Dr. Kusum Kumari Bharti

Final Report

8 December 2020

Sales Distribution Using Plant Propagation Algorithm (P.P.A.)

Team Members:

Anshul Singh – 2018301

Satya – 2018069

Sai Kiran - 2018152

Abstract:

These days, optimization problems arising in industry are some of the hardest, often because of the tight specifications of the products involved. They are almost invariably constrained and they involve highly nonlinear, and non-convex functions both in the objective and in the constraints.

Plant Propagation Algorithm (P.P.A) on continuous problems seems to be very competitive. On this selection of problems, our algorithm is found to be as good as and in some cases superior to other algorithms.

In our project we explore the Plant Optimization technique (Strawberry Plant) which will help us find the efficient way for the distribution of the salesperson's goods, to a various people while maintaining the efficiency, during the process.

Introduction:

We know that plant propagation is the process in which we grow new plants from a variety of sources: seeds, cuttings, and other plant parts.

Although there are already many good algorithms and heuristics for Optimization and search problems, Plant propagation algorithm (P.P.A.) which is a global optimization, emulates the strategy that plants deploy to survive by colonising new places which have good conditions for growth. Plants, like animals, survive by overcoming adverse conditions.

Exploration, which is covering the whole search space and **Exploitation** which refer to the property of searching for local optima, near good solutions, are the two main properties that global Optimization algorithms ought to have.

Our Algorithm:

The nature inspired algorithm, i.e., Plant Propagation Algorithm (P.P.A.), which has recently been introduced, emulates the way plants, in particular the strawberry plants, propagates.

The Strawberry algorithm is an exemplar P.P.A. which can be seen as a multipath following algorithm.

Now below we see how a strawberry plant and possibly any plant which propagates through runners will do to optimize its survival.

1) If it is in a good spot of the ground, with enough water, nutrients, and light, then it won’t change its location and will send many short runners that will give new strawberry plants and occupy the neighbourhood as best they can for its survival.

2) If, on the other hand, the mother plant is in a spot that is poor in water, nutrients, light, or any one of these necessary for a plant to survive, then it will try to find a better spot for its offspring. Therefore, it will send few long runners to explore distant neighbourhoods. We can also assume that it will send only a few, since sending a long runner is a big investment for a plant which is in a poor spot.

3) We may further assume that the quality of the spot (abundance of nutrients, water, and light) is reflected in the growth of the plant.

P.P.A. is attractive because, among other things, it is simple to describe and implement. It has been demonstrated to work well on both unconstrained and constrained continuous optimization problems.

Plant propagation optimization in Sales Distribution

The issue here is the representation of a plant which itself depicts a solution. So, we can apply Hamiltonian cycle (tour) of the complete graph representation.

The distance that separates the tours is defined as the number of exchanges to transform one tour into another.

The step-by-step method is described below:

1) We firstly sort the tours by their tour lengths; a pre-determined number of the tours is taken amongst the ones that have good short lengths.
2) Then short runners are sent from these plants, i.e. new neighbouring tours are generated from them. The 2-opt move is then implemented by removing 2 edges from the current tour and exchanging them with the other 2 edges.
3) Similarly, long runners are implemented by applying a k-opt rule with k > 2.
4) If, in this process, shorter tours are preferred and kept, then it will converge to potentially better solutions than it started with.

The parameters used in PPA are the **population size, NP,** which is the number of strawberry plants and **Ni is the normalized form.** The algorithm uses the objective function value at different plant positions **Xi,** where **i = 1,…….,NP**.

After all individuals/plants in the population have sent out their allocated runners, new plants are evaluated and the whole increased population is sorted. To keep the population constant, individuals with lower growth are eliminated. The number of runners allocated to a given plant is proportional to its fitness as in

$$n^i_\alpha = \lceil n_{\max} N_i \alpha \rceil, \quad \alpha \in (0, 1). \tag{1}$$

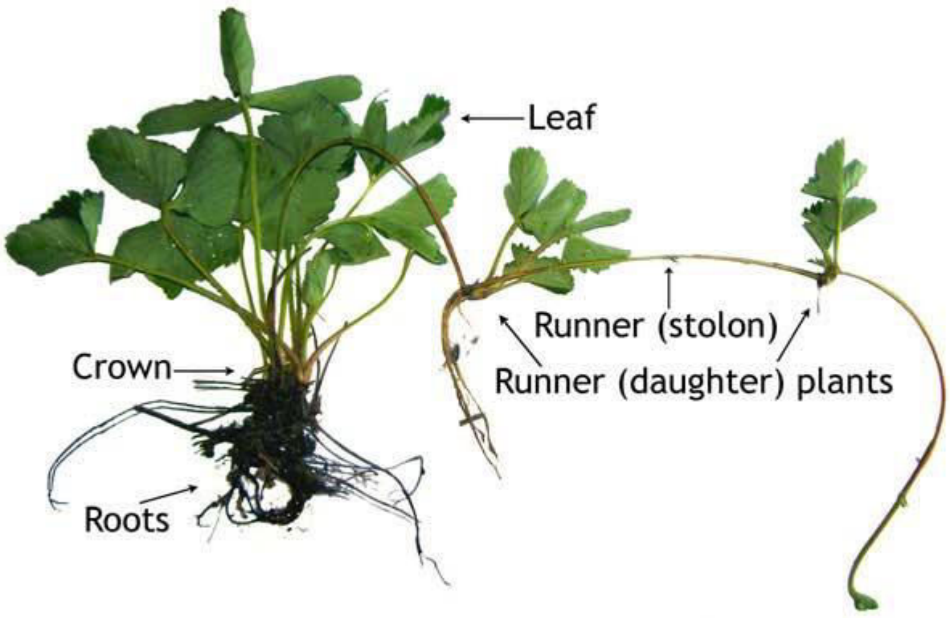
Every solution X_i generates at least one runner and the length of each such runner is inversely proportional to its growth as in (2) below:

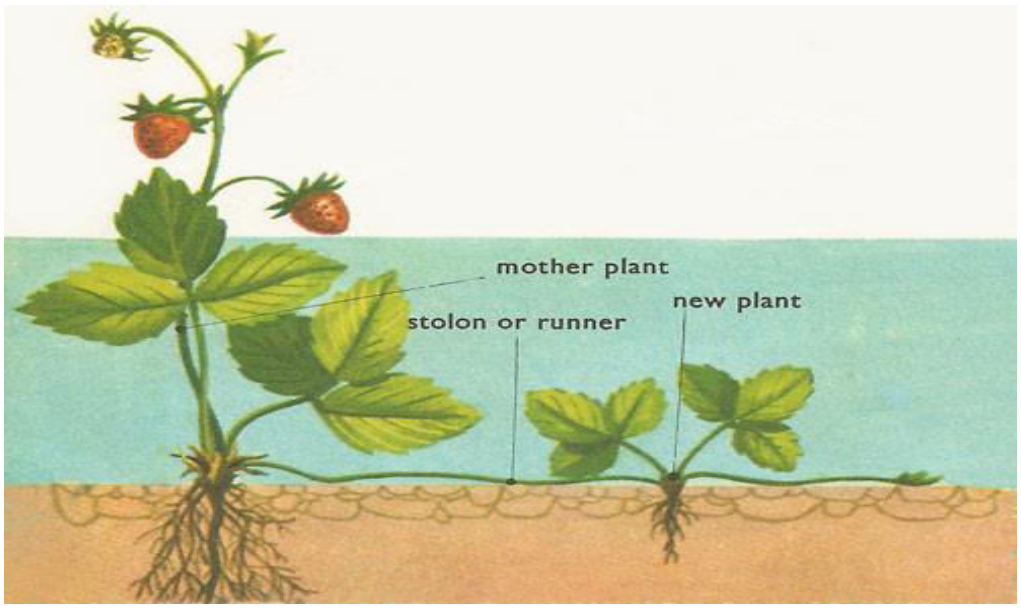
$$dx^i_j = 2(1 - N_i)(\alpha - 0.5), \quad \text{for } j = 1, \ldots, n, \tag{2}$$

where n is the problem dimension. Having calculated dx^i, the extent to which the runner will reach, the search equation that finds the next neighbourhood to explore is

$$y_{i,j} = x_{i,j} + (b_j - a_j)\, dx^i_j, \quad \text{for } j = 1, \ldots, n. \tag{3}$$

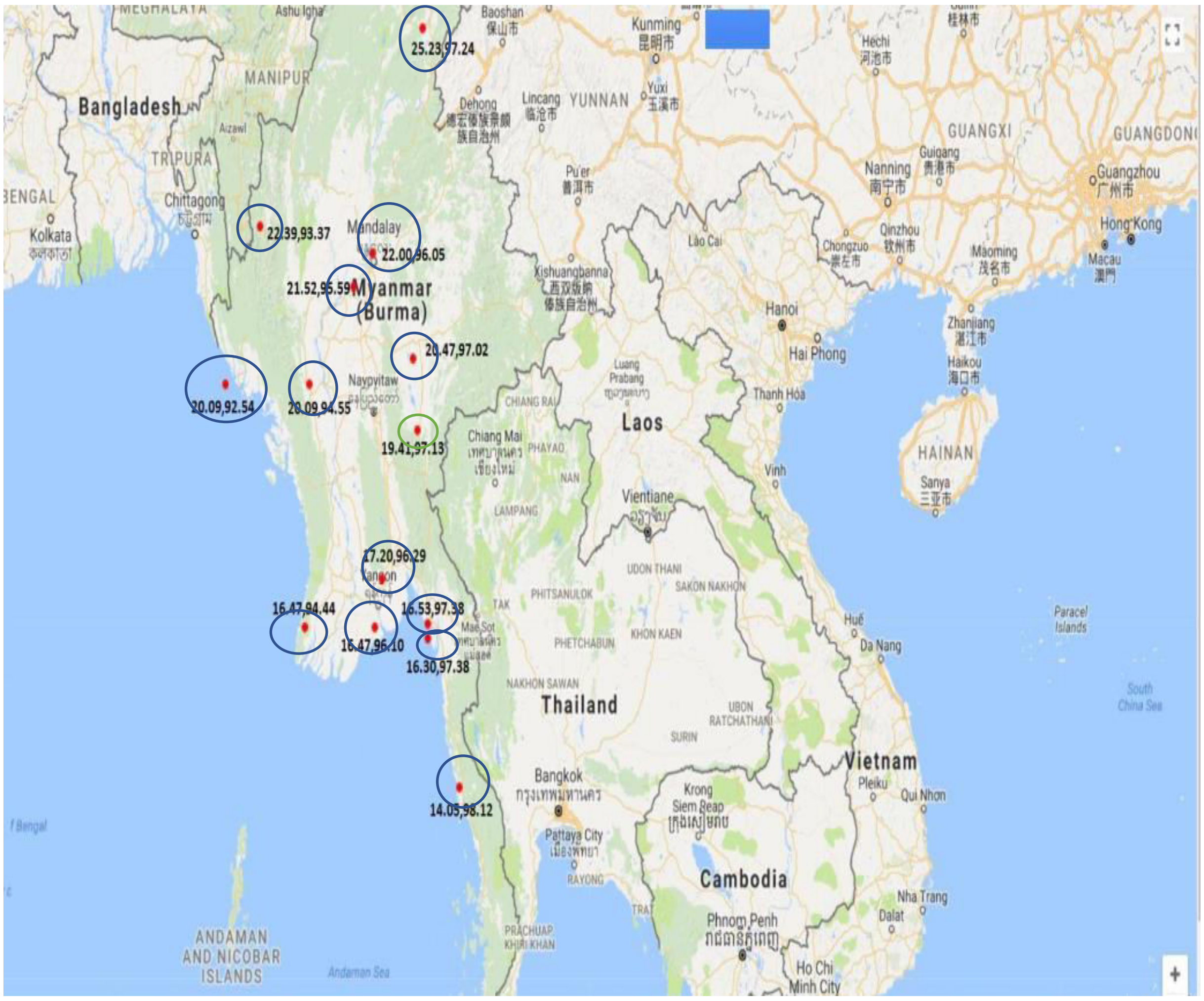
If the bounds of the search domain are violated, the point is adjusted to be within the domain $[a_j, b_j]$, where a_j and b_j are lower and upper bounds delimiting the search space for the jth coordinate.





IMPLEMENTATION AND PROCEDURE:

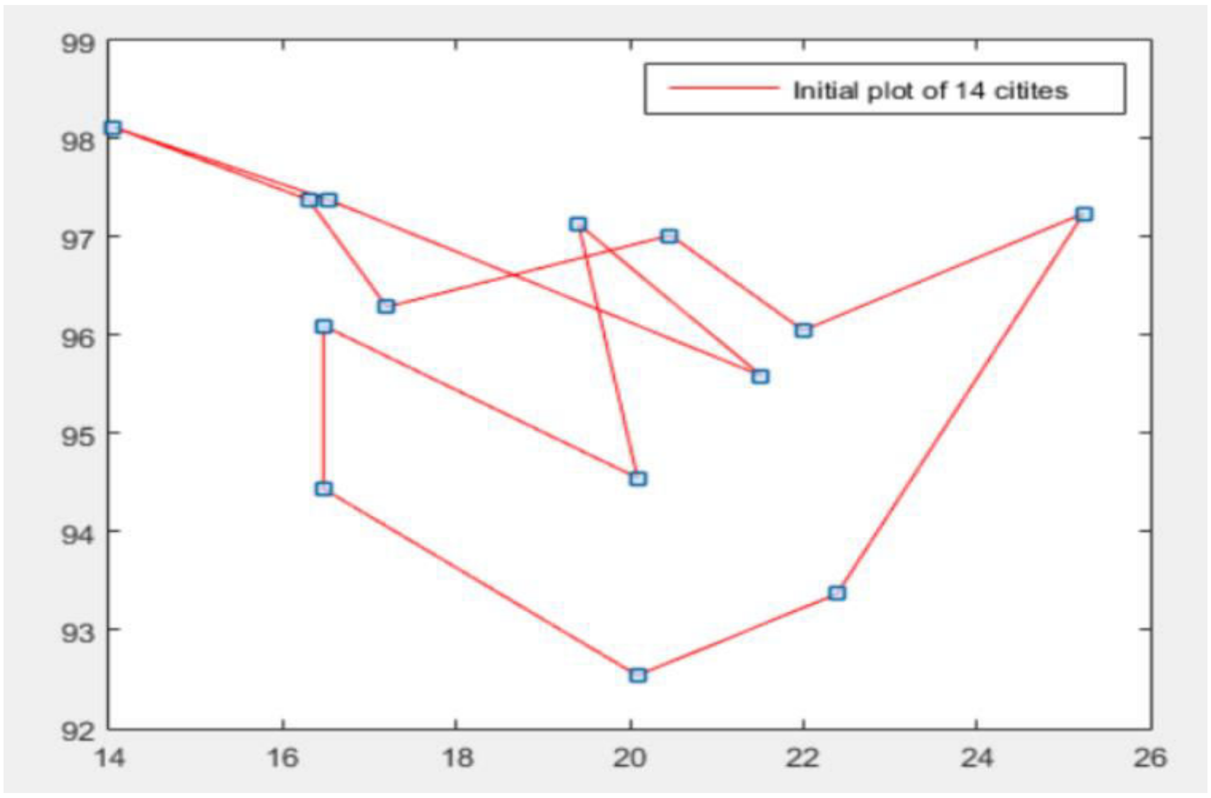
First, on the map, we select any number of points, let's take-14, that we have circled and we take the location of all the points and put in a csv file, so that we can access from the code to work upon it for our problem.



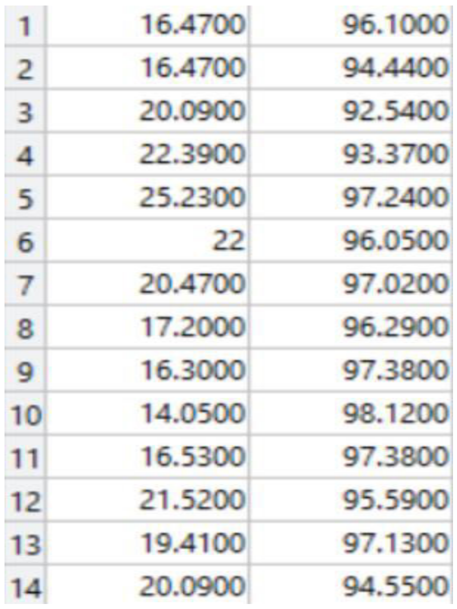
The final circled coordinates on the map are as followed and are in load_data.csv file:

```
1  16.47,96.10
2  16.47,94.44
3  20.09,92.54
4  22.39,93.37
5  25.23,97.24
6  22.00,96.05
7  20.47,97.02
8  17.20,96.29
9  16.30,97.38
10  14.05,98.12
11  16.53,97.38
12  21.52,95.59
13  19.41,97.13
14  20.09,94.55
```

1) **Initially we obtain the lines on the scatterplot for our selected 14 points.**
2) **Then we generate a population; Total Locations = 14, and P = Xi, i = 1, . . . , NP of valid tours where, NP=50.(NP*total_locations)**



OUR C.S.V. FILE



	X	Y
1	16.4700	96.1000
2	16.4700	94.4400
3	20.0900	92.5400
4	22.3900	93.3700
5	25.2300	97.2400
6	22	96.0500
7	20.4700	97.0200
8	17.2000	96.2900
9	16.3000	97.3800
10	14.0500	98.1200
11	16.5300	97.3800
12	21.5200	95.5900
13	19.4100	97.1300
14	20.0900	94.5500

Now we will be exploiting by the short runner first and then with long runners' technique and eventually come up with the final result/Path for our problem.

Short Runner

Main Path: 2 4 8 1 5 7 14 3 10 9 11 13 12 6

Main Path Cost: 44.9611435667493

Short Path: 2 4 8 1 6 7 14 3 10 9 11 13 12 5

Short Path Cost: 47.8227529664884

Long Runner

Main Path: 6 5 13 14 2 4 12 7 10 3 11 9 8 1

Main Path Cost: 48.8701495669961

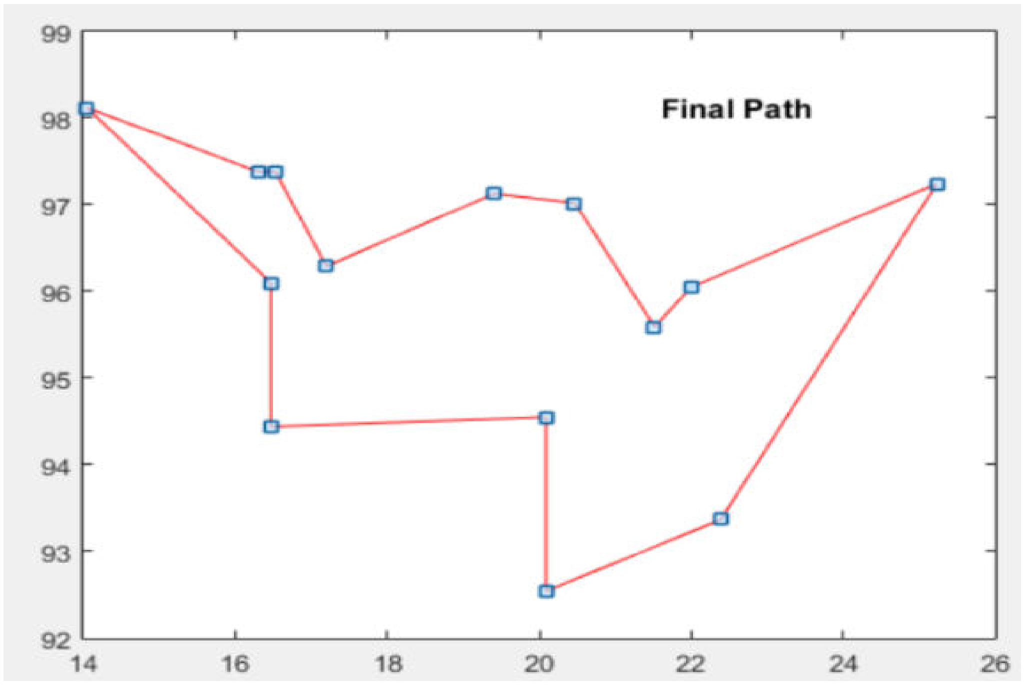
Long Path: 6 14 13 12 2 4 5 7 10 3 11 9 8 1

Long Path Cost: 51.6121726742571

Final Path

Final Path: 6 5 4 3 14 2 1 10 9 11 8 13 7 12

Final Path Cost: 30.8785



Code for P.P.A.: (We have implemented it in MATLAB)

```
%We read the data from the inputs we have selected from the C.S.V. file.
[coordinates] = xlsread('load_data.csv');


%Then we plot the cities
plot(coordinates(:,1),coordinates(:,2));
%find out the rowsize and the column size
[rowsize, colsize] = size(coordinates);
citySize = rowsize;
last = citySize+1;
ic = zeros(last,2);
for i=1:citySize
    %v = r1(i,1);
    ic(i,1) = coordinates(i,1);
    ic(i,2) = coordinates(i,2);
end
%now add the starting city coordinate
ic(last,1) = coordinates(1,1);
ic(last,2) = coordinates(1,2);
%plot the cities
plot(ic(:,1),ic(:,2));
% delcare a matrix for the city distance
data = zeros(rowsize,rowsize);
%now calculate the distance among the cities
for i=1:rowsize
    for j=1:rowsize
        distance = sqrt((coordinates(i,1)-coordinates(j,1))^2+(coordinates(i,2)-coordinates(j,2))^2);
        data(i,j) = distance;
    end
end


N = 50;
m = rowsize; %number of city
```

```
kmax = 100;
y = 50;  %arbitrary value of y
r1 = rand(m,N);

for i=1:N
    r1(:,i)= randperm(m,m);
end
for test_no = 1:kmax
    f = zeros(1,N);
    %Now calculating the path cost
    for a=1:N
        pcost =0;
        j=2;
        loopval = m-1;
        for i=1:loopval
            row = r1(i,a);
            col = r1(j,a);
            pcost = pcost+data(row,col);
            j=j+1;
        end
        first = r1(1,a);
        last = r1(14,a);
        pcostfst_last = data(first,last);
        totalPcost = pcost + pcostfst_last;
        %fprintf('Total path cost of this solution: %f\n', totalPcost);
        f(1,a) = totalPcost; %Adding the path cost of 1st and last
    end
    %sort the cost
    f_sorted = sort(f,'ascend');
    %then sort the solution with respect to the sorted value
    for j=1:N
        r1(:,j) = r1(:,find(f==f_sorted(j),1));
    end

    %Now generate short runners for 10% TOP best solution
```

```
shortVal = N/10;


for i=1:shortVal
    %take an arbitray number of y
    %y = randperm(m,1);
    short = ceil(y/i);
    srunner = rand(m,short); %short runners
    for j=1:short
        rdnNo = randi(m,2);
        %swap the random indexed city to generate new solution from short
        %runners to exploit
        tmp = r1(:,i);     %temporarily keep the column of r1

        swapPos1 = rdnNo(1,1);
        swapPos2 = rdnNo(2,2);
        swap = tmp (swapPos1);
        tmp (swapPos1) = tmp (swapPos2);
        tmp (swapPos2) = swap;
        srunner(:,j)= tmp; %generate short number of short runner

        %calculate the path cost
        q=2;
        tmppcost=0;
            for p=1:loopval
                row = srunner(p,j);
                col = srunner(q,j);
                tmppcost = tmppcost+data(row,col);
                q=q+1;
            end
            first = srunner(1,j);
            last = srunner(14,j);
            pcostfst_last = data(first,last);
            tmppcost = tmppcost + pcostfst_last;
            %fprintf('Total path cost of this short runner: %f\n', tmppcost);
        if f_sorted(i)>tmppcost
```

```
           r1(:,i) = srunner(:,j);
        else
            %Ignore srunner
        end
    end
end

%Now work for long runner

for i=shortVal+1:N
    %lrunner = rand(m,1);
    randomlong = randperm(m,3); %generate the value of k to apply k-opt rule k>2
    lrunner = r1(:,i); %temporarily keep the column of r1
    swap = lrunner(randomlong(1));
    lrunner(randomlong(1)) = lrunner(randomlong(2));
    lrunner(randomlong(2)) = lrunner(randomlong(3));
    lrunner(randomlong(3)) = swap;
    %calculate the path cost
        q=2;
        tmppcost=0;
            for p=1:loopval
                row = lrunner(p,1);
                col = lrunner(q,1);
                tmppcost = tmppcost+data(row,col);
                q=q+1;
            end
            first = lrunner(1,1);
            last = lrunner(14,1);
            pcostfst_last = data(first,last);
            tmppcost = tmppcost + pcostfst_last;
        if f_sorted(i)>tmppcost
           r1(:,i) = lrunner;
        else
            %Ignore srunner
        end
    end
fc = zeros(last,2);
for i=1:rowsize
    v = r1(i,1);
    fc(i,1) = coordinates(v,1);
    fc(i,2) = coordinates(v,2);
end
%now add the starting city coordinate
v = r1(1,1);
fc(15,1) = coordinates(v,1);
fc(15,2) = coordinates(v,2);

plot(fc(:,1),fc(:,2));
fprintf('\n so far obtained best solution is %f\n', f_sorted(1,1));
soln = r1(:,1);
soln = soln.';
display(soln);
end
soln = r1(:,1);
soln = soln.';
display(soln);
display(f_sorted(1,1));
plot(fc(:,1),fc(:,2));
```

Final Results:

We get the final results for our problem, with 14 different locations on the map as selected by us:

The final cost - 30.875

and

The final Path – 6 ,5 ,4, 3, 14, 2, 1, 10, 9, 11, 8, 13, 7, 12

AND

We can see that the result that we have achieved from the scatterplot in the very end is a very optimized result from the Strawberry plant propagation algorithm (P.P.O.), compared to the initial scatterplot shown, that we used by the long runner and short runner technique, on 14 different locations, that helped the distribution of sales in an efficient manner, as much as possible for the salesman.

COMPARATIVE ANALYSIS WITH MODIFIED PARTICLE SWARM OPTIMIZATION (M.P.S.O):

We conducted a second test and the results are compared to those obtained by the Modified P.S.O.

There are four versions of P.S.O. studied by us.

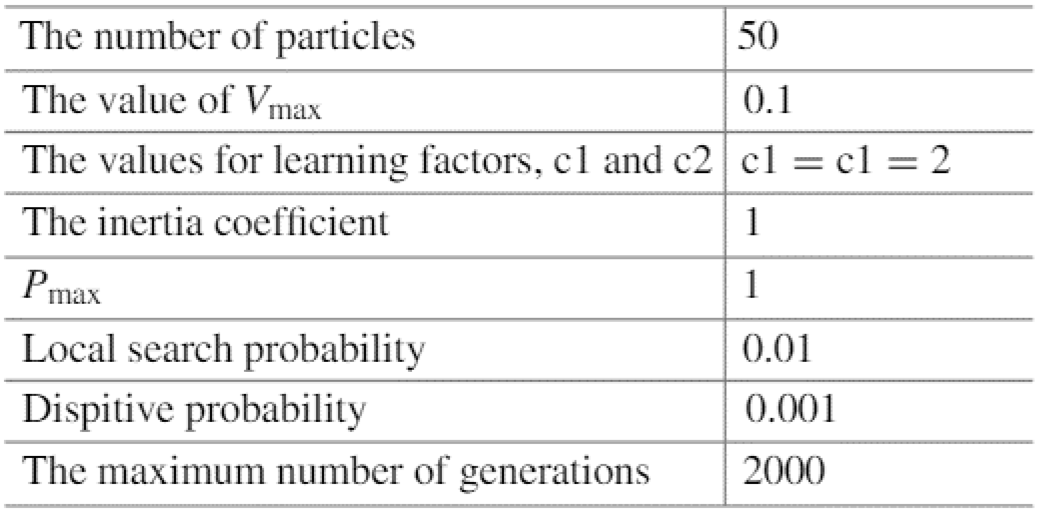
1. PSO-TS: ***The PSO based on space transformation***

2. PSO-TS-2opt: ***PSO-TS combined with 2-opt local search***

3. PSO-TS-CO: ***PSO-TS with chaotic operations***

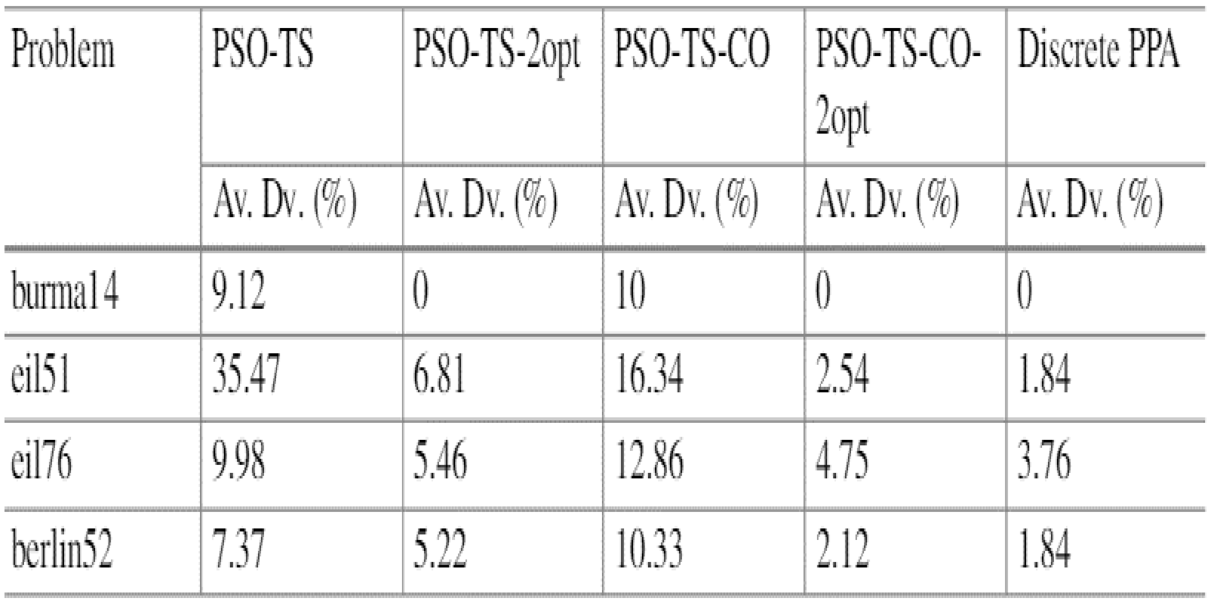
4. PSO-TS-CO-2opt: ***PSO-TS combined with CO and 2-op***

The parameter values used by us in the P.S.O. experiment is listed below:



The number of particles	50
The value of $V_{\max}$	0.1
The values for learning factors, c1 and c2	c1 = c1 = 2
The inertia coefficient	1
$P_{\max}$	1
Local search probability	0.01
Dispitive probability	0.001
The maximum number of generations	2000

All algorithms have been applied to all our instances of 14 different locations. Each algorithm was run 10 times for each problem. The comparison between **All Modified Particle Swarm Optimizations** and **Plant Propagation Algorithm** for solving our problem of sales distribution is described below:



Problem	PSO-TS	PSO-TS-2opt	PSO-TS-CO	PSO-TS-CO-2opt	Discrete PPA
	Av. Dv. (%)	Av. Dv. (%)	Av. Dv. (%)	Av. Dv. (%)	Av. Dv. (%)
burma14	9.12	0	10	0	0
eil51	35.47	6.81	16.34	2.54	1.84
eil76	9.98	5.46	12.86	4.75	3.76
berlin52	7.37	5.22	10.33	2.12	1.84

The problem column is basically the different datasets(inputs) that we took after dividing our original data, for the better understanding of the comparison analysis between the algorithms.

From these experiments and the results, **it is very clear that the Plant propagation Algorithm outperformed all 4 Modified P.S.O. on all instances in terms of solution quality.**

CONCLUSION:

We have implemented Strawberry plant propagation optimization to help the salesperson distribute his goods in an efficient manner as suggested by the algorithm.

The results are compared to those obtained with another algorithm i.e., Modified Particle Swarm Optimization and the results were very clear from the comparative analysis. **Plant Propagation optimization is found either near best known solutions or optimal ones to most of them which have been discovered for this issue.**

Indeed, the recorded evidence in the large majority of cases, points to the overwhelming superiority of Plant Propagation Algorithm. Further improvements and testing are being carried out on a more extensive collection of test problems including discrete ones for a completely optimized result from the algorithm.

CONTRIBUTIONS:

Respected Ma'am,

All 3 of us have worked as a great and hardworking team and have contributed equally in the making of this final report.

In specific:

The research on the Plant propagation algorithm and working on the solution for our problem through the code was done by Anshul Singh (2018301) and Sai Kiran (2018152).

The comparative analysis using the particle swarm optimization and the dataset handling, with an equal hand in research work was done by Satya (2018069).

In overall making of this entire report as very representable; it includes the hand of us all together.

Thank you.

REFERENCES & ACKNOWLEDGEMENTS:

[1] https://www.researchgate.net/publication/252321319_Nature-Inspired_Optimisation_Approaches_and_the_New_Plant_Propagation_Algorithm

[2] Sulaiman,M., Salhi, A., Selamoglu, B.I.,Kirikchi, O.B.: A plant propagation algorithm for constrained engineering optimisation problems. Mathematical Problems in Engineering 627416, 10 pp (2014). doi:10.1155/2014/627416

[3] https://www.semanticscholar.org/paper/A-Seed-Based-Plant-Propagation-Algorithm%3A-The-Model-Sulaiman-Salhi/b9123912695c052ca5654b441ffd314f59e7daf7

[4] https://books.google.com.bh/books?id=TwvNCwAAQBAJ&pg=PA61&lpg=PA61&dq=Sulaiman,M.,+Salhi,+A.:+A+Seed-based+plant+propagation+algorithm:+the+feeding+station+model.+Sci+World+J+(2015)&source=bl&ots=94-_neKNn2&sig=ACfU3U2ALYdZnGDrQt4iJAP4AFqXzN3k6Q&hl=en&sa=X&ved=2ahUKEwio6dOUxK3tAhV-QxUIHS6EAoIQ6AEwBXoECAcQAg#v=onepage&q=Sulaiman%2CM.%2C%20Salhi%2C%20A.%3A%20A%20Seed-based%20plant%20propagation%20algorithm%3A%20the%20feeding%20station%20model.%20Sci%20World%20J%20(2015)&f=false

THE END