HW1\_EC

Tournament selection

tournament = c ("0101110111","0111001001","0011001001","1110101100","1000010010","0000100010")  
set.seed(42)  
  
random\_tournament = sample(tournament,6)  
f = vector()  
  
for (i in 1:length(random\_tournament) ) {  
 f[i]= table(strsplit(random\_tournament[i],"")) [2]   
}  
  
mating\_pool= vector()  
  
#slinding window of 3 elements  
  
for (i in 1:length(f) ) {  
 sliding\_window = seq(from=i,to=i+2,by=1)  
 overflow= i+2 - length(f)  
 if (overflow > 0) {  
 sliding\_window = c(sliding\_window[c(1:(3 - overflow))],1:overflow)  
 }  
 mating\_pool[i] = max(f[sliding\_window] )  
}

# • How many copies of each chromosome are present in the mating pool?

table(mating\_pool)

## mating\_pool  
## 6 7   
## 3 3

# • What is the average fitness of the chromosomes in the mating pool?

mean(mating\_pool)

## [1] 6.5

# If the tournament size is reduced to one, what is the probability that the chromosome 1110101100

appears in the mating pool?

#if the ournament size is reduced to one means there is no comptetition so it will be 1/n and in this case 1/6 probability to appear in the mating pool

# • If the tournament size is increased to five, what is the probability that the chromosome 0111001001

appears in the mating pool?

tournament = c ("0101110111","0111001001","0011001001","1110101100","1000010010","0000100010")  
set.seed(42)  
  
random\_tournament = sample(tournament,6)  
f = vector()  
  
for (i in 1:length(random\_tournament) ) {  
 f[i]= table(strsplit(random\_tournament[i],"")) [2]   
}  
  
mating\_pool= vector()  
  
#slinding window of 3 elements  
  
for (i in 1:length(f) ) {  
 sliding\_window = seq(from=i,to=i+4,by=1)  
 overflow= i+4 - length(f)  
 if (overflow > 0) {  
 sliding\_window = c(sliding\_window[c(1:(5 - overflow))],1:overflow)  
 }  
 mating\_pool[i] = max(f[sliding\_window] )  
}  
  
mating\_pool

## [1] 7 6 7 7 7 7

# 0 since it must the top 2 in oder to appear

Whole arithmetic crossover

x = c(0.23, 0.57, 0.29, 0.44, 0.44)  
  
y = c(0.63, 0.82, 0.62, 0.15, 0.51)

#ui= alpha\* xi + (1-alpha)\*yi  
alpha= 0.5  
u\_0.5 = (alpha \* x) + (1-alpha)\*y  
v\_0.5= (alpha\* y) + (1-alpha)\*x  
#results  
u\_0.5

## [1] 0.430 0.695 0.455 0.295 0.475

v\_0.5

## [1] 0.430 0.695 0.455 0.295 0.475

alpha= 0.1  
u\_0.1 = (alpha \* x) + (1-alpha)\*y  
v\_0.1= (alpha\* y) + (1-alpha)\*x  
#results  
u\_0.1

## [1] 0.590 0.795 0.587 0.179 0.503

v\_0.1

## [1] 0.270 0.595 0.323 0.411 0.447

3 Selection

Population = c("AAOOEIIIOEA", "EEOUEO" ,"UUIOOAAIEEO","AAAEEEIIIOUUU","AUEIOUOOEEIIUIA")  
fitness = vector()   
for (i in 1:length(Population)){  
 fitness[i] = sum(floor(table(strsplit(Population[i],""))/2))  
  
}

## • Calculate the probabilities of selecting each one of these individuals, based on a proportional selection

(based on the fitness).

prop\_selection= print(fitness / sum(fitness))

## [1] 0.1904762 0.0952381 0.2380952 0.1904762 0.2857143

## • Calculate the probabilities of selecting each one of these individuals, based on a linear ranking selection

with C = 2.

r = vector()  
c = 2  
n\_1 = (length(Population)-1)  
  
for (i in 1:length(fitness) ){  
   
 r[i] = which(sort(fitness) == fitness[i])[1] -1   
  
}  
  
linear\_ranking\_fitness = print(r \* (c/n\_1))

## [1] 0.5 0.0 1.5 0.5 2.0

## • Calculate the probabilities of selecting each one of these individuals, based on an exponential ranking

selection with m = 3.

m = 3  
  
exponential\_ranking\_fitness = print(m \* (r/n\_1) ^ (m-1))

## [1] 0.1875 0.0000 1.6875 0.1875 3.0000