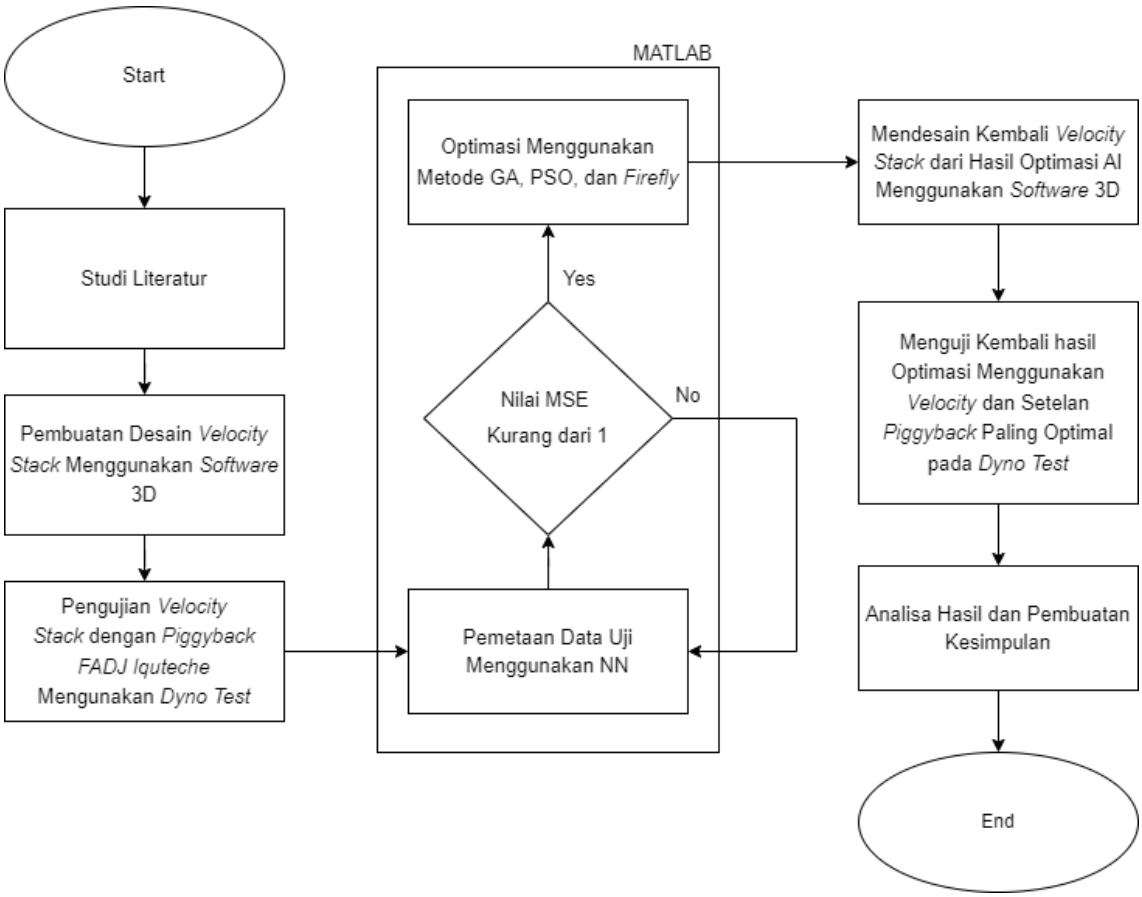
3. METODOLOGI PENELITIAN

3.1 *Flowchart* Penelitian

Berikut merupakan metode penelitian yang akan digunakan untuk optimasi desain *velocity stack* dengan *piggyback* menggunakan metode *artificial intelligence* (AI).



Gambar 3.1 *Flowcart* Penelitian

3.2 Tahap Studi Literatur

Pada tahap studi literatur akan dilakukan pengumpulan data guna memulai penelitian optimasi desain *velocity stack* pada performa Yamaha Aerox 155 dengan *piggyback* dengan menggunakan *Neural Network* (NN) dengan algoritma *Genetic Algorithm* (GA), *Particle Swarm Optimization* (PSO), dan *Firefly Algorithm* (FA). Data yang diperlukan meliputi data spesifikasi kendaraan dan data hasil pengujian performa kendaraan sebelum pemasangan *velocity stack* dan *piggyback*.

Pengambilan data kendaraan akan dilakukan menggunakan *dyno test* yang berada pada Lab. Kometra. Saat melakukan proses *dyno*, kondisi kendaraan harus berada pada kondisi prima agar hasil yang didapat sesuai. Selain itu, pada saat proses *dyno* dilakukan, pastikan bahwa kendaraan sudah terikat dengan baik pada strap penahan agar kendaraan tidak terlepas pada saat proses *dyno* dilakukan. Pada Gambar 3.2 ditampilkan proses melakukan *dyno*.

Gambar 3.2 Posisi Kendaraan Diatas Mesin *Dyno* untuk Proses Pengambilan Data

3.3 Tahap Pembuatan Desain *Velocity Stack* dan Pemasangan *Piggyback*

3.3.1 Desain Velocity Stack

Pembuatan desain *velocity stack* akan dilakukan dengan menggunakan *software* CAD dengan nama Autodesk Inventor Professional 2024. Pada desain yang akan dibuat akan meliputi bagian *inlet diameter, outlet height*, dan *lip radius.* Pada bagian *inlet diameter*, akan dibuat dengan diameter 60mm, 70mm, dan 80mm. Kemudian pada bagian *outlet height* akan dibuat dengan ukuran 60mm, 80mm, dan 100mm. Berikutnya pada bagian *lip radius* akan digunakan ukuran 15mm, 20mm, dan 25mm. Dari rentang pada masing-masing parameter tersebut, dalam pembuatan *velocity stack* akan dilakukan persilangan agar didapat kombinasi dari ketiga variabel dari masing-masing rentang ukuran.

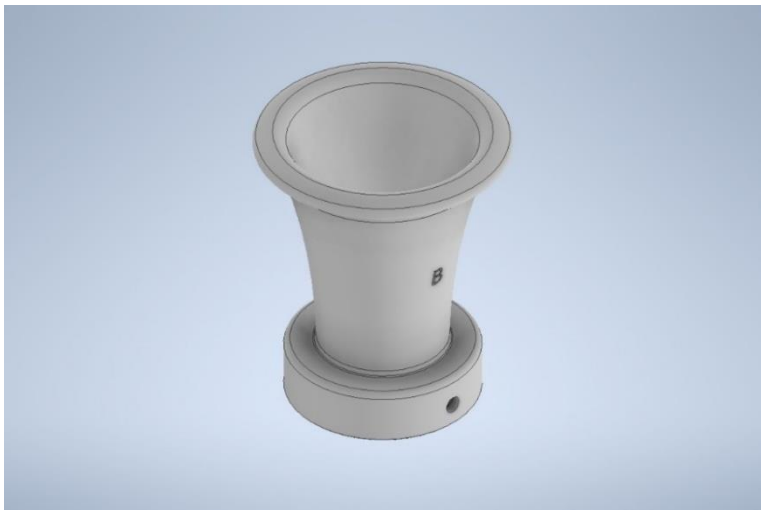
Hasil persilangan dapat dilihat pada Tabel 3.1, dimana dari kombinasi tersebut didapatkan sebanyak 27 desain *velocity stack*. Setelah desain telah selesai dibuat, akan dilakukan proses cetak menggunakan pencetak 3D dengan menggunakan *filament* PLA+.

Tabel 3.1 Kombinasi Desain Velocity Stack

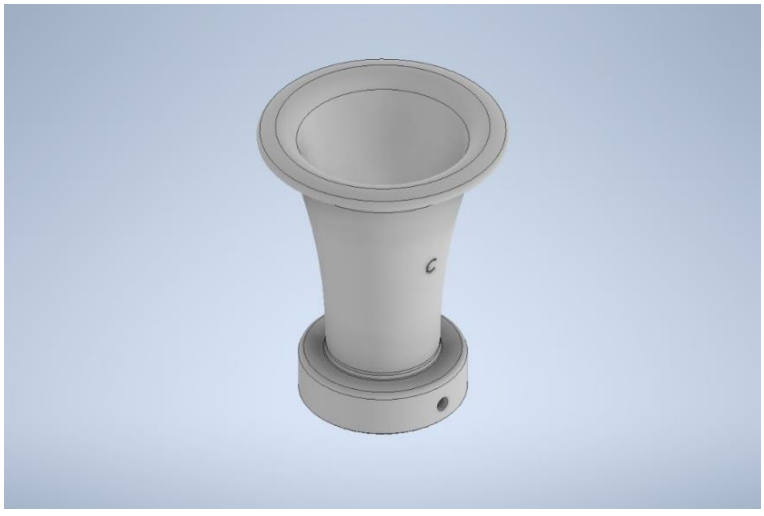
Bagian	Height (mm)	Inlet (mm)	Lip Radius (mm)
A	60	60	15
B	80	70	20
C	100	80	25
D	60	80	25
E	80	60	15
F	100	70	20
G	60	70	20
H	80	80	25
I	100	60	15
J	60	60	25
K	80	70	15
L	100	80	20
M	60	70	15
N	80	80	20
O	100	60	25
P	60	80	20
Q	80	60	25
R	100	70	15
S	60	60	20
T	80	70	25
U	100	80	15
V	60	70	25
W	80	80	15
X	100	60	20
Y	60	80	15
Z	80	60	20
AA	100	70	25

Pada kombinasi yang dilakukan, dapat dilihat untuk contoh parameter *velocity stack* dengan ketinggian yang berbeda-beda pada Gambar 3.3, Gambar 3.4, dan Gambar 3.5.

Gambar 3.3 Contoh Hasil Desain Parameter *Velocity Stack* dengan Ketinggian 60mm



Gambar 3.4 Contoh Hasil Desain Parameter *Velocity Stack* dengan Ketinggian 80mm



Gambar 3.5 Contoh Hasil Desain Parameter *Velocity Stack* dengan Ketinggian 100mm

3.3.2 Pemasangan Parameter *Velocity Stack*

Pada penelitian yang akan dilakukan, perlu mengetahui posisi atau letak pemasangan parameter *velocity stack* agar tidak membahayakan kondisi mesin. Sehingga pemasangan diletakkan pada bagian dalam *filter box*, sehingga udara yang masuk melalui *velocity stack* merupakan udara yang telah tersaring oleh elemen penyaring udara. Pada Gambar 3.5 merupakan letak pemasangan pada bagian *filter box*, sedangkan pada Gambar 3.6 menampilkan posisi pemasangan saat elemen penyaring udara juga terpasang.

Gambar 3.6 Posisi Pemasangan Parameter Velocity Stack

Gambar 3.7 Posisi Parameter Velocity Stack Saat Filter Udara Terpasang

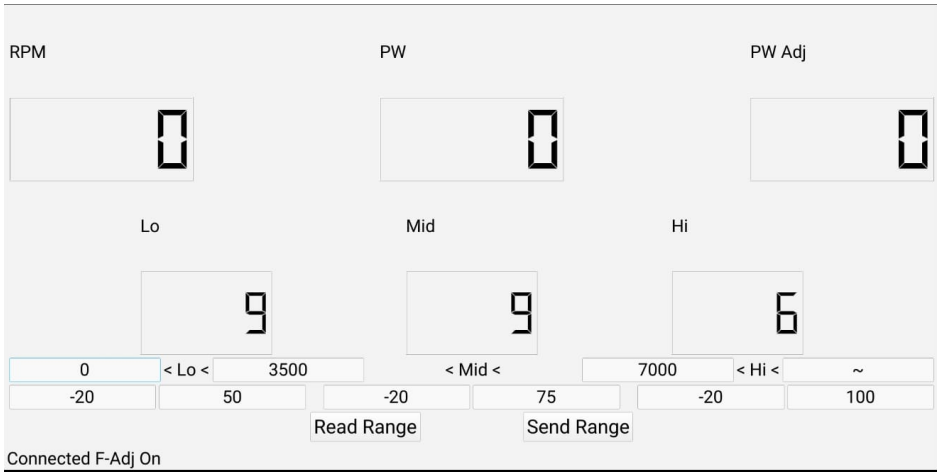
3.3.3 Pemasangan Parameter *Piggyback*

Beberapa parameter *piggyback* yang dapat divariasikan antara lain: low, medium, dan high. Pada variabel low akan disetel pada rentang -1 dan -2. Berikutnya pada variabel medium, akan disetel pada rentang -1 dan -3. Kemudian pada variabel high, akan disetel pada rentang -1 dan -4. Berikutnya dari angka penyetelan akan dilakukan persilangan agar didapat kombinasi dari ketiga variabel. Hasil kombinasi antar variabel pada *piggyback* dapat dilihat pada Tabel 3.2, dimana pada hasil kombinasi tersebut didapatkan 8 data setelan. Pengujian akan dilakukan diatas mesin *dyno test* yang terletak pada Lab. Kometra guna mendapatkan data daya dan torsi.

Tabel 3.2 Data Setelan pada *Piggyback*

Low (%)	Med (%)	Hi (%)
-1	-1	-1
-1	-3	-1
-1	-3	-2
-1	-1	-2
-3	-3	-2
-3	-3	-1

-3	-1	-1
-3	-1	-2



Gambar 3.8 *User Interface* pada Parameter *Piggyback*

3.4 Tahap Pengujian Velocity Stack dan Piggyback

Dalam pengujian yang dilakukan, akan melibatkan hasil desain *velocity stack* dan setelan *piggyback* agar didapatkan sejumlah data. Variasi parameter *piggyback* juga perlu untuk dilakukan dengan tujuan menemukan kombinasi parameter yang optimal untuk meningkatkan performa mesin Yamaha Aerox 155. Kombinasi antara parameter *velocity stack* dengan parameter *piggyback* menghasilkan sebanyak 216 data pengujian.

3.5 Tahap Pemetaan Menggunakan *Neural Network* (NN)

Data hasil pengujian *velocity stack* dengan *piggyback* akan digunakan untuk melakukan pemetaan sebagai hubungan antara variabel masukan dan keluaran yang akan diterapkan pada metode GA, PSO, dan FA. Pada tahap pemetaan, akan didapatkan nilai *Mean Square Error* (MSE). Nilai MSE yang dihasilkan oleh *Neural Network* (NN) akan memiliki hasil yang berbeda-beda, menyesuaikan dengan percobaan kombinasi antara banyaknya jumlah neuron, arsitektur, fungsi aktivasi, dan algoritma pembelajaran. Kemudian untuk dapat melakukan simulasi data menggunakan NN, akan digunakan *software* bahasa pemrograman yaitu MATLAB dengan versi R2023b.

Untuk dapat memasukan data pengujian kedalam NN, data harus dituliskan terlebih dahulu kedalam *software* lembar kerja yaitu Microsoft Excel. Setelah semua data sudah tertulis lengkap pada Microsoft Excel, ekstensi folder penyimpanan harus disimpan dalam bentuk nilai berbatas koma (.csv). Berikutnya untuk folder data pengujian dan folder bahasa program harus menjadi satu folder agar memudahkan untuk mengidentifikasi apabila terjadi masalah saat

dilakukan simulasi pada MATLAB. Untuk penulisan Bahasa program NN, adalah sebagai berikut:

```
%importing data
data = readmatrix('Daya.csv');
p = data(:,1:6)';
tdaya = data(:,7)';
t = tdaya;
```

Data dibaca dari file CSV bernama 'Daya.csv'. Kolom pertama hingga keenam dari data digunakan sebagai input (p), dan kolom ketujuh sebagai target (tdaya).

```
%visualization of the data
histogram (tdaya,7);
title('Daya(HP)');
```

Menampilkan histogram dari data daya (tdaya).

```
%Normalisasi Factor
pfinal = zeros(6,size(p,2));
for i = 1:6
    pfinal(i,:) = (p(i,:)-min(p(i,:)))/(max(p(i,:))-min(p(i,:)));
end
```

Data input (p) dinormalisasi ke dalam rentang 0-1

```
%Train Neural Network
%Iterate Training
perfnew = 100;
f3 = figure;
f4 = figure;
f5 = figure;
for i=1:10
    init_perf = inf;2
    net = feedforwardnet([21 21 21 21]);
    net.layers{1}.transferFcn = 'tansig';
    net.layers{2}.transferFcn = 'logsig';
    net.layers{3}.transferFcn = 'tansig';
    net.layers{4}.transferFcn = 'logsig';
    net.divideParam.trainRatio    = 90/100; %rasio data yang digunakan untuk training
    net.divideParam.valRatio      = 5/100;  %rasio data yang digunakan untuk validasi
    net.divideParam.testRatio     = 5/100;  % rasio data yang digunakan untuk testing
    net.trainParam.epochs         = 10000;  % pilihan pertama u/ stopping parameter
    net.trainParam.goal           = 1e-9;   % pilihan kedua u/ stopping parameter
    [net,tr] = trainlm(net,pfinal,t);       % pfinal = data input, t = target lm
    output = sim(net,pfinal);               % prediksi output BPNN
    perf = perform(net, output, t);         % MSE
    if perf<perfnew
        netnew = net;
        perfnew = perf;

        y = net(pfinal);
        e = t-y;
        figure(f3);
```

```
        ploterrhist(e,'bins',30)
        figure(f4);
        plotperform(tr)
        figure(f5);
        plottrainstate(tr)
    end
end
output = sim(netnew,pfinal);                % prediksi output BPNN
perf = perform(netnew, output, t);          % MSE
view(net)                                   % show net
perfnew;
```

Dilakukan iterasi pelatihan neural network sebanyak 10 kali. Neural network feedforward dengan 4 lapisan tersembunyi (21 neuron tiap lapisan) dibuat. Fungsi aktivasi yang digunakan adalah 'tansig' untuk lapisan tersembunyi dan 'logsig' untuk lapisan output. Pembagian data untuk pelatihan, validasi, dan pengujian diatur. Pelatihan dilakukan dengan algoritma Levenberg-Marquardt ('trainlm'). Performa jaringan dievaluasi dengan Mean Squared Error (MSE). Jika performa jaringan baru lebih baik, jaringan dan performa baru disimpan. Grafik performa, histogram error, dan status pelatihan ditampilkan.

```
%komparasi hasil prediksi dengan data original
numberorder = (1:216);
plot(numberorder,t(1,:),'-b',numberorder,output(1,:),'-r');
title('Ori vs. Prediksi Daya');
xlabel('Data ke-n');
ylabel('Nilai output (%)');
legend('f(x)-ori','f(x)-prediksi daya)');
```

Dilakukan plot untuk membandingkan hasil prediksi dengan data asli.

3.6 Tahap Keputusan Penggunaan Metode GA, PSO, dan FA

Pada tahap keputusan penggunaan metode GA, PSO, dan FA ditentukan dari nilai *Mean Square Error* (MSE) terendah. Apabila nilai MSE berada dibawah 1, maka akan dilanjutkan ke tahap optimasi, jika tidak, maka akan kembali melakukan pemetaan data uji menggunakan NN hingga mendapat nilai MSE dibawah 1. Setelah mendapat nilai MSE dibawah 1, folder yang disimpan adalah netnew. Netnew merupakan data hubungan antar variabel masukan dan keluaran yang akan dibaca oleh metode GA, PSO, dan FA. Untuk masing-masing data netnew dapat di simpan dengan nama folder netnewDaya untuk folder daya, dan netnewTorsi untuk folder torsi.

3.7 Tahap Optimasi Menggunakan Metode GA, PSO, dan FA

Dalam tahap optimasi menggunakan metode GA, PSO, dan FA akan digunakan *software* bahasa pemrograman yaitu MATLAB dengan versi R2023b. Dimana pada tahap ini masing-masing metode akan mengeluarkan hasil desain *velocity stack* terbaik dan hasil setelan *piggyback* yang paling optimal. Untuk masing-masing bahasa program, antara lain:

GA

```
%% initiation GA
nVar=6;                 % Number of Decision Variables

VarSize=[1 nVar];       % Decision Variables Matrix Size

VarMin= 0;              % Lower Bound of Variables
VarMax= 1;              % Upper Bound of Variables
```

nVar: Jumlah variabel keputusan. VarSize: Ukuran matriks variabel keputusan. VarMin dan VarMax: Batas bawah dan atas dari variabel keputusan. Parameter GA seperti jumlah iterasi (MaxIt), ukuran populasi (nPop), probabilitas crossover (pc), probabilitas mutasi (pm), dan lainnya diatur.

```
%% GA Parameters

MaxIt=100;              % Maximum Number of Iterations

nPop=216;               % Population Size

pc=0.5;                 % Crossover Percentage
nc=2*round(pc*nPop/2);  % Number of Offsprings (also Parnets)
gamma=0.4;              % Extra Range Factor for Crossover

pm=0.3;                 % Mutation Percentage
nm=round(pm*nPop);      % Number of Mutants
mu=0.1;                 % Mutation Rate

ANSWER=questdlg('Choose selection method:','Genetic Algorith',...
    'Roulette Wheel','Tournament','Random','Roulette Wheel');

UseRouletteWheelSelection=strcmp(ANSWER,'Roulette Wheel');
UseTournamentSelection=strcmp(ANSWER,'Tournament');
UseRandomSelection=strcmp(ANSWER,'Random');

if UseRouletteWheelSelection
    beta=8; % Selection Pressure
end
if UseTournamentSelection
    TournamentSize=3;   % Tournamnet Size
end

%% Initialization

empty_individual.Position=[];
empty_individual.Cost=[];

pop=repmat(empty_individual,nPop,1);

for i=1:nPop

    % Initialize Position
    pop(i).Position=unifrnd(VarMin,VarMax,VarSize);

    % Evaluation

pop(i).Cost=abs(sim(netnewDaya,(pop(i).Position)'))+abs(sim(netnewTorsi,(pop(i).Posit
ion)'));
```

```
end

% Sort Population
Costs=[pop.Cost];
[Costs, SortOrder]=sort(Costs, 'descend');
pop=pop(SortOrder);

% Store Best Solution
BestSol=pop(1);

% Array to Hold Best Cost Values
BestCost=ones(MaxIt,1);

% Store Cost
Best=pop(1).Cost;
WorstCost=pop(end).Cost;
```

Populasi awal diinisialisasi dengan nilai acak antara VarMin dan VarMax. Evaluasi biaya (cost) setiap individu dalam populasi.

```
%% Main Loop

for it=1:MaxIt

    % Calculate Selection Probabilities
    if UseRouletteWheelSelection
        P=exp(-beta*Costs/Best);
        P=P/sum(P);
    end

    % Crossover
    popc=repmat(empty_individual,nc/2,2);
    for k=1:nc/2

        % Select Parents Indices
        if UseRouletteWheelSelection
            i1=RouletteWheelSelection(P);
            i2=RouletteWheelSelection(P);
        end
        if UseTournamentSelection
            i1=TournamentSelection(pop,TournamentSize);
            i2=TournamentSelection(pop,TournamentSize);
        end
        if UseRandomSelection
            i1=randi([1 nPop]);
            i2=randi([1 nPop]);
        end

        % Select Parents
        p1=pop(i1);
        p2=pop(i2);

        % Apply Crossover
        [popc(k,1).Position,
popc(k,2).Position]=Crossover(p1.Position,p2.Position,gamma,VarMin,VarMax);

        % Evaluate Offspring
        % popc(k,1).Cost=sim(netnew,(popc(k,1).Position)');
        % popc(k,2).Cost=sim(netnew,(popc(k,2).Position)');
```

```
popc(k,1).Cost=abs(sim(netnewDaya,(popc(k,1).Position)'))+abs(sim(netnewTorsi,(popc(k
,1).Position)'));

popc(k,2).Cost=abs(sim(netnewDaya,(popc(k,2).Position)'))+abs(sim(netnewTorsi,(popc(k
,2).Position)'));
    end
    popc=popc(:);


    % Mutation
    popm=repmat(empty_individual,nm,1);
    for k=1:nm

        % Select Parent
        i=randi([1 nPop]);
        p=pop(i);

        % Apply Mutation
        popm(k).Position=Mutate(p.Position,mu,VarMin,VarMax);

        % Evaluate Mutant
        % popm(k).Cost=sim(netnew,(popm(k).Position)');

popm(k).Cost=abs(sim(netnewDaya,(popm(k).Position)'))+abs(sim(netnewTorsi,(popm(k).Po
sition)'));
    end

    % Create Merged Population
    pop=[pop
         popc
         popm]; %#ok

    % Sort Population
    Costs=[pop.Cost];
    [Costs, SortOrder]=sort(Costs,'descend');
    pop=pop(SortOrder);

    %Update Best Cost
    Best=min(Best,pop(1).Cost);

    % Truncation
    pop=pop(1:nPop);
    Costs=Costs(1:nPop);

    % Store Best Solution Ever Found
    BestSol=pop(1);

    % Store Best Cost Ever Found
    BestCost(it)=BestSol.Cost;

end
```

Pada setiap iterasi: Perhitungan probabilitas seleksi. Crossover antara pasangan individu. Mutasi pada beberapa individu. Gabungan antara populasi, offspring dari crossover, dan individu yang mengalami mutasi. Penyortiran populasi berdasarkan biaya. Penyimpanan solusi terbaik yang pernah ditemukan (BestSol) dan biayanya. Truncation: Memotong populasi kembali menjadi ukuran semula. Penyimpanan biaya terbaik setiap iterasi.

```
%% Results
figure;
semilogy(BestCost,'LineWidth',2);
xlabel('Iteration');
ylabel('Optimasi');
grid on;
Daya=sim(netnewDaya,(BestSol.Position)');
Torsi=sim(netnewTorsi,(BestSol.Position)');
```

Plot dari biaya terbaik terhadap iterasi. Prediksi daya (Daya) dan torsi (Torsi) dari solusi terbaik.

PSO

```
Stop_criteria = 1e-12;  % min error
N = 100;                % N jumlah partikel, semakin banyak semakin bagus
maxit = 216;            % maxit = max iteration
dim = 6;                % jumlah parameter/variabel yang mau dicari - x saja
upbnd = 1;              % upper bound
lwbnd = 0;              % lower bound
% assume w = 1 and c1 = c2 = 2
w = 1;
c1 = 2;
c2 = 2;

% initializing location and velocities
x = rand(N,dim)*(upbnd-lwbnd) + lwbnd;
v = rand(N,dim);
f = zeros(N,1);
[brs,kol] = size(x);     % baris kolom

% bobot inersia
for i = 1:brs
    f(i) = (abs(sim(netnewDaya,(x(i,:))')))+(abs(sim(netnewTorsi,(x(i,:))')));
end
it = 1;
Pbest = x;
fbest = f;
[maxf,idk] = max(f);
Gbest = x(idk,:);
lastbest = [];
maxftot = [];

while it < maxit
    r1 = rand;
    r2 = rand;
    for j = 1:brs
        v(j,:) = w*v(j,:) + r1*c1.*(Pbest(j,:)-x(j,:)) + r2*c2.*(Gbest-x(j,:));
        x(j,:) = x(j,:) + v(j,:);
        %adhering nilai pertama
        if x(j,1)>upbnd
            x(j,1)=upbnd;
        elseif x(j,1)<lwbnd
            x(j,1)=lwbnd;
        end
        %adhering nilai kedua
        if x(j,2)>upbnd
            x(j,2)=upbnd;
        elseif x(j,2)<lwbnd
            x(j,2)=lwbnd;
        end
        %adhering nilai ketiga
```

```
        if x(j,3)>upbnd
            x(j,3)=upbnd;
        elseif x(j,3)<lwbnd
            x(j,3)=lwbnd;
        end
        f(j) = (abs(sim(netnewDaya,(x(i,:))')))+(abs(sim(netnewTorsi,(x(i,:))')));
    end

    %update Pbest
    changerow = f>fbest;
    fbest = fbest.*(1-changerow) + f.*changerow;
%     Pbest(find(changerow),:) = x(find(changerow),:);
    Pbest = Pbest.*(1-changerow) + x.*changerow;
    [maxf,idk] = max(fbest);
    maxftot = cat(1,maxftot,maxf);
    Gbest = Pbest(idk,:);
    if sum(var(Pbest)) < Stop_criteria % stop iteration
        break
    end
    it = it+1;
    lastbest = Gbest;
end
disp(['var opt = ' num2str(Gbest(1,1)) ' ' num2str(Gbest(1,2)) ' '
num2str(Gbest(1,3))]);
disp(['Nilai Terbesar = ' num2str(maxf)]);
disp('Daya Terbesar = ');
sim(netnewDaya,(Gbest(1,:))')
disp('Torsi Terbesar = ');
sim(netnewTorsi,(Gbest(1,:))')
plot(maxftot)
grid on
xlabel('Generation');ylabel('Fitnessvalues')
```

FA

```
% Parameter dari Velocity dan Piggyback
% a bp cp^2 Lb Up
data= [0 1
       0 1
       0 1
       0 1
       0 1
       0 1]; % ====> Velocity dan Piggybyack
load = 3500;                    % Total
n_data = length (data(:,1));    % Jumlah Data
mwlimits = data(:,1:2);         % Pmax dan Pmin
Ub = ctranspose(mwlimits(:,2)); % Upper Boundary
Lb = ctranspose(mwlimits(:,1)); % Lower Boundary
```

'data': Array yang berisi data kecepatan (velocity) dan piggyback.

'load': Total beban.

'n_data': Jumlah data.

'mwlimits': Batas atas dan bawah dari data.

'Ub dan Lb': Batas atas (Upper Boundary) dan batas bawah (Lower Boundary).

```
% Parameter Kunang - Kunang
n = 20;
iter = 1;
```

```
iterm = 100;
dim = n_data;
alfa = 0.009;
beta = 0.009;
gama = 0.009;
```

Parameter-parameter algoritma Kunang-Kunang seperti ‘n’ (jumlah kunang-kunang), ‘iter’ (iterasi awal), ‘iterm’ (jumlah iterasi maksimum), ‘alfa’, ‘beta’, ‘gama’.

```
%%%==== Pembangkitkan Data Awal ====

% Inisialisasi Awal Populasi Sesuai Boundaries
for i=1:n
    firefly(i,:)=Lb+(Ub-Lb).*rand;
end
```

Inisialisasi populasi kunang-kunang awal dengan nilai acak sesuai dengan batas atas dan bawah.

```
%%%========Fungsi Objective =====

% Cost function thrtmal generator units
for i=1:n
    %fireflyvalue(i,:)=sim(netnewDaya,(firefly(i,:))');
    fireflyvalue(i,:)=sim(netnewTorsi,(firefly(i,:))');
end
```

Menghitung nilai fungsi objektif (biaya) untuk setiap kunang-kunang dalam populasi.

```
% Pencarian Cahaya Kunang-Kunang Paling Terang Sementara
[soll] = max(fireflyvalue);
bl = find(fireflyvalue==soll);
best = firefly(bl,:);
```

Mencari kunang-kunang dengan biaya terbesar sebagai solusi awal.

```
%%%============== Light Intensity & Attractiveness ====

% Inisialisasi Light Intensity = Kemampuan
for i=1:n
    lightintensity(i)=1/(1+fireflyvalue(i));
end
lightintensity=(lightintensity');
beta=beta*ones(n,dim); % Inisialisasi Beta ke Tiap Firefly

% Inisialisasi Attractiveness
for i=(1:n)
    for k=(1:dim)
        for j=(bl)
            r(i,k)=norm(firefly(i,k)-firefly(j,k)); % Jarak Antara Firefly i dengan
Firefly Paling Terang
```

```
            beta(i,k)=1*exp(-gama*(r(i,k)^2)); % Attractiveness Setiap Firefly
        end
    end
end
```

Menghitung intensitas cahaya dan daya tarik antar kunang-kunang. Atur nilai daya tarik berdasarkan jarak antara kunang-kunang dan kunang-kunang terang.

```
%%%==================Proses Iterasi=================

hfig = figure;
hold on
title('Grafik Konvergensi');
hbestplot = plot(1:iterm,zeros(1,iterm),'-');
xlabel('Iterasi');
ylabel('Biaya');
hold off
drawnow;

while iter <= iterm
    for i=1:n
        maxfirefly(i,:)=0;
        penaltyfactor(i)=0;
    end

    for k=(1:dim)
        for i=1:n
            if lightintensity(bl)>lightintensity(i)
                % Firefly ke-k Pada Populasi i Bergerak Menuju Firefly ke-k Pada
Populasi J
                firefly(i,k)=firefly(i,k)+((beta(i,k))*exp(-
gama*(r(i,k)^2)*(firefly(bl,k)-firefly(i,k)))+(rand*1/2))*alfa;
            end
        % Update Attractiveness Baru
        r(i,k)=norm(firefly(i,k)-firefly(bl,k));
        beta(i,k)=1*exp(-gama*(r(i,k)^2));
        end
    end

 % Cek Jika Firefly Melebihi Boundaries
 for i=(1:n)
     for k=(1:dim)
         if firefly(i,k)>(mwlimits(k,2))
            % Jika Firefly Melebihi Ub, Buat Firefly = Ub
            firefly(i,k)=mwlimits(k,2);
         elseif firefly(i,k)<(mwlimits(k,1))
            % Jika Firefly Dibawah LB, Buat Firefly = Lb
            firefly(i,k)=mwlimits(k,1);
         end
     end
 end



 % Masukkan ke Objective Function
 for i=1:n
    fireflyvalue(i,:)=sim(netnewTorsi,(firefly(i,:))');
 end
```

Program ini menggunakan jaringan saraf tiruan (netnewTorsi) untuk mengevaluasi biaya setiap kunang-kunang.

```
% Ambil Nilai Firefly Paling Terang
[sol_fa]=max(fireflyvalue);
bl=find(fireflyvalue==sol_fa);
best_firefly=firefly(bl,:);

for i=1:n
    lightintensity(i)=1/((i)); % Check Kemampuan Firefly yang Baru
end
plotvector = get(hbestplot,'Ydata');
plotvector(iter)= (sol_fa);
set(hbestplot,'Ydata',plotvector);
drawnow
iter = iter + 1;
end
```

Menampilkan grafik plot

3.8 Tahap Mendesain Kembali *Velocity Stack* dari Hasil Optimasi AI

Pada tahap ini, akan digunakan *software* CAD dengan nama Autodesk Inventor Professional 2024 untuk melakukan desain hasil optimalisasi. Dalam melakukan desain Kembali, ukuran dari *inlet diameter, outlet height*, dan *lip radius* akan mengikuti hasil optimasi desain *velocity stack* yang paling optimal dari masing-masing metode. Untuk hasil yang dikeluarkan oleh masing-masing metode akan berupa angka normalisasi, dimana angka tersebut harus diubah dengan memasukan pada persamaan berikut:

$$\frac{x-x1}{x2-x1} = \frac{y-y1}{y2-y1} \qquad (1.1)$$

Untuk variabel y1, merupakan batas bawah dari data yang diambil, sedangkan untuk variabel y2, merupakan batas atas dari data yang diambil. Kemudian pada variabel x2 merupakan angka hasil normalisasi yang dihasilkan oleh Matlab. Sedangkan untuk variabel x merupakan hasil yang akan digunakan sebagai konfigurasi pada parameter *velocity stack* dan parameter *piggyback.*

Setelah desain telah selesai dibuat, akan dilakukan kembali proses cetak menggunakan pencetak 3D dengan menggunakan *filament* PLA+.

3.9 Tahap Pengujian Hasil Optimasi Desain *Velocity Stack* dan Setelan *Piggyback*

Dalam tahap pengujian hasil optimasi, akan digunakan hasil desain *velocity stack* terbaik dengan setelan *piggyback* yang paling optimal. Pada konfigurasi penyetelan variabel pada *piggyback*, angka yang dikeluarkan oleh masing-masing metode akan berupa hasil normalisasi, dimana angka tersebut akan diubah dengan persamaan pada rumus 1.1. Sehingga

akan didapat *output* berupa daya dan torsi dari hasil optimasi desain *velocity stack* dengan *piggyback* pada Yamaha Aerox 155.

3.10 Tahap Analisa Hasil

Pada tahap analisa hasil, akan dilakukan analisa berupa perbandingan antara hasil prediksi yang telah dikeluarkan oleh masing-masing metode (GA, PSO, dan FA) dengan kondisi pengujian di lapangan. Setelah didapat hasil perbandingan, maka akan dapat dilihat berapa besarnya *error* yang dihasilkan.

3.11 Tahap Pembuatan Kesimpulan

Pada tahap pembuatan kesimpulan, akan dituliskan hasil perbandingan antara hasil prediksi dari masing-masing metode AI dengan pengujian di lapangan. Dari data hasil perbandingan tersebut akan diketahui besaran *error* pada masing-masing metode. Apabila hasil *error* telah diketahui, maka dapat disimpulkan metode mana yang paling sesuai untuk diterapkan dalam proses optimalisasi desain parameter *velocity stack* dengan parameter *piggyback.*