Computational Statistics Computer Lab 6

Muhaiminul Islam, Hussnain Haidar

2022-12-19

Question 1: Genetic algorithm

In this assignment, we will try to perform one-dimensional maximization with the help of a genetic algorithm.

1. Define the function

$$f(x) = \frac{x^2}{e^x} - 2 * exp(-(9 * sinx)/(x^2 + x + 1))$$

```
f= function(x)
{
  result= ((x^2)/exp(x))-2*exp(-(9*sin(x))/(x^2+x+1))
  return(result)
}
```

2. Define the function crossover(): for two scalars x and y it returns their "kid" as (x+y)/2.

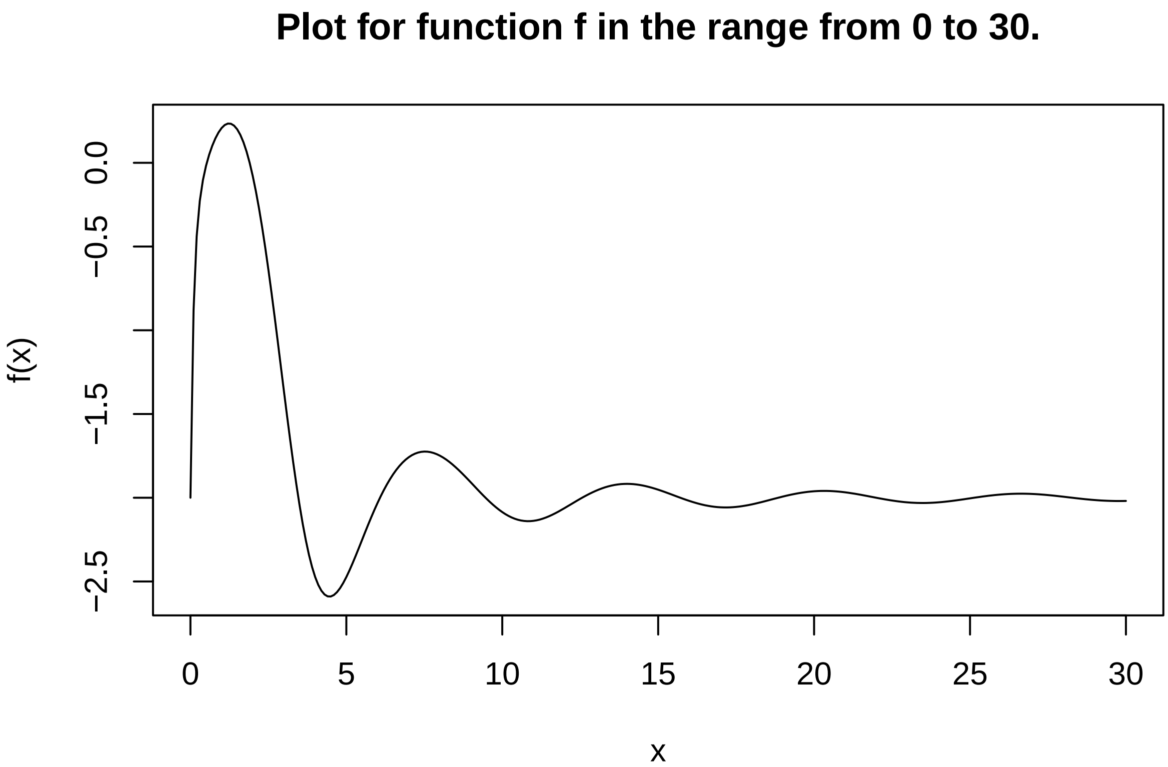
```
crossover= function(x,y){

  kid= (x+y)/2
  return(kid)

}
```

3. Define the function mutate() that for a scalar x returns the result of the integer division x^2 mod 30.

```
mutate= function(x){

  result= (x^2)%%30
  return(result)

}
```

4(a)

```r
x= seq(0,30,0.1)
plot(x, f(x), main="Plot for function f in the range from 0 to 30.", ylab = "f(x)", type= "l")
```



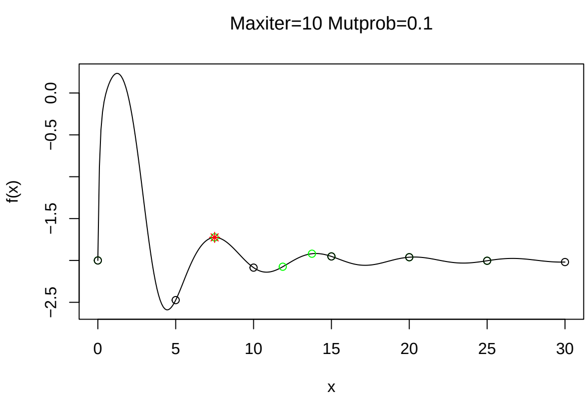
Visually we can see that the maximum value for the function f can be found at around x= 2

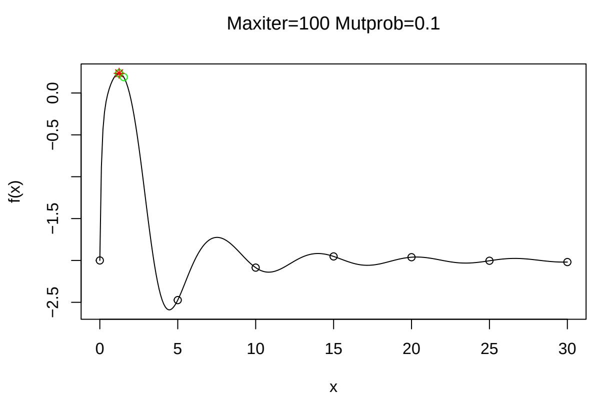
4(b)-4(d)

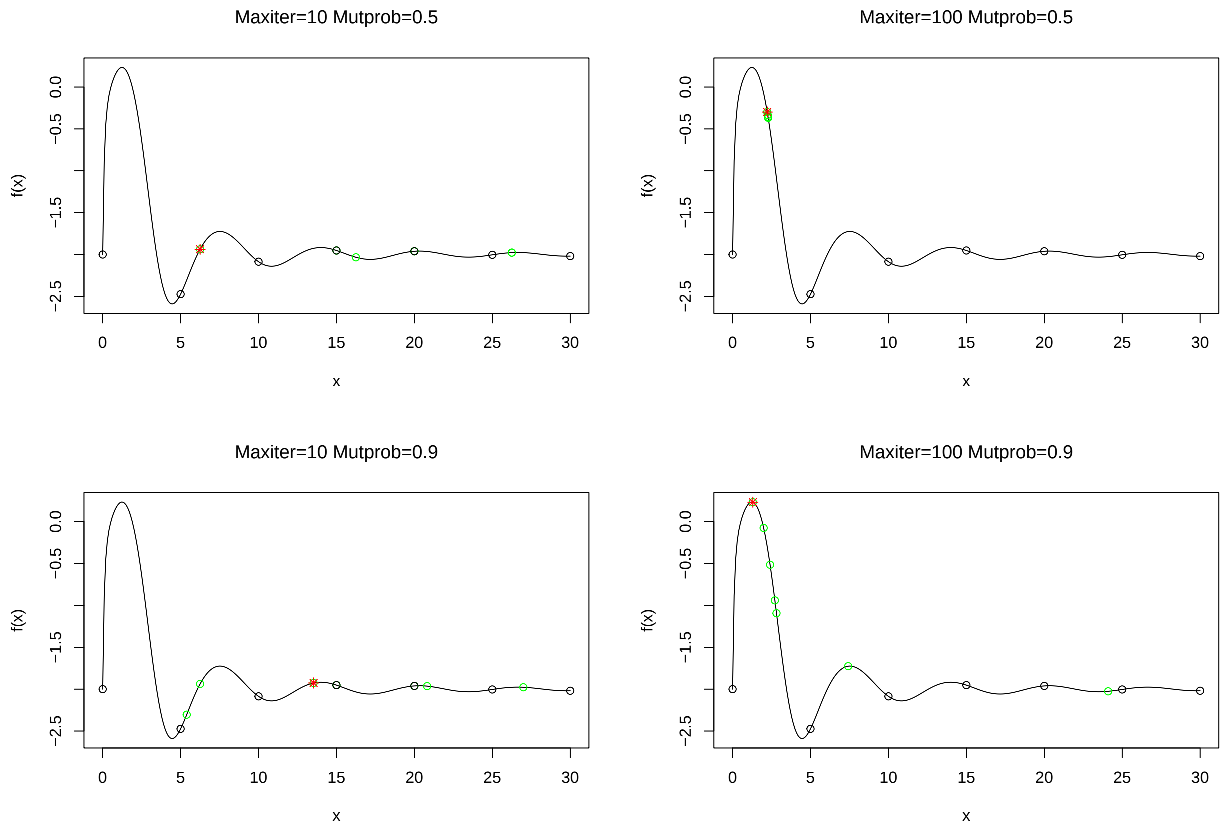
```r
genetic= function(maxiter,mutprob){
  x= seq(0,30,0.1)
  plot(x, f(x), main=substitute(paste('Maxiter=',maxiter,' Mutprob=', mutprob)), ylab = "f(x)", type= "
  X= seq(0,30,5)
  Values= f(X)
  mx= c()
  for(i in 1:maxiter){
    parents= sample(1:7, 2)
    victim= order(Values)[1]
    new_kid= crossover(X[parents[1]], X[parents[2]])

    new_kid= ifelse(mutprob> runif(1),mutate(new_kid),new_kid)
```

```
    X[victim]= new_kid

    Values= f(X)
    # print(Values)
    mxp= X[which.max(Values)]
    mxv= f(mxp)


  }
  # print(mxv)

  #last generation after iterations
  points(X, Values, col="green")
  #max value in the graph with red point
  points(mxp, mxv, col="red", pch=8)

  # comparing with initial values
  X_initial= seq(0,30,5)
  Values_initial= f(X_initial)
  points(X_initial, Values_initial, col="black")



}
```

In the below graphs black dots represent values of initial population, green dots represent values of population after final iteration, and the red dot represent max value after final iteration.







from above graphs we can see that genetic algorithm fails to find global maximal value when number of iterations are 10 even when we set mutation probability to 0.9, but when we increase number of iterations to 100 we see that the population moves toward global maximum value. More iteration means more possibility to converge to global maximal as strong parents are remaining after iterations with higher mutation probability.