HW1_EC

Jesus Eider Diaz Moraila

A00828174

1 Tournament selection (20%)

For this exercise, assume that the chromosomes are coded as ten-bit strings, and that the fitness of each chromosome is calculated as the sum of ones in the chromosome. For example, the chromosomes 1011110111 and 0001001001 have a fitness of 8 and 3, respectively.

Imagine that the following population of six individuals has already been shuffled:

Population
0101110111
0111001001
0011001001
1110101100
1000010010
0000100010

Apply a tournament selection of size three to create the mating pool (assume the version of the tournament selection that slides the tournament window) and answer the following questions:

```
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

random_tournament
```

```
## "0101110111" "1000010010" "0000100010" "1110101100" "0111001001" "0011001001"

#3 of the 1st and 3 of the 4th
```

• 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
```

2 Whole arithmetic crossover (10%)

Given two real-valued chromosomes $\vec{x} = \{0.23, 0.57, 0.29, 0.44, 0.44\}$ and $\vec{y} = \{0.63, 0.82, 0.62, 0.15, 0.51\}$, calculate the resulting offspring if the whole arithmetic operator is applied. Consider two cases: first with $\alpha = 0.5$ and later with $\alpha = 0.10$.

```
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 for alpha = 0.5
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 for alpha = 0.1
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 (15%)

For this exercise, assume that the chromosomes are coded as character strings of a variable length and that the fitness of each chromosome is calculated as the sum of the repetitions of characters in the chromosome. The possible values for the genes are the vowels in the Spanish alphabet: 'A', 'E', . . . , 'U'. For example, the strings `EIAAO` and `AEEIOEEUAUAEE` have a fitness of 1 (since only 'A' is repeated once) and 8 (since 'A' is repeated twice, 'E' is repeated five times and 'U' is repeated once), respectively.

Imagine that you are given the following population of five individuals:

Population
AAOOEIIIOEA
EEOUEO
UUIOOAAIEEO
AAAEEEIIIOUUU
AUEIOUOOEEIIUIA

```
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
```

4 Selecting the best representation (15%)

A company wants to determine the best representation for a planning problem. The problem this company tries to solve requires to find a sequence of 12 actions in order to move a robot from point A to point B (the robot is located on a grid along with some obstacles). The robot has four available actions in its repository: `UP`, `DOWN`, `LEFT`, and `RIGHT`. Any of these actions takes exactly one second to execute. Since you are a well-known expert on genetic algorithms, this company is requesting your help to make a choice that maximizes the opportunities that its genetic algorithm implementation fins a good-quality solution for the problem. Given the limited amount of resources for the project, they have only considered two representations for its genetic algorithm implementation: binary and integer-based representations. Also, they have already decided that, regardless of the chosen representation, they will use linear ranking selection and one-point crossover, and that $p_c = 1$ and $p_m = 0$.

Just hours before you submit your report with the recommended representation (and the corresponding justification), you are informed that a new action will be added to the robot's repository: `JUMP` (which allows the robot to jump over the obstacles in the direction of the previous move). How would this change affect your decision on the representation to recommend? Justify your answer.
Hint: To compare the two representations, focus on the size of the search space. Remember that you are not requested to implement the genetic algorithm, just to analyze what is requested.

In order to represent in binary we need a solution that is formed of the following tuple (0,1,0,0,0) where each tuple means a command in this case this encoding means a DOWN action (1,0,0,0,0) means a UP action (0,0,0,0,1) means a JUMP action an so on, the final solution representation will be (1,0,0,0,0) twelve times meaning we have a 2 ^(5*12) possible solutions or a search space of 115292150 4606846976 maybe less if we don't accept a solution that have two ones in a window of five, like (1,0,0,0,1) this is not a possible piece of a solution.

On the other hand we have a integer representation that is simply look like (1,2,1,2,4,5,3,3,1,1,2,1) where 1 means UP, 2 means DOWN an so on, in this case we have a 5^12 solutions or a search space of 244140625 .

5 Analysis (40%)

Analyze the following scenarios. For each one of them, provide a brief but solid analysis.

- What would you think of using elitism in a steady-state genetic algorithm? Do you think it is a good idea? What would be the benefits?
- Imagine that linear ranking selection (using $C = 2$) is executed on a population of n chromosomes. What would be the effect on the selection probabilities if the fitness of every chromosome in the population is multiplied by a constant k?
- Imagine that exponential ranking selection (using $m = 3$) is executed on a population of n chromosomes. What would be the effect on the selection probabilities if the fitness of every chromosome in the population is multiplied by a constant k?
- Given two real-valued chromosomes of length five, $\vec{x} = \{x_1, x_2, x_3, x_4, x_5\}$ and $\vec{y} = \{y_1, y_2, y_3, y_4, y_5\}$, what would be the result of combining $\vec{x}$ and $\vec{y}$ through a whole arithmetic crossover with $\alpha = 1.0$? How would the offspring change if $\alpha = 0$?

1) I f we are concern about rapid convergence the combination of steady-state and elitism will be a good cho ice, since every cycle we substitute the worst individuals and then passing the best individual every time, this will cause every cycle the population will be more likely to be akin the “elite solution” , on the other h and this rapid convergence may cause that the algorithm gets stuck in a local optima because this combin ation is not in favor of diversity.

2) Nothing since the fitness is calculated by the relative ranking of the population and if we multiply the who le fitness by any constant the fitness ranking is preserved

3) Same as 2) we no compute the selection based on the direct fitness but in the ranking, and because the ra nking is conserved if the fitness is multiplied by a constant, everything remains the same

4) The result offspring of both alpha values will be the parents in both cases, since the crossover formula dic tates the proportion of each parent in the offspring, thus using a alpha=1 will give us 100% proportion of the first parent + 0% of the second parent for the first offspring and 0% of the first parent and 100% for t he second parent for the second offspring. And for the case of alpha = 0 it’s the same but the order is inver se in the offspring.