**RC25538 (IRE1507-013) July 13, 2015**
**Computer Science**

# IBM Research Report

## Automation of Daisy and Crop Yield Optimization

**Pramod Kumar**
IBM Research
Smarter Cities Technology Centre
Mulhuddart
Dublin 15, Ireland

and

Indian Institute of Technology, Kharagpur

**Research Division**
**Almaden – Austin – Beijing – Cambridge – Dublin - Haifa – India – Melbourne - T.J. Watson – Tokyo - Zurich**

# AUTOMATION OF DAISY
# AND
# CROP YIELD OPTIMIZATION

By
Pramod Kumar
IIIrd year undergraduate student
Of
Indian Institute of Technology, Kharagpur

## Acknowledgement

I wish to express my sincere gratitude to Mr. Kieran Deegan, IBM Smart Cloud Performance Manager for providing me an opportunity to do my internship and project work with "**IBM Ireland**."

My sincere thanks also goes to Dr. Sean McKenna for offering me an opportunity to work in his research group and leading me on diverse exciting projects.

I would like to express my sincere gratitude to my advisor Dr. Seshu Tirupathi for the continuous support of my research project, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this research project. I could not have imagined having a better advisor and mentor for my research project.

I would like to thank Dr. Beat Buesser for the encouragement and support through the learning process of this project.

## Introduction:

Daisy is a soil-plant-atmosphere system model that is designed to simulate water balance, heat balance, solute balance and crop production in agro-ecosystems [1]. All these process are dictated by the various management strategies by the farmer and this is an important driving variable in model. Weather data, the other important driving variable, is provided by in-house IBM research project called Deep Thunder that models hyper-local, short-term forecasting [2]. In addition to these driving variables, the user is required to define the relevant soil data, vegetation and other data parameters that dictate the physical process of the agro-ecosystem.

Daisy requires few user inputs to simulate crop-production in agro-ecosystem. Daisy generates simulation results based on historical data of important variable and provide crop yield & influencing factor in production.
Daisy is most compatible with windows operating system so one of the project objective was to automate daisy to make it platform independent. Along with daisy automation, Our objectives were to identify

i) Optimal crop management strategies,
ii) Optimizing external factors
iii) Understanding crop growth behaviour with driving variables

This projects was designed to run for 6 weeks only to meet academic requirements, and so our initial focus was on automation and then move to optimization use cases.

Programming Language : Python
Platform & Software : Windows 2007 & Daisy

## What is Daisy?

“Daisy is a soil-plant-atmosphere system model designed to simulate water balance, heat balance, solute balance and crop production in agro-ecosystems subjected to various management strategies. The water balance model comprises a surface water balance and a soil water balance. The soil water balance includes water flow in the soil matrix as well as in macrospores. Furthermore, it includes water uptake by plants and a model drainage to pipe drain. The heat balance model simulates soil temperature and freezing and melting in the soil. The solute balance model simulates transport, sorption and transformation processes. The crop production model simulates plant growth and development, including the accumulation of dry matter and nitrogen in different plant parts. Competition for light, water and nitrogen between plant species are also simulated.”

“In Daisy, weather data is used as driving variable. The minimum weather data requirement is daily values of global radiation, air temperature and precipitation. However, much more detailed information can be utilized by the model, e.g. hourly values of global radiation, air temperature, relative humidity, wind speed, and precipitation.”

“The input to Daisy is daily or hourly weather data (at least precipitation, global radiation, and temperature, much more can be used if available), management information (sow/harvest, tillage operations, as well as data and amounts of irrigation, fertilizer and pesticide applications), and finally soil quality (texture, humus content).” [1]

## Why is automation of daisy important?

Daisy, a soil-plant-atmosphere model package uses the past weather data and plot the graphs to analyze crop production. User requries a weather data file which is provided by IBM research project called “Deep Thunder” as one of the input file for daisy to simulate crop production.
In order to run full simulation, user needs to provide another input file which contains information about plowing date, seed bed preparation date, clay type, fertilizer and other influencing factors for crop production simulation. Based on user inputs for different factors, user gets crop growth and yield information but still user cannot say much about best possible combination of plowing date, fertilizer amount and other paramters which can maximize crop yield. For this, we need to run optimzation program to identify maximum yield parameters.
As per my understanding, running full daisy simulation is manual and a bit tedious to get all results. Along with this, Daisy data processing is operating system dependent [running on windows only]. There are many reasons to automate daisy, Let me list few of them here

A) To make it platform independent
B) Running full end-to-end simulation without any human intervention on any platform
C) Generating simulation results
D) Reducing human error risk

Daisy Automation consist following three steps:
1. Generating daisy file with user parameters
2. Running daisy file
3. Plotting and subplotting for daisy output files

## Why Python

Initially I started looking at output files to understand the data structure. Output files consist of many columns having information about crop height, timestamp, water stress, nitrogen stress and many other contributing factors which can impact crop growth and yield. I then studied the requirements of writing programs to generate Daisy scripts and automating the execution of these scripts.

As per the research project I had to find or write a tool which can generate Daisy scripts based on user input, execute them and then generate required user plots to understand crop growth behaviour with time and other contributing factors.

I started looking at available tools and custom packages to complete all the three tasks but I did not find any existing solution to satisfy project need. After discussing with my project mentor, I started writing the solution in python. Python is an open source programming language which has strong inbuild plotting packages and libraries for file editing. Python works very well across the operating system, hence we opted to build our solution in python.

I was new to python but I accepted the challenge and started looking for resources to learn python and started writing sample codes. When I was writing my script, I had faced many interesting problems such as reading file data as column data file, joining multiple column and creating a column for dates. To resolve such problems, I wrote my script in such a way that it can read the file word by word, and can convert year, month, day and hour column data into dates. In addition, I got exposure to various libraries like Numpy and Scipy.

## 1. Generating daisy file with user parameters:

Daisy programs require setup files which are typically written by the user. The main drawback for these set up files is the requirement of following the syntax of these setup files. The initial aim of the project was to make a user friendly interface through python to provide the input parameters required for the set up file. Upon execution of this python file, the user is asked for input parameters for the crop model that needs to be studied. This program has two advantages:

1. The interface is uniform for both windows and unix environments.
2. The user does not have to learn the syntax of the daisy program to run the crop models.

Some of the input parameters required from the user are fertilizer weight, plowing date, seed bed preperation date, simulation date and clay type which impact crop growth and yield.
As the first part of automation, I am generating the daisy scripts/files with dynamic variable parameters which impact crop growth. Given below, is a sample input file taken from the Daisy library with the input parameters given by the user marked in blue. The python script that I generated fills in these values and completes the daisy script. For example, the sample input file gets the variable parameters like plowing date, seed bed preparation date and fertilizers from the user.

Here is sample input file:
**Sample Input file:**

```
;;; test.dai -- Sample file using the Daisy libraries.
(description "Simulation for use in tutorial.")
;; Use standard parameterizations.
(input file "tillage.dai")
(input file "crop.dai")
(input file "log.dai")
;; Weather data.
(weather default "<Weather data file>")
;; We have some very sandy soil.
(defhorizon Ap FAO3

"Andeby top soil."
(clay <User_Input_Variable> [%])
(silt <User_Input_Variable> [%])
(sand <User_Input_Variable>[%])
(humus <User_Input_Variable>[%])
(C_per_N <User_Input_Variable>[g C/g N])
(dry_bulk_density <User_Input_Variable>[g/cm^3]))
(defhorizon C Ap
"Andeby C horizon."
(humus <User_Input_Variable>[%]))
;; We build the column from the horizons.
(defcolumn Andeby default
"The B.And farm, Andeby, 2002."
(Soil (horizons (<User_Input_Variable>[cm] Ap) (-2.5 [m] C))
(border <User_Input_Variable>  [m])
(MaxRootingDepth <User_Input_Variable> [cm]))
(OrganicMatter (init (input <User_Input_Variable> [kg C/ha/y])
(root <User_Input_Variable> [kg C/ha/y])
(end <User_Input_Variable> [cm])))
```

```
(Groundwater deep))
;; Use it.
(column Andeby)
;; Simulation start date.
(time <User_Input_Variable>)
(manager activity
(wait (at <User_Input_Variable>))
(plowing)
(wait (at <User_Input_Variable>))
(fertilize (mineral (weight <User_Input_Variable> [kg N/ha])
(NH4_fraction <User_Input_Variable> [])))
(wait (at <User_Input_Variable>))

(sow "<User_Input_Variable>"))
(wait (or (crop_ds_after "<User_Input_Variable>" 2.0)
(at <User_Input_Variable>)))
(harvest "<User_Input_Variable>")
(wait (at <User_Input_Variable>))
(fertilize (mineral (weight <User_Input_Variable> [kg N/ha])
(NH4_fraction <User_Input_Variable> [])))
(wait (at <User_Input_Variable>))
(harvest "Grass"
(stub 8.0 [cm]) ;Leave 8 cm stub.
(stem 1.00 [])) ;Harvest everything above stub.
(wait (at <User_Input_Variable>))
(stop))
;; Create these log files.
(output harvest
("N Balance" (when monthly)
(from 0 [m]) (to -1 [m]))
("Crop Production"
(set "$crop" "Spring Barley")
(where "sbarley.dlf"))
(checkpoint (when (at 1987 8 7 6))))
```

## 2. Running daisy file:

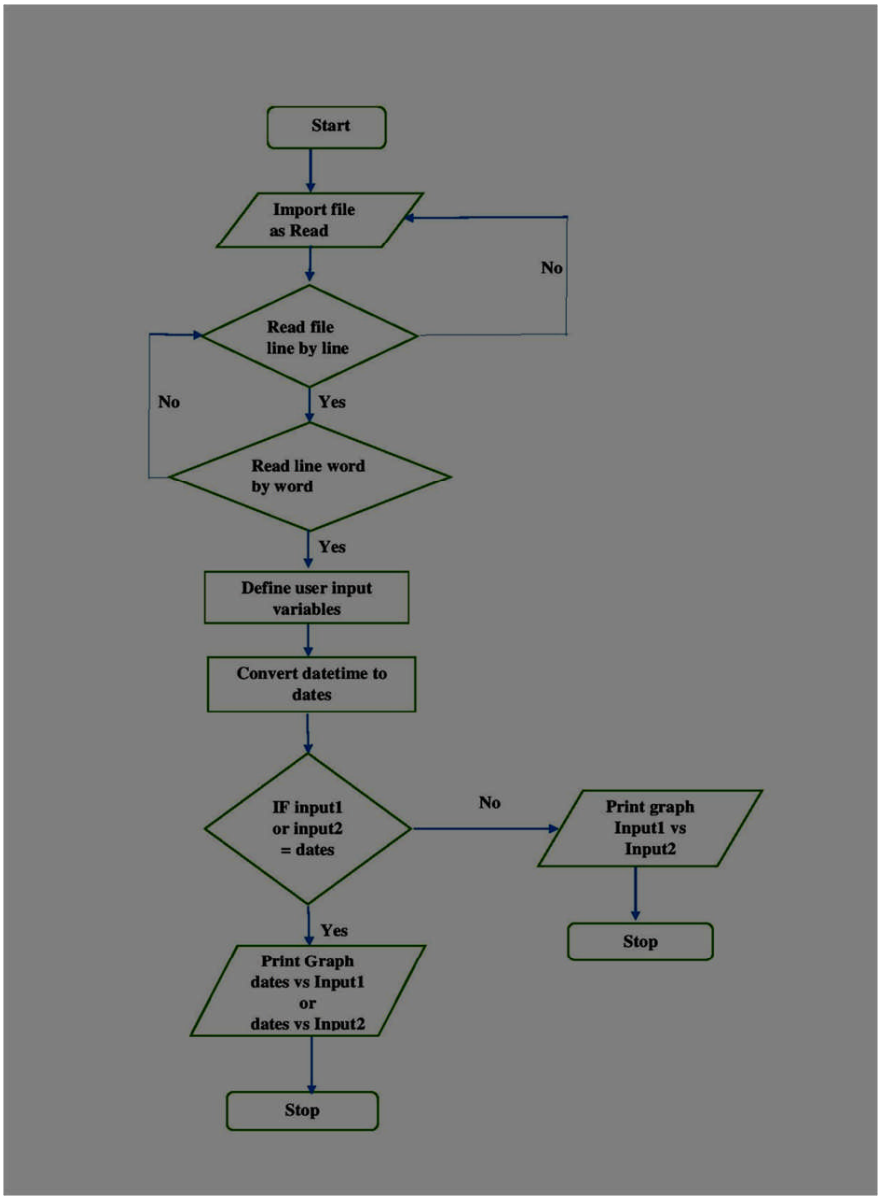
The second part of the automation is to run the daisy script generated above. This was done using the "*subprocess*" command in python. This command takes inputs of the daisy exuecutable path and the daisy script that needs to be run. Using python for execuing the file again makes the automation OS independent even for execution of the script and also ensures a single unifying interface for the whole program. Upon execution of the Daisy script, based on the input parameters, Daisy generates output which contains information about crop growth with time, crop yield, crop height with time, soil water content with time, soil NO3 content with time and many other interesting parameters.
I use these output files in plotting graphs to understand crop growth behaviour with time and other contributing factors, for optimizing the different factors that are effecting crop yield.

## 3. Plotting and subplotting of output files:

The generated output files are text files containing information about crop production, harvest data,

soil data. So, the final stage of automation is to generate user defined plots from the output data generated through the Daisy script.

**Flow chart to understand python script for plotting graphs:**



**Flowchart - 1**

I wrote python scripts which uses one of the output files [generated from daisy] as input file to plot graphs. When user runs the python scripts, it will ask the user to enter information to plot on X and Y-axis, then the script plots graphs according to user inputs. For sub plotting, user will have to enter four inputs two for subplot1 and two for subplot2 as what user wants to plot on X and Y-axis of plot1 and plot2.

For example, if the user inputs are “dates” and “Crop” then user will get “dates vs Crop” Graph

[Fig.1].

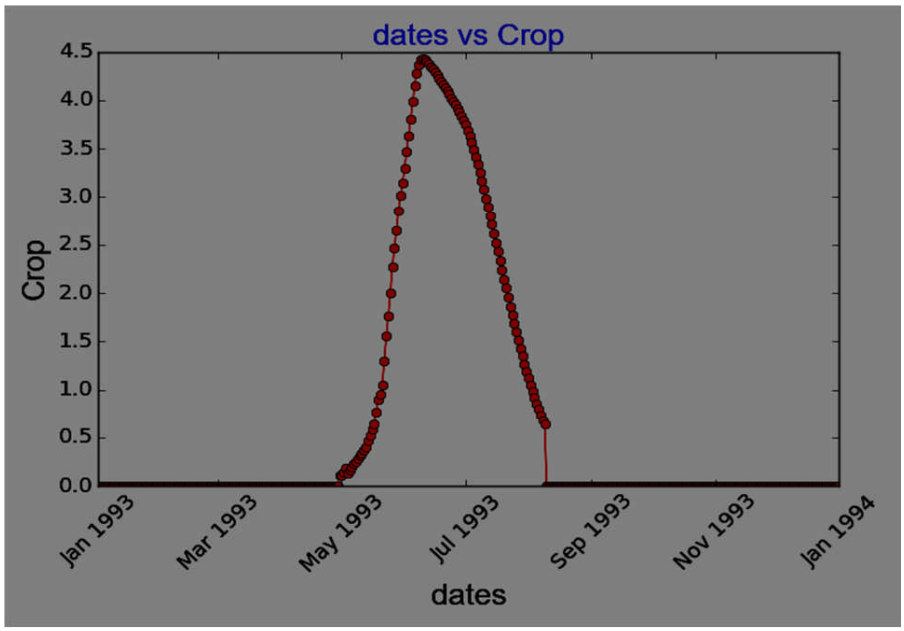

dates vs Crop
Crop
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Jan 1993
Mar 1993
May 1993
Jul 1993
Sep 1993
Nov 1993
Jan 1994
dates

**Fig. 1**

If the user inputs are “NSOrg” and “water_stress” then user will get “NSOrg vs water_stress” graph [Fig.2]. For subplotting, if user inputs for plot1 are “dates” and “Height” and for plot2 inputs are “Depth” and “WLeaf” then user will get “dates vs Height” and “Depth vs Wleaf” graph [Fig.3]. This plotting aspect is automated for all the output variables, with the user not requiring to program in between. The number of output variables are 36 and the program is automated to plot any of the variables on either X or Y-axis.

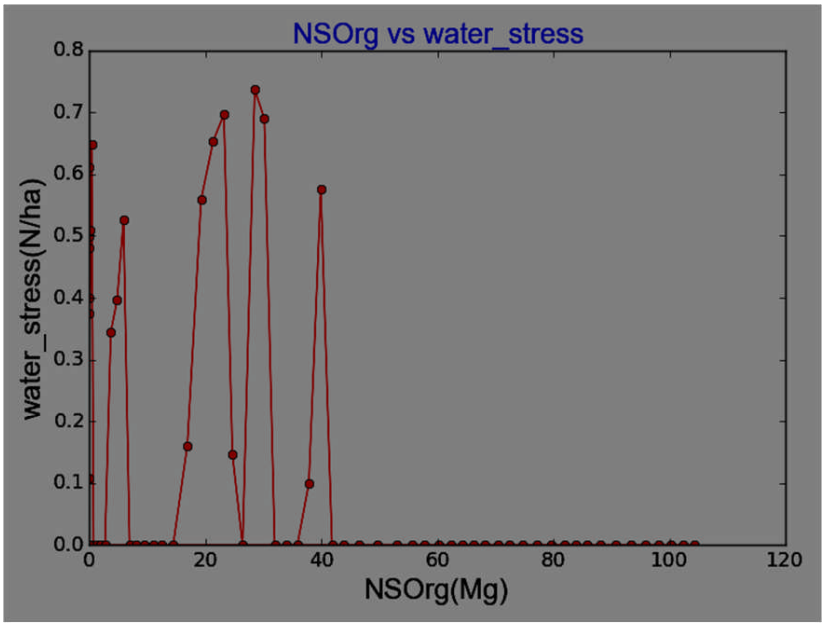



**Fig. 2**

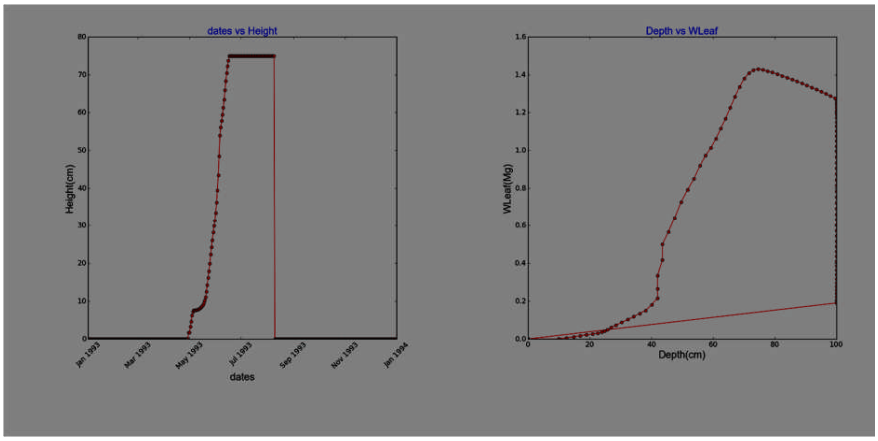
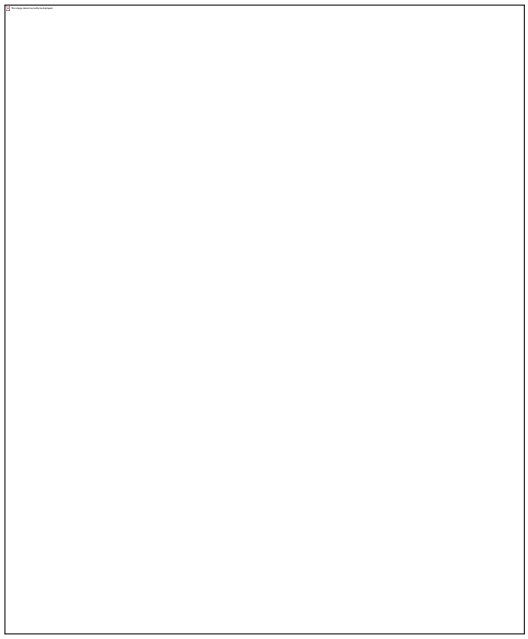

dates vs Height
Height(cm)
dates
Jan 1993
Mar 1993
May 1993
Jul 1993
Sep 1993
Nov 1993
Jan 1994
Depth vs WLeaf
WLeaf(Mg)
Depth(cm)

**Fig.3**



**Flowchart - 2**

Flowchart-2 gives the overall structure of the automation. The user just sees a single unifying python script to execute which performs all the three tasks, namely, generation of Daisy script, execution of Daisy script and plotting of output variables in well formatted pdf format.

## Crop Yield Optimization:

Optimization is performed to get the best possible combination of plowing date, seed bed preparation date, fertilizer date and other contributing factors that are effecting crop growth, to maximize crop yield. We are using genetic algorithm to optimize crop yield.

Genetic Algorithms (GAs) are adaptive methods which may be used to solve search and optimisation problems. Genetic algorithm is a heuristic method based on the principles of natural selection and “survival of the fittest". Genetic algorithms are useful when search space is “very large.” For example, GAs can be used to design bridge structures, for maximum strength/weight ratio, or to Travelling salesman problem. Genetic algorithm solves problem with multiple solutions, it is easy to understand and always provide an answer and answer gets better with time.

Important steps involved in Genetic Algorithm are:

a) Initialization
b) Selection
c) Reproduction
    - Crossover
    - Mutation
d) Termination Criteria

There are many resources that describe the theory and applications of genetic algorithms [3]. In this report, I'll try to describe the various aspects of genetic algorithms with a simple example which I think will provide a more easily understandable introduction to Genetic algorithms.

Example of Genetic Algorithm

$$\textit{Maximize}\ \ \mathbf{f(x)} = \mathbf{x^2}$$

$$\mathbf{0 < x < 6}$$

Consider,
Population size = 3
Crossover probability = .7
Mutation probability= .1

**Step1: Initialization**

Randomly generate initial population of chromosomes (binary strings). Then convert binary strings into relative real value. Each bit of chromosome is called gene.
In genetic algorithm, population size is number of chromosomes in one generation.

Randomly generated set of chromosomes
0 0 0
1 0 1
1 1 0

Calculate decoding value and converting it to its real relative value

$$x = x_{\min} + ((x_{\max} - x_{\min})/(2^l - 1))\mathrm{DV}$$

*l= number of bits in chromosome*
*x= relative real value*

Decoded Value( DV) relative real value ( x)

0 0 0 = $0*2^2+0*2^1+1*2^0 = 0$ 0

1 0 1 = $1*2^2+0*2^1+1*2^0 = 5$ 3.57

1 1 0 = $1*2^2+1*2^1+0*2^0 = 6$ 5.14

**Step2: Selection**

We are calculating value of objective function. Then converting that value to fitness. which is supposed to be proportional to “ability" of the individual which that chromosome represents. More fit chromosome will have more responsibility for being parents than less fit chromosomes. We are replacing less fit chromosomes to more fit chromosomes.

| Chromosome | DV | x | Fitness = f(x) | *Probability | *Expected count | Actual count |
|---|---|---|---|---|---|---|
| 0 0 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 0 1 | 5 | 3.57 | 12.75 | 0.33 | 1 | 1 |
| 1 1 0 | 6 | 5.14 | 26.42 | 0.67 | 2.03 | 2 |
| Sum<br>Avg<br>Max | | | 39.17<br>13.05<br>26.42 | 1<br>.33<br>.67 | | |

**Table -1**

*Probability = *Fitness / Total Fitness*
*Expected Count = *Probability / Average Probability*

Actual count for chromosome1 is 0. Chromosome1 is less fit. As chromosome3 has highest actual count, it will be parent chromosome. Chromosome1 is replaced by chromosome3 (more fit/parent). After replacing, we generate a new set of chromosomes called mutant.

**Step3: Crossover**

Crossover takes two individuals, and cuts their chromosome strings at some randomly chosen position, to produce two “head" segments, and two “tail" segments. The tail segments are then swapped over to produce two new full length chromosomes.

| Mutant | Crossover point | Offspring after crossover | x value | f(x) |
|---|---|---|---|---|
| 1 1 0 | 0 | 1 1 0 | 5.14 | 26.42 |
| 1 0 \|1 | 2 | 1 0 0 | 3.42 | 11.7 |
| 1 1 \|0 | 2 | 1 1 1 | 6 | 36 |
| Sum<br>Avg<br>Max | | | | 74.12<br>24.71<br>36 |

**Table -2**

Crossover probability indicates how many chromosomes are used in reproduction (crossover). If crossover probability is 1, then all chromosomes in population is reproduced. If crossover

probability is 0, then there is no reproduction (crossover) in poputlation.

Crossover is a convergence operation which is intended to pull the population towards a local/global minimum/maximum.

**Step4: Mutation**

Mutation operator is based on mutation probability. It produces valid chromosomes and it is main operator for global search as it checks each and every bit of chromosome. Provided gene needs to be mutated, we make a bit-flip change for gene, i.e., (1 to 0) or (0 to 1).

| Offspring after crossover | Offspring after mutation | x value | f(x) |
|---|---|---|---|
| 1 1 0 | 0 1 0 | 1.71 | 2.92 |
| 1 0 0 | 1 0 0 | 3.42 | 11.7 |
| 1 1 1 | 1 1 1 | 6 | 36 |
| Sum<br>Avg<br>Max | | | 50.62<br>16.87<br>36 |

**Table -3**

Mutation probability indicates how many gene to be mutated ( bit flip change), in population of chromosome.

Mutation is a divergence operation. It is intended to occasionally break one or more members of a population out of a local minimum/maximum space and potentially discover a better minimum/maximum space.

**Step5: Termination Criteria**

Achievement of maximum number of generations as given in problem
No improvement for some generations
A solution is found that satisfies minimum criteria

**Crop Optimization – Specific Example:**

Consider a specific farmer management strategy as given below:

Plowing date - 1st , January
Seed_Bed_Preparation date - 2nd , January
Spring barley harvesting date - 1st , September
Fertilizer amount – 1 [kg N/ha]
Now, the farmer wants to find the best possible fertilizing date for which the crop yield is maximum.

In this case the plowing date, seed bed preparation date and the harvesting dates are fixed and we are trying to find the best possible fertilizer date(when we should use the fertilizer) for which crop yield is maximum.

I am randomly generating set of chromosomes in python script and converting them to their relative real values. Then I am converting these real values to their respective month and day. The computed month and day values are given as input to the daisy file. After replacing date and month, with recently generated date and month value, daisy file is executed to get the harvesting data. In order to automate the full process, we are using the "*subprocess"* function in python script to run daisy software which generates the daisy.log output file. In the next step, we are scanning daisy.log to find harvesting yield data. In order to run genetic algorithms in my optimization program, I used DEAP framework[4].

To find the optimal value for fertilizing date, I ran the python script multiple times with varying population size and number of iterations. For the first iteration, chromosomes are generated randomly and chromosomes for the following iterations are obtained by doing crossover and mutations with the probabilities mentioned below. Program compares each chromosome yield and returns the maximum yield as final output.

Mutation probability = .2
Crossover probability = .9
no. of bits in chromosome = 7

Here is my sample result:

| S. No. | Population size | ngnen ( no of iterations) | Runs (population size*ngen) | Fertilizing date | Harvesting yield |
|---|---|---|---|---|---|
| 1 | 5 | 10 | 50 | 19th , February | 5.04358 |
| 2 | 7 | 10 | 70 | 19th, February | 5.04358 |
| 3 | 10 | 20 | 200 | 17th , February | 5.0437 |

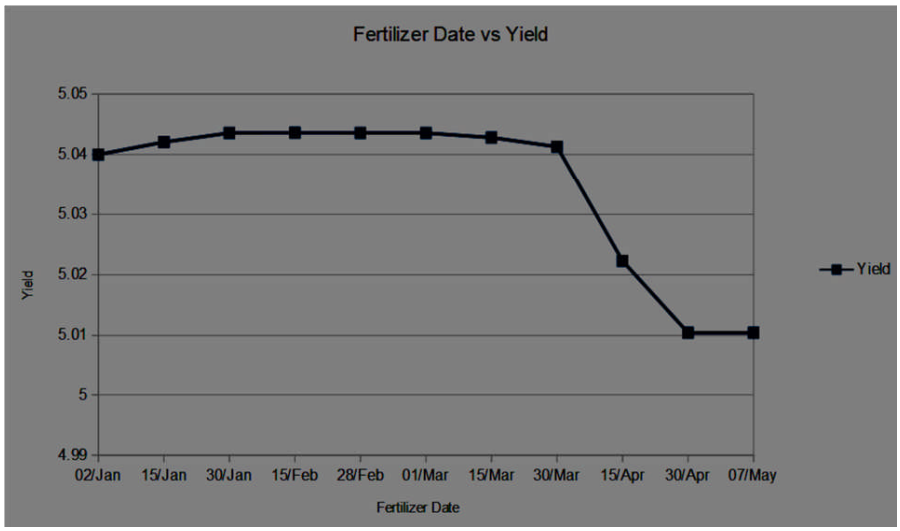
**Table -4**

As number of runs(population size * ngnen) are increasing, we are getting close to optimal solution (see Table-4).

By using genetic algorithm, we get very close to the optimal value in 50 or 70 runs but if we are not using genetic algorithm then it would take 127 runs (number of days in between, we want the optimal fertilizer date to be) to get the optimal value.

To check if I am getting the right result from optimization program, I changed the fertilizing date in daisy file, manually and I got the following result(Table -5).

**Table -5**


Fertilizer Date vs Yield
5.05
5.04
5.03
5.02
5.01
5
4.99
Yield
02/Jan
15/Jan
30/Jan
15/Feb
28/Feb
01/Mar
15/Mar
30/Mar
15/Apr
30/Apr
07/May
Fertilizer Date
Yield

**Future work for this project:**

1) Find the optimal harvesting yield for other parameters.
2) Find the optimal harvesting yield for more than one parameter.
3) Including uncertianties of weather data and other time varying parameterizations for the crop yeild.

## References


[1] Søren Hansen. Daisy, a flexible soil-plant-atmosphere system model.

[2] Lloyd A Treinish, Anthony P Praino, and Zaphiris D Christidis. Implementation of mesoscale numerical weather prediction for weather-sensitive business operations. In *Proceedings of the Nineteenth Intenational Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology*, 2003.

[3] David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning.

[4] DEAP computation framework "*https://github.com/deap/deap*".