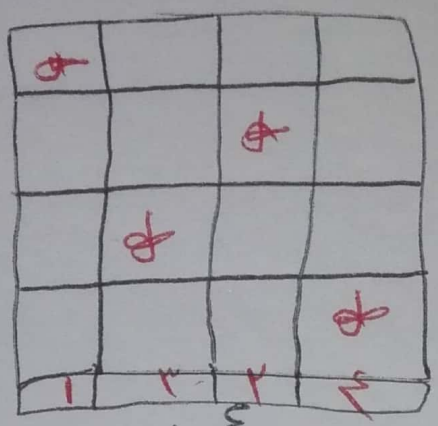
۶) می‌آییم از بین این حالت‌ها بهترین را انتخاب می‌کنم و نسل بعدی را می‌سازم که می‌شود بهترین نسل‌هایی که از قبل داشتم.

* (در تکامل آن موجودیت‌هایی به نسل بعد می‌روند که fitness آن‌ها بهتر باشد) *

Scanned by CamScanner

مثلاً برای مسئله‌ی ۴-وزیر به صورت زیر عمل می‌کنیم:

هرکدام از state ها را به عنوان یک رشته کد کنیم. یکی از ساده‌ترین کدگذاری‌ها این است که در هر ستون بگیم این وزیر در خانه‌ی شماره چند هست، چون اینجا برای ۴ وزیر هست رشته‌ای از طول ۴، که مقدارش integer هست را در نظر می‌گیریم.



۴ وزیر

یک آرایه تک بعدی ایجاد می‌کنیم که به تعداد ستون‌های صفحه شطرنج فضا دارد.

- جمعیت هر نسل می‌تواند تعداد کروموزوم‌ها را تعیین کند.
- جمعیت اولیه از انتخاب رندومی از کروموزوم‌ها ایجاد می‌شود.
- الگوریتم ژنتیک ابتدا باید جمعیت اولیه را تولید کرده و سپس سعی در بهبود این جمعیت کند.

برای مسئله‌ی ۸ وزیر جمعیت به صورت تصادفی تولید می‌شود و وزیرها به صورت تصادفی روی صفحه شطرنج قرار می‌گیرند.

Scanned by CamScanner

```
random_chromosome = lambda size: [random.randint(1,size) for _ in range(size)]
random_chromosome.__doc__='making random chromosomes'
```

Scanned by CamScanner

تابع fitness: برای محاسبه تعداد جفت‌های وزیر که به هم کارد می‌دهند تعریف می‌کنیم.

ما باید تابع Fitness را برای همه‌ی افراد جمعیت (کروموزوم‌ها) محاسبه کنیم.

Scanned by CamScanner

```python
def fitness(chromosome,maxFitness=None):
    n = len(chromosome)
    if maxFitness==None:
        maxFitness=(n*(n-1))/2
    horizontal_collisions = sum([chromosome.count(queen)-1 for queen in chromosome])/2

    left_diagonal=[0]*(2*n)
    right_diagonal=[0]*(2*n)
    for index,chrom in enumerate(chromosome):
        left_diagonal[index+chrom-1]+=1
        right_diagonal[n-index+chrom-2]+=1

    diagonal_collisions = 0
    for i in range(2*n-1):
        counter = 0
        if left_diagonal[i]  > 1: counter += left_diagonal[i]-1
        if right_diagonal[i] > 1: counter += right_diagonal[i]-1
        diagonal_collisions+= counter / (n-abs(i-n+1))

    return int(maxFitness-(horizontal_collisions + diagonal_collisions))
```

Scanned by CamScanner

سپس باید احتمال انتخاب از تابع fitness را محاسبه کنیم .

Scanned by CamScanner

```
probability=lambda chromosome, fitness,maxFitness=1: fitness(chromosome)/maxFitness
```

Scanned by CamScanner

در مرحله‌ی بعد، به طور تصادفی دو جفت را برای تولید مثل براساس حالتی که روی مرحله‌ی قبل حساب کردیم انتخاب می‌کنیم.

تابع انتخاب تصادفی ما:

Scanned by CamScanner

```python
def random_pick(population, probabilities):
    # tmp={tuple(i):j for i,j in zip(population, probabilities)}
    # return list(max(tmp,key=lambda x:tmp[x]))
    r = sum(probabilities)*random.random()
    upto = 0
    for c, w in zip(population, probabilities):
        upto+=w
        if upto>=r: return c
    raise RuntimeError("This is unreachable state :(")
```

Scanned by CamScanner

Crossover : یک نقطه تصادفی را در طول رشته انتخاب می‌شود و حالت های جدیدی با استفاده از آن ساخته می‌شود.

هر فرزند با انتخاب تکه اول از یکی از والدین و تکه دوم از والد دیگر به وجود می‌آید.

Scanned by CamScanner

```python
def reproduce(x, y):
    assert len(x)==len(y)
    '''doing cross_over between two chromosomes'''
    c = random.randint(0,len(x)-1)
    return x[:c]+y[c:]
```

Scanned by CamScanner

مرحله‌ی بعد Mutation (جهش):

به صورت تصادفی چند ژن را تغییر می‌دهیم و کروموزم های جدید به دست آمده جمعیت جدید ما را خواهند ساخت.

Scanned by CamScanner

```
def mutate(x):
    '''randomly changing the value of a random index of a chromosome'''
    n = len(x)
    x[random.randint(0,n-1)]=random.randint(1,n)
    return x
```

Scanned by CamScanner

تابع queen_genetic :

ابتدا احتمالات را بااستفاده از تابع probability به دست می اورد.

سپس در یک حلقه For بااستفاده از تابع pick_random دو تا از بهترین کروموزم ها را انتخاب کرده .بعد بااستفاده از تابع reproduceدو کروموزوم جدید ایجاد میشود .

در مرحله بعدی یک عدد تصادفی ایجاد کرده با استفاده از random.random()ان را با mutationProbability مقایسه میکنیم .

اگر mutationProbability کمتر از عدد تصدفی ایجاده شده بود , فرزندان ایجاد شده را به تابع mutate جهش میدهیم سپس انها را در لیست population_new قرار میدهیم . این تابع تا زمانی ادامه پیدا میکند که fitness کروموزوم ها به حداکثر خود برسد یا تمام جمعیت را پیمایش کند .در اخر جمعیت جدید را برمیگرداند

```
def genetic_queen(population, fitness,maxfitness,mutationProbability = 0.03):
    new_population = []
    probabilities = [probability(n, fitness,maxfitness) for n in population]
    probabilities = [probability(n, fitness,maxfitness) for n in population]
    for _ in population:
        x = random_pick(population, probabilities) #best chromosome 1
        y = random_pick(population, probabilities) #best chromosome 2
        child = reproduce(x, y) #creating two new chromosomes from the best 2 chromosomes
        if random.random() < mutationProbability:
            child = mutate(child)
        new_population.append(child)
        if fitness(child) == maxfitness: break
    return new_population
```

Scanned by CamScanner

```
- - Q -
Q - - -
- - - Q
- Q - -

solution found after 1 generations
 Q =queens  and  - =hoom empty
 sample output for 4 queens
```

Scanned by CamScanner

```
Q - - - -
- - Q - -
- - - - Q
- Q - - -
- - - Q -

solution found after 2 generations
 Q =queens  and  - =hoom empty
 sample output for 5 queens
```

Scanned by CamScanner