Half Fueled in SCM Code:

%%

clc; clear all;

%% GIS Data Loading

coastline = shaperead('Export\_Output\_3.shp');

coastline.X = deg2km(coastline.X)-5635;

coastline.Y = deg2km(coastline.Y)-2450;

%% Grid Creation

stepkm = 10;

y\_step = 48:stepkm:477-stepkm;

y\_step = y\_step(:);

x\_step = 69:stepkm:665-stepkm;

x\_step = x\_step(:);

x\_temp = [0 0];

lx = length(x\_step);

ly = length(y\_step);

for w = 1:lx

T = [x\_step(w)\*ones(ly,1) y\_step];

x\_temp = [x\_temp; T];

end

%% Region of Interest Processing

q=1;

while q <= length(x\_temp)

if inpolygon(x\_temp(q,1),x\_temp(q,2),coastline.X,coastline.Y) == 0

x\_temp(q,:) = [];

q

else

q

q = q+1;

end

end

%% Model Implementation

i = 1;

n = 1;

count = 1;

x(1,:) = [72 240];

d\_avg = 250;

lrange = 0.99;

urange = 1.01;

m=1;

stp = 0;

while m <= length(x\_temp)

for i=1:size(x,1)

d(i) = dist(x\_temp(m,1),x\_temp(m,2),x(i,1),x(i,2));

end

if size(x,1) == 1

if d>lrange\*d\_avg && d<urange\*d\_avg

x = [x; x\_temp(m,:)];

count = count + 1

m=1;

if count == stp

m = length(x\_temp)+5;

n = length(x\_temp)+5;

end

end

else

[M1,I1] = min(d);

d\_temp = d;

d\_temp(I1) = inf;

[M2,I2] = min(d\_temp);

if M1>lrange\*d\_avg && M1<urange\*d\_avg && M2>lrange\*d\_avg && M2<urange\*d\_avg

x = [x; x\_temp(m,:)];

count = count + 1

m=1;

if count == stp

m = length(x\_temp)+5;

n = length(x\_temp)+5;

end

end

end

m=m+1;

if m == length(x\_temp)+1

n=1;

% m=1;

while n <= length(x\_temp)

for i=1:size(x,1)

d(i) = dist(x\_temp(n,1),x\_temp(n,2),x(i,1),x(i,2));

end

[M1,I1] = min(d);

d\_temp = d;

d\_temp(I1) = inf;

[M2,I2] = min(d\_temp);

if M1>lrange\*d\_avg && M1<urange\*d\_avg && M2>urange\*d\_avg

x = [x; x\_temp(n,:)];

count = count + 1

m=1;

n = length(x\_temp)+5;

if count == stp

n = length(x\_temp)+5;

m = length(x\_temp)+5;

end

else

n = n+1;

end

end

end

end

% New station parameters

new\_station\_min\_distance = 125

% Initializing array to store new stations

new\_stations = [];

% Find midpoints between adjacent existing stations and check if a new station can be placed there

for i = 1:size(x, 1)

for j = i+1:size(x, 1)

% Calculate the distance between the current pair of stations

dist\_between\_stations = norm(x(i,:) - x(j,:));

% Only consider this pair if the distance is less than twice the minimum distance for existing stations

if dist\_between\_stations < 2 \* d\_avg

% Find the midpoint between this pair of stations

mid\_point = (x(i,:) + x(j,:)) / 2;

% Check if the midpoint is within the polygon

if inpolygon(mid\_point(1), mid\_point(2), coastline.X, coastline.Y)

% Flag to check the validity of the new station

valid\_new\_station = true;

% Check the distance between the midpoint and all other new stations

for k = 1:size(new\_stations, 1)

if norm(new\_stations(k,:) - mid\_point) < new\_station\_min\_distance

valid\_new\_station = false;

break;

end

end

% If the new station is valid, add it to the list of new stations

if valid\_new\_station

new\_stations = [new\_stations; mid\_point];

end

end

end

end

end

%% Plotting the results

figure

mapshow(coastline); % Plot the shape data

grid on; grid minor;

hold on

plot(x(:,1),x(:,2),'rx','LineWidth',2) % Plot the existing stations

% Only plot new stations if the array is not empty

if ~isempty(new\_stations)

plot(new\_stations(:,1),new\_stations(:,2),'go','LineWidth',2) % Plot the new stations

end

xlabel('Km')

ylabel('Km')

title('Stations Plotted')

function d = dist(x1,y1,x2,y2)

d = sqrt((x1-x2)^2 + (y1-y2)^2);

end