APPLYING THE FIREFLY ALGORITHM TO TRAFFIC MATRIX ESTIMATION PROBLEM

Authors: Ali Zaman[1]
[1]Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST), Karachi, Pakistan
zaman.ali1883@gmail.com

Received: 07-February-2021 / Revised: 01-March-2021 / Accepted: 30-December-2021
Karachi Institute of Economics and Technology || Technology Forces Journal, Issue 2, Volume 3, 2021

ABSTRACT

Computational problems of NP nature are of utmost interest to computerscientists as these problems are not easily solvable incorporating deterministic algorithms. Non deterministic approaches which make use of random numbers to solve these problems have become a point of focus for many scientists as they offer a viable solution to many problems of this type. Nature inspired swam intelligence is one such approach which incorporates abstracted algorithms from nature which seems to be constantly solving these types of problems. In this project the focus will be on firefly algorithm which was proposed by Xin She Yang in 2008[1]. The paper which were used for reference was [2] and [3] which applied this algorithm to QAP (quadratic asignment poblem), and used an elephant and mice approach to predict correct traffic estimate respectively. The main focus of this Project is not to get better results but comparable results as it is of exploratory nature. The problem of interest in this paper is Traffic estimation problem which is ubiquitious in the area of networking. Data was used from the Abilene dataset.

Key Words: Genetic Algorithm, Computer Networks, Traffic Matrix Estimation, Traffic Engineering (TE), Quality of Service (QoS)

1. INTRODUCTION

Traffic matrix estimation is an underdetermined problem of linear algebraic nature. In [3] a very interesting approach was proposed to solve this problem. The solution mainly comprised of two steps which can be guessed by the name first elephants than mice. The first step comprised of estimation of the matrix using gaussian approximation the second step was to use this estimate for a local search in its proximity while minimizing the L2 norm. The focus of this paper is replacing the second step with another approach which uses swarm intelligence. Swarm intelligence is a class of algorithms which uses some steps to explore and exploit data gained from random steps to converge towards a good or viable solution for problems of NP nature. Of the many algorithms proposed in this class the one which will be implemented in this paper is the firefly algorithm which was proposed by Xin She Yang [1]. The algorithm according to its creater is inspired by the behavious of fireflies which are attracted to each other by the light which they produce. The factors that are involved in this production and attraction of light are the ability of the firefly and the absorption of light in the environment due to some factors and some random factors which can be modeled but cannot be exactly predicted. So the abstraction of the process implies existence of exploration and exploitation of some solution space which is common to all swarm intelligence algorithms as implies by [4].

In the remainder of this paper the following sequence will be followed. First Problem of traffic matrix estimation will be discussed followed by an explaination of the firefly algorithm, next a discussion will be presented highlighting the issues which were dealt with while applying the proposed solution to the problem in consideration. This will be followed by an explaination of the grid search which was used to search the optimum values of hyperparameters. The conclusion will highlight the limitations and the direction of further mork which can be done.

2. Traffic Matrix Estimation Problem

The problem of traffic estimation can be folrmulated mathematically by the system of Linear equations Y=AX. Here Y represent the link flows i.e traffic from one router to another. This is an observable quantity and the size of Y is m*t where m is the number of links and t is the time stamp for which the traffic ws measures. A is a binary routing matrix of size m*n, where m is the umber of links and n is the number of od flows. X is the traffic matrix of size n*t where n is the numer of od flows and t is the time stamps in Y oer which measurement was performed. In the system of equations Y and A are observable and X is unobservable, secondly Y's m dimension is far less than X's n dimension thus making the problem of under-determined nature.

In [3] it was proposed that because of over dispersion, network traffic cannot be estimated using one distribution (Gaussian, Poisson, Negative binomial). This has resulted because the nature of the type of services has become heterogenious. Over dispersion which makes estimation difficult is more of a problem for mice flows. So in the first step the elephant flows which are closer to Gaussian approximation are estimated then in the second step a bounded value estimation of the result of the first step is used to calculate the over dispered mice flows. This improved prediction by four orders of magnitude.

The data used is from the Abilene dataset. For the first step a generalized linear model is used to calculate regression coefficients relating the link counts and the od flows. In the second step fmincon from matlab is used to minimize the L2 norm of the (predicted values*binary matrix)-(observed values from Abilene dataset) and the L2 norm of the (predicted value) - (regression coefficients) as this represents the elephant flows in our estimation.

Finally the results are represented graphically showing that the prediction is very close to the actual values.

3. Firefly Algorithm

Fireflies for mating purposes use patterns of light to attract each other. This light is generated by a chemical process of bioluminescence. The process is natural, thus allowing for space of the assumption that the patterns and the intensity need not be same in all fireflies and certain intensities and patterns may to lead to a higher probability of being selected for mating thus the probability of certain genes surviving become higher. The environment plays the role on this brightness of light because light gets absorbed in the environment, the light emitted by fireflies is usually only visible in a radius in a few hundred meters. So the farther a firefly is from another firefly the lesser likely it is tht one will be attracted towards another. The the relationship netween light intensity Io and the distance d is an inverse relationsip. And this relationship must also take into account the absorption of the light in the atmosphere. But determinism lies saldomly in nature, and it is also true for this attraction between fireflies so there is a certain unexplainable randomness in the behavious as well.

Allowing the brightness to be replaced by a value that corresponds to the result of some minimization or maximization function, and replacing fireflies with input values to those function transforms this natural process into one which can be modelled in a program. This abstraction allows us to see the firefly algorithm to search in a solution space that can consisit of intital random solutions which eventually converge to a better solution. The algorithmic representation is as follows[1].

Firefly Algorithm

Objective function $f(\mathbf{x})$, $\quad \mathbf{x} = (x_1, ..., x_d)^T$
Generate initial population of fireflies $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
Light intensity I_i *at* $\mathbf{x}_i$ *is determined by* $f(\mathbf{x}_i)$
Define light absorption coefficient γ
while $(t < MaxGeneration)$
for $i = 1 : n$ *all n fireflies*
  for $j = 1 : i$ *all n fireflies*
    if $(I_j > I_i)$, *Move firefly i towards j in d-dimension;* **end if**
    Attractiveness varies with distance r via $\exp[-\gamma r]$
    Evaluate new solutions and update light intensity
  end for j
end for i
Rank the fireflies and find the current best
end while
Postprocess results and visualization

Attractiveness is calculated as (Io/(1+absorption*distance^2))

The new solution is calculated as follows

att = attractiveess,fi = firefly(i), fj = firefly(j), e = exploration coeffecient, r = random generated firefly, nf = new firefly

nf = (1-att)fi +(att)fj+(e)r

4. Applying the Algorithm to Traffic Matrix Estimation

Initially the algorithm was applied to the dataset without using the generalized linear model. What was realized was that the solution space is too large to be searched for without an intital estimate. So the glm solution had to be incorporated in the algorithm as well. For this purpose an alteration was made such that the intital fireflies after being generated randomly are made closer to the estimate og glmfit using the folrmula for calculation of new fireflies. The att variable is changed with a value estimation_move which represents how much a solution is to be moved from the random guess towards the glmfit estimate. First some random od flows are generated incorporating the maximum and minimum values that are present in the Abiline dataset after wards they are moved towards the estimate. The constraints are also applied using the upper and lower bound as was done in [3]. The code is as follows

```
function f=generateFireFlies(maxv,minv,population)

global Amatrix;

global b;

fireflies=[];

random_od_flows=[];

for i = 1:1:population

for j = 1:1:length(maxv)

min=minv(j);

max=maxv(j);

random_od_flows(j)=min+rand(1,1)*(max-min);

end
```

```
while (Amatrix*random_od_flows' > b)
for j = 1:1:length(maxv)
min = minv(j);
max = maxv(j);
random_od_flows(j)=min+rand(1,1)*(max-min);
end
end
fireflies(:,i)=random_od_flows;
end
f=fireflies;
function fireFlies = generateFireflies_with_estimate(maxv,minv,population,lb,ub,exploration,estimated_firefly,estimation_move)
ffs=generateFireFlies(maxv,minv,population);
for i = 1:1:size(ffs,2)
ffs(:,i) = ((1-estimation_move)*ffs(:,i)) + (estimation_move*estimated_firefly) + (exploration*generateFireFlies(maxv,minv,1));
end
ffs = applyConstraintsOnFireflies(lb,ub,ffs);
fireFlies = ff
```

The distance, attribute is calculated by simply calculating the L2 norm between two fireflies and attractiveness is calculated using the approach proposed in [1] and [2]. Calulating brightness was a bit tricky because the magnitude of the errors was expolentially large as it is between quantities which are large by nature. It was realized the simply using the L2 norm between the predicted and actual value from Abiline dataset did not work as this was a huge number. the inverse of this number represented brightness. This resulted in all brightnesses becoming equal to almost zero making the move from one firefly to another almost impossible to mimic as proposed by the original algorithm. The problem was overcome by using logarithms of the L2 norms. The code is as follows

```
function br = fireflyBrightness(x)
global Amatrix;
global b;
A=(Amatrix*x-b);
f1=norm(A,2);
br=1/log(f1);
```

The rest of the algorithm is as was proposed by [1]

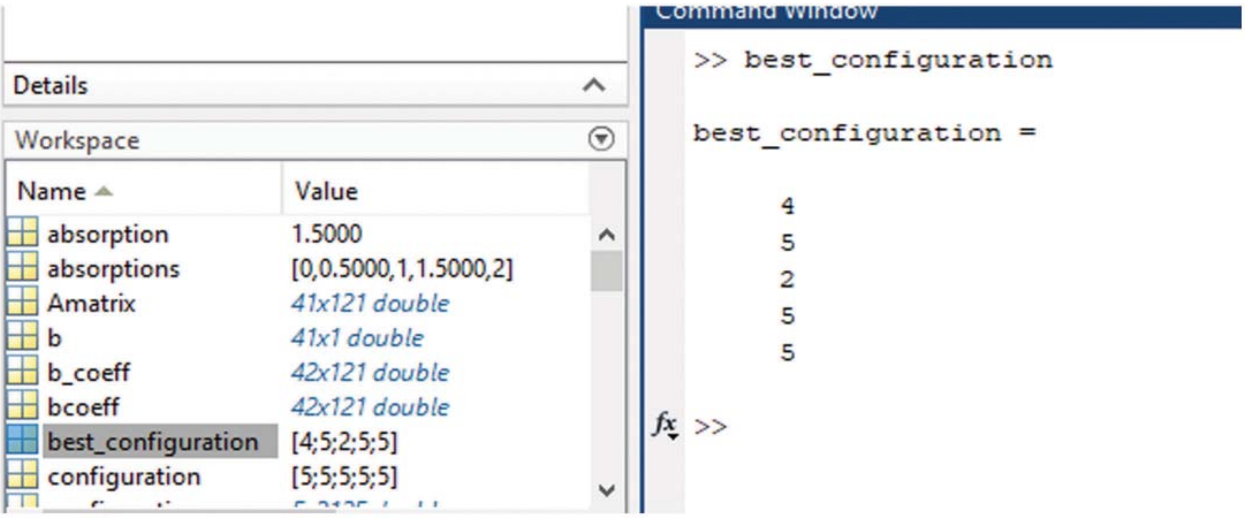
```
function sol = myFireflyAlgo (maxv,minv,absorption,population,generations,lb,ub,exploration,estimated_firefly,estimation_move)
bestFireflies=[];
for i = 1:1:generations
ffs = generateFireflies_with_estimate (maxv,minv,population,lb,ub,exploration,estimated_firefly,estimation_move);
all_generated_fireflies_for_g_i=ffs;
for j = 1:1:population
for k = 1:1:population
if (fireflyBrightness(ffs(:,k)) > firefly Brightness (ffs(:,j)))
fDistance = fireflyDistance(ffs(:,j),ffs(:,k));
beta = attractiveness (fireflyBrightness (ffs(:,k)),fDistance,absorption);
ffs(:,j) = ((1-beta)*ffs(:,j)) + (beta*ffs(:,k)) + (exploration*(generateFireFlies(maxv,minv,1)));
ffs(:,j) = apply Constraints On Fireflies (lb,ub,ffs(:,j));
all_generated_fireflies_for_g_i = [all_generated_fireflies_for_g_i ffs(:,j)];
end
end
end
bestFireflies = [bestFireflies getBestFirefly(all_generated_fireflies_for_g_i)];
end
sol = getBestFirefly(bestFireflies);
```

5. Seaching for the optimum hyperprameters

Running the algorithm while it was being made gave an intuitive understanding of the boundries of the hyperparameters. So a grid search was conducted using nested loops and these boundries as follows

```
absorptions = [0 0.5 1 1.5 2];
populations = [3 5 7 9 11];
generations_list = [50 100 150 200 250];
explorations = [0.1 0.001 0.0001 0.00001 0.000001];
estimated_firefly = X0;
estimation_moves = [0.8 0.85 0.9 0.95 0.99];
configurations=[];
errors = [];
for i = 1:1:length(absorptions)
for j = 1:1:length(populations)
for k = 1:1:length(generations_list)
for l = 1:1:length(explorations)
for m = 1:1:length(estimation_moves)
absorption = absorptions(i);
population = populations(j);
generations = generations_list(k);
exploration = explorations(l);
estimation_move = estimation_moves(m);
finalsolution (:,2)= myFireflyAlgo (maxv, minv, absorption, population, generations, lb,ub, exploration,estimated_firefly,estimation_move);
error=norm((finalsolution(:,1)-finalsolution(:,2)),2);
configuration = [i j k l m]';
        errors = [errors error];
            configurations = [configurations configuration];
end
end
end
end
end
[value,index]=min(errors);
best_configuration = configurations(:,index);
```

Runing the searh gave the following best configuration



So a viable configuration is

Absorption = 1.5

Population = 11

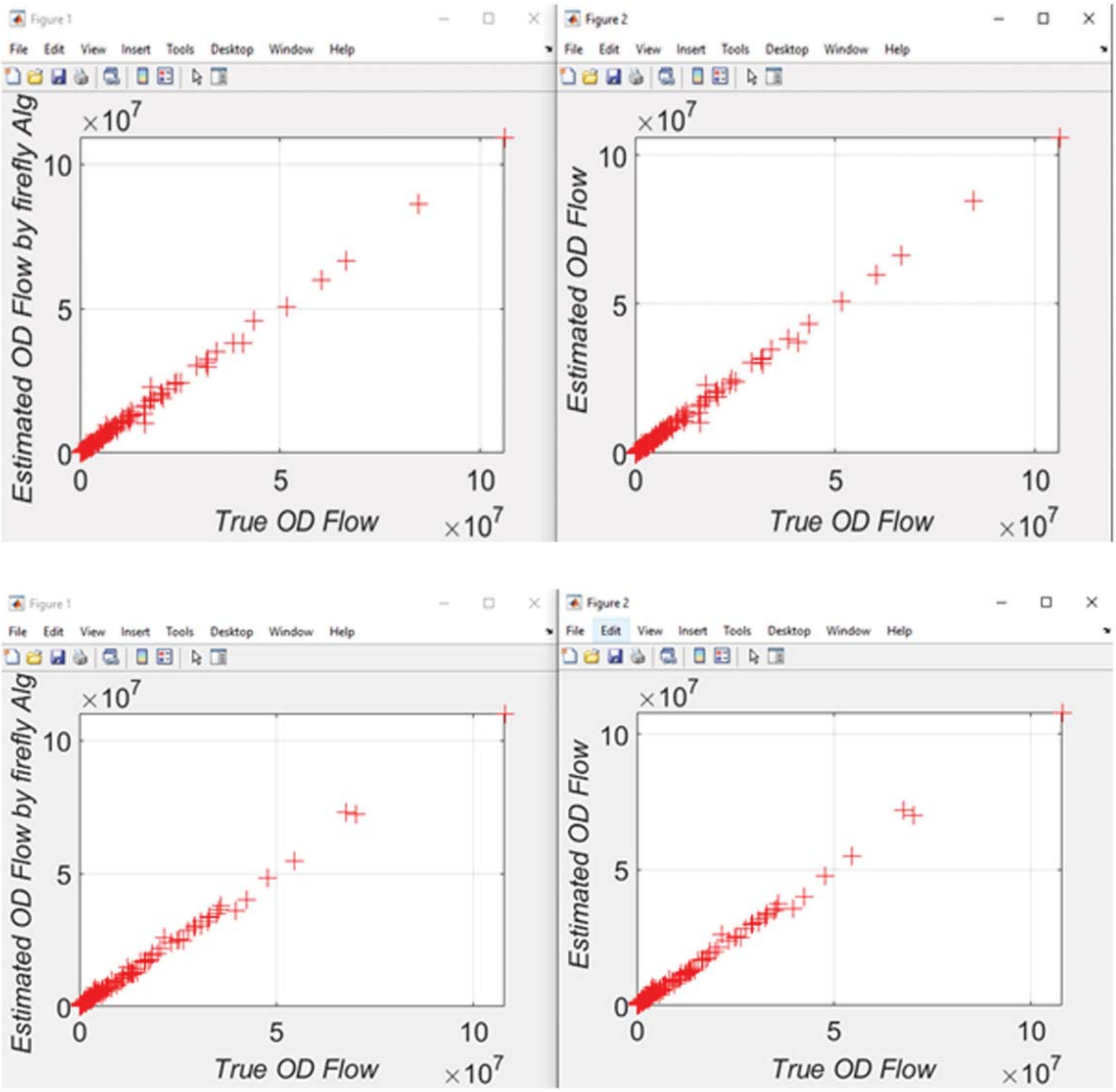
Generations = 100

Exploration = 0.000001

Estimation_move = 0.99

This grid search implies that the movement of the fireflies towards each other should not be a simple inverse of the brightness, it should be a bit less than that but not as less as it would be had the absorption coeffecient been two. This allows for more exploration of the solution space. The poplation should be 11 or more. More exploration is required for this parameter. The generations parameter indicated that starting the whole process again and agin need not lead to more fruitation. The exploration and exploitation parameters indicate that the generated fireflies should initially be as close to the elephant estimate as it can be.

Now a graphical comparison between the accuracy of the results achieved by the firefly algorithm and the ones achieved by [3] are shown. A total of eleven sample time stamps have been selected arbitrarily. Three from the training data and eight from the test data



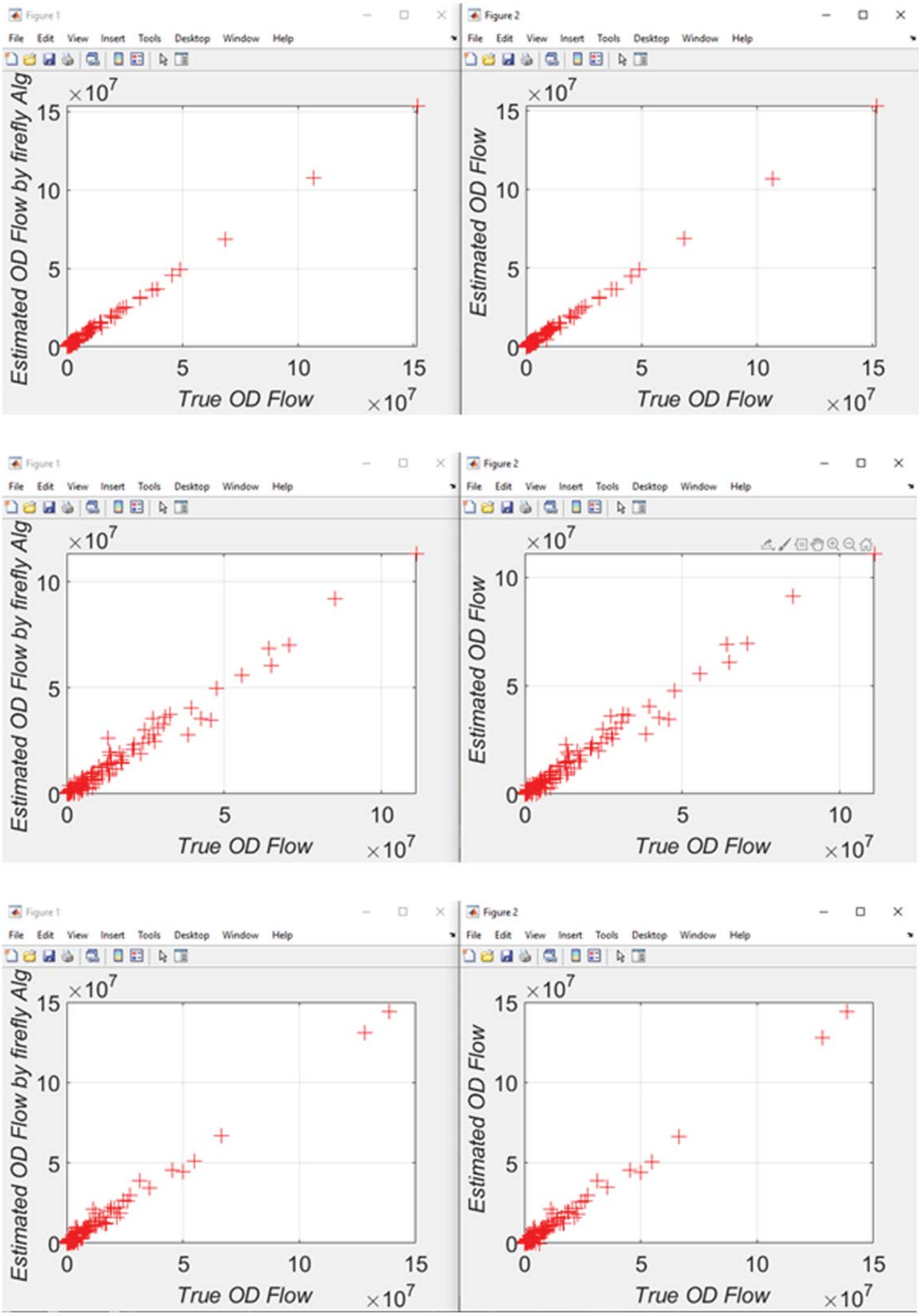


Figure 1
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow by firefly Alg
True OD Flow
Figure 2
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow
True OD Flow

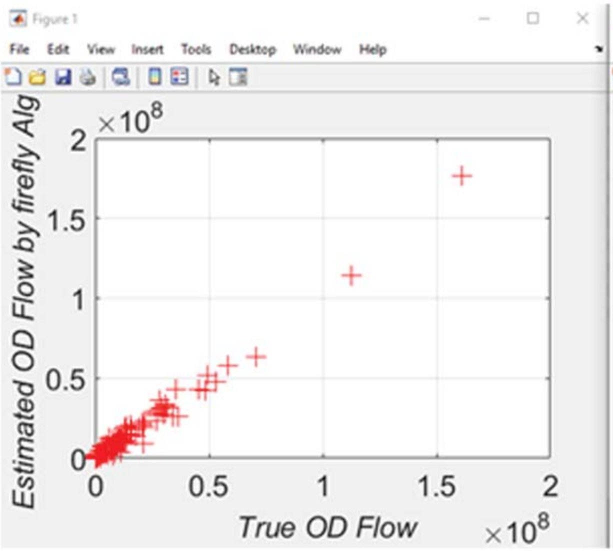


Figure 1
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow by firefly Alg
True OD Flow
$\times 10^8$
$\times 10^8$

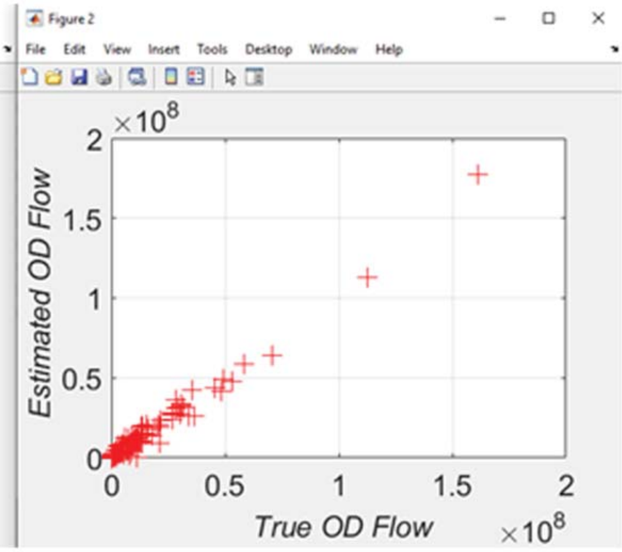


Figure 2
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow
True OD Flow
$\times 10^8$
$\times 10^8$

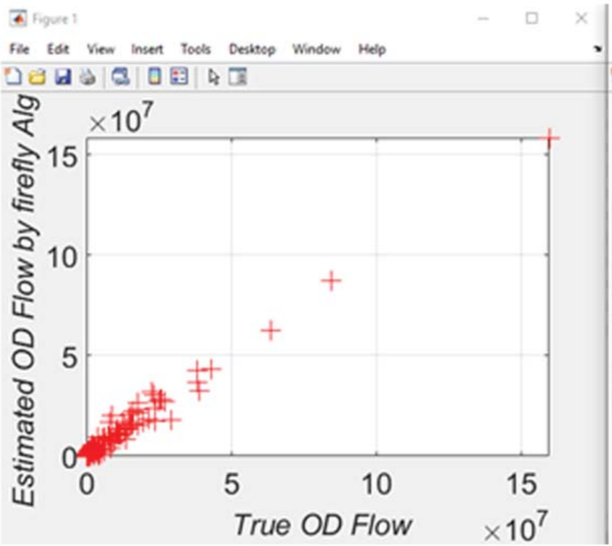


Figure 1
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow by firefly Alg
True OD Flow
$\times 10^7$
$\times 10^7$

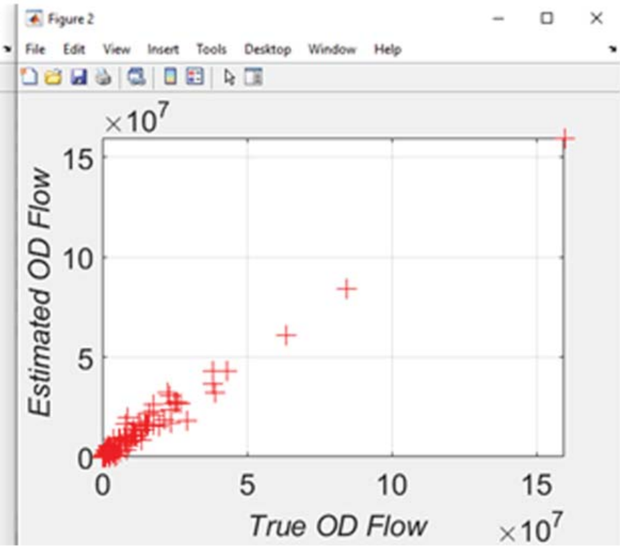


Figure 2
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow
True OD Flow
$\times 10^7$
$\times 10^7$

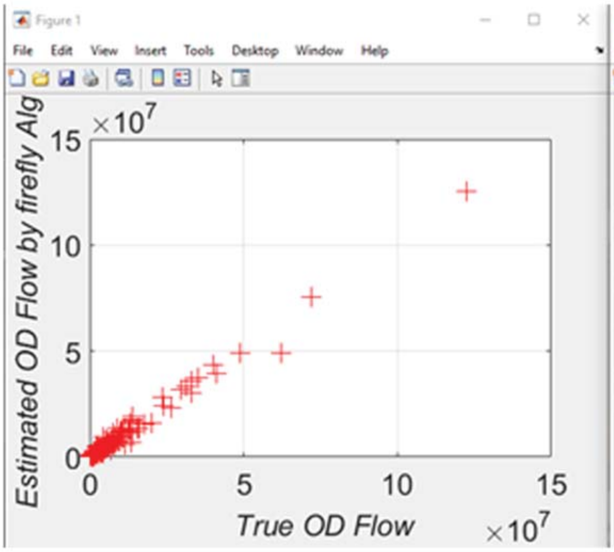


Figure 1
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow by firefly Alg
True OD Flow
$\times 10^7$
$\times 10^7$

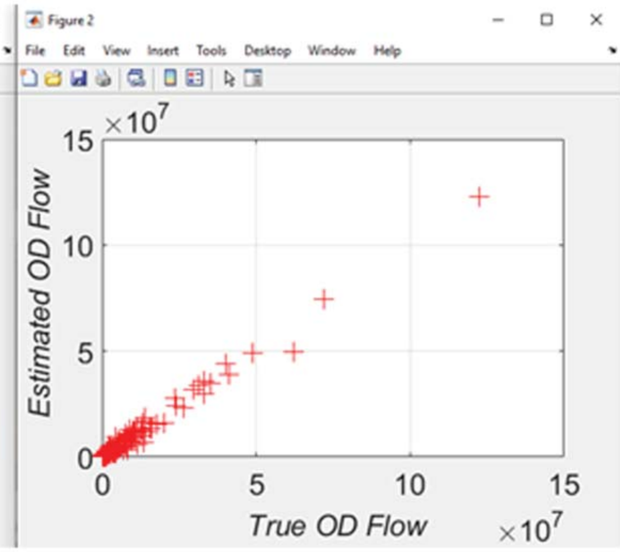


Figure 2
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow
True OD Flow
$\times 10^7$
$\times 10^7$

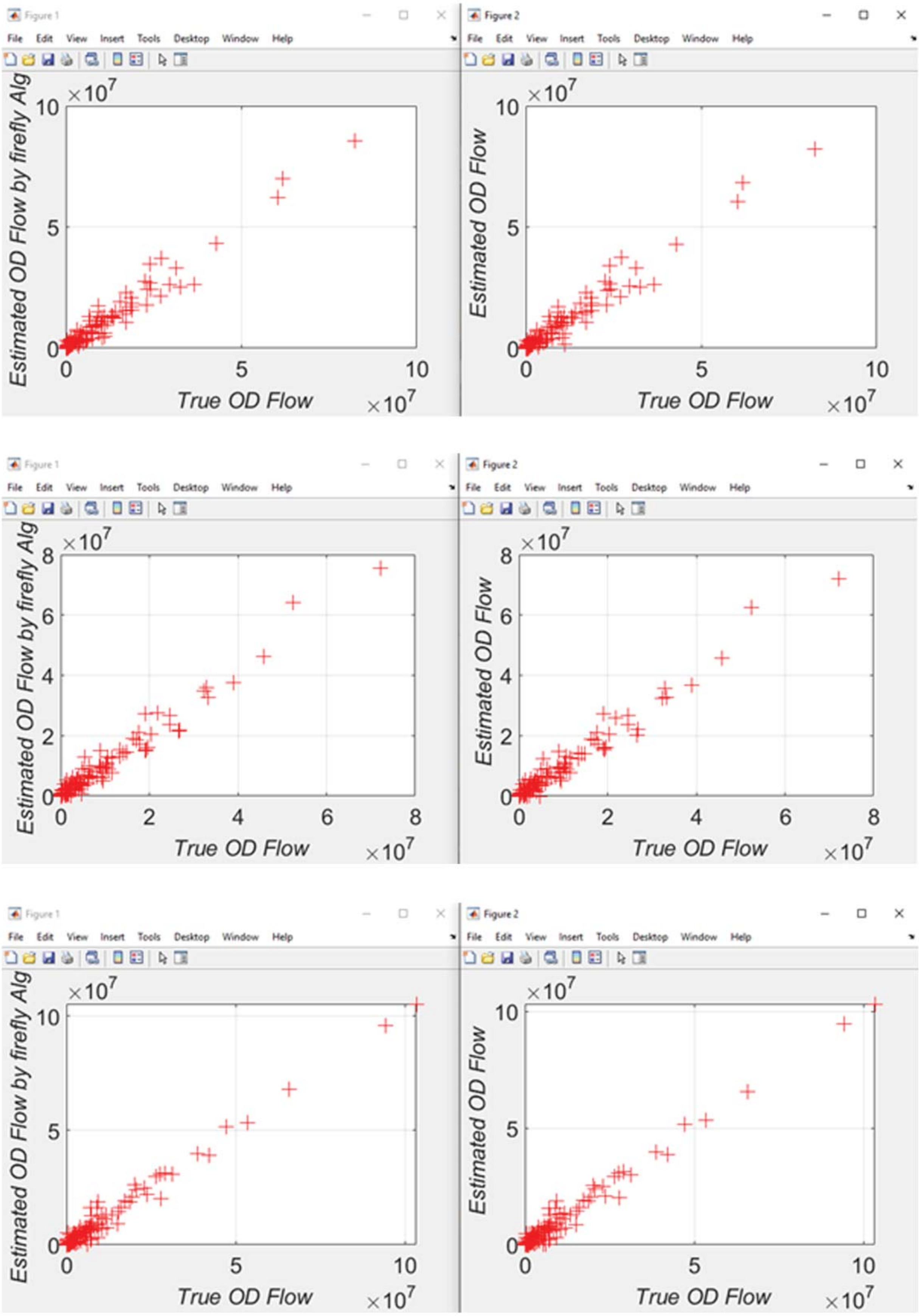


Figure 1
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow by firefly Alg
True OD Flow
Figure 2
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow
True OD Flow
Figure 1
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow by firefly Alg
True OD Flow
Figure 2
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow
True OD Flow
Figure 1
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow by firefly Alg
True OD Flow
Figure 2
File Edit View Insert Tools Desktop Window Help
Estimated OD Flow
True OD Flow

6. Conclusion, flaws and further work

The applied firefly algorithm can achieve the same amount of accuracy as the fmincon algorithm used by matlab.

One major flaw in this study is that fireflies are generated in each generation while in the original algorithm the initialization is done before the generations loop allowing the fire flies generated more time to converge. thus allowing more of the solution space in a localization to be searched. Secondly selecting the best configuration of fireflies shouldn't depend on which configuration gave the best firefly, it should depend upon which cofiguration gave best fireflies on average, unfortunatly this was not done.

A resursive version can be made which gradually lessens the generation,population and exploation after a firefly has been generated going through all generations and then thisfire fly can be used next time as the estimated solution. The original firelfy algorithm which was proposed in [1] has gone through evolution itself, thus the evolved version which show more promise whould also be aplplied to this problem so that more accuracy then [3] can be achieved.

References

[1] Xin-She Yang. Firefly algorithm for multimodal optimization. Stochastic Algorithms: Foundation and Application, 5th, 2009.

[2] Karel Durkota. "implementaion of a discrete firefly algorithm for the QAP problem Within the SEAGE framework" Bachlors thesis. Faculty of Electrical Engineering. Czech University of Prague

[3] R A Memon, S M Atif, S Qazi. First elephants then mice: A two stage robust estimation technique for traffic matrix in large cloud networks in presence of over dispersion. Dept of Electronix and power Engineering, College of Engineering. NUST,PAF KIET

[4] D.P.F Cruz, R.D.Maia, L.N.D Castro. A critical discussion into the core of swarm intelligence algorithms. Springer Nature 2019