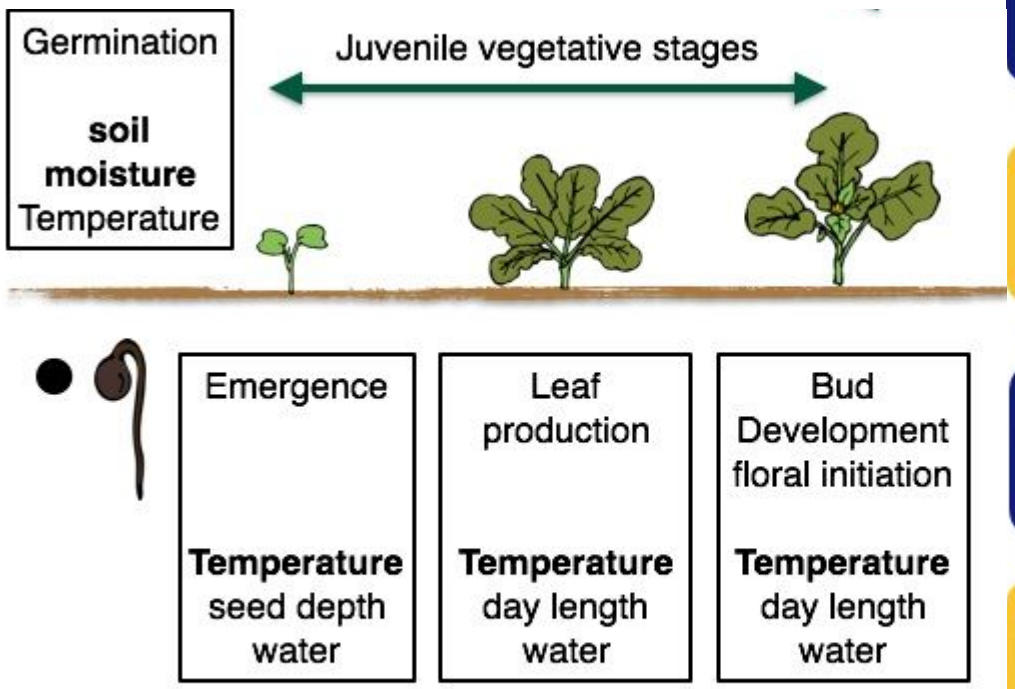
Nutrient Solution for Juvenil Stages In Hydroponics using Genetic Algorithm

Bryan Andi Gerrardo (19/448695/PPA/05778)
Deffa Rahadiyan (19/448699/PPA/05782)
Nadhifa Sofia (19/448721/PPA/05804)

Background



Plant

Nutrient is an important

Each growth stage needs different nutrient

Juvenil stage is a vital stage

Genetic Algorithm

Problem

The amount of nutrients can be given to plants, but the amount of nutrients that are suitable to plant needs are known, consequently the growth of plants in the juvenile stage is not optimal.

Purpose

Designing a system to determine the right nutrients in the juvenile stage using Genetic Algorithms.

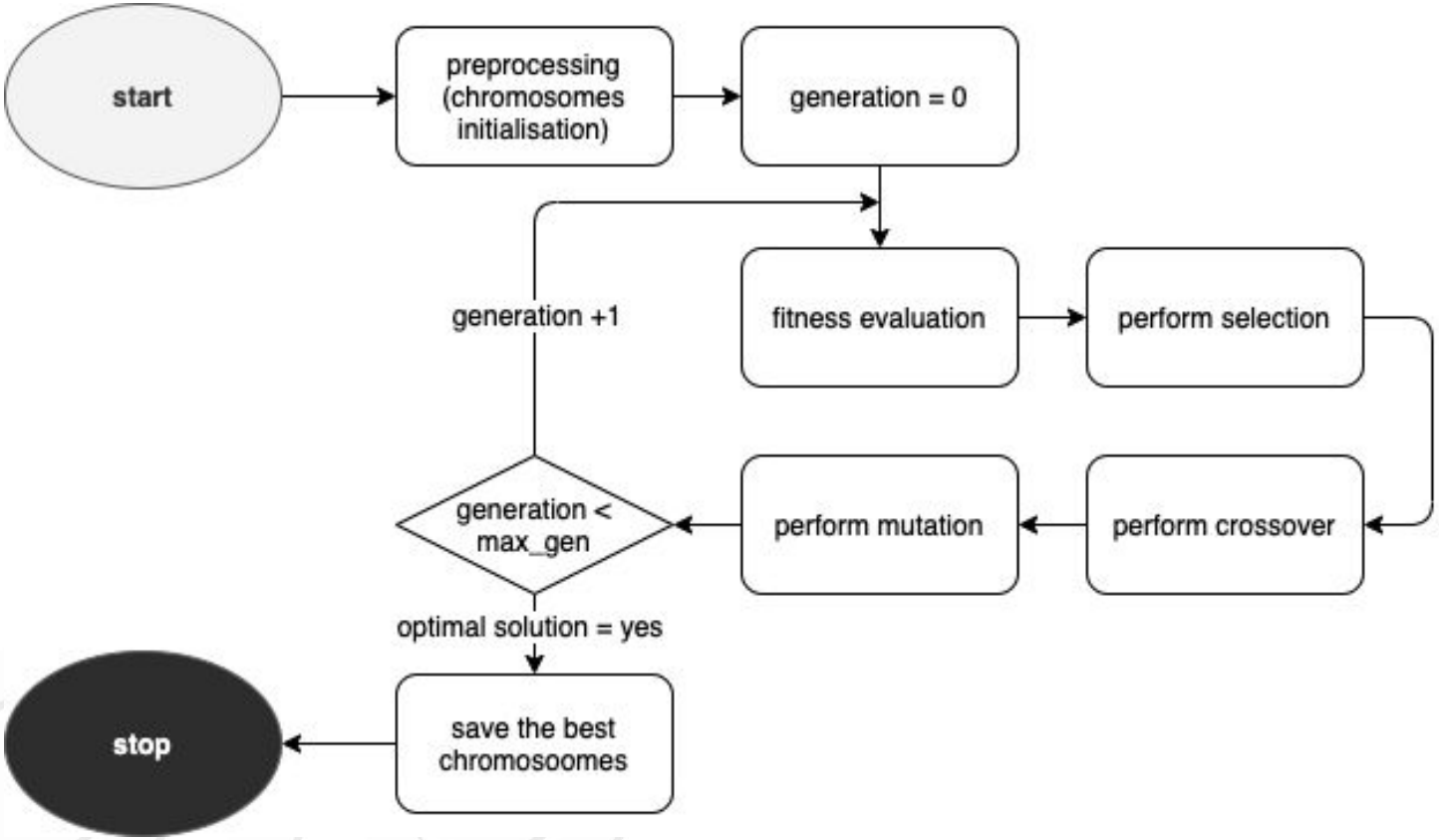
Benefit

Plants can grow well in the juvenile stage

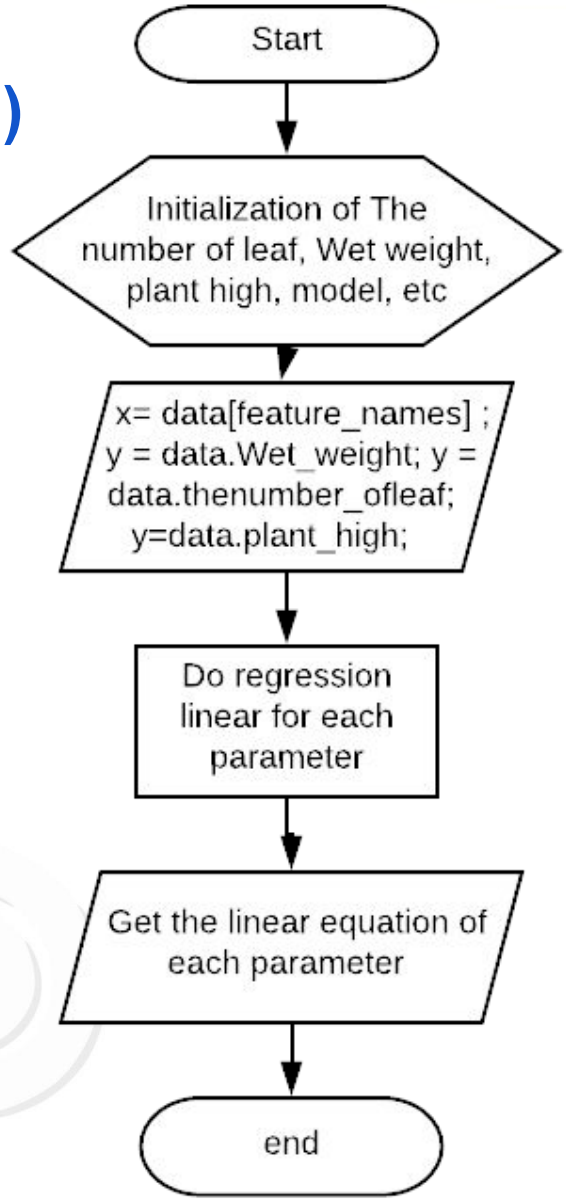
Problem scope

- The research is just for juvenil stage in 2 weeks after planting
- Plant that will be used is pepper (Annum Capsicum L.)
- Nutrient that will be analysis is Macronutrient (Nitrogen, Phosphor, Kalium)
- The plant characteristics that will be analysis is plant high, the number of leaf, and Wet weight
- The number of chromosome is 50.

Main flowchart



Flowchart for Preprocessing (Linear Regression)



Characteristics of success seedling

- The number of leaf are more than 4 leaves
- plant high is more than 15cm
- Wet weight of plant is more than 8 gr

Initialization (Value Encoding)

Chromosome[1] = [a;b;c;d] = [X1;X2,X3,X4]
Chromosome[2] = [a;b;c;d] = [X1;X2,X3,X4]
Chromosome[3] = [a;b;c;d] = [X1;X2,X3,X4]
Chromosome[4] = [a;b;c;d] = [X1;X2,X3,X4]
Chromosome[5] = [a;b;c;d] = [X1;X2,X3,X4]

.
.
.

Chromosome[50] = [a;b;c;d] = [X1;X2,X3,X4]

Note : a=kalium; b=nitrogen; c=phosphorus; d=water.

Fitness function

fitness function has to be something that measures how good the solution is. In this system, the greater value will be a good solution.

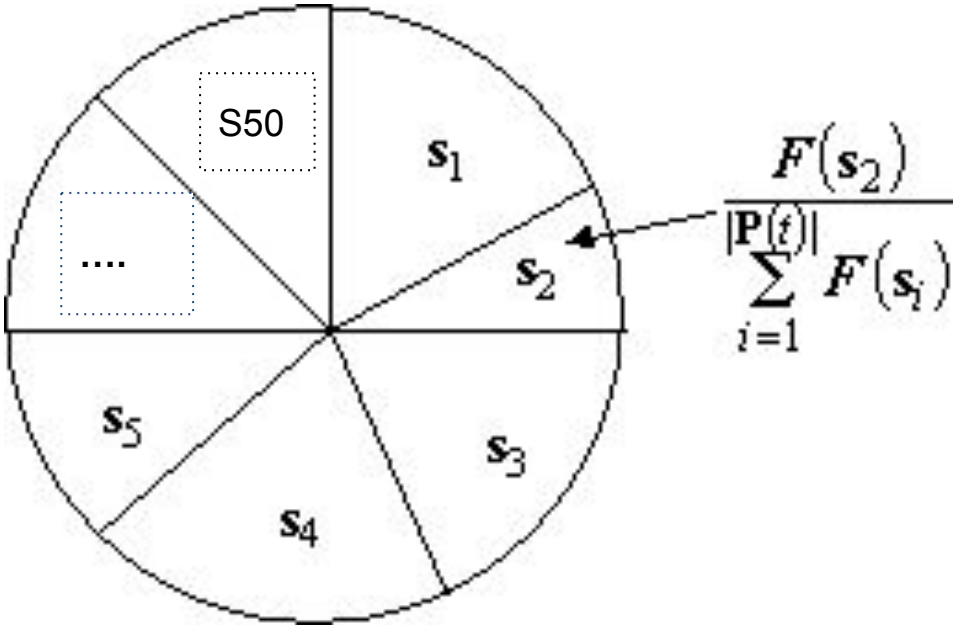
$$\text{fitness} = (x1 - y1) + (x2 - y2) + (x3 - y3)$$

Note :

x1 = objective value of Plant high
x2 = objective value of The number of leaf
x3 = objective value of Wet weight
y1 = plant high setpoint(in control condition)
y2 = the number of leaf setpoint (in control condition)
y3 =Wet weight setpoint (in control condition)

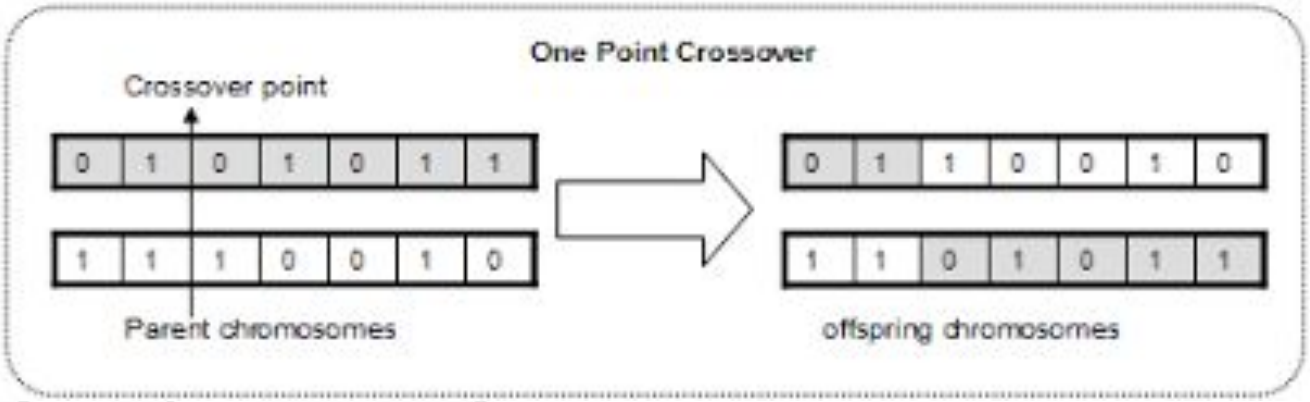
Selection using Roulette Wheel

Individual	Fitness
S1	a
S2	b
S3	c
S4	d
S5	e
...	...
S50	z

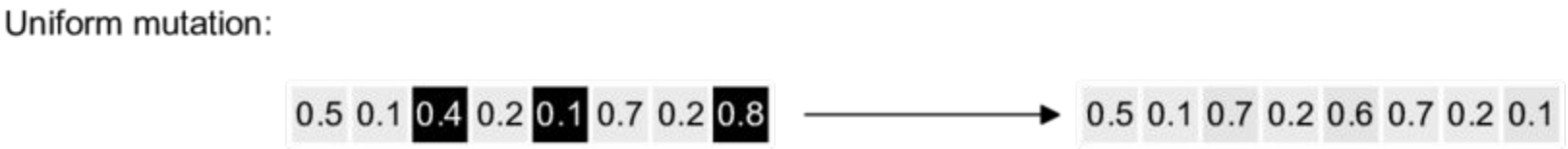


Crossover using 1-Cut-Point

- Choose one position in chromosomes randomly.
- Child 1 consist of a head chromosomes of parent 1 and tail of parent 2 .
- Child 2 consist of a head chromosomes of parent 2 and tail of parent 1



Uniform Mutation



In uniform mutation we select a random gene from our chromosome, let's say xi and assign a uniform random value to it. Let xi be within the range [ai,bi] then we assign U(ai,bi) to xi U(ai,bi) denotes a uniform random number from within the range [ai,bi].

```
Algorithm –
1.    Select a random integer number i from [1,n]
2.    Set xi to U(ai,bi).
```

Updating Generation

If the **generation < max_gen**, assume that max_gen = 5, then we should continue the generation looping by using :

- fitness evaluation
- perform selection
- perform crossover
- perform mutation

If the generation reaches the max_gen that is defined in advance, then the engine will **save the best chromosomes.**

IMPLEMENTATION GA TO DETERMINE NUTRIENT SOLUTION IN SEEDLING STAGE

Plant data in control condition

NO	Indicator	Value
1	Average of plant high	5,65
2	Average of the number of leaves	8,50
3	Average of plant weight	0,37

Plant Data in varying nutrient content

No	Kalium	Nitrogen	Postassium	Air	Tinggi_Tanaman	Jumlah_Daun	Berat_Basah
1	11.00	10.00	5.00	1,171.00	16.96	4	4.37
2	9.00	19.00	6.00	1,473.00	20.54	8	18.62
3	7.00	4.00	1.00	1,187.00	17.44	6	1.19
4	6.00	4.00	12.00	912.00	18.49	7	2.20
5	1.00	15.00	4.00	1,286.00	15.17	6	2.95
6	15.00	10.00	13.00	951.00	13.54	8	4.90
7	19.00	1.00	4.00	665.00	22.62	8	12.26
8	17.00	6.00	19.00	1,198.00	17.18	5	4.09

...

...

...

95	15.00	17.00	6.00	913.00	25.12	8	18.19
96	15.00	16.00	4.00	506.00	17.08	6	3.31
97	14.00	1.00	20.00	605.00	21.25	8	16.16
98	16.00	8.00	14.00	1,142.00	20.47	6	8.04
99	20.00	19.00	7.00	538.00	11.52	4	2.52
100	4.00	8.00	9.00	987.00	10.33	8	12.36

Preprocessing : Linear equation for each parameters

Plant_high = 0.384k + 0.014n + 0.164p + 0.010w + 0.012

Num_leaf = 0.178k + 0.059n + 0.183p + 0.007w + 0.010

Weight = 0.080k + 0.096n + 0.287p + 0.005w + 0.012

Note :

k = Kalium

n = Nitrogen

w = Water

p = Phosphor

Initiation Implementation

```
# Initializing n = Number of chromosome
n = 10

# Initialization of chromosomes
# chromosomes
chromosome_abmix = np.random.randint(1,21 ,(n,3))
chromosome_water = np.random.randint(500,1501 ,(n,1))
chromosome = np.append(chromosome_abmix,
                       chromosome_water,
                       axis=1)

epoch = 0
# Initializing g = Number of generation
g = 5
```

Selection Implementation

```
def selection(fitness,chromosome):
  # Calculating the total of fitness function
  total = fitness.sum()
  # Calculating Probability for each chromosome
  prob = fitness/total

  # Selection using Roulette Wheel
  # Calculating Cumulative Probability
  cum_sum = np.cumsum(prob)

  # Generating Random Numbers in the range 0-1
  Ran_nums = np.random.random((chromosome.shape[0]))

  # Making a new matrix of chromosome for calculation purpose
  chromosome_2 = np.zeros((chromosome.shape[0],4))
  ...
```

Selection Implementation (Cont'd)

```
def selection(fitness,chromosome):
  ...
  for i in range(Ran_nums.shape[0]):
      for j in range(chromosome.shape[0]):
          if Ran_nums[i]  < cum_sum[j]:
              chromosome_2[i,:] = chromosome[j,:]
              break

  chromosome = chromosome_2.astype(int)
  return(chromosome)
```

Crossover implementation

```
def crossover(chromosome):
    # CROSSOVER
    R = np.random.random((chromosome.shape[0]))
    # Crossover Rate
    pc = 0.25
    flag = R < pc

    # Determining the cross chromosomes
    cross_chromosome = chromosome[[(i == True) for i in flag]]
    len_cross_chrom = len(cross_chromosome)

    # Calculating cross values
    cross_values = np.random.randint(1,4,len_cross_chrom)

    cpy_chromosome = np.zeros(cross_chromosome.shape)
```

Crossover implementation (Cont'd)

```
def crossover(chromosome):
    ...
    # Performing Cross-Over
    # Copying the chromosome values for calculations
    for i in range(cross_chromosome.shape[0]):
        cpy_chromosome[i , :] = cross_chromosome[i , :]

    if len_cross_chrom == 1:
        cross_chromosome = cross_chromosome
    else :
        for i in range(len_cross_chrom):
            c_val = cross_values[i]
            if i == len_cross_chrom - 1 :
                cross_chromosome[i , c_val:] =
cpy_chromosome[0 , c_val:]
            else :
                cross_chromosome[i , c_val:] =
cpy_chromosome[i+1 , c_val:]
```

Crossover implementation (Cont'd)

```
def crossover(chromosome):
    ...
    index_chromosome = 0
    index_newchromosome = 0
    for i in flag :
        if i == True :
            chromosome[index_chromosome, :] =
            cross_chromosome[index_newchromosome, :]
            index_newchromosome = index_newchromosome + 1
        index_chromosome = index_chromosome + 1

    return(chromosome)
```

Mutation implementation

```
# Calculating the total no. of generations
    a ,b = chromosome.shape[0] ,chromosome.shape[1]
    total_gen = a*b

    # Mutation rate = pm
    pm = 0.1
    no_of_mutations = int(np.round(pm * total_gen))

    # Calculating the Generation number
    gen_num = np.random.randint(0,total_gen, no_of_mutations)
```

Mutation implementation (Cont'd)

```
for i in range(no_of_mutations):
        a = gen_num[i]
        row = a//4
        col = a%4
        if (col==3) :
 # Generating a random number which can replace the selected
chromosome to be mutated for Gen 1,2,3
            Replacing_num = np.random.randint(500,1501)
        else :
# Generating a random number which can replace the selected
chromosome to be mutated for Water Gen or Gen 4
            replacing_num = np.random.randint(1,21)
        chromosome[row , col] = Replacing_num

    return(chromosome)
```

Evaluation of Implementation

```
"""
tinggi_tanaman(x) = 0,3839 k + 0,0143 n + 0,1642 p + 0,0097 a
+ 0,0122
jumlah_daun(x) = 0,1779 k + 0,0589 n + 0,1826 p + 0,0067 a +
0,0099
berat_basah(x) = 0,0798 k + 0,0961 n + 0,2868 p + 0,0049 a +
0,0125
"""

# Initializing n = Number of chromosome
n = 10
# Initialization of chromosomes
# chromosome
chromosome_abmix = np.random.randint(1,21 ,(n,3))
chromosome_water = np.random.randint(500,1501 ,(n,1))
chromosome = np.append(chromosome_abmix,
                       chromosome_water, axis=1)
print("First Chromosomes : \n",chromosome)

epoch = 0
# Initializing g = Number of generation
g = 5
```

Evaluation of Implementation

```
while epoch <  g :

    #Measure Fitness
    fitness =  getFitness(chromosome)

    #Apply SELECTION on chromosome
    chromosome = selection(fitness,chromosome)

    print("Chromosomes after selection : \n",chromosome)

    chromosome = crossover(chromosome)

    print("Chromosomes after crossover : \n",chromosome)

    chromosome = mutation(chromosome)

    print(" Chromosomes after mutation : \n" , chromosome)

    epoch = epoch + 1
    print("\n")
```

Initiation Result

```
In [20]: runfile('D:/Kuliah/KK
KKPM')
First Chromosomes :
 [[  20    3    1  540]
 [  12    6   15 1321]
 [   1    2   18 1318]
 [   7   20    8  522]
 [  19    1   10 1450]
 [  11    4    5 1357]
 [   3    5    1  920]
 [   4   13   18 1323]
 [  11    1   20  687]
 [   1   15   11 1489]]
```

Selection Result

```
Total Fitness : 232.14800000000002
Probability :
 [0.04807278 0.14131933 0.11445716 0.0425806  0.15587039 0.11265658
 0.03959543 0.13460379 0.08846942 0.12237452]
Cumulative Sum :
 [0.04807278 0.18939211 0.30384927 0.34642986 0.50230026 0.61495684
 0.65455227 0.78915606 0.87762548 1.        ]
Random Numbers
: [0.25189781 0.62878681 0.81941443 0.88784457 0.64617552 0.87584071
 0.13807343 0.48309349 0.5434454  0.87148473]
Chromosomes after selection :
 [[   1    2   18 1318]
 [   3    5    1  920]
 [  11    1   20  687]
 [   1   15   11 1489]
 [   3    5    1  920]
 [  11    1   20  687]
 [  12    6   15 1321]
 [  19    1   10 1450]
 [  11    4    5 1357]
 [  11    1   20  687]]
```

Crossover Result

```
Random Values :
 [0.86983262 0.82894541 0.86034812 0.55927658 0.47760462 0.23874565
 0.06220378 0.13663914 0.3764001  0.74549245]
Flagged Chromosome :
 [False False False False False  True  True  True False False]
Cross chromosome :
 [[  11    1   20  687]
 [  12    6   15 1321]
 [  19    1   10 1450]]
Cross Values or Cut Point :
 [3 2 2]
Chromosomes after crossover :
 [[   1    2   18 1318]
 [   3    5    1  920]
 [  11    1   20  687]
 [   1   15   11 1489]
 [   3    5    1  920]
 [  11    1   20 1321]
 [  12    6   10 1450]
 [  19    1   20  687]
 [  11    4    5 1357]
 [  11    1   20  687]]
```

Mutation result

```
Total Gen : 40
Total Mutations : 4
Gen for Mutation :  [12 17 39  3]
Mutation in Gen : 0 of Chromosome : 3
Replaced value :  3
Mutation in Gen : 1 of Chromosome : 4
Replaced value :  13
Mutation in Gen : 3 of Chromosome : 9
Replaced value :  742
Mutation in Gen : 3 of Chromosome : 0
Replaced value :  693
 Chromosomes after mutation :
 [[   1    2   18  693]
 [   3    5    1  920]
 [  11    1   20  687]
 [   3   15   11 1489]
 [   3   13    1  920]
 [  11    1   20 1321]
 [  12    6   10 1450]
 [  19    1   20  687]
 [  11    4    5 1357]
 [  11    1   20  742]]
```

Final Result after n Generation

IPython console

Console 1/A

```
------------------------------After 5 Generation---------------------------------
Chromosomes after 5 generation :
 [[  18   10   15 1042]
 [  18    9    7 1445]
 [  18   10   14 1445]
 [  18   10   15  820]
 [  18   14    8 1042]
 [  18    2   19 1032]
 [  18   10   15 1445]
 [   4    2   19  929]
 [   4   10   15 1042]
 [   4    2   19 1032]]
Fitness :
 [24.846 28.173 31.44  20.516 21.129 25.62  32.891 15.579 17.35  18.124]
Best fitness is in chromosome : 6
With Chromosome :  [  18   10   15 1445]
Thus, the best ABMix Composition are consist of:
 18 ml Kalium,  10 ml Nitrogen, and 15 ml Phosphor, with usage of water 1445 ml
```

Conclusion

The nutrient solution for Juvenil Stages in hydroponics project is proposed to use Genetic Algorithm since it is commonly used to **generate high-quality solutions** for **optimization problems** and **search problems** (in terms of the **optimum amount of ABmix composition consists of K, N, P_4, H_2O)**.

As there less human calculations in Genetic Algorithm, this project is suggested by using GA. Genetic algorithm itself simulates the process of natural selection which means those species **who can adapt to changes** in their environment **are able to survive** and **reproduce** and **go to next generation.**

To sum up, if the **generation** has achieved the **maximum generation** or the most optimum result, then the **system will stop**. Otherwise, continue the generation process by updating fitness evaluation, individuals selection, crossover and mutation like this project.

Thank you! :)

Mengakar Kuat
Menjulang
Tinggi