Instituto Politécnico Nacional

ESC Superior de Cómputo

"EVOLUTIONARY COMPUTING" CIRCLE AND SQUARE PROBLEM

Jorge Luis Rosas Trigueros

Saldaña Aguilar Gabriela

18/10/16

THEORETICAL FRAMEWORK

Evolutionary algorithms have been shown to be successful for a wide range of optimization problems. While these randomized search heuristics work well for many optimization problems in practice, a satisfying and rigorous mathematical understanding of their performance is an important challenge in the area of genetic and evolutionary computing. It has been proven for various combinatorial optimization problems that they can be solved by evolutionary algorithms in reasonable time using a suitable representation together with mutation operators adjusted to the given problem. The representations used in these papers are different from the general encodings working with binary strings as considered earlier in theoretical works on the runtime behavior of evolutionary algorithms. the chosen representations reflect some properties of partial solutions of the problem at hand that allow to obtain solutions that can be extended to optimal ones for the considered problem.

Dynamic programming is a well-known algorithmic technique that helps to tackle a wide range of problems. A general framework for dynamic programming has been considered by e. g. Woeginger and Kogan . The technique allows the design of efficient algorithms, that solve the problem at hand to optimality, by extending partial solutions to optimal ones.

Framework for Evolutionary Algorithms An evolutionary algorithm consists of different generic modules, which have to be made precise by the user to best fit to the problem. Experimental practice, but also some theoretical work, demonstrate that the right choice of representation, variation operators, and selection method is crucial for the success of such algorithms. We assume that the problem to be solved is given by a multi-objective function g that has to be optimized. We consider simple evolutionary algorithms that consist of the following components. The algorithm (see Algorithm 2) starts with an initial population P0.

During the optimization the evolutionary algorithm uses a selection operator sel(·) and a mutation operator mut(·) to create new individuals. The d-dimensional fitness function together with a partial order par on R d induce a partial order dom on the phenotype space, which guides the search. After the termination of the EA, an output function out(·) is utilized to map the individuals in the last population to search points from the original search space.

The first thing we need to ask ourselves is ¿What I´m going to evaluate with my genetic algorithm?

R= We need to evaluate how good is a solution for the given problem, it means that we will have a bunch of solutions and we want the better one. In short Chromosome=Possible solution.

CIRCLE USING GENETIC ALGORITHMS

We are given a set of 2D points that correspond to a perimeter of a given circunference, our task is to find the radius and the center of this circle using this points. We are going to create randomly a chromosome that will contain the h,k,r (xcentro,ycentro,radio) values of the circle then we will calculate the radious of each one by using the formula:

raux=(x-h)**2 +(y-k)**2 so we can find out the difference between the expected radious and the radious we are generating, this difference is returned as the error, eventually we are going to get closer to the real value some iterations beyond.

SQUARE USING GENETIC ALGORITHMS

We are given a set of 2D points that correspond to a perimeter of a given square, our task is to find the side length and the center of this square using this points. We are going to create randomly a chromosome that will contain the h,k,l (xcentro,ycentro,side length) values of the square then we will calculate the side length of each one.

Finding the largest value of x within the points and the largest value of y then we do the same but for the minimun value in x and y finally we find the difference between this two values by substracting :
Xl=MaxX – MinX and   Yl =MaxY - MinY we took the bigger one L=max(Xl,Yl) and with this length divided by 2 we get the center of the square in (L/2, L/2). Finally we return the error that is the difference between the real values and the calculated values for h,k,L.

EQUIPMENT AND MATERIAL.

SOFTWARE: The program is made in Python

BODY

We were told to write down the problems mentioned above.

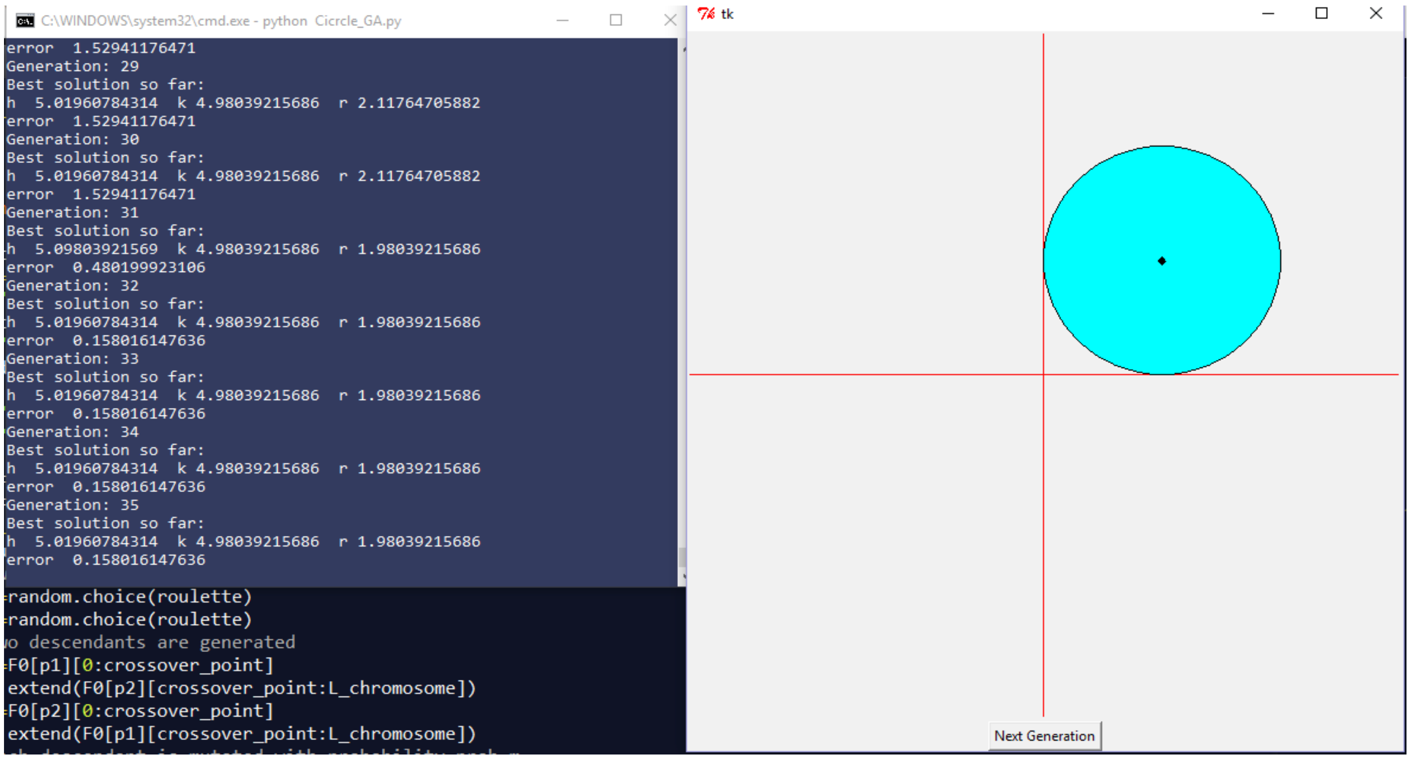
Circle

Measures using points=[(5,3),(5,7),(7,5)] probabilities are in red and number of generations (in blue) we went through to find the real values.

`(see e. g. [1, 2]).`

	20	50	70
0.25	Best solution so far: h 7.05882352941 k 4.86274509804 r 2.33333333333 error 11.0484429066	Best solution so far: h 5.01960784314 k 4.94117647059 r 2.0 error 0.545174932718	Best solution so far: h 5.01960784314 k 4.98039215686 r 2.0 error 0.234525182622
0.5	Best solution so far: h 5.33333333333 k 4.98039215686 r 1.76470588235 error 2.33064206075	Best solution so far: h 5.13725490196 k 4.98039215686 r 1.86274509804 error 1.09919261822	Best solution so far: h 5.13725490196 k 4.98039215686 r 1.86274509804 error 1.09919261822
0.75	Best solution so far: h 5.09803921569 k 5.09803921569 r 1.92156862745 error 0.849673202614	Best solution so far: h 5.01960784314 k 5.01960784314 r 2.0 error 0.234525182622	Best solution so far: h 5.01960784314 k 5.01960784314 r 2.0 error 0.234525182622

F1:Circle



```
error  1.52941176471
Generation: 29
Best solution so far:
h  5.01960784314  k 4.98039215686  r 2.11764705882
error  1.52941176471
Generation: 30
Best solution so far:
h  5.01960784314  k 4.98039215686  r 2.11764705882
error  1.52941176471
Generation: 31
Best solution so far:
h  5.09803921569  k 4.98039215686  r 1.98039215686
error  0.480199923106
Generation: 32
Best solution so far:
h  5.01960784314  k 4.98039215686  r 1.98039215686
error  0.158016147636
Generation: 33
Best solution so far:
h  5.01960784314  k 4.98039215686  r 1.98039215686
error  0.158016147636
Generation: 34
Best solution so far:
h  5.01960784314  k 4.98039215686  r 1.98039215686
error  0.158016147636
Generation: 35
Best solution so far:
h  5.01960784314  k 4.98039215686  r 1.98039215686
error  0.158016147636
```

```
random.choice(roulette)
random.choice(roulette)
o descendants are generated
F0[p1][0:crossover_point]
extend(F0[p2][crossover_point:L_chromosome])
F0[p2][0:crossover_point]
extend(F0[p1][crossover_point:L_chromosome])
```

F2: Graphical representation

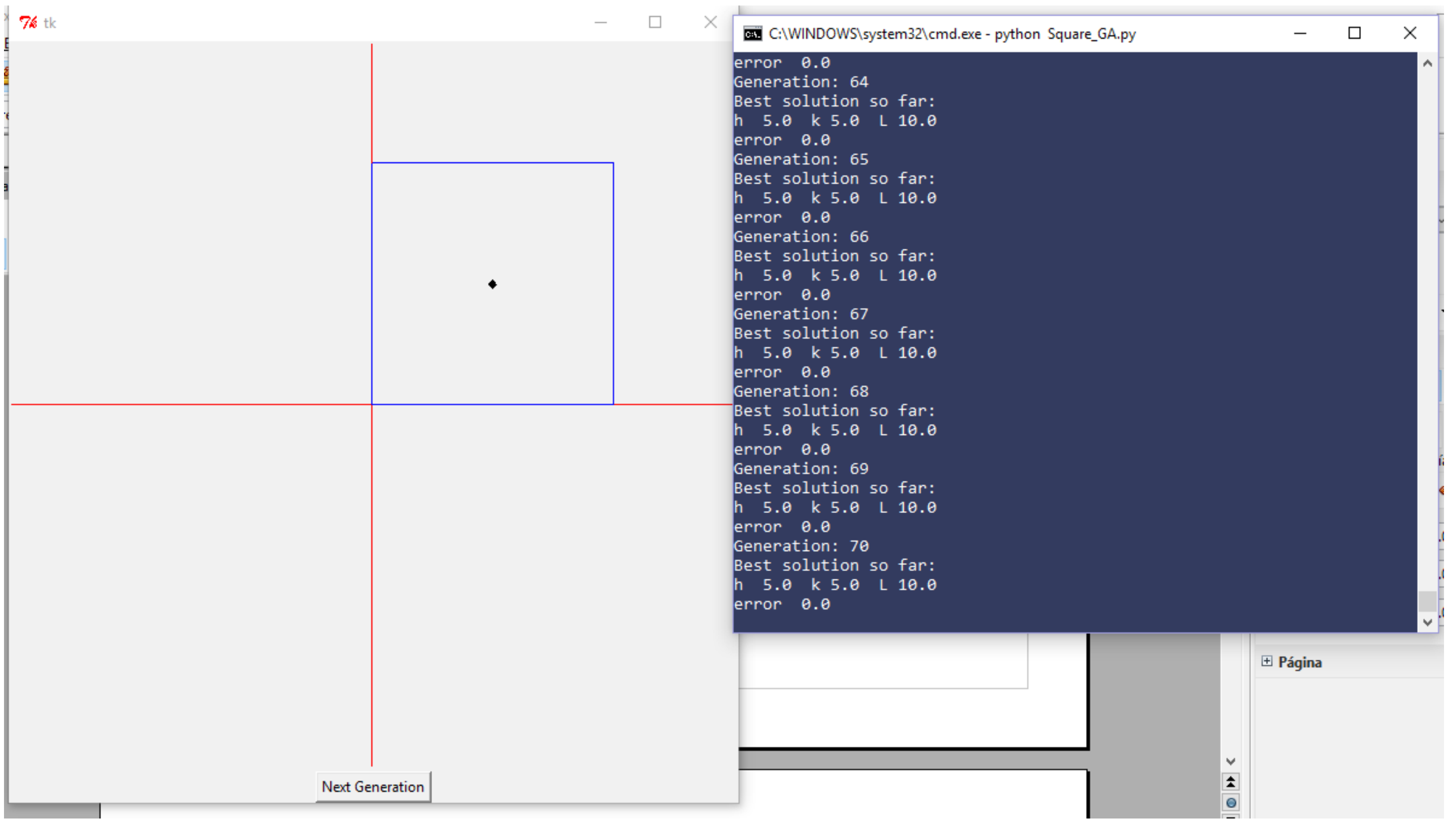
Square

Measures using points=[(0,3),(5,0),(3,10)] probabilities are in red and number of generations (in blue) we went through to find the real values.

(see e. g. [3, 4]).

	20	50	70
0.25	Best solution so far: h 3.607843137725 k 4.90196078431 L 10.0 error 1.49019607843	Best solution so far: h 3.74509803922 k 4.98039215686 L 10.0 error 1.27450980392	Best solution so far: h 3.74509803922 k 5.0 L 10.0 error 1.25490196078
0.5	Best solution so far: h 4.92156862745 k 5.0 L 9.96078431373 error 0.117647058824	Best solution so far: h 5.0 k 5.0 L 10.0 error 0.0	Best solution so far: h 5.0 k 5.0 L 10.0 error 0.0
0.75	h 5.0 k 4.98039215686 L 10.0 error 0.0196078431373	Best solution so far: h 5.0 k 5.0 L 10.0 error 0.0	Best solution so far: h 5.0 k 5.0 L 10.0 error 0.0

F3: Square



```
error  0.0
Generation: 64
Best solution so far:
h  5.0  k 5.0  L 10.0
error  0.0
Generation: 65
Best solution so far:
h  5.0  k 5.0  L 10.0
error  0.0
Generation: 66
Best solution so far:
h  5.0  k 5.0  L 10.0
error  0.0
Generation: 67
Best solution so far:
h  5.0  k 5.0  L 10.0
error  0.0
Generation: 68
Best solution so far:
h  5.0  k 5.0  L 10.0
error  0.0
Generation: 69
Best solution so far:
h  5.0  k 5.0  L 10.0
error  0.0
Generation: 70
Best solution so far:
h  5.0  k 5.0  L 10.0
error  0.0
```

F4:Graphical representation

CODE***

SQUARE

```
def f(x):
   global points
   L_points=len(points)
   h, k, R=x
   error=0

   mX= points[0][0]
   mY= points[0][1]
   nX= points[0][0]
   nY= points[0][1]
   #maximun
   for i in range(0,L_points):
      if mX < points[i][0] :
         mX = points[i][0]

   for i in range(0,L_points):
      if mY < points[i][1] :
         mY = points[i][1]
```

```
    #minimun

    for i in range(0,L_points):
        if nX > points[i][0] :
            nX = points[i][0]

    for i in range(0,L_points):
        if nY > points[i][1] :
            nY = points[i][1]

    lX=mX- nX
    lY=mY- nY
    L=max(lX,lY)
    h=L/2
    k=L/2

    hc, kc, Lc=x #chromosome solution

    error+=abs(L- Lc )
    error+=abs(h- hc )
    error+=abs(k- kc )
    #print "Error",error

    return error
```

CIRCLE

```
def f(x):
    global points
    h,k,R=x
    error=0
#se usa la formula (x-h)(x-h) + (y-k)(-k)=r*r para obtener el radio generado
    for index in range(len(points)):
        raux2=(points[index][0]-h)**2 +(points[index][1]-k)**2
#comparamos las diferencias de cada uno de los radios
#que generan de los puntos, si la diferencia en radios es minima
#esos puntos
        error+=abs(raux2-R**2)
    return error;
```

CONCLUSION

I liked it the way we took the class in the lab like if it was free style, but with the help of the professor, also it is good to explain what will be the next topic to cover in the lab so the students can find it out and investigate a little bit before entering the class because it could be confusing.

BIBLIOGRAPHY

Evolutionary Computing class