CE 602: OPTIMIZATION METHODS

Term Project Report

On

Optimization model on Reservoir based Irrigation system

Under the guidance of:

Prof. Rajib Kumar Bhattacharya

Submitted by:

Jenny Laura RVS (216104106)

Albert E Ngullie (224104602)

Saraswati Harivenu Nair (224104605)

Ila Hari (224104612)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Optimization is defined as an act, process, or methodology of making something (such as a design, system, or decision) perfect, functional, or as effective as possible.The formulation of an optimization problem can be done using two methods, classical and non-classical techniques. The classical optimization method starts with the initial solution and converges to the optimal solution with each successive iteration. This convergence depends onthe selection of the initial approximation. But it is not necessary that we would always have knowledge about the initial solution. In such situations, we use non-classical optimization techniques. Some of these methods are inspired by nature.

Most real-world optimizations are highly non-linear and multimodal, under complex constraints. Different objectives are often conflicting. To solve such problems, we use metaheuristic optimization that deals with metaheuristic algorithms.

Algorithms with stochastic components are often referred to as metaheuristics. We will follow Glover's convention and call all modern nature-inspired algorithms *metaheuristics* (Glover 1986, Glover and Kochenberger 2003). Loosely speaking, *heuristic* means *to find* the solution *by trial and error*. Here *meta-* means *beyond* or *higher level*, and metaheuristics generally perform better than simple heuristics. The word "metaheuristic" was coined by Fred Glover inhis seminal paper (Glover 1986), and a metaheuristic can be considered a "master strategy thatguides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality" (Glover and Laguna 1997). In addition, all metaheuristic algorithms use a certain trade-off of randomization and local search. Quality solutions to difficult optimization problems can be found in a reasonable amount of time, but there is no guarantee that optimal solutions can be reached. Almost all metaheuristic algorithms tend to be suitable for global optimization.

Particle swarm optimization is a metaheuristic technique that helps in solving non-linear non-convex optimization problems. A population-based search algorithm that is inspired by the social behaviour of birds within a flock.

For example, if we consider a group of birds searching for food in an area, and there is only one location where the food is available. This location would be the solution to the problem. The birds would not have any prior knowledge about the location. But they do know how farthe food is from their present location. So, the best strategy here is to follow the bird that is closest to the food.

CHAPTER 2: OBJECTIVE FUNCTION FORMULATION

2.1 Problem Statement:

Consider a reservoir that is on a non-perennial river. The reservoir is used to supply water only for irrigation purposes during the Rabi season. The crop calendar and cropping pattern for the study area has been given in the following tables. The maximum height of the masonry dam is 34.6 m. The gross reservoir storage capacity is 77.84 million cubic meters (MCM), live storage capacity is 70.4 MCM, and dead storage capacity is 7.80 MCM. The Gross Commanded Area GCA is 14,222, and Culturable Command Area CCA is 9848 ha. (Ultimate Potential 12,507 ha) with an intensity of irrigation of 127%.

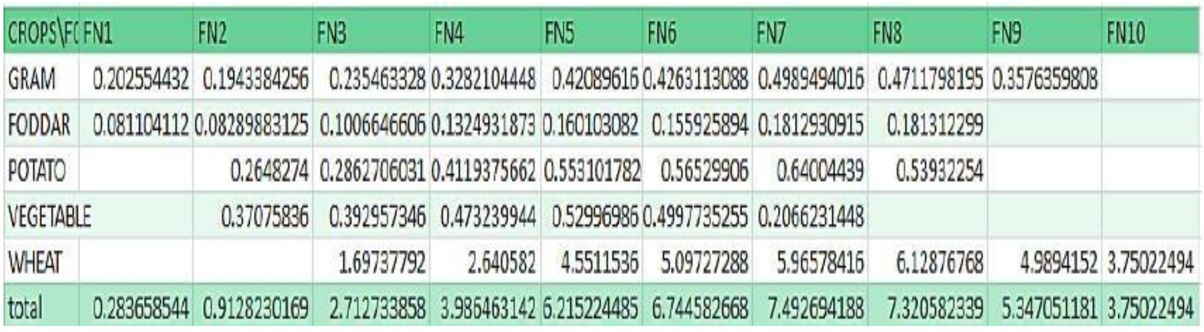
Table 2.1.1: Crop Calendar

Sr. No	CROP	TIME DURATION	DAY
1	GRAM	16 OCT - 28 FEB	135
2	POTATO	1 NOV – 15 FEB	105
3	VEGETABLE	1 NOV – 31 JAN	90
4	WHEAT	16 NOV – 15 MAR	120
5	FODDER	16 OCT – 15 FEB	120

Table 2.1.2: Cropping Pattern

Sr. No.	CROP	%	AREA
1	GRAM	5	492
2	POTATO	6	590
3	VEGETABLE	6	590
4	WHEAT	54	5320
5	FODDER	2	197
	TOTAL		7189

Table 2.1.3: Demand of Crops



CROPS\F(	FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9	FN10
GRAM	0.202554432	0.1943384256	0.235463328	0.3282104448	0.42089616	0.4263113088	0.4989494016	0.4711798195	0.3576359808	
FODDAR	0.081104112	0.08289883125	0.1006646606	0.1324931873	0.160103082	0.155925894	0.1812930915	0.181312299		
POTATO		0.2648274	0.2862706031	0.4119375662	0.553101782	0.56529906	0.64004439	0.53932254		
VEGETABLE		0.37075836	0.392957346	0.473239944	0.52996986	0.4997735255	0.2066231448			
WHEAT			1.69737792	2.640582	4.5511536	5.09727288	5.96578416	6.12876768	4.9894152	3.75022494
total	0.283658544	0.9128230169	2.712733858	3.986463142	6.215224485	6.744582668	7.492694188	7.320582339	5.347051181	3.75022494

2.2 Objective function:

This study developed an optimization model for a reservoir-based irrigation system on a non-perennial river. This reservoir is used to supply water only for irrigation purposes during the Rabi season. The Rabi season lasts from October to March. The crop calendar and the cropping pattern during this period have been provided. The major objective of this optimization model is the maximum efficient utilization of water. This amount must be such that it is sufficient for the healthy growth of crops but at the same, time there should not be excess usage of water. The demand of each crop per unit hectare area for every fortnight has been tabulated and given to us as prior data. The area utilized for each crop has also been provided. As water is released from the reservoir to fulfil the demand for the crops for each fortnight, the storage in the reservoir decreases. The storage level is a function of time where time is measured in the number of fortnights that pass. The difference between the release of water and the demand of crops needs to be minimized.

The objective function is formulated by the equation:

$$\boldsymbol{min\, f(x)} = \sum_{nc=1}^{NC} \sum_{t=1}^{N} ky_t^{nc} \left(\boldsymbol{R_{t,nc}} - \boldsymbol{D_{t,nc}}\right)^2 + \sum_{t=1}^{N} \left(\mathbf{S_t}(1 - \mathbf{B} * \mathbf{e_t}) - \mathbf{S_{t+1}}(1 + \mathbf{B} * \mathbf{e_t}) + \mathbf{I_t} - \sum_{nc=1}^{NC} \mathbf{R_{t,nc}} - \mathbf{A_o} * \mathbf{e_t} \right)^2 \tag{7}$$

Where,

$R_{t,nc}$ = release in the period t = 1, 2..., T for crop nc = 1, 2, ..., NC (MCM)

$D_{t,nc}$ = demand to sustain the crop nc = 1,2,..., NC in the period t=1, 2..., T (MCM)

ky_t^{nc} = yield response factor for time period t= 1, 2..., T of crop nc = 1, 2..., NC

S_t = Storage at time period t= 1, 2..., T (MCM)

S_{t+1}= Storage at time period t+1 (MCM)

I_t = Inflow at time period t= 1, 2..., T (MCM)

A_o and B = Regression constant correlating surface area (Ha) and storage value

e_t= Rate of evaporation at each fortnight in (mm)

2.2.1 Irrigation Demand Constraints

The releases for irrigation should be more than or equal to zero to sustain the crops and also at the same time this should not exceed the maximum irrigation demand to produce the targeted yield

$0 \leq \boldsymbol{R_{t,nc}} \leq \boldsymbol{D_{t,nc}}$

$R_{t,nc}$ is release for irrigation in the period t= 1, 2..., T of crop nc = 1,2,3...,NC

$D_{t,nc}$ is irrigation demand to sustain the crop nc = 1,2,3...,NC in the period t= 1, 2..., T

2.2.2 Reservoir Storage – Capacity Constraints

The reservoir storage in each fortnight should not be less than the dead storage, and should not be more than the live storage of the reservoir

$Smin \leq St \leq Smax$

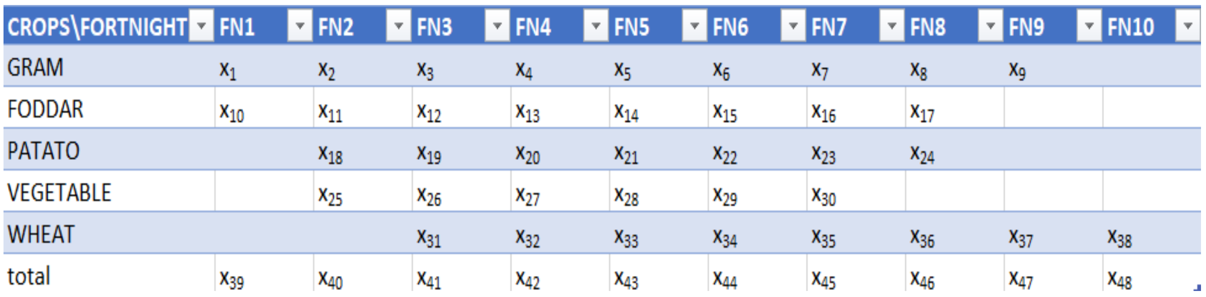
Smax : Live storage of the reservoir in MCM.

Smin : Dead storage of the reservoir in MCM.

Table 2.2.1: Terms using regression constants for each fortnight

FORTNIGHT	(1-B*et/100)	(1+B*et/100)	(A0/100)*et
FN1	0.99325	1.00675014	0.17028
FN2	0.994113	1.00588675	0.1485
FN3	0.994783	1.00521723	0.1316106
FN4	0.994806	1.005194468	0.1310364
FN5	0.994404	1.005595552	0.1411542
FN6	0.99455	1.005449561	0.1374714
FN7	0.993624	1.006375743	0.1608354
FN8	0.993448	1.006552345	0.1652904
FN9	0.993627	1.006373388	0.160776
FN10	0.992151	1.007849	0.198

Table 2.2.2: Variables representing the release of water in each fortnight



CROPS\FORTNIGHT	FN1	FN2	FN3	FN4	FN5	FN6	FN7	FN8	FN9	FN10
GRAM	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	
FODDAR	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}		
PATATO		x_{18}	x_{19}	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}		
VEGETABLE		x_{25}	x_{26}	x_{27}	x_{28}	x_{29}	x_{30}			
WHEAT			x_{31}	x_{32}	x_{33}	x_{34}	x_{35}	x_{36}	x_{37}	x_{38}
total	x_{39}	x_{40}	x_{41}	x_{42}	x_{43}	x_{44}	x_{45}	x_{46}	x_{47}	x_{48}

The function that has been developed, comprises two parts. The first part is the actual objectivefunction and the second part is the penalized function of the constraints. This division into two parts helps in reducing the number of constraints and the run-time of the code developed.

The incorporation of the constraints has been done as an objective function itself, along with the primary objective function that we have formulated. Here, the central idea is to decrease the amount of error between the release and the demand. In total, there are 48 variables from x_1 to x_{48}. Out of which the first 38 variables are representing the release for every fortnight. The last 10 variables denote the storage of water in the reservoir at the end of every fortnight.

For all the variables pertaining to the release of water, the lower bound will be zero and the upper bound will be the maximum demand mentioned in the table. For the storage variables, the lower limit will be the dead storage in the reservoir and the upper limit will be the live storage. After every fortnight, some amount of water has been released, and pending water will be used to fulfil the water demand of the next fortnight. Hence for the n^{th} storage variable, the upper limit will be the lower limit of $(n-1)^{th}$ variable and the lower limit will be the upper limit of $(n+1)^{th}$ variable.

CHAPTER 3: OPTIMIZATION ALGORITHM

3.1 Particle swarm optimization algorithm

Particle Swarm Optimization (PSO) algorithm, inspired by nature, has been introduced by Kennedy and Eberhart. As a fast and potentially good method, this algorithm has been extended to such applications as function optimization, control system, parameters optimization of the fuzzy system and neural network etc.

In this algorithm, each point in the problem space is taken as a particle and all the points as a swarm, like a flock of birds or a school of fish rushing towards the food or their habitats.The particles flow through a multidimensional search space, with a specific position and velocity for the purpose of computing the objective function of an optimization problem. The best personal and global fitness positions are computed over each iteration of running the PSO algorithm. The position and velocity of each particle are updated according to the calculated fitness functions until the optimal solution is obtained.

In studying the behaviour of social animals with the artificial life theory, for how to construct the swarm artificial life systems with cooperative behaviour by computer, Millonas proposed five basic principles (van den Bergh 2001):

- Proximity: the swarm should be able to carry out simple space and time computations.
- Quality: the swarm should be able to sense the quality change in the environment and response it.
- Diverse response: the swarm should not limit its way to get the resources in a narrow scope. (4) Stability: the swarm should not change its behaviour mode with every environmental change.
- Adaptability: the swarm should change its behaviour mode when this change is worthwhile.

Let $x_i(t)$ denote the position of particle *i* in the search space at time t; unless otherwise stated, tdenotes discrete time steps. The position of the particle is changed by adding a velocity, $v_i(t)$, to the current position, i.e.,
$x_i(t+1) = x_i(t) + v_i(t+1)$ (1)

The latter term of velocity will be obtained as;

$v_i(t+1) = w \times v_i(t) + C_1 \times r_1 \times (p_i^{lb} - x_i(t)) + C_2 \times r_2 \times (p_i^{gb} - x_i(t))$ (2)

In equation (2) C_1 and C_2 denote the weighting coefficients for the personal best and global bestpositions, respectively; p_i^{lb} denotes the ith particle's best-known position; p_i^{gb} represents the best position known to the swarm, and r_1 and r_2 denotes a random number between 0 and 1, sampled from a uniform distribution. C_1 and C_2 should be tuned to obtain optimal solutions.

The velocity vector is what drives the optimization process. It comprises of three components;momentum ($w \times v_i(t)$), cognitive ($C_1 \times r_1 \times (p_i^{lb} - x_i(t))$ and social ($C_2 \times r_2 \times (p_i^{gb} - x_i(t))$.

- Momentum Component:
  It means that the particle has confidence in its current moving state and conducts inertial moving according to its own velocity, so parameter ω is called inertia weight.
  This term gives us information about the memory of the previous flight direction. This prevents the particle from changing its direction drastically.

- Cognitive Component:
  The second part depends on the distance between the particle's current position and its own optimal position, called the "cognitive" item. It means the particle's own thinking, i.e., particle's move resulting from its own experience. Therefore, parameter C_1 is called cognitive learning factor (also called cognitive acceleration factor).
- Social Component:
  The third part relies on the distance between the particle's current position and the global (or local) optimal position in the swarm, called "social" factor. It means the information share and cooperation among the particles, namely particle's moving coming from other particles' experience in the swarm. It simulates the move of good particle through the cognition, so the parameter C_2 is called social learning factor (also called social acceleration factor).

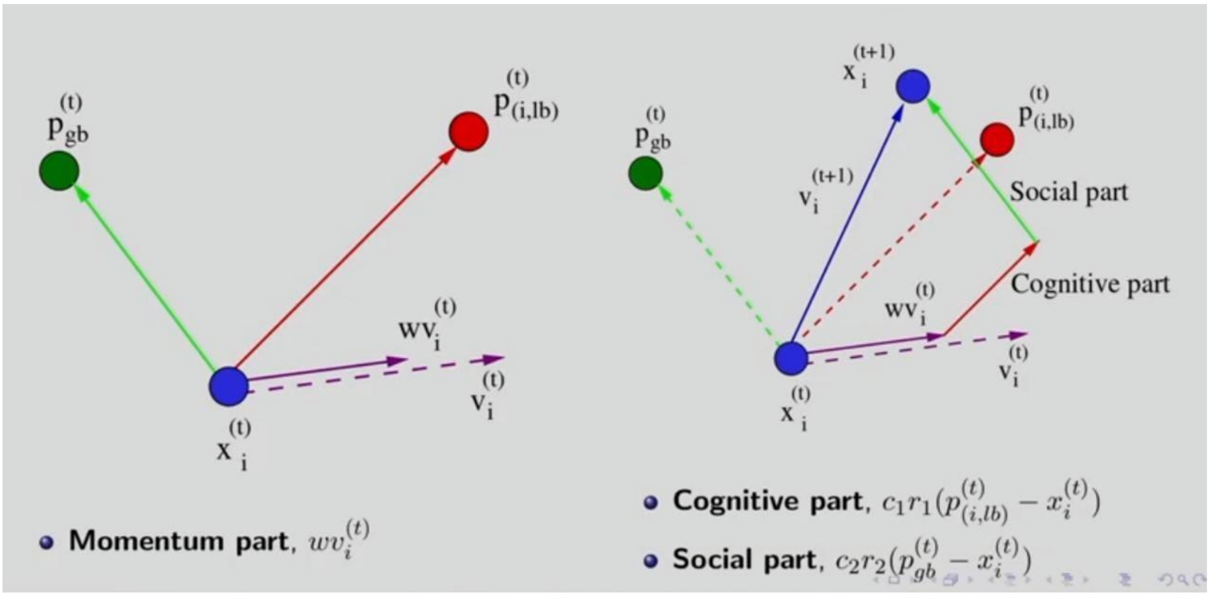

$p_{gb}^{(t)}$
$p_{(i,lb)}^{(t)}$
$wv_i^{(t)}$
$v_i^{(t)}$
$x_i^{(t)}$
Momentum part, $wv_i^{(t)}$
$x_i^{(t+1)}$
$v_i^{(t+1)}$
Social part
Cognitive part
Cognitive part, $c_1r_1(p_{(i,lb)}^{(t)} - x_i^{(t)})$
Social part, $c_2r_2(p_{gb}^{(t)} - x_i^{(t)})$

Figure 3.1.1: Geometric Illustration of Velocity components

Originally, two PSO algorithms were developed, which differ in the size of their neighbourhoods. These two algorithms, namely the *gbest* and *lbest* PSO, are summarized in the following sections.

3.2 Global Best PSO

For the global best PSO, or gbest PSO, the neighborhood for each particle is the entire swarm. The social network employed by the gbest PSO reflects the star topology. For the star neighbourhood topology, the social component of the particle velocity update reflects information obtained from all the particles in the swarm. In this case, the social information is the best position found by the swarm, referred to as p_i^{gb}. Considering minimization problems, the global best position p_i^{gb}, at time step t, is defined as,

$p_i^{gb} \in \{ p_0^{gb}(t), \ldots, p_{ns}^{gb}(t)\} | f(p_i^{gb}(t)) = \min\{f(p_i^{gb}(t)), \ldots, f(p_{ns}^{gb}(t))\}$ (3)

where n_s is the total number of particles in the swarm.

It is important to note that the definition in equation (3) states that p_i^{gb} is the best position discovered by any of the particles so far – it is usually calculated as the best personal best position. The global best position can also be selected from the particles of the current swarm, in which case;

$p_i^{gb} = \min\{f(x_0(t)), \ldots, f(x_{ns}(t))\}$ (4)

3.2.1 Algorithm for gbest PSO

First, we create and initialize an n_x-dimensional swarm;

```
repeat
        for each particle $i = 1, \ldots, n_s$ do
                //set the personal best position
                if $f(xi) < f(p_i^{lb})$ then
                $p_i^{gb} = xi$;
                end
                //set the global best position if $f(p_i^{lb}) < f( p_i^{gb})$ then
                        $p_i^{lb} = p_i^{gb}$;
                end
        end
        for each particle $i = 1, \ldots, n_s$ do
                update the velocity using equation (2);
                update the position using equation (1);
        end
until stopping condition is true;
```

3.3 Local Best PSO

The local best PSO, or lbest PSO, uses a ring social network topology where smaller neighbourhoods are defined for each particle. The social component reflects information exchanged within the neighbourhood of the particle, reflecting local knowledge of the environment. With reference to the velocity equation, the social contribution to particle velocity is proportional to the distance between a particle and the best position found by the neighbourhood of particles.

The local best particle position, p_i^{lb} , i.e. the best position found in the neighbourhood Ni, is defined as

$p_i^{lb} (t + 1) \in \{Ni|f(p_i^{lb} (t + 1)) = \min\{f(x)\}, \forall x \in Ni\}$ (5)

with the neighbourhood defined as

$Ni = \{ p_i^{lb} -n_{Ni}(t), p_{t-nNi}^{lb} (t), \ldots, p_{i-1}^{lb} (t), p_i^{lb}, p_{i+1}^{lb}, \ldots, p_{t+nNi}^{lb} (t)\}$ (6)

for neighbourhoods of size n_{Ni}. The local best position will also be referred to as the neighbourhood's best position.

3.3.1 Algorithm for lbest PSO

```
First, we create and initialize an $n_x$-dimensional swarm;
repeat
    for each particle $i = 1, . . . , n_s$ do
        //set the personal best position
        if $f(xi) < f(p_i^{lb})$ then
        $p_i^{gb} = xi$;
        end
        //set the global best position if $f(p_i^{lb}) < f( p_i^{gb})$ then
            $p_i^{lb} = p_i^{gb}$;
        end
    end
    for each particle $i = 1, . . . , n_s$ do
        update the velocity using equation (2);
        update the position using equation (1);
    end
until the stopping condition is true;
```

3.4 Generalized Framework of PSO Algorithm

```
1. Solution representation
2. Input: t = 1 (Generation counter), Maximum allowed generations = T
3. Initialize random swarm (P(t));
4. Evaluate (P(t)); (Evaluate objectives, constraints and assign fitness
5. While t<=T do
6. Update $x_i^{pbest}$ of each (i) and find $x_i^{gbest}$
7. For (i=1; i<=N, i++) do
8. Update velocity ($v_i(t+1)$)
9. Update position $x_i (t+1)$
10. Evaluate $x_i (t+1)$ and include it in P(t+1)
11. End for
12. t:=t+1;
13. end while
```

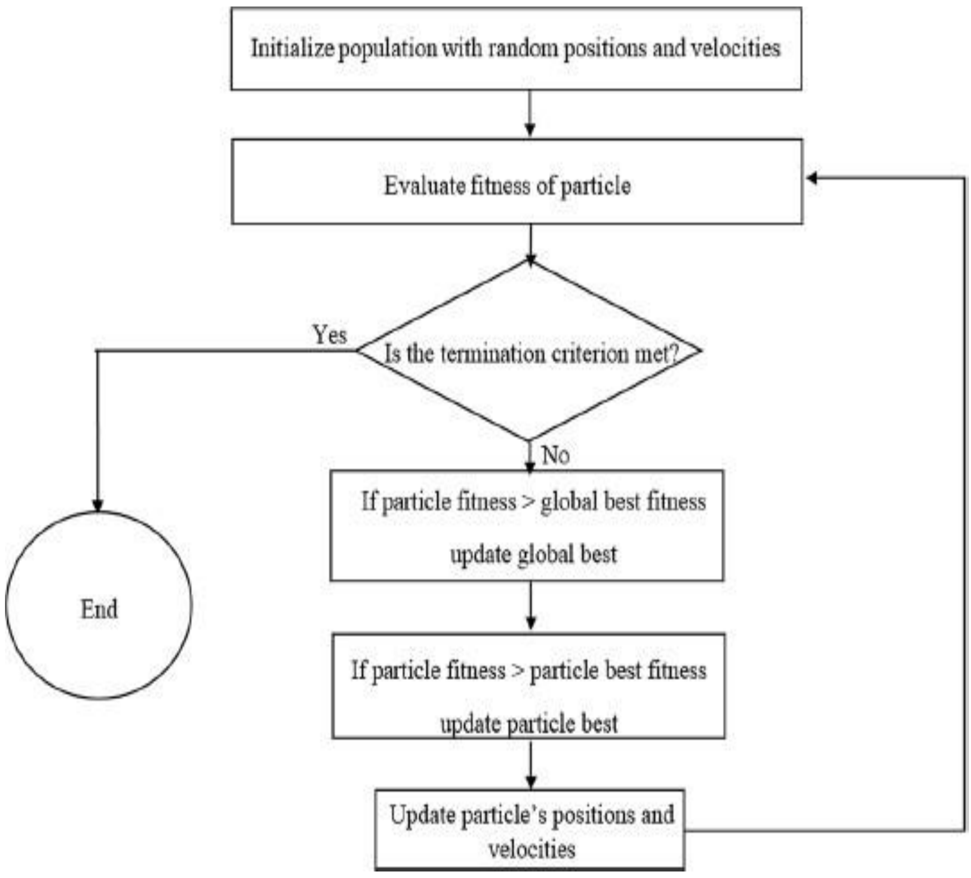

Initialize population with random positions and velocities
Evaluate fitness of particle
Is the termination criterion met?
Yes
No
End
If particle fitness > global best fitness
update global best
If particle fitness > particle best fitness
update particle best
Update particle's positions and velocities

Figure 3.4.1: Flow chart of PSO Algorithm

CHAPTER 4: WORKING OF THE ALGORITHM

The computation of the optimal value of the minimisation problem is carried out in Matlab R2021a. The conceptualisation of the problem involves the generation of individual scripts for the objective function and the algorithm (Particle Swarm Algorithm). The optimization problem has also been solved using f_{min} search algorithm with an optimization tool in Matlab R2021a. The details of the algorithm and the application used are discussed in the section as follows.

4.1. Particle Swarm Optimisation

In the objective function script, the variables have been assigned the bounds within which their respective values are destined to be calibrated. The script also includes the expansion of the objective function with the inclusion of the constants as mentioned below. In the Algorithm Script, the number of decision variables has been set to 61 which includes the sum of release variables (50) and storage variables ((10 +1) variables to represent the remaining water level in the reservoir). The maximum limit of the variable has been set to 70.04 MCM which corresponds to the live storage capacity of the reservoir.

The population count is set to 100 and the number of iterations is set to 100 with inertia weight,w = 1 , wdamp = 0.99 , cognitive learning factor, C1 is set to 1.5 and social learning factor C2is set to 2 which satisfies the constraint of the sum of social and cognitive learning factor is tobe less than 4. Runtime of 10 is provided. The iterative loop initializes the position of the particle and the initial velocity of the particle. The evaluation of the initial position and the velocity is carried through incorporating the cost function of the optimization problem. The personal best and the global best are updated and the current position of the particle is recorded by the algorithm. This process continues till the objective function is minimized. The scripts for the objective function and the algorithm is attached in the sub-sections below. A set of codes have been run corresponding to the varying iteration counts of 100,200,300 and 500 with a varying population count of 100, 200, 300 and 500.

4.1.1 Script for Objective Function

```
function ObjVal=RELEASE2(Foods)

load ('DATA.mat')

%load ('initialvalue.mat') %for fmin algorithm

%%Assigning the values

 x1 = Foods(:,1); x2 = Foods(:,2); x3 = Foods(:,3); x4 = Foods(:,4);

 x5 = Foods(:,5); x6 = Foods(:,6); x7 = Foods(:,7); x8 = Foods(:,8);

 x9 = Foods(:,9); x10 = Foods(:,11); x11 = Foods(:,12); x12 = Foods(:,13);

 x13 = Foods(:,14); x14 = Foods(:,15); x15 = Foods(:,16); x16 = Foods(:,17);

 x17 = Foods(:,18); x18 = Foods(:,22); x19 = Foods(:,23); x20 = Foods(:,24);

 x21 = Foods(:,25); x22 = Foods(:,26);  x23 = Foods(:,27);  x24 = Foods(:,28);

 x25 = Foods(:,32);  x26 = Foods(:,33);  x27 = Foods(:,34);  x28 = Foods(:,35);
```

```
x29= Foods(:,36);  x30 = Foods(:,37); x31 = Foods(:,43); x32 = Foods(:,44);
  x33= Foods(:,45); x34 = Foods(:,46);  x35 = Foods(:,47); x36 = Foods(:,48);
  x37 = Foods(:,49);  x38= Foods(:,50);  x39= Foods(:,51); x40= Foods(:,52);
  x41= Foods(:,53);  x42= Foods(:,54);  x43= Foods(:,55);  x44= Foods(:,56);
  x45= Foods(:,57);  x46= Foods(:,58);  x47= Foods(:,59); x48 = Foods (:,60);
Objective=0.2*(x1-0.202554432)^2+0.317*(x2-0.194338426)^2 ...
    +0.608*(x3-0.235463328)^2+0.813*(x4-0.328210445)^2+0.8*(x5-0.42089616)^2+ ...
    0.666*(x6-0.426311309)^2+0.388*(x7-0.498949402)^2+0.25*(x8-0.47117982)^2+ ...
    0.2*(x9-0.357635981)^2+0.2*(x10-0.081104112)^2+0.25*(x11-0.082898831)^2+ ...
    0.375*(x12-0.100664661)^2+0.525*(x13-0.132493187)^2+0.581*(x14-0.160103082)^2+
 ...
    0.544*(x15-0.155925894)^2+0.515*(x16-0.181293092)^2+0.5*(x17-0.181312299)^2+ ...
    0.45*(x18-0.2648274)^2+0.43*(x19-0.286270603)^2+0.38*(x20-0.411937566)^2+ ...
    0.393*(x21-0.553101783)^2+0.515*(x22-0.56529906)^2+0.585*(x23-0.64004439)^2+ ...
    0.388*(x24-0.53932254)^2+1.1*(x25-0.37075836)^2+1.01*(x26-0.392957346)^2+ ...
    0.83*(x27-0.473239944)^2+0.65*(x28-0.52996986)^2+0.43*(x29-0.499773526)^2+ ...
    0.2*(x30-0.206623145)^2+0.2*(x31-1.69737792)^2+0.32*(x32-2.640582)^2+ ...
    0.52*(x33-4.5511536)^2+0.6*(x34-5.09727288)^2+0.6*(x35-5.96578416)^2+ ...
    0.575*(x36-6.12876768)^2+0.545*(x37-4.9894152)^2+0.5*(x38-3.75022494)^2+ ...
    ((70.04*0.99325)-(x39*1.00675014)-((x1+x10)-0.17028))^2 +...
    ((x39*0.994113)-(x40*1.00588675)-((x2+x11+x18+x25)-0.1485))^2 + ...
    ((x40*0.994783)-(x41*1.00521723)-((x3+x12+x19+x26+x31)-0.1316106))^2 + ...
    ((x41*0.994806)-(x42*1.005194468)-((x4+x13+x20+x27+x32)-0.1310364))^2 + ...
    ((x42*0.994404)-(x43*1.005595552)-((x5+x14+x21+x28+x33)-0.1411542))^2 + ...
    ((x43*0.99455)-(x44*1.005449561)-((x6+x15+x22+x29+x34)-0.1374714))^2 + ...
    ((x44*0.993624)-(x45*1.006375743)-((x7+x16+x23+x30+x35)-0.1608354))^2 + ...
    ((x45*0.993448)-(x46*1.006552345)-((x8+x17+x24+x36)-0.1652904))^2 + ...
    ((x46*0.993627)-(x47*1.006373388)-((x9+x37)-0.160776))^2 + ...
    ((x47*0.992151)-(x48*1.007849)-((x38)-0.198))^2;
```

```
ObjVal=Objective;
End
```

4.1.2. Script for Particle Swarm Optimization

```
clc;
clear;
close all;

tic
load ('DATA.mat')

%% Problem Definition
CostFunction=@(x) RELEASE2(x);
nVar=61;
VarSize=[1 nVar];
VarMin=0;
VarMax= 70.04;
lb=LB;
ub=UB;

%% PSO Parameters
MaxIt=100;
nPop=100;
w=1;
wdamp=0.99;
c1=1.5;
c2=2.0;
VelMax=0.1*(VarMax-VarMin);
VelMin=-VelMax;
runtime = 10;
```

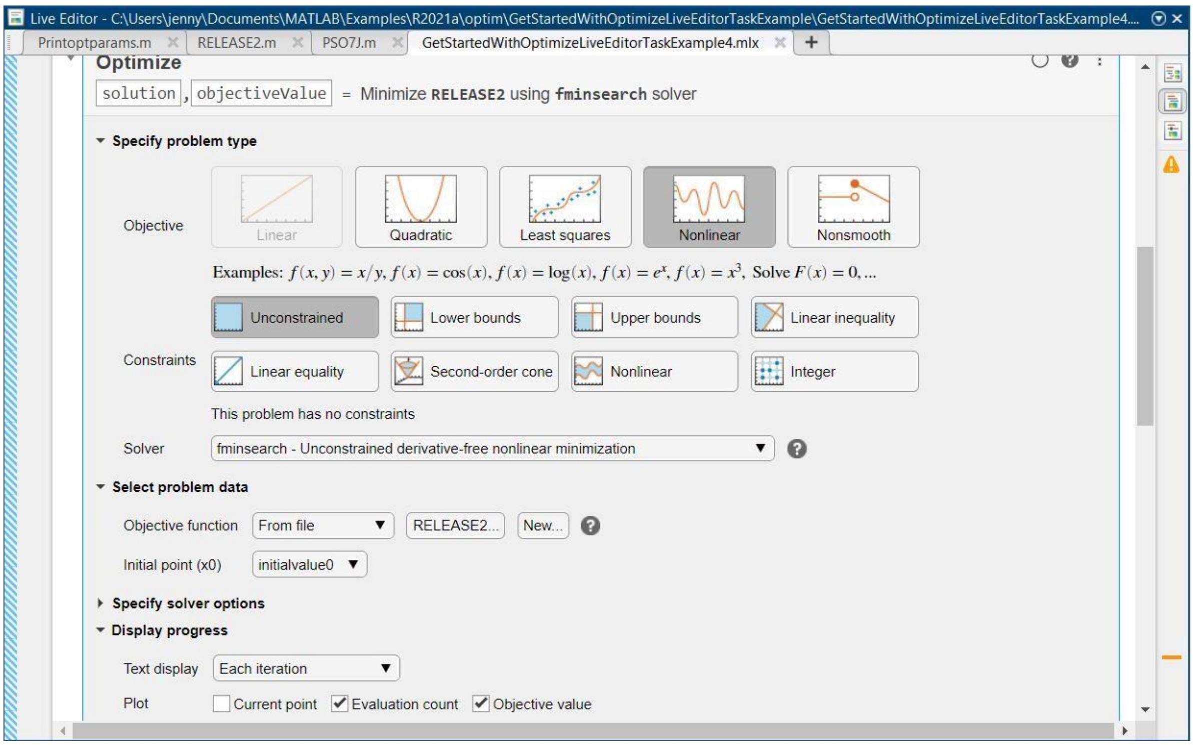
```
for r=1:runtime
%% Initialization
empty_particle.Position=[];
empty_particle.Cost=[];
empty_particle.Velocity=[];
empty_particle.Best.Position=[];
empty_particle.Best.Cost=[];
particle=repmat(empty_particle,nPop,1);
GlobalBest.Cost=inf;
for i=1:nPop

    % Initialize Position
    for j=1:nVar
        particle(i).Position(1,j)=unifrnd(lb(j,1),ub(j,1));
    end

    % Initialize Velocity
    particle(i).Velocity=zeros(VarSize);

    % Evaluation
    particle(i).Cost=CostFunction(particle(i).Position);

    % Updation of Personal Best
    particle(i).Best.Position=particle(i).Position;
    particle(i).Best.Cost=particle(i).Cost;

    % Updation of Global Best
    if particle(i).Best.Cost<GlobalBest.Cost

        GlobalBest=particle(i).Best;
    end

end
BestCost=zeros(MaxIt,1);
%% PSO Main Loop
for it=1:MaxIt


    for i=1:nPop

        % Update Velocity
        particle(i).Velocity = w*particle(i).Velocity ...
            +c1*rand(VarSize).*(particle(i).Best.Position-particle(i).Position) ...
            +c2*rand(VarSize).*(GlobalBest.Position-particle(i).Position);

        % Apply Velocity Limits
        particle(i).Velocity = max(particle(i).Velocity,VelMin);
        particle(i).Velocity = min(particle(i).Velocity,VelMax);

        % Update Position
        particle(i).Position = particle(i).Position + particle(i).Velocity;

        for j=1:nVar
        IsOutside=(particle(i).Position(1,j)<lb(j,1) | particle(i).Position(1,j)>ub(j,1));
        particle(i).Velocity(IsOutside)=-particle(i).Velocity(IsOutside);

        % Apply Position Limits
        particle(i).Position(1,j) = max(particle(i).Position(1,j),lb(j,1));
        particle(i).Position(1,j) = min(particle(i).Position(1,j),ub(j,1));
        end
        % Evaluation
        particle(i).Cost = CostFunction(particle(i).Position);

        % Update Personal Best
        if particle(i).Cost<particle(i).Best.Cost

            particle(i).Best.Position=particle(i).Position;
            particle(i).Best.Cost=particle(i).Cost;

            % Update Global Best
            if particle(i).Best.Cost<GlobalBest.Cost

                GlobalBest=particle(i).Best;
                Globalbestloc=particle(i).Best.Position;

            end

        end

    end

    BestCost(it)=GlobalBest.Cost;
    Glmin(r, it)=BestCost(it);
    GlParams(r,it,:)=Globalbestloc;
    time(r,it)=toc;
    disp(['Iteration ' num2str(it) ': Best Cost = ' num2str(BestCost(it))]);

    w=w*wdamp;
```

```
end

BestSol = GlobalBest;

GlobalMins(r)=BestSol;

end


 save all1j
```

4.2. fminsearch Algorithm

The fminsearch algorithm uses a simplex of n + 1 points for n-dimensional vectors x. The algorithm first makes a simplex around the initial guess x0 by adding 5% of each component $x_0(i)$ to x_0, and using these n vectors as elements of the simplex in addition to x0. (The algorithm uses 0.00025 as component (i) if $x_0(i) = 0$.) Then, the algorithm modifies the simplex repeatedly.

The objective is defined as an unconstrained nonlinear, due to the incorporation of the penalty function in the parent objective function itself. fminsearch has been incorporated from the available list of solver. The objective function script has been included as the problem data input file. The best solutions generated from PSO algorithm corresponding to the minimum value of population and iteration count have been used as initial values as mentioned in figure 4.2.1. and the minimum optimal value of the function is obtained.



Live Editor - C:\Users\jenny\Documents\MATLAB\Examples\R2021a\optim\GetStartedWithOptimizeLiveEditorTaskExample\GetStartedWithOptimizeLiveEditorTaskExample4....

Printoptparams.m | RELEASE2.m | PSO7J.m | GetStartedWithOptimizeLiveEditorTaskExample4.mlx

Optimize

solution, objectiveValue = Minimize **RELEASE2** using **fminsearch** solver

Specify problem type

Objective: Linear | Quadratic | Least squares | [✓ Nonlinear] | Nonsmooth

Examples: $f(x,y) = x/y$, $f(x) = \cos(x)$, $f(x) = \log(x)$, $f(x) = e^x$, $f(x) = x^3$, Solve $F(x) = 0$, ...

Constraints: [✓ Unconstrained] | Lower bounds | Upper bounds | Linear inequality | Linear equality | Second-order cone | Nonlinear | Integer

This problem has no constraints

Solver: fminsearch - Unconstrained derivative-free nonlinear minimization

Select problem data

Objective function: From file | RELEASE2... | New...

Initial point (x0): initialvalue0

Specify solver options

Display progress

Text display: Each iteration

Plot: ☐ Current point ☑ Evaluation count ☑ Objective value

Figure 4.2.1: fminsearch solver for optimization.

CHAPTER 5 - RESULTS

The global minima of the objective function, corresponding to the number of decision variables of 61, which includes the sum of release variables (50) and storage variables (10), have been obtained using Particle Swarm Optimization Algorithm. Also, the optimal value of the same has been generated using the fminsearch Algorithm. The corresponding results are provided in the upcoming sections.

5.1. Particle Swarm Optimization

A) The minimal function value corresponding to the population x iteration values of 100 x 100, 100 x200, 100 x 300, 100 x 500 , 200 x 100, 200 x 200 , 200 x 300, 200 x 500 , 300 x 100, 300 x 200, 300 x 300 , 300 x 500 , 500 x 100, 500 x 200 , 500 x 300 and 500 x 500 are tabulated below.

ITERATIONS	POPULATION			
	100	**200**	**300**	**500**
100	0.792394601	0.179192498	0.149431502	0.242256062
200	0.291731186	0.058215665	8.98E-06	0.188885135
300	0.073359625	0.19306388	0.078610843	0.006093017
500	0.135426317	0.020384081	0.130783575	0.018724316

Table 5.1. 1. Global Minimal values of the optimization function in MCM.

B) The lowest value of the objective function is 8.98E-06 MCM which corresponds to a population value of 300 for the iteration count of 200. The release variables in correspondence to the minimal value is as provided below.

CROPS\FORTNIGHT	FN1	FN2	FN3	FN4	FN5
GRAM	0.202554428	0.194337003	0.235463328	0.328210445	0.419959727
FODDAR	0.081103958	0.082898826	0.100664661	0.132493187	0.158900694
POTATO		0.263199489	0.286270515	0.411926844	0.553005632
VEGETABLE		0.37075836	0.392880964	0.472501106	0.52996986
WHEAT			1.69737792	2.640582	4.551117775

Table 5.1. 2. Release values for the crops for 1 - 5 fortnights in MCM.

CROPS\FORTNIGHT	FN6	FN7	FN8	FN9	FN10
GRAM	0.426302819	0.498949393	0.471179819	0.356986642	
FODDAR	0.155911997	0.180190579	0.180984438		
POTATO	0.564078054	0.63874517	0.539210909		
VEGETABLE	0.499773526	0.206623145			
WHEAT	5.097272875	5.965078106	6.12876768	4.989414506	3.75022494

Table 5.1.3. Release values for the crops for 6 - 10 fortnights in MCM.

C) The storage variables for the minimal objective value of 8.98E-06 MCM is as provided below.

FORTNIGHTS	STORAGE
FN1	67.42371286
FN2	64.15669153
FN3	59.65942769
FN4	52.95832226
FN5	45.81470411
FN6	37.95227985
FN7	30.35033185
FN8	24.81354815
FN9	20.90256012
FN10	36.94731499

Table 5.1.4. Storage Values for fortnights 1 - 10 in MCM.

5.2. fminsearch Algorithm

A) The minimal value of the objective function for 20000 iterations is 0.00875786 as mentioned in the figure below.

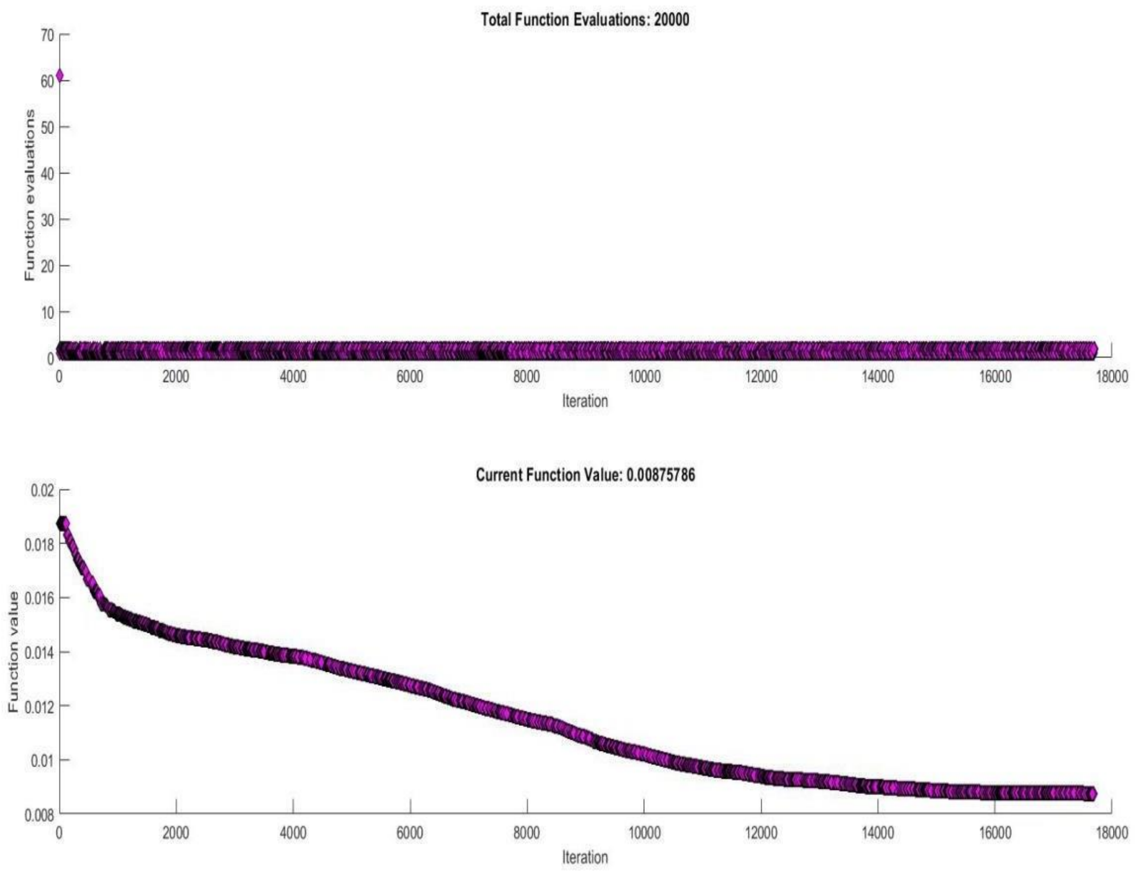


Figure. 5.2.1. Output showing minimal objective value for fminsearch algorithm.

CHAPTER 6: DISCUSSIONS

The minimal value of the objective function for crop yield optimization is carried out with the incorporation of particle swarm optimization for different sets of population values and iteration counts. fminsearch is also used to calculate the same for an iterative count of 20000.

1. For PSO, From table 5.1.1. it is concluded that, despite witnessing some refinement in the optimal value with the increase in the population and iterative counts, the results are still inconclusive about the trend it follows. This is because the PSO algorithm initializes based on random values.

2. The increase in runtime is experienced with the increase in either population or iteration count.

3. As PSO is based on random initial values, every run of the same code generates different optimal values and corresponding release and storage values, which is a disadvantage.

4. fminsearch resulted in the minimum value of 0.00875786 for 20000 iterations, which reveals that with the increase in iterations PSO might also converge to a lower value

REFERENCES

1. **Ghanim M. Alwan** (2016), “Optimization Techniques”,
2. **Mahsa Jahandideh-Tehrani & Omid, Bozorg-Haddad & Hugo A. Loáiciga** (2020), “Application of Particle Swarm Optimization to water management: an introduction and overview”,
3. **Dongshu Wang, Dapei Tan and Lei Liu** (2017), “Particle Swarm Optimization algorithm: an overview”, *Springer-Verlag Berlin Heidelberg 2017*
4. Matlab documentation
5. **Dr. Deepak Sharma**, IIT Guwahati, “Evolutionary Computation for Single and Multi-Objective Optimization”, NPTEL Lecture 8: Particle Swarm Optimization (PS