Genetic Algorithms

حل مسائل به کمک الگوریتم های ژنتیک

نویسنده مقاله : آرش گرامی فرد

arash_gramifard@yahoo.com

چکیده مقاله :

الگوریتم های ژنتیک با ایده گرفتن از طبیعت وروند تکاملی رشد کردند ویکی از مهمترین کاربرد های ان درمسائل Tsp می باشد.
تا به حال ایده های فراوانی در این زمینه داده شده است در اینجا به چند نمونه ازچگونگی ترکیب این ایده ها ومشکلات انها اشاره خواهد شد.

هر الگوریتم ژنتیک اصولا از سه پروسه اساسی تشکیل میشود :

۱.ازدواج Recombination
۲.جهش ژنی Mutation
۳.انتخاب مجدد Selection

که هر کدام از این پروسه ها میتواند انواع وشکل های متفاوتی داشته باشند که در ادامه بیشتر با انها آشنا خواهید شد.

کلمات کلیدی :

جهش ژنی-الگوریتم ژنتیک- کروموزم- ژن
Genetic algorithm-Recombination-Mutation-Selection-Roulette wheel-tournament crossover-Chromosome-jene

مقدمه :

در بیشتر کلاس های درسی ،آموزش به صورت تئوریک می باشدو من پس از گذراندن برنامه نویسی همروند به این فکر افتادم که روند حل مسائل از طریق الگوریتم های ژنتیک رادر عمل ببینم و ان را در قالب یک برنامه پیاده سازی کنم و در این راه نقاط قوت و ضعف تکنیک های مختلف و چگونگی ترکیب روشهای مختلف را با هم بیازمایم.

الگوریتم های ژنتیک :

این الگوریتم ها مبتنی برروند تکاملی می باشند که بر گرفته ازنظام طبیعت است.نسلهای موجودات قوی تر بیشتر زندگی می کنند ونسلهای بعدی نیز قوی تر می شوند یا به عبارت دیگر میتوان گفت : طبیعت افراد قوی تر(شایسته تر) را برای زندگی بر می گزیند. حال با چنین ایده ای می توان به نحوی مسئله را شبیه سازی کامپیوتری کرد.
مزیت الگوریتم های ژنتیکی در این است که برای مسائلی که جواب قطعی وجود ندارد ویا نمیتوان از راه حل های معمول به جواب رسید طی یک روند تکاملی مجموعه ای از نزدیکترین جواب ها را به بهترین جواب را نشان خواهد داد .

فرم کلی الگوریتم های ژنتیک :

```
Main( )
Initialize and Evaluate  population of size  M          ایجاد نسل اولیه
      While (not stop condition)
            Select parents from population               انتخاب والدها
            Recombine parents to produce L offspring     ترکیب مجدد والد ها
            Mutate each of the L offspring               جهش ژنی
            Select new population of size M again        انتخاب مجدد نسل جدید
       End while
End
```

حال مسئله فروشنده دوره گرد را تبدیل به یک مسئله ژنتیک می کنیم.

مسئله فروشنده دوره گرد(TSP (traveler sales man problem :

این مسئله به این شکل است که یک فروشنده دوره گرد می خواهد از تعدادی شهر مشخص رد بشود وبه شهر خود باز گردد به نحوی که طول مسیر کمترین شود.

برای راحت تر نشان دادن مسئله میتوان شهر ها را روی یک صفحه XY رسم کرد و فاصله مختصاتی را به عنوان فاصله شهر ها در نظر گرفت.

D=((x(city1)-x(city2)) ^ 2 + (y(city1)+y(city2)) ^ 2) ^ 0.5

اگرچه با بسط دادن مسئله در فضای R3(XYZ) میتوان فاصله را برای یک فروشنده بین سیاره ای استفاده کرد.

D=((x(city1)-x(city2)) ^ 2 + (y(city1)+y(city2)) ^ 2 + (z(city1)-z(city2) ^ 2) ^ 0.5

حل این مسئله از طریق راه حل های معمول نیاز به چک کردن !(n-1)حالت دارد که از عهده کامپیوتر خارج است و مقرون به صرفه هم نیست پس از الگوریتم ژنتیک استفاده می شود.
در اینجا شهر ها ژن ها را تشکیل می دهند و جواب ها(مسیر ها) به شکل کروموزم نشان داده می شوند.

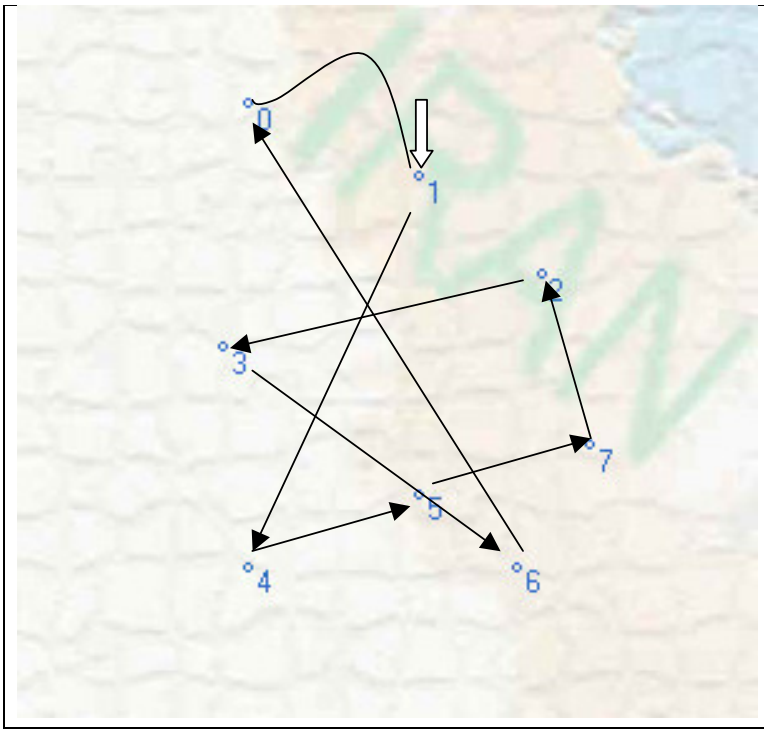
Make_cities( ) شهر ها را تولید می کند.
فاصله هر شهر را تا شهر دیگر را بدست اورده و مقادیر را در یک آرایه ذخیره می کند.

Evaluate_cities_distance( )

ایجاد نسل اولیه (Initialize and Evaluate population of size M)

کروموزوم های نسل اول را بوجود می اورد.

Make_chromosome( )




```
Function Make_chromosome( )
chromosome = chrom_no
Dim a As Integer
For i = 0 To chromosome
    For j = 0 To tedad
    Randomize
    a = Rnd() * tedad
    chorom(i, j) = a
Next
End Function
```

Sample chromosome

1	4	5	7	2	3	6	0

شکل (۱)

کروموزوم ها می توانند ناقص باشند به این مفهوم که به یکی از شهر ها نرفته باشد یا به یک شهر بیش از یک بار رفته باشد.
در برنامه می توان به شکلهای مختلف از ورود این کروموزوم هاجلوگیری کرد.

بعد از ساخت کرومزوم ها نوبت به تعیین شایستگی هر یک از آنها می رسد.

```
Evaluate_chromosome()
```

ومقادیر را در ارایه دیگری ذخیره می شود و با دادن شماره کروموزم میزان شایستگی برگردانده می شود.
شایستگی باید متناسب با جواب مسئله باشد.
در اینجا کوتاه بودن مسیر مهم است و رد شدن از تمامی شهر ها.

```
Function fitnes(r)
fit = 0
mtekrar = ?
```

در اینجا با این متغیر میزان اهمیت تکراری بودن شهرها مشخص می شود.

```
 mtoul = ?
```

و با این متغیر میزان اهمیت فاصله مشخص می شود.

```
er2 = 0
er = 0
```

محاسبه تعداد خطاها(تکراری بودن شهرها)

```
For e = 0 To tedad - 1
      er2 = er2 + er
       er = 0
      For t = e + 1 To tedad
     If crom2(r, e) = crom2(r, t) Then er = er + 1
 Next t, e

toul = 0
```

محاسبه طول مسیر

```
For e = 0 To tedad - 1
 p = crom2(r, e)
 p2 = crom2(r, e + 1)
 toul1 = (evxy(p, p2))
 toul = toul + toul1
next e
```

تبدیل مسیر به یک دور(وصل شدن ابتدا به انتها)

```
p3 = crom2(r, 0)
 toul = toul + evxy(p2, p3)
 fit2 = ((er2 * mtekrar + toul * mtoul) / 100) + 1
```

تا اینجا خطاها وفاصله ها محاسبه شده است وبا معکوس کردن ان یک عدد شایستگی وجود خواهد داشت که هر چه مسیر و خطا کمتر باشد شایستگی بیشتر است . این عدد بین ۰ و ۱ میباشد.

```
fit = 1 / fit2

 End Function
```

انتخاب والدها (Select Parents) :

باید دو کروموزم به عنوان والد انتخاب شود.

۱. Random selection

۲. Roulette wheel

۳. Ranking

۴. K-tournament selection

Random selection :

با استفاده از یک تابع تصادفی ۲ کروموزم انتخاب می شود.

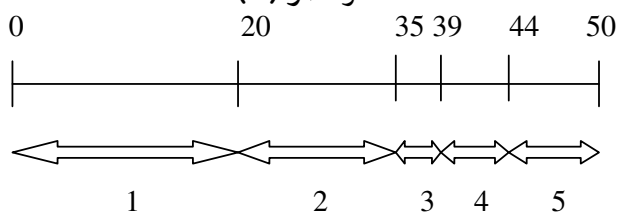
```
cr1 = Int(Rnd() * corom_no)
cr2 = Int(Rnd() * corom_no)
```

Roulette wheel :

در این روش ابتدا مجموع شایستگی ها را حساب کرده سپس یک عدد را به طور تصادفی بین صفر و مجموع شایستگی ها انتخاب می کند مثلا اگر کروموزم ها ای با شایستگی های زیر وجود داشته باشد فضای انتخاب چنین میشود.

Sum_of_fitness=50

نمودار (۱)



شماره کروموزم	۱	۲	۳	۴	۵
میزان شایستگی	۲۰	۱۵	۴	۵	۶

جدول (۱) Fitness table

```
Function select_parent( )
For i = 0 To corom_no
        Sum = Sum + evcrom(i)
        evrolet(i) = Sum
      Next i
      For k = 0 To corom_no
        pi = Rnd() * Sum
        pi2 = Rnd() * Sum
        ro2 = 100
        ro1 = 100
        While (pi > evrolet(ro1))
          ro1 = ro1 + 1
        Wend
        ro1 = ro1 - 1
       While (pi2 > evrolet(ro2))
          ro2 = ro2 + 1
        Wend
        ro2 = ro2 – 1
```

نتایج انتخاب در دو متغیر ro1,ro2 ذخیره می شوند.

```
End Function
```

با این شیوه احتمال انتخاب با میزان شایستگی نسبت مستقیم دارد.

: Ranking selection

کروموزم ها بر حسب شایستگی مرتب می شوند وبرنامه به ترتیب انتخاب را انجام می دهد.

: K-tournament selection

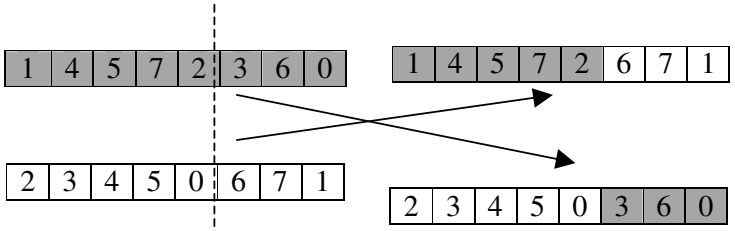
از بین k کروموزم تصادفی کروموزمی که شایستگی بیشتری دارد انتخاب می شود.

ازدواج یا ترکیب مجدد(Recombination) :

one point cross over.1
k-point cross over.2
advanced cross over.3

:One point cross over

دو کروموزم از یک نقطه شکسته شده وبا هم ترکیب می شوند.



شکل (۲)

نقطه شکست می تواند تصادفی باشد.
این روش برای بخش های پایانی (نسل های پیشرفته) بسیار ضعیف است.
چرا که کروموزم های ناقص تولید می کند.

```
findcross (0)
For i = 0 To cross
    crom2(k* 2), i) = crom(crit1, i)
Next i
jj = 0
For jj = cross + 1 To tedad
    crom2(k* 2, jj) = crom(cr2, jj)
Next jj
For i = 0 To cross
    crom2(k, i) = crom(cr2, i)
Next i
For joj = cross + 1 To tedad
  crom2(k, joj) = crom(crit1, joj)
Next joj
```

پس با هر ترکیب دو کروموزم جدید ایجاد می شود.

: K-point cross over

این شیوه نیز مشابه شیوه قبلی می باشدبا این تفاوت که نقاط شکست بیشتر از یک نقطه می باشد.

: Advanced cross over

این شیوه بسیار کارا می باشد و همواره کاربرد دارد.در این شیوه ژن اول رااز یکی از دو کروموزم برمی دارد و دو ژن بعدی ان را که در دو کرومزوم آمده است را پیدا کرده و ان ژنی که فاصله کمتری تا ژن ما دارد را به عنوان ژن بعدی کروموزم جدید انتخاب می کند.این عملیات تا کامل شدن کروموزم ادامه پیدا خواهد کرد.

نکته : این شیوه در سیستم هایی که کروموزم ناقص دارند قابل استفاده نمی باشد.

```
Cr1=?
Cr2=?
crom2(chorom_no+k, 0) = crom(cr1, 0)
p = crom2(cr1, 0)
        p2 = p
         n1 = findnext(cr1, p)
          n2 = findnext(cr2, p)

         For j = 0 To tedad - 1
             While ((check(chorom_no +k, j, n1)))
               n1 = findnext(cr1, p)
               p = n1
             Wend

             While ((check(chorom_no +k, j, n2)))
               n2 = findnext(cr2, p)
               p = n2
             Wend
               crom2(k+chrom_no, j + 1) = betterxy(p2, n1, n2)
         Next
```

City	0	1	2	3	4
0	0	18	12	20	13
1	18	0	25	16	32
2	12	25	0	14	55
3	20	16	14	0	30
4	13	32	55	30	0

جدول (۲) cities distance

در جدول بالا فاصله شهر ۲ تا شهر ۳ مقدار ۱۴ می باشد.
در زیر دو کروموزم دیده می شود که به روش پیشرفته با توجه به جدول بالا ترکیب می شوند.

0	1	2	3	4

4	2	1	3	0

نتیجه ی حاصل کروموزمی با مشخصات زیر می شود:

0	4	1	3	2

شکل (۳)

جهش ژنی(Mutation) :

Random mutation .1
Swap .2

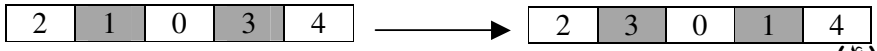
Random mutation :

یک راه این است که برنامه یک کروموزم را به طور اتفاقی انتخاب می کند(از کروموزم های فرزند نه والد)
سپس یک ژن ازآن را به طور تصادفی انتخاب می کند.
یک ژن دیگر به طور اتفاقی انتخاب کرده و جایگزین آن ژن می کند.
راه دیگر این است که تمام ژن های کروموزم ها را گذر می کند و در هر مقطع یک عدد تصادفی در نظرمی گیرد و اگر از یک عدد خاص بالا تر بود ان ژن کروموزم را به روش بالا جهش می دهد .حسن این روش در این است که با مشخص کردن عدد میتوان نشان داد که چند درصد ژن کروموزم ها جهش پیدامی کنند.
این دو شیوه معمولا موجب بوجود آمدن کروموزم های ناقص می شوند و در مقاطع پایانی مخرب می باشند.

```
For i = 0 To corom_no * 2
 For j = 0 To tedad
  Randomize
  Dim r As Integer
  Dim allel As Integer
  allel = Rnd() * tedad
  If (r > عدد خاص) Then crom2(i+corom_no, j) = allel
  Next j, i
```

Swap :

این شیوه بر خلاف شیوه قبل بسیار کاربردی است.بدین صورت که دو ژن را از یک کروموزم گرفته وجای ان دو را با هم عوض می کند . برای کارایی بیشتر میتوان بعد از جابجایی میزان شایستگی را در کروموزم قبلی و بعدی مقایسه کرد و اگر میزان شایستگی بهتر نشده بود دوباره کروموزم را به وضعیت قبلی برگرداند.



شکل (۴)

```
For i = corom_no To (corom_no * 3)
      Randomize
      c1 = Int(Rnd() * tedad)    :    c2 = Int(Rnd() * tedad)
      fitnes (i)
      fi1 = fit
      p = swap_c(i, c1, c2)
       fitnes (i)
       fi2 = fit
      If (fi1 > fi2) Then p = swap_c(i, c1, c2)
    Next i
```

انتخاب نسل جدید (Selection) :

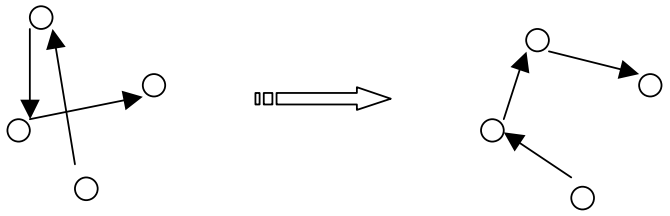
نحوه عمل انتخاب جهت تولید نسل جدید (به عنوان نسل اولیه در پروسه)همانند انتخاب والد ها می باشد با این تفاوت که تعداد دفعات آن برابر با تعداد کروموزم های نسل اولیه است.
نکته دیگری که قابل توجه است جلوگیری از Diverge شدن های ناگهانی می باشد چرا که عمل انتخاب در هر حال به صورت یک تابع احتمال است و امکان دارد که حتی بهترین کروموزم هم به نسل بعدی راه پیدا نکند بدین منظور در هر نسل می توان بهترین کروموزم را به صورت دستی به نسل بعد منتقل کرد.

شرط پایانی(Stop condition) :

یکی از راهها محدود کردن تعداد نسلها می باشد که میتوان مشخص کرد که برتامه تا چند نسل پیش برود.
راه دیگر گذاشتن محدودیت زمانی است .
و شاید بهترین راه چک کردن بهترین کروموزم هر نسل باشد که در صورت تکرار شدن آن به میزان k-times عملیات را تمام می کند.

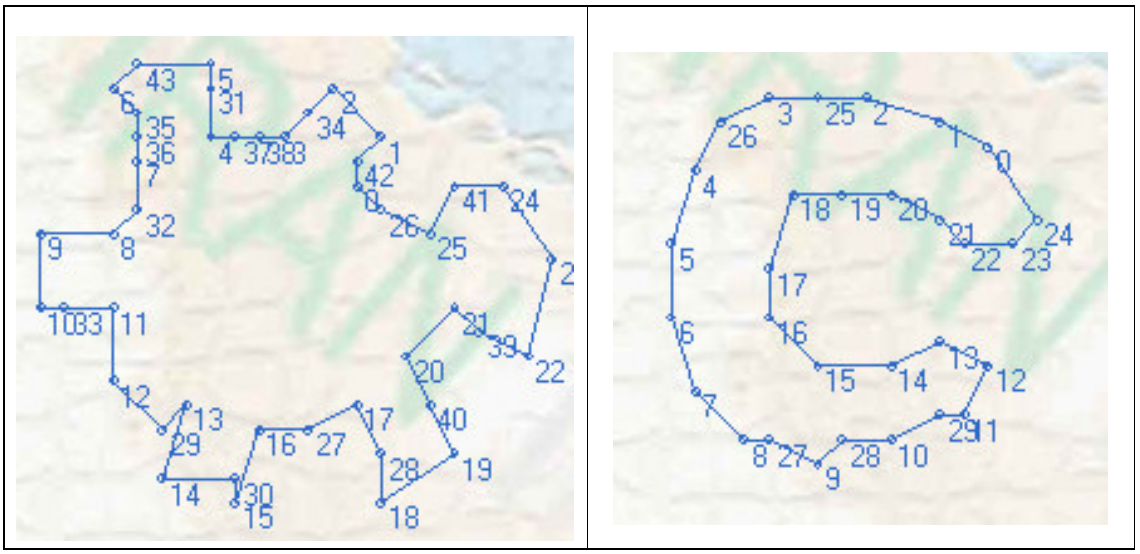
نکات دیگر در باره الگوریتم های ژنتیک:

القاء یک کروموزم خوب بطور دستی میتواند در رسیدن به جواب اصلی بسیار مفید واقع شود.
مثلا اضافه کردن یک کروموزم که از یک ژن شروع شده و به نزدیک ترین ژن بعدی که در کروموزم نیامده پیوند می خورد.
کار دیگری که میتوان در این زمینه انجام داد زمانی که بهترین کروموزم دیگر تغییری نمی کند میتوان یک موج جهش را روی ان اعمال کرد.
بدین صورت که یک دوره Swap روی ژنهای کنار هم انجام داد .
این شیوه موجب باز شدن گره ها می شود.

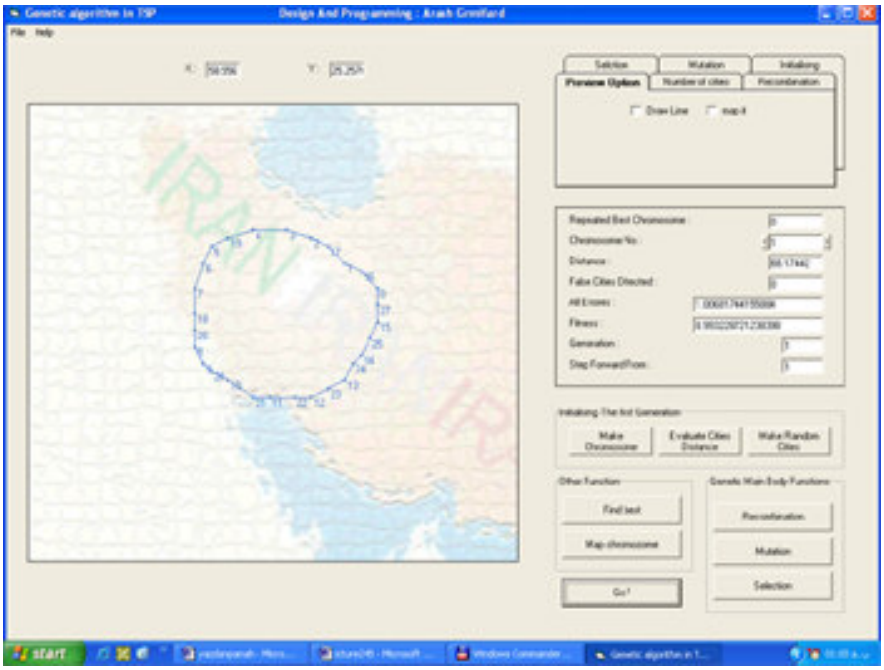


شکل (۵)

نتایج حاصل شده پس از اجرای برنامه:



شکل (۶)



یک نما از برنامه TSP که در حال حاضر موجود است

شکل (۷)

: Acknowlage

از استادم جناب آقای مهندس حفاری که من را در این تحقیق یاری کردند تشکر می کنم.

منابع (References) :

http://gaslab.cs.unr.edu/docs/techreports/gong/journal.html

http://www.generation5.org/content/2001/tspapp.asp

http://cs.felk.cvut.cz/~xobitko/ga/

http://cs.felk.cvut.cz/~xobitko/ga/cromu.html