Application of Firefly Algorithm for optimizing of cost function

Ananyae kumar bhartari

October 31, 2022

Contents

1 Introduction

Most conventional or classic algorithms are deterministic. For example, the simplex method in linear programming is deterministic. Some deterministic optimization algorithms used the gradient information; they are called gradient-based algorithms. For example, the well-known Newton-Raphson algorithm is gradient-based, since it uses the function values and their derivatives, and it works extremely well for smooth unimodal problems. But such methods have problems when the objective function(The function whose optimum we want to find) is dis continuous or has a discrete domain. Another problem with calculating derivatives is that for a multi dimensional support the Jacobian will be difficult to calculate. In such case a non-gradient algorithm is preferred. Non-gradient based or gradient-free algorithms do not use any derivative, only the function values.
For stochastic algorithms, in general we have two types: heuristic and meta-heuristic, though their difference is small. Loosely speaking, heuristic means "to find" or "to discover by trial and error." Quality solutions to a tough optimization problem can be found in a reasonable amount of time, but there is no guarantee that optimal solutions will be reached. This is good when we do not necessarily want the best solutions but rather good solutions that are easily reachable.
Further development of heuristic algorithms is the so-called meta-heuristic algorithms. Here meta means "beyond" or "higher level," and these algorithms generally perform better than simple heuristics. In addition, all meta-heuristic algorithms use certain trade offs of randomization and local search. It is worth pointing out that no agreed definitions of heuristics and meta-heuristics exist in the literature; some use the terms heuristics and meta-heuristics interchangeably. However, the recent trend tends to name all stochastic algorithms with randomization and local search as meta-heuristic. Here we also use this convention. Randomization provides a good way to move away from local search to search on a global scale. Therefore, almost all meta-heuristic algorithms tend to be suitable for global optimization. We are going to look at meta-heuristic and put them to use to solve problems that will be difficult to solve with gradient based algorithms.

2 Firefly Algorithm

2.1 Firefly Behavior

This algorithm is based on the behavior of firefly in nature. There are about 2000 firefly species, and most fireflies produce short, rhythmic flashes. The pattern of flashes is often unique for a particular species. The flashing light is produced by a process of bio-luminescence; the true functions of such signaling systems are still being debated. The two fundamental functions of such flashes are to attract mating partners (communication) and to attract potential prey. In addition, flashing may also serve as a protective warning mechanism to remind potential predators of the bitter taste of fireflies.

The rhythmic flashing, rate of flashing, time interval between flashing plays and important role during the mating. Moreover the light intensity at a particular distance r from the light source obeys the inverse-square law. That is to say, the light intensity I decreases as the distance r increases in terms of $I \propto \frac{1}{r^2}$. Furthermore, the air absorbs light, which becomes weaker and weaker as the distance increases. But the luminescence is enough to communicate over hundred's of meter.

2.2 Assumptions of Algorithms

We have to conceptualize certain rules so as to govern the movements of firefly in the environment, the basic 3 rules that govern FA are:

- All the fireflies are attracted toward each other, they are assumed to be unisex.
- The attractiveness of each firefly is proportional to their brightness, this means that a brighter firefly will attract all those who are less bright than it.
  Moreover their brightness decrease as the distance between them increases.
- The brightness of the firefly will be determined by the objective function(the function whose maxima we want to find).
  For example when we want to find the maxima the objective function can be chosen to calculate intensity of fireflies. If we are want to find minima then negative of objective function can be used to calculate intensity.

2.3 Variation in Intensity

It has been observed that Light intensity varies with distance, intuitively things that are further away seem to be less bright. In the simplest form intensity is inversely proportional to the distance.

$$I(r) \propto \frac{I_s}{r^2} \tag{2.1}$$

Here I_s represents the intensity of the source and r represents the distance between source and observer. For a medium with fixed absorption constant γ the approximate form becomes.

$$I(r) = I_0 e^{-\gamma r} \tag{2.2}$$

Here I_0 represents the initial intensity. But to account for the inverse square relationship we can modify the equation 2.2 as

$$I(r) = I_0 e^{-\gamma r^2} \tag{2.3}$$

Because a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, we can now define the attractiveness β of a firefly by

$$\beta = \beta_0 e^{-\gamma r^2} \tag{2.4}$$

where β_0 is the attractiveness at r = 0. Since it is often faster to calculate $\frac{1}{1+r^2}$ than an exponential function, this function, if necessary, can conveniently be approximated as

$$\beta = \beta_0 \frac{1}{1+\gamma r^2} \tag{2.5}$$

For all practical purposes a general form of attractiveness is of some form

$$\beta = \beta_0 e^{-\gamma r^m}, m \geq 1 \tag{2.6}$$

2.4 Position Updating

We have to propose the movement of each firefly in the environment. Let see for a specific case, say at t time the i^{th} firefly brightness is lesser than j^{th} firefly one, then according to our system j^{th} will attract the i^{th}. Now new position of the i^{th} firefly is given by

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \epsilon_i \tag{2.7}$$

Here x_i^t represents the position of i^{th} at time step t.
r_{ij} is the distance between i^{th} and j^{th} firefly.
ϵ_i is a vector of random numbers drawn from a Gaussian distribution or uniform distribution
α is constant that controls the randomization.

When be have a many individuals then a single firefly may be attracted towards many of many individuals in such a case the update equation

$$x_i^{t+1} = x_i^t + \sum_{j \in J_i} \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \epsilon_i \tag{2.8}$$

Here J_i is the set of index all such fireflies that are brighter than i^{th} firefly.

2.5 Controlling randomization

When we apply FA , we want want that once the fireflies reach the global optimum they should not move or move less so that other fireflies nearby can reach it, this implies that the randomization needs to be controlled.
The random walk part is basically help the fire flies in exploring their neighbourhood so its makes perfect sense that when the firefly is at optimum it should not explore alot.
This can be achieved by using α that decreases with time hence after a considerable time the exploration part is negligible. We have chosen

$$\alpha(t) = \theta \alpha(t-1), 0 \leq \theta \leq 1 \tag{2.9}$$

Here θ is a constant and $\alpha(t)$ goes to 0, as time steps(t) tend to ∞. Mostly θ is chosen in range [.9,1].

2.6 Pseudo-code

Here we present the Pseudo code for Firefly algorithm.

Objective function f(x) where x is d-dimensional $x = (x_1, x_2, ...x_d)$
Generate initial position of each firefly x_i for n fireflies
Define all the constants γ,β_0,α and number of time steps(T)
Calculate the Intensity (I) of each firefly considering initial position
for `1<=t<=T` **do**
  for `1<=i<=n` **do**
    for `1<=j<=n` **do**
      $temp \leftarrow 0$ ▷ temp stores the weighted displacement
      if $I_j \geq I_i$ **then**
        $temp \leftarrow temp + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t)$
      end if
      $x_i^t \leftarrow x_i^{t-1} + temp + \alpha * \epsilon_i$ ▷ Calculate the new position
      Evaluate new solutions and update light intensity
    end for j
  end for i
end for t
Rank the fireflies and find the current global best g_*

Here global best g_* means the point having the maximum value(Maxima).

2.7 Implementation

In this section we are going to provide the implementation of FA on Eggcrate function(objective function) in 2D case. Eggcrate function is given by

$$f(x,y) = x^2 + y^2 + 25(sin(x)^2 + sin(y)^2) \tag{2.10}$$

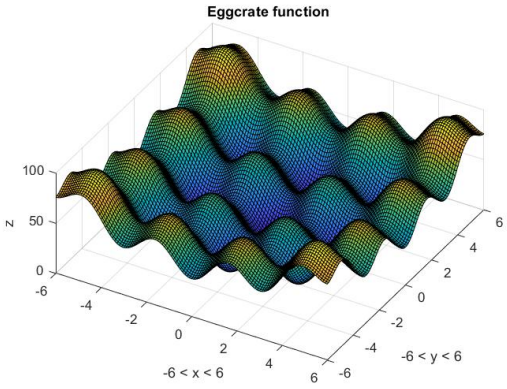


Figure 1: Eggcrate function

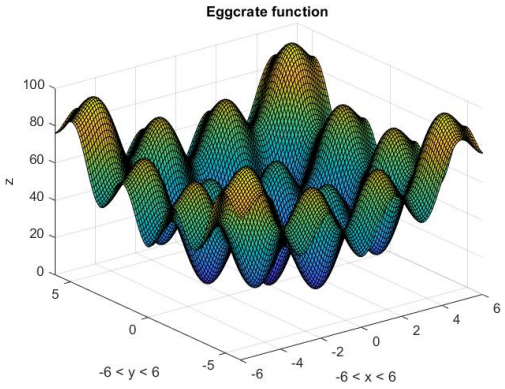


Figure 2: Eggcrate function

We want to find the minima hence we will look at negative of the Objective function.

2.7.1 Constraints

We are going to find the minima in $(x,y) \in [-4,4] \times [-4,4]$

- Number of firefly(n) is chosen to be 50
- Time steps(T) are chosen to be 20
- α =.2
- β_0 =.5
- γ =1
- θ =.95

2.7.2 Code

To see the complete code visit :- https://github.com/ananyae-24/TBC/blob/main/Matlab/Firefly_algorithm.m

2.7.3 Result

The FA performed very well and was and the global minima was given at (-0.0607,0.0453) which is close to (0,0) the actual global minima.
The contour is also shown below which shows both the initial and final position of fireflies.

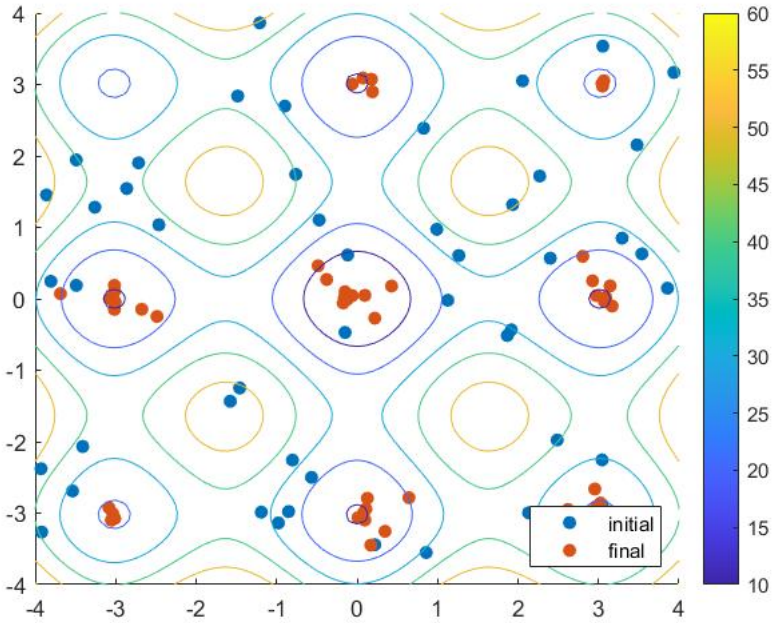


Figure 3: Couture Plot with initial and final positions

An observation evident from the plot is that FA also finds local optimums.

2.8 Discretize FA

In some case our state space is not continuous hence we have the need to discretize FA . Many variety of approaches have been developed in literature and problem statement plays an important role to determine which approach to use.

2.8.1 Logistic approach

Logistic function is a very well know function given by

$$S(x) = \frac{1}{1+exp(-x)} \tag{2.11}$$

an interesting property of this function is that it gives values between [0,1] hence the output can be intuitively treated as a probability.
We can develop a probabilistic rule based on its output

$$x = \sum_{j \in J_i} \beta_0 e^{-\gamma r_{ij}^2} (x_j^t - x_i^t) + \alpha \epsilon_i \tag{2.12}$$

$$S(x) < u \implies x_i^{t+1} = x_i^t - 1 \tag{2.13}$$

$$S(x) \geq u \implies x_i^{t+1} = x_i^t + 1 \tag{2.14}$$

Here u is a user chosen constant. Another extension of this approach will be to use u_1, u_2 and assigning -1,0,1 respectively for intervals $[0,u_1]$,(u_1,u_2),$[u_2,1]$

2.8.2 Ceil and Floor

Some times taking greatest integer less than and smallest integer greater than work for problems but it is not considered a good approach as it does not work in most of the cases.

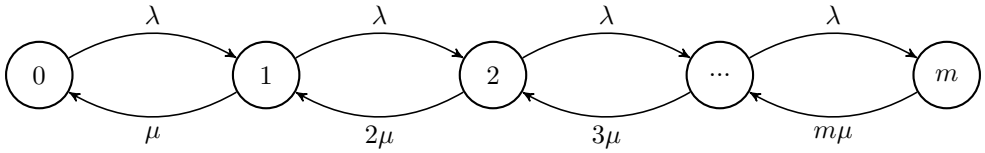
3 Problem Statement 1

3.0.1 Introduction

Here we consider the models with exponential service times, in which the arrival process is a Poisson one. The mean arrival rate (per unit of time) at each system is denoted by λ and the parameter of the service time is denoted by μ. Traffic intensity ρ is the ratio of arrival λ to service rate μ.
We have m parallel identical servers. In this system arriving customer is served if at least one server is available. When all servers are occupied the newly arriving customer departs the queuing systems without being served this implies that queuing size is 0. The customers arriving when all the severs are occupied are lost.

3.0.2 Deriving Objective Functions

In this session we are going to derive the objective function We start by demonstrating the above queue as a Poisson process:



Let π_i denote the steady state probability of being in state i. Using Chapman Kolmogorov equations:

$$\begin{aligned} 0 &= -\lambda\pi_0 + \mu\pi_1 \\ 0 &= \lambda\pi_0 - (\lambda+\mu)\pi_1 + 2\mu\pi_2 \\ 0 &= \lambda\pi_{n-1} - (\lambda+n\mu)\pi_n + (n+1)\mu\pi_{n+1} \quad \forall n \in 1,...m-1 \\ 0 &= \lambda\pi_{m-1} - m\mu\pi_m \end{aligned} \tag{3.1}$$

Mover over as they are probabilities they also satisfy:

$$\sum_{k=0}^{m} \pi_k = 1 \tag{3.2}$$

$\pi_i = A\frac{\rho^i}{i!}$ satisfy the 3.1, where A is some constant. Value of A is determined by 3.2. $\therefore A = \frac{1}{\sum_{k=0}^{m} \frac{\rho^i}{i!}}$
The steady-state probability that the newly arriving customers are lost:

$$\pi_l = \pi_m = \frac{\frac{\rho^m}{m!}}{\sum_{k=0}^{m} \frac{\rho^i}{i!}} \tag{3.3}$$

In the case of M/M/m/-/m queuing systems with losses we seek the number of servers that maximizes the overall profits. Here we get the following objective function:

$$r\rho(1-\pi_w) - cm \tag{3.4}$$

Here r denotes the cost related to jobs in system and $r\rho$ can be interpreted as total cost of traffic given that the traffic is completely accepted. Thus overall rate of profit will be $r\rho(1-\pi_w)$, here c denotes the loss due to server depreciation and as we have m servers overall loss will be cm.

3.0.3 Objective Function

In our case for **M/M/m/-/m queuing system with losses** the objective function is defined as

$$max_m\, f(m) = max_m\, \rho(1 - \frac{\rho^m}{m! \sum_{k=0}^{m} \frac{\rho^k}{k!}}) - cm \tag{3.5}$$

The mean arrival rate (per unit of time) at each system is denoted by λ and the parameter of the service time is denoted by μ. Traffic intensity ρ is the ratio of arrival λ to service rate μ.
Here r denotes the cost related to jobs in system and c denotes the cost of server depreciation.

3.0.4 Constraints

We are going to use the FA algorithm and use exponential function for Intensity variation 2.4 with $\alpha = 0.5$, $\beta_0 = 0.2$, $\gamma = .01$.
In the case of M/M/m/-/m system the number of fireflies is 20 and the maximum generation of fireflies is 50, so the total number of functions evaluation is 1000.

3.0.5 Code

To see the complete code visit :-
https://github.com/ananyae-24/TBC/blob/main/Matlab/Firefly_algorithm_m.m

3.0.6 Results

the results are shown is tabular format below, For each experiment r=2, c=1. and initial position of fireflies was assigned an integer between 1 to 150.

λ	μ	optimum value of m	f(m) 3.5
10	2	5	2.1513
10	5	2	0.4000
60	2	32	22.2240
60	5	13	7.2824

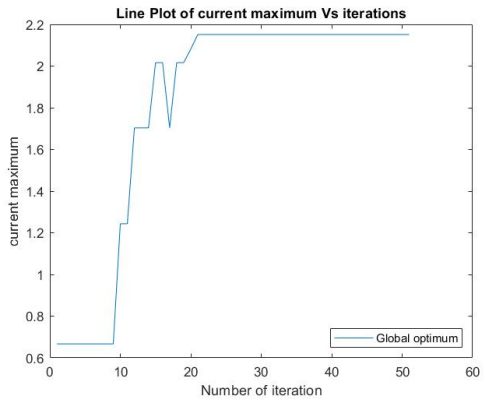


Figure 4: $\lambda = 10, \mu = 2$

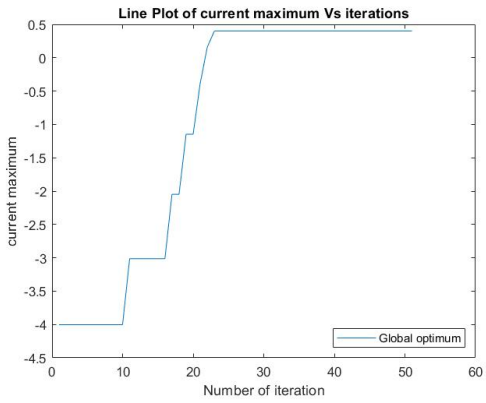


Figure 5: $\lambda = 10, \mu = 5$

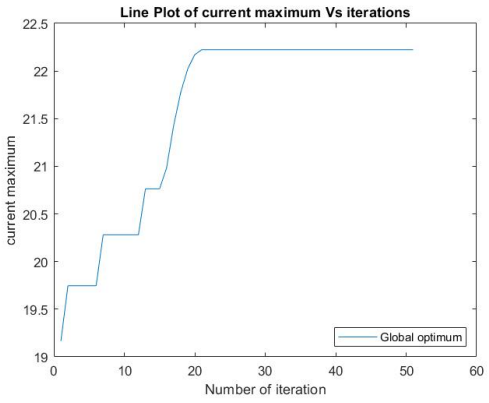


Figure 6: $\lambda = 60, \mu = 2$

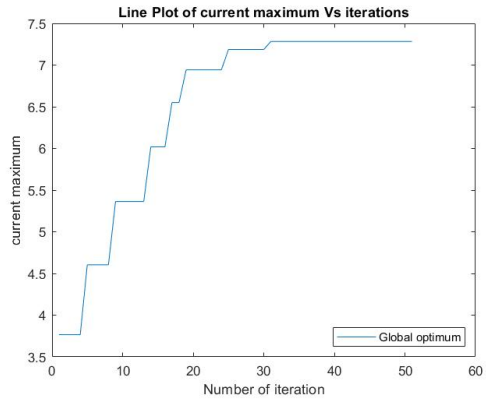


Figure 7: $\lambda = 60, \mu = 5$

Each of the above graph shows convergence and the result was cross verified with the actual answer which matched perfectly with our result.

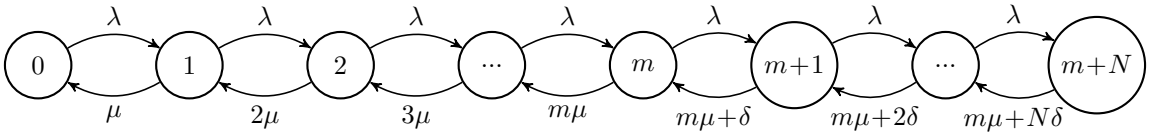
4 Problem Statement 2

4.1 Introduction

Here we consider the models with exponential service times, in which the arrival process is a Poisson one. The mean arrival rate (per unit of time) at each system is denoted by λ and the parameter of the service time is denoted by μ. Traffic intensity ρ is the ratio of arrival λ to service rate μ.
We have m parallel identical servers and the given time there can be $m+N$ customers(m being served and N waiting). If newly arriving customers find $m+N$ customers in systems, they are lost. The customers are served in FIFO queuing discipline.
Each job arriving to the system has its own maximal waiting time T_w. This time is assumed to be with an exponential distribution with parameter δ. If the time, which a job would have to wait for accessing a server exceeds T_w, then it departs from the system after time T_w.

4.2 Deriving Objective Functions

In this session we are going to derive the objective function We start by demonstrating the above queue as a Poisson process:



Let π_i denote the steady state probability of being in state i. Using Chapman Kolmogorov equations:

$$\begin{gathered}
0=-\lambda\pi_0+\mu\pi_1 \\
0=\lambda\pi_0-(\lambda+\mu)\pi_1+2\mu\pi_2 \\
0=\lambda\pi_{n-1}-(\lambda+n\mu)\pi_n+(n+1)\mu\pi_{n+1} \quad \forall n\in\{1,...m-1\} \\
0=\lambda\pi_{m-1}-(\lambda+m\mu)\pi_m+(m\mu+\delta)\pi_{m+1} \\
0=\lambda\pi_{n-1}-(\lambda+m\mu+(n-m)\delta)\pi_n+(m\mu+(n-m+1)\delta)\pi_{n+1} \quad \forall n\in\{m+1,...,m+N-1\} \\
0=\lambda\pi_{m+N-1}-(m\mu+N\delta)\pi_{m+N}
\end{gathered} \tag{4.1}$$

Mover over as they are probabilities they also satisfy:

$$\sum_{k=0}^{m+N}\pi_k=1 \tag{4.2}$$

Define

$$\pi_0=\frac{1}{\sum_{k=0}^{m}\frac{\rho^k}{k!}+\frac{\rho^m}{m!}\sum_{r=1}^{N}\frac{\rho^r}{\prod_{n=1}^{r}(m+n\frac{\delta}{\mu})}} \tag{4.3}$$

The solution to linear system is:

$$\begin{gathered}
\pi_i=\pi_0\frac{\rho^i}{i!} \quad \forall i\in\{0,1,...m\} \\
\pi_i=\pi_0\frac{\rho^m}{m!}\frac{\rho^{i-m}}{\prod_{r=1}^{i-m}(m+r\frac{\delta}{\mu})} \quad \forall i\in\{m+1,m+2,...,m+N\}
\end{gathered} \tag{4.4}$$

The probability that jobs will be lost because of exceeding the time limit is the following:

$$\pi_w=\sum_{r=1}^{N}\frac{\delta}{\lambda}r\pi_{r+m} \tag{4.5}$$

Here $\frac{\delta}{\lambda}$ is the rate of loss of job, so if we have r jobs then the total rate becomes $\frac{\delta r}{\lambda}$.
The steady-state probability that the newly arriving customers are lost:

$$\pi_{m+N}=\pi_0\frac{\rho^{m+N}}{m!\prod_{r=1}^{N}(m+r\frac{\delta}{\mu})} \tag{4.6}$$

The probability that jobs will be lost as the queue is full is given by:

$$\pi_l = \pi_w \pi_{m+N} \tag{4.7}$$

In the case of M/M/m/FIFO/m+N queuing systems with losses we seek the number of servers that maximizes the overall profits. Here we get the following objective function:

$$c_1\lambda(1-\pi_l) - c_2(m+N) \tag{4.8}$$

Here c_1 is the profit in each job and $c_1\lambda$ can be interpreted as rate of making profit given that the jobs are accepted. Thus overall rate of profit will be $c_1\lambda(1-\pi_l)$. here c_2 denotes the loss due to server depreciation and as we have $m+N$ servers overall loss will be $c_2(m+N)$.

4.3 Objective Function

In our case for **M/M/m/FIFO/m+N queuing system with losses** the objective function is defined as :

$$max_m\, f(m) = max_m\, c_1\lambda(1-\pi_l) - c_2(m+N) \tag{4.9}$$

The mean arrival rate (per unit of time) at each system is denoted by λ and the parameter of the service time is denoted by μ. Traffic intensity ρ is the ratio of arrival λ to service rate μ.
Each job arriving to the system has its own maximal waiting time T_w. This time is assumed to be with an exponential distribution with parameter δ.
Here c_1 denotes the cost related to jobs in system and c_2 denotes the cost of server depreciation.

4.4 Constraints

We are going to use the FA algorithm and use the approximation of exponential function for Intensity variation 2.5 with $\alpha = 0.2$, $\beta_0 = 0.2$, $\gamma = 1$.
In the case of M/M/m/FIFO/m+N system the number of fireflies is 40 and the maximum generation of fireflies is 150, so the total number of functions evaluation is 6000.

4.5 Code

To see the complete code visit :-
https://github.com/ananyae-24/TBC/blob/main/Matlab/Firefly_algorithm_m_N.m

4.6 Results

the results are shown is tabular format below, For each experiment N=10, c_1,$\delta = 5$ and $c_2 = 3$. and initial position of fireflies was assigned an integer between 1 to 200.

λ	μ	N	optimum value of m	f(m) 4.9
100	2	10	26	382.4839
150	2	10	41	583.5755
100	20	10	4	455.9393

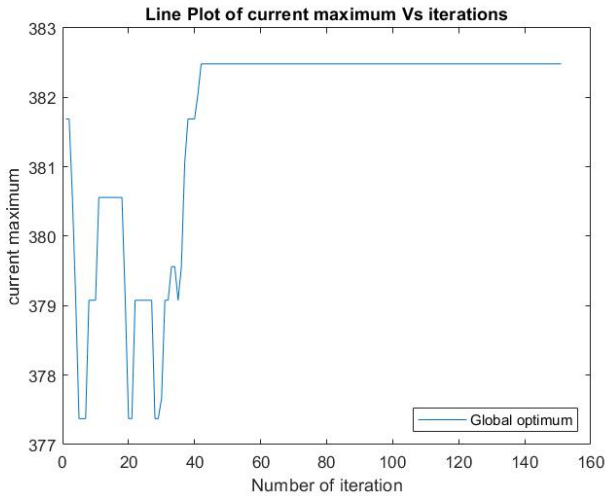


Figure 8: $\lambda = 100, \mu = 2, N = 10$

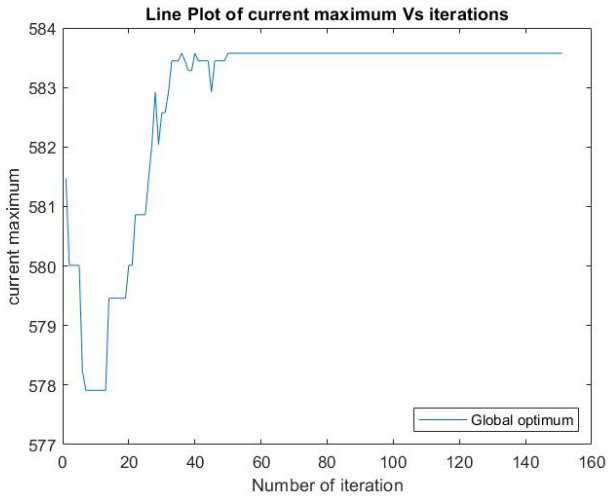


Figure 9: $\lambda = 150, \mu = 2, N = 10$

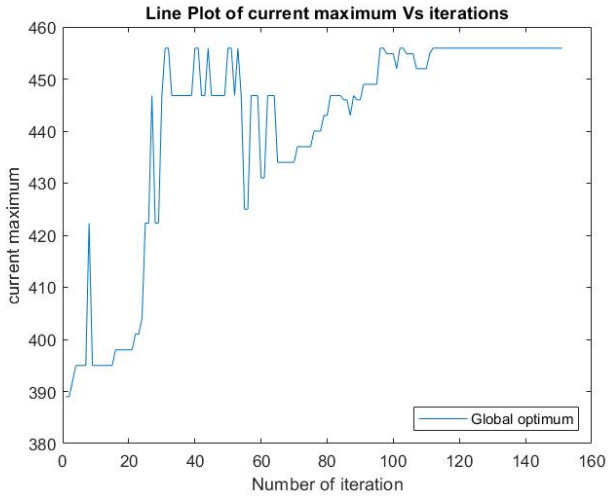


Figure 10: $\lambda = 100, \mu = 20, N = 10$

Each of the above graph shows convergence and the result was cross verified with the actual answer which matched perfectly with our result.

5 Problem Statement 3

5.1 Introduction

The flow shop scheduling problem (FSSP) is normally classified as a complex combinatorial optimization problem, in which there is a set of n jobs (1,2,...,n) to be processed in a set of m machines (1,2,...,m) in the **same order**. We normally look for a special sequence of processing the jobs in the machines to minimize one or more criteria such as minimization of makespan, mean flow, etc. There are different most commonly used criteria such as the minimization of the total completion time or makespan of the schedule (C_{max}) which is sometimes referred to as maximum flow time or F_{max} . The processing times needed for the jobs on the machines are assumed to be non-negative and deterministic denoted as t_{ij} , with i = 1,2,..,n and j = 1,2,..,m.
Although the optimal solution of the flow shop scheduling problem can be determined in polynomial time when m=2 (Johnson's algorithm which in of order nlog(n) where n is the number of jobs), the general form of this kind of problem is known to be NP-Complete in the strong sense when m$\geq$3 and generally for m$\geq$3 n! schedules need to be considered. That is why the problem is somewhat restricted in by not allowing job passing. In this case, "only" n! schedules must be considered and the problem is then known as permutation flow shop which is classified as n/m/P/ Fmax or as F/prmu/ Cmax .

5.1.1 Objective Functions

In a Flow Scheduling Problem the order of jobs processed by the machine is fixed. We are going to demonstrate this my an example.
Consider the following instance of problem F3$||C_{max}$. The T_{ij} matrix is provided which tells the complete information about how much time is required for job j to be processed in machine i.

j	t_{1j}	t_{2j}	t_{3j}
1	4	2	1
2	3	6	2
3	7	2	3
4	1	5	8

t_{ij} represents the time required to process job j on machine i
If every machine processes the jobs in the order (1,2,3,4), then the completion times of three operations of job 1 can be calculated as follows:

j	C_{1j}	C_{2j}	C_{3j}
1	4	4+2=6	6+1=7
2	4+3=7	max{7,6}+6=13	max{13,7}+2=15
3	7+7=14	max{14,13}+2=16	max{16,15}+3=19
4	14+1=15	max{15,16}+5=21	max{21,19}+8=29

Here C_{ij} represents the time required to complete job j in machine i.
So using this above method we can calculate the make-span time for each permutation and the objective is to find that permutation for which the make-span time is minimum.

5.1.2 Mathematical Model

In this section, a mathematical model for permutation flow shop scheduling problem is presented. The assumptions of the model are as follows:

- All n jobs to schedule are independent and available for processing at time zero.
- Machine cannot process two jobs at the same time.
- Each job is processed on at most one machine at a time.
- Setup times for the operations are sequence-independent and are included in processing times.
- There is only one of each type of machine.
- No more than one operation of the same job can be executed at a time.

Parameters:

n: Number of jobs

m: Number of machines

i: Machine index, (i=1,..,m)

j: Job index, (j=1,…,n)

k: Order index, (k=1,…,n)

t_{ij} : Processing time of job j on machine i

Decision variables:

C_{ijk} : Completion time of job j on machine i in kth order

x_{jk} : Binary variable taking value 1 if job j is processed in kth order and 0, otherwise.

The mathematical model for minimizing the make-span is as follow:

$$min \sum_{j=1}^{n} C_{mjn} x_{jn} \tag{5.1}$$

Constraints:-

- The job does not start on a machine until it finishes processing on the previous machine, which can be formulated as:-

$$\sum_{k=1}^{n} (C_{(i+1)jk} - t_{(i+1)j}) x_{jk} \geq \sum_{k=1}^{n} C_{ijk} x_{jk} \tag{5.2}$$

- The job's predecessor has completed processing on that machine, which can be formulated as:-

$$\sum_{j=1}^{n}(C_{ij(k+1)}-t_{ij})x_{j((k+1)} \geq \sum_{j=1}^{n} C_{ijk}x_{jk} \tag{5.3}$$

- Each sequence position is filled with only one job, which can be formulated as:-

$$\sum_{j=1}^{n} x_{jk} = 1 \tag{5.4}$$

- Each job is assigned to only one position in the job sequence, which can be formulated as:-

$$\sum_{k=1}^{n} x_{jk} = 1 \tag{5.5}$$

- Completion time is a positive real number:-

$$C_{ijk} \geq 0 \quad \forall i,j,k \tag{5.6}$$

- x_{jk} can be 1 or 0:-

$$x_{jk} \in 0,1 \quad \forall j,k \tag{5.7}$$

- Relationship between the binary variables and the completion time variables, there exist M such that:-

$$C_{ijk} \leq M x_{jk} \quad \forall i,j,k \tag{5.8}$$

5.1.3 Initial position

We know from Linear Algebra that each permutation can be represented as an permutation matrix.
A **permutation matrix** is a square binary matrix that has exactly one entry of 1 in each row and each column and 0s elsewhere. Example of 3×3 permutation matrix is:-

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{5.9}$$

The above matrix will be represented the permutation (2,3,1). Therefore a good idea is to use permutation matrix as initial position of fireflies.

5.1.4 Discretization

We can clearly see that the domain will be discrete as the valid position of fireflies are valid permutation which are n! in number. So after each update 2.8 we not necessarily will get a a permutation matrix therefore there is a need to convert the updated position to a valid permutation matrix which can be achieved using logistic function2.11. We explain this approach by a toy example using a 3×3 matrix. Suppose after updating the position of firefly we get:-

$$P = \begin{bmatrix} -.2 & .9 & .4 \\ .1 & -.25 & .8 \\ .05 & -.45 & .33 \end{bmatrix} \tag{5.10}$$

We apply logistic function to each entry of $P=(p_{ij})$ to get Q. Say Q=(q_{ij}) where $q_{ij} = \frac{1}{1+exp(p_{ij})}$.

$$Q = \begin{bmatrix} 0.450 & 0.711 & 0.599 \\ 0.525 & 0.438 & 0.690 \\ 0.512 & 0.389 & 0.582 \end{bmatrix} \tag{5.11}$$

Note:-Rounding up-to 3 decimal place for convenience not actually needed while coding

Now q_{ij} can be interpreted as the probability of taking the value of zero to one. In our example q_{12} has 71.1% probability of taking value 1.
Therefore there are 3 ways in which the matrix Q can be converted to a valid permutation matrix.

5.1.4.1 Column wise

We can iterate over the columns find the maximum value of probability and change that entry to 1 and rest to 0. Care one has to take is remove the rows in which 1 already exist. In out example final matrix will look like:-

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

5.1.4.2 Row wise

We can iterate over the rows find the maximum value of probability and change that entry to 1 and rest to 0. Care one has to take is remove the columns in which 1 already exist. In out example final matrix will look like:-

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{5.13}$$

5.1.4.3 Element wise

We find the max element in the matrix and change it to 1, we repeat this process by excluding the rows and columns in which 1 already exist. In out example final matrix will look like:-

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{5.14}$$

5.1.5 Constraints

We are going to use the FA algorithm and use the approximation of exponential function for Intensity variation 2.5 with $\alpha = .5$, $\beta_0 = 2$, $\gamma = .2$,$\theta = 1$. Element-wise approach was used to convert to a valid permutation matrix.
In the our case of the number of fireflies is 5 and the maximum generation of fireflies is 1000, so the total number of functions evaluation is 5000.

5.1.6 Code

To see the complete code visit :-
https://github.com/ananyae-24/TBC/blob/main/Flowscheduling/FlowScheduling.ipynb

5.1.7 Results

We are going fix the number of jobs to 9 and machine to 5 and randomly generate a T_{ij} a 5×9 matrix with $t_{ij} \in 1,2,...,200$.
We are going to compare FA naive approach that is checking all permutations and present the final result in a tabular format.

Result using FA	Time(sec) by FA	Actual C_{max}	Time(sec) by naive approach
1384	1.959248	1383	11.983689
1336	1.996897	1336	10.110663
1376	1.959713	1351	10.124528
1491	1.925532	1464	10.229164
1379	1.996422	1376	10.154174
1331	1.970159	1321	11.565649

Note for this attempt Number of jobs=9 Number of Machine=5

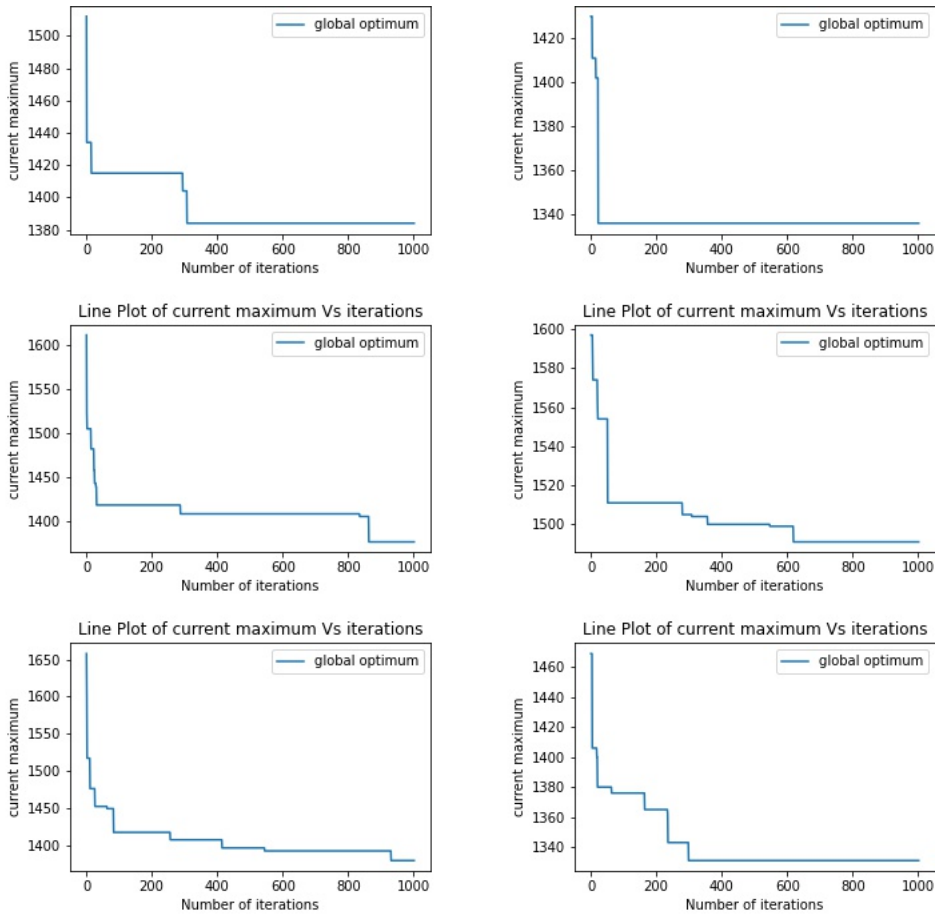


Figure 11: Result of each attempt are plotted

Each of the above graph shows convergence and results are near true value.

6 Conclusion

The firefly algorithm is a very powerful technique used to solve the problems of queuing systems optimization. It is a simple method and easy to implement. In this paper we have tested this algorithm to the multi-objective maximization problem of cost function. The parameters of firefly algorithm such as the absorption coefficient, the population of fireflies and the number of iterations depend on the optimized problem. We can successfully conclude that the firefly algorithm works great much faster than naive approach and try and give best solution.
Future work will focus on application of other meta heuristics such as the bee algorithm and particle swarm optimization in queuing and scheduling optimization problems.

7 References

- X.S. Yang, Nature-Inspired Metaheuristic Algorithms, Luniver Press, London, 2008.
- X. S. Yang, "Firefly algorithms for multimodal optimization", Stochastic Algorithms: Foundations and Applications, SAGA, Lecture Notes in Computer Sciences 5792, 169–178 (2009).
- S. Łukasik and S. Żak, "Firefly algorithm for continuous constrained optimization task", Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems LNCS 5796, 97–106 (2009).
- M.K. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, "A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems", Int. J. Industrial Eng. Computations 1, 1–10 (2010).
- G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, Queueing Networks and Markov Chains. Modeling and Performance Evaluation with Computer Science Applications, John Wiley&Sons, London, 1998.
- B. Filipowicz, Modeling and Optimization of Queueing Systems. Volume 1, Markovian Systems, T. Rudkowski Publishing House, Cracow, 1999, (in Polish).
- B. Filipowicz and J. Kwiecień, "Queueing systems and networks: models and applications", Bull. Pol. Ac.: Tech. 56 (4), 379–390 (2008).

- Ch.H. Lin and J.Ch. Ke, "Optimization analysis for an infinite capacity queueing system with multiple queue-dependent servers: genetic algorithms", Int. J. Computer Mathematics 88 (7), 1430–1442 (2011).
- X.S. Yang, Firefly Algorithm – Matlab files, http://www.mathworks.com/matlabcentral/fileexchange/29693-firefly-algorithm (2011).
- Dong, X., Huang, H., Chen, P. (2009). An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion, Computers Operations Research, 36, 1664 -1669.
- Farahmand Rad, S., Ruiz, R., Boroojerdian, N. (2009). New high performing heuristics for minimizing makespan in permutation flowshops, Omega, 37, 331 – 345
- Garey, M., Johnson, D., Sethi, R. (1976). The complexity of flowshop and jobshop scheduling, Mathematics of Operations Research, 1 (2), 117–129.
- Johnson, S., 1954. Optimal two- and three-stage production schedules with setup times included, Naval Research Logistics Quarterly, 1, 61.
- Lian, Z., Gu, X., Jiao, B. (2006). A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan, Applied Mathematics and Computation, 175, 773–785.
- Mohammad Kazem Sayadia , Reza Ramezaniana and Nader Ghaffari-Nasaba (2010) A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems
- J. KWIECIEŃ and B. FILIPOWICZ (2012) Firefly algorithm in optimization of queuing systems

Annexure-II

DECLARATION

I/We hereby declare that the work presented in the project report entitled Application of Firefly Algorithm for optimizing of cost function ..... contains my own ideas in my own words. At places, where ideas and words are borrowed from other sources, proper references, as applicable, have been cited. To the best of our knowledge this work does not emanate from or resemble other work created by person(s) other than mentioned herein.

Name and Signature Ananyae Kumar Bhartari

Date: 31/10/22

Santipada Shovai

01/11/2022