# FINAL REPORT

COEN 283 - Operating Systems

# JOB SHOP SCHEDULING OPTIMIZATION USING FIREFLY ALGORITHM

TEAM : FIREFLY

**Team Members:**
Deepank Chinnam
Manoj Parihar
Surag Suresh Yalaburgi
Vismit Patel

# Table of Contents

# List of Figures

# List of Tables

## Preface

The proposal lists out the details as per the requirements of ‘COEN 283 - Operating Systems’. The theme of this project is ‘Optimizing the Job Shop Scheduling problem using Firefly Algorithm’.

This project focuses on the famous ‘Job Shop Scheduling problem’, and attempts to solve the problem. We will conclude the project by displaying its effectiveness in various scenarios by looking at performance parameters.

The benchmark algorithm for measuring the job shop scheduling problem is traditional Algorithm(Shortest time first). We will measure the performance parameters like MakeSpan, Maximal Machine Workload to verify the efficiency and display the result using visual graphs like Gantt Charts.

The main aim of this project is to make everyone understand the improvements a certain algorithm brings over other traditional approaches by visually looking at them in action.

## Acknowledgement

We would like to extend our gratitude towards Prof. Elkady, who inspired us to make this project. We would also like to thank Santa Clara University which provided us the opportunity to learn the subject and apply its concept into making this project. Also the contribution of TA’s cannot be overlooked, who took their time to provide us prompt feedback and help us improve.

## Abstract

Job Shop scheduling is a famous scheduling problem which is categorized as one of the NP-hard problem. Firefly Algorithm is one such metaheuristic algorithm that attempts to improve the performance involving job shop scheduling.

Job Shop Scheduling - It is a common scenario where there are many workstations available and there are many tasks coming in. Each machine is capable of doing at most one job. What should be the optimum way in which these machines should take up tasks so that they waste minimum possible time. Any scheduling algorithm comes with two main constraints.

1. Each operation of a job must follow a process sequence.
2. Each process can be processed on assigned machines only.

This project attempts to visualize the job scheduling problem, first in it's most traditional approach i.e. Shortest time first and compare the results with our firefly algorithms. Benchmarking methods are used to evaluate and study the performance.

# Introduction

## Objective

The main objective of this project is to discuss and analyze FireFly Algorithm over traditional approaches for solving the Job Shop Scheduling algorithm. We explore the advantages and disadvantages of these algorithms over each other and discuss their feasibility.

## What is the problem?

The main motivation lies around the problem that we know many approaches to tackle the scheduling algorithms, but we cannot imagine, each and every algorithm just by looking at it in theory. This project attempts to visualize the process and show us what kind of heuristic improves which aspect of performance parameters.

## Why is this project Related to the Class?

Scheduling is one of core aspect of any Operating System. Managing resources, optimizing performance and improving task completion, all these have evolved from Operating Systems. Job shop problem is something we can see in many real world applications, where many machines combine to perform some jobs. Each machine takes care of different operations. Their core idea however comes from job scheduling used in Operating Systems.

## Problem Statement

Suppose there are many machines (*m*) where each machine is responsible for a specific operation. There are several jobs (*n*) that need to be performed. Each job arrives at a certain time (*t*). Also each job has operations it needs to perform in order.  The problem is to schedule the jobs on the machines so as to minimize the total time of schedule i.e. the time it takes from when the jobs are first started until all the jobs are completed. There are certain constraints:

1. No other operation can be performed on that job until the previous operation for that job has completed.
2. A machine can work on one operation on a single job at a time.
3. A task once started, must be completed. (No pre-emption).

## Area or Scope of Investigation

Our main area of investigation is to simulate the distributed environment, where there are many machines that can only do certain types of jobs. Then we create a scheduler that accepts jobs and based on whether the machine it has to operate on is empty or not, sends them for processing to that machine. These actions are logged on and the changes on various performance parameters are then shown using graphs and bar charts.

# Theoretical Basis and Literature Review

## Definition of the Problem

The aim of this problem is to analyze when to apply what heuristic so that we may optimize the problem of job scheduling in a distributed environment. Mainly we will be looking at three basic performance factors:

a. Makespan: The makespan is the total length of the schedule i.e. when all the jobs have finished processing from the time the first job started its processing.
b. Maximal Machine Workload: The maximum working time a job is spent on any machine.
c. Total Workload of Machines: Total working time over all machines.

## Theoretical Background of the Problem & Related Research

The simplest variation of job scheduling involves a set of ‘n’ jobs, that need to be performed. In a distributed environment, however, there are many machines which can be assigned a certain task and more machines work in parallel so that it takes lesser amount of time.

If there are m machines, each of which is required to perform n independent operations. The combination can be potentially exploded up to (n!)m operational sequences. Job shop scheduling is one of the most famous scheduling problems. Therefore it becomes important to measure the performance of these tasks and get an optimum strategy.

# Hypothesis

1. Any operation cannot be interrupted in between its process.
2. The operation order between jobs are arbitrary.
3. All jobs have same priority
4. Any job can be processed at the starting time of the scheduling.
5. Process time is the total time in which setup time and transfer time are included.

We measure overall Makespan (time of total completion of all jobs), and time a job spends on a certain machine, maximal workload using different scheduling algorithms outlined. From them, we build various comparison diagrams comparing these algorithms and establishing their strengths and weaknesses.

## Methodology

### Firefly Algorithm for Solving Job Shop

The main steps involved in Firefly Algorithms are as follows,

1. Initializing a swarm of fireflies, determined by the flashing intensity.
2. Doing pair wise comparison of the light intensity, move the firefly with less intensity towards high intensity one. Moving distance is directly dependent on the attractiveness.
3. A new firefly is evaluated after moving and light intensity is updated.
4. We update the best solution in every iteration.
5. The pair wise comparison is done till the end condition is met.
6. The Final solution that we get is the most optimal one.

Table 1 contains example of FJSP with three jobs and four machines. Here job J1 has three operations, J2 has two operations, J3 has three operations. In table you can find different time to run the same operation on different machine. This sequence of the job should not be change for execution.

| Job | Position | Operation | M1 | M2 | M3 | M4 |
|---|---|---|---|---|---|---|
| J1 | 1 | O (1, 1) | 2 | - | 1 | 6 |
|  | 2 | O (1, 2) | 5 | 3 | - | 2 |
|  | 3 | O (1, 3) | - | 2 | 4 | - |
| J2 | 4 | O (2, 1) | 7 | - | - | 11 |
|  | 5 | O (2, 2) | 4 | 4 | 12 | 8 |
| J3 | 6 | O (3, 1) | 2 | - | 7 | 9 |
|  | 7 | O (3, 2) | 3 | 5 | 8 | 1 |
|  | 8 | O (3, 3) | 4 | 3 | - | 5 |

Table 1: Example Flexible Job Shop Problem

## Model

- Makespan (Cm) : The total time taken to completely process all the jobs i.e the total length of the schedule.
- Maximal machine Workload (Wm) : The maximum amount of working time spent on any machine.
- Total Workload of machines (Wt) : The total working time combined over all the machines.

$$\min f_1 = \max_{1 \le k \le m} (C_k)$$

Minimisation of maximal completion time (makespan) of the machines

$$\min f_2 = \max_{1 \le k \le m} (W_k)$$

Minimisation of maximal machine critical workload

$$\min f_3 = \sum_{k=1}^{m} W_k$$

Minimisation of total workload of machines

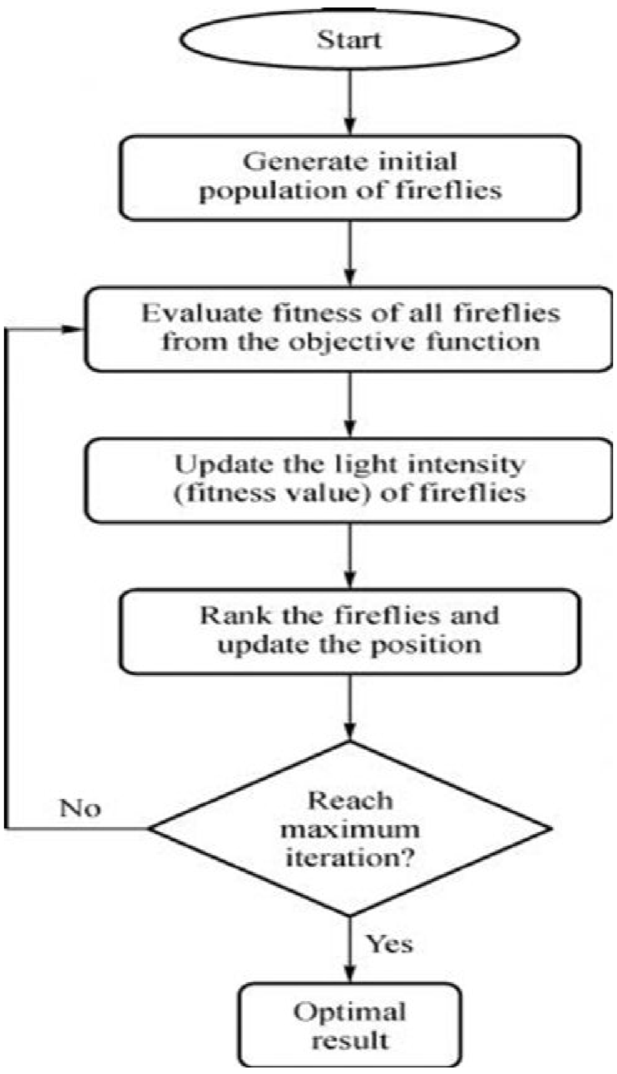
$$\text{Minimize } F(c) = W_1 \times f_1 + W_2 \times f_2 + W_3 \times f_3$$

Subject to:

$$W_1 + W_2 + W_3 = 1, \quad 0 \le W_1, W_2, W_3 \le 1$$

F(c) denotes the combined objective function value of a schedule. W1, W2 and W3 represent the weight coefficients.

## Flow Diagram of the algorithm


Start
Generate initial population of fireflies
Evaluate fitness of all fireflies from the objective function
Update the light intensity (fitness value) of fireflies
Rank the fireflies and update the position
Reach maximum iteration?
No
Yes
Optimal result

The flow diagram illustrates the flow of the firefly algorithm. Initialization is done in the starting phase. Then, each firefly is evaluated in the population. If the termination criteria is met the the solution with minimal value is the outputted. If not, the positions of the firefly is updated and repeated till the termination. In the local search  step, we search the neighbourhood for the solution.

## Firefly rules

1. All fireflies are unisex, which means that they are attracted to other fireflies regardless of their sex
2. Attractiveness of a firefly is proportional to their brightness; thus for any two flashing fireflies, the less brighter one will move towards the brighter one. The attractiveness is proportional to the brightness and they both increase as their distance decreases. If there is no brighter one than a particular firefly, it will move randomly
3. The brightness of a firefly is determined by the value of the objective function

## Firefly algorithm - Pseudo Code

```
Objective function f(x), x = (x_i, . . . , x_d)^T
Generate initial population of fireflies x_i(i = 1, 2, . . . ,n)
Light intensity I_i at x_i is determined by f (x_i)
Define light absorption coefficient $\gamma$
while  (t < Max  Generation)
    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
```

## Governing Factors

### Distance

The distance between any two fireflies i and j, at positions $x_i$ and $x_j$, respectively, can be defined as a Cartesian distance:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2}$$

where $x_{i,k}$ is the kth component of the spatial coordinate $x_i$ of the *i*th firefly, and d is the number of dimensions.

### Attractiveness

The attractiveness of a firefly is determined by its light intensity. Each firefly has its distinctive attractiveness β, which implies how strong it attracts other members of the swarm. The form of attractiveness function of a firefly is the following monotonically decreasing function.

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

where r is the distance between two fireflies, $\beta_0$ is the attractiveness at r=0 and γ is a fixed light absorption coefficient.

### Movement

The movement of a firefly i, which is attracted by a more attractive (i.e. brighter) firefly j, is given by the following equation.

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha(rand - 1/2)$$

where the first term is the current position of a firefly, the second term is due to attraction and the third term is a randomisation with α being the randomisation parameter, while rand is a random number generator uniformly distributed in the space [0, 1]. The γ value ensures the firefly algorithm to search for the global solution in the large search space.

## Solution representation

The FJSP problem is a combination of machine assignment and operation scheduling decisions, so the solution can be expressed by the assignment of operations on machines and the processing sequence of operations on the machines. In this study, we used improved A-string (machine assignment vector) and B-string (operation scheduling vector) representation for each firefly that could be used to solve the multi-objective FJSP efficiently

**Machine assignment:** An array of integer values is used to represent a machine assignment vector. The length of the array is equal to the sum of all operations of all jobs. Each integer value equals the index of the array of alternative machine set of each operation.

**A - Machine Assignment Vector (MAV)**

| Operation | O (1, 1) | O (1, 2) | O (1, 3) | O (2, 1) | O (2, 2) | O (3, 1) | O (3, 2) | O (3, 3) |
|---|---|---|---|---|---|---|---|---|
| Machine | 3 | 4 | 2 | 1 | 1 | 1 | 4 | 2 |

**Operation scheduling:** Operation scheduling vector has the same length as the machine assignment vector. It consists of a sequence of job numbers in which job number i occurs $n_i$ times. It can avoid creating an infeasible schedule when replacing each operation by the corresponding job index.

**B - Operation Scheduling Vector (OSV)**

| Sequence | 4 | 6 | 1 | 2 | 5 | 7 | 8 | 3 |
|---|---|---|---|---|---|---|---|---|
| Operation | O (2, 1) | O (3, 1) | O (1, 1) | O (1, 2) | O (2, 2) | O (3, 2) | O (3, 3) | O (1, 3) |

## Firefly Evaluation

Each firefly is represented by a machine assignment vector and an operation scheduling vector. Using the permutation of these vectors in the population each firefly is evaluated to determine the objective function. The objective function value of each firefly is associated with the light intensity of the corresponding firefly.

In this work, the evaluation of the goodness of schedule is measured by the combined objective function, which can be calculated using

$$\text{Minimize } F(c) = W_1 \times f_1 + W_2 \times f_2 + W_3 \times f_3$$

Subject to:

$$W_1 + W_2 + W_3 = 1, \quad 0 \leq W_1, W_2, W_3 \leq 1$$

Below table shows the initial firefly generation with calculated combined objective function value for the example problem given in Table 1.

| Population $P$ | Initial firefly generation | | Objective function | | | |
|---|---|---|---|---|---|---|
| | A-string | B-string | $f_1$ | $f_2$ | $f_3$ | $F(c)$ |
| 1 | 3 4 2 1 1 1 4 2 | 4 6 1 2 5 7 8 3 | 15 | 13 | 22 | 15.8 |
| 2 | 3 4 2 1 1 1 4 2 | 4 6 1 2 5 7 3 8 | 13 | 13 | 22 | 14.8 |
| 3 | 3 4 2 1 2 1 4 2 | 4 6 1 5 7 2 8 3 | 16 | 9 | 22 | 15.1 |
| 4 | 3 2 2 4 2 1 4 1 | 1 6 2 4 7 3 5 8 | 16 | 12 | 28 | 17.2 |
| 5 | 3 4 3 1 2 1 4 2 | 6 7 1 2 4 5 3 8 | 16 | 9 | 24 | 15.5 |
| 6 | 3 4 3 1 2 1 4 2 | 4 6 1 2 7 5 8 3 | 14 | 9 | 24 | 14.5 |
| 7 | 3 4 3 1 2 1 4 2 | 1 4 2 6 7 8 3 5 | 17 | 9 | 24 | 16.0 |
| 8 | 3 4 2 1 2 1 4 2 | 6 4 7 1 8 5 2 3 | 15 | 9 | 22 | 14.6 |
| 9 | 3 4 2 1 2 1 4 2 | 1 2 4 6 3 5 7 8 | 14 | 9 | 22 | 14.1[a] |
| 10 | 1 4 2 1 1 1 4 2 | 6 4 7 1 5 8 2 3 | 15 | 15 | 23 | 16.6 |

## Solution Updation

In firefly algorithm, firefly movement is based on light intensity and comparing it between two fireflies. The attractiveness of a firefly is determined by its brightness, which in turn is associated with the encoded objective function.

Distance: There are two possible ways to measure the distance between two permutations: (a) Hamming distance and (b) the number of the required swaps of the first solution in order to get the second one.

In A-string, the distance between any two fireflies i and j, at positions xi and xj, respectively, can be measured using Hamming distance. The Hamming distance between two permutations is the number of non-corresponding elements in the sequence. The distance between two permutations in the B-string can be measured using swap distance. The swap distance is the number of minimal required swaps of one permutation in order to obtain the other one.

**Attraction and movement:**

$$x_i = \beta(r)(x_j - x_i)$$

The attraction step can be broken in two sub-steps β-step and α-step interchangeable. The β step must be computed before the α-step while finding the new position of the firefly.

$$x_i = x_i + \alpha(rand - 1/2)$$

The steps involved in β-step are as follows:
Let *d* the difference between the elements of best firefly and other firefly *r* Hamming distance for A-string and swap distance for B string

**Step 1:**
All necessary insertions in the machine assignment vector and all necessary pair-wise exchanges the operation scheduling vector are stored in d.

**Step 2:**
Counting the number of insertions in machine assignment vector and number of pair-wise exchanges in operation scheduling vector between two firefly solutions will give the hamming distance and swap distance respectively, which is stored in r.

**Step 3:**
Compute β probability using $\beta(r) = \beta_0/(1 + \gamma r^2)$

**Step 4:**
Random number rand( ) is generated in the range (0,1) according to the number of counts in the set d of machine assignment vector and operation scheduling vector.

**Step 5:**

If rand( )≤β, then the corresponding insertion in the machine assignment vector and pair-wise exchange in the operation scheduling vector is performed on the elements of the current firefly. This procedure moves the current firefly to the global best position, which is controlled by the β probability.

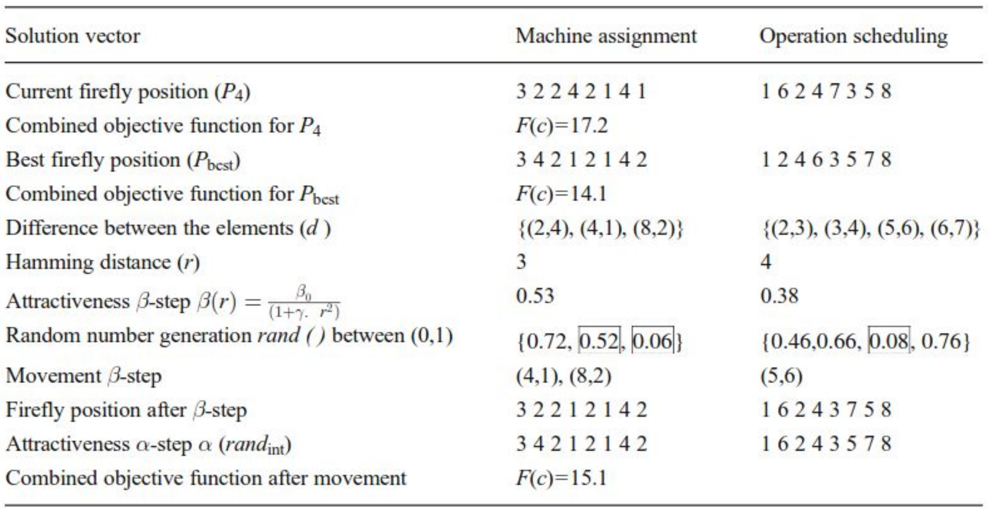
**α-Step:**

The random movement of firefly α (rand−1/2) is approximated as α (randint), given by

$$x_i = x_i + \alpha(rand_{\text{int}})$$

In machine assignment vector α-step is applied by randomly choosing an element position using α (randint) and the machine assignment component of that position is replaced with a new machine having minimum processing time among the set of available machines. In operation scheduling vector α-step is applied by randomly choosing an element position by using α (randint) and swap with another position in the string which is also chosen at random.

The below table illustrates the firefly position update procedure for population 1 in the first generation. The parameters used in this illustration are as follows: β0=1, γ=0.1, α=1. The improvement of objective function value F(c) from 17.2 to 15.1 shows the movement of firefly from its current solution to the best firefly solution. The objective function of each firefly population obtained in the first generation replaces its previous value (P), if its current solution is better than the previously stored firefly solution. The best solution of the first iteration replaces the global best firefly solution ($P_{best}$) if it is better than the previously stored global best solution. The procedure is repeated until the termination criterion is satisfied. In this study, the termination criterion is the total number of generations.



| Solution vector | Machine assignment | Operation scheduling |
|---|---|---|
| Current firefly position ($P_4$) | 3 2 2 4 2 1 4 1 | 1 6 2 4 7 3 5 8 |
| Combined objective function for $P_4$ | $F(c)$=17.2 | |
| Best firefly position ($P_{\text{best}}$) | 3 4 2 1 2 1 4 2 | 1 2 4 6 3 5 7 8 |
| Combined objective function for $P_{\text{best}}$ | $F(c)$=14.1 | |
| Difference between the elements ($d$ ) | {(2,4), (4,1), (8,2)} | {(2,3), (3,4), (5,6), (6,7)} |
| Hamming distance ($r$) | 3 | 4 |
| Attractiveness $\beta$-step $\beta(r) = \frac{\beta_0}{(1+\gamma.\ r^2)}$ | 0.53 | 0.38 |
| Random number generation *rand ( )* between (0,1) | {0.72, $\boxed{0.52}$, $\boxed{0.06}$} | {0.46,0.66, $\boxed{0.08}$, 0.76} |
| Movement $\beta$-step | (4,1), (8,2) | (5,6) |
| Firefly position after $\beta$-step | 3 2 2 1 2 1 4 2 | 1 6 2 4 3 7 5 8 |
| Attractiveness $\alpha$-step $\alpha$ ($rand_{\text{int}}$) | 3 4 2 1 2 1 4 2 | 1 6 2 4 3 5 7 8 |
| Combined objective function after movement | $F(c)$=15.1 | |

## Termination Condition

Fireflies will evolve in each iteration (generation). Finally all firefly will move towards the brighter firefly and try to be as bright as brightest. This means global optima is reached. The sequence derived from the state will the best one.

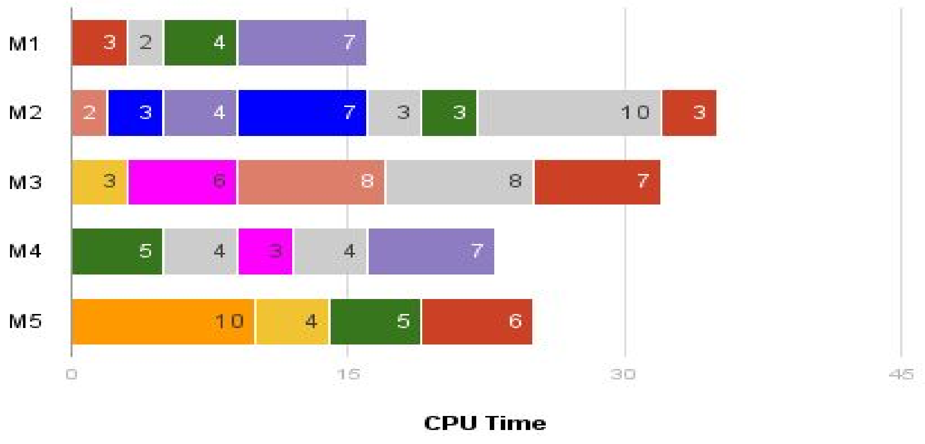
## Comparison of Firefly with SJF algorithm

Consider a problem with 8 jobs and 5 machines to be optimized using Firefly and SJF algorithms

| Job | $O_{ij}$ | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ |
|---|---|---|---|---|---|---|
| $J_1$ | $O_{1,1}$ | 5 | 3 | – | – | – |
| | $O_{1,2}$ | – | 7 | – | – | – |
| $J_2$ | $O_{2,1}$ | – | – | 6 | – | – |
| | $O_{2,2}$ | – | – | – | 3 | 4 |
| $J_3$ | $O_{3,1}$ | – | 4 | 6 | – | – |
| | $O_{3,2}$ | 7 | – | – | – | – |
| | $O_{3,3}$ | – | – | – | 7 | – |
| $J_4$ | $O_{4,1}$ | – | – | – | – | 10 |
| $J_5$ | $O_{5,1}$ | – | – | – | 5 | – |
| | $O_{5,2}$ | 4 | 5 | 8 | – | – |
| | $O_{5,3}$ | – | – | – | 6 | 5 |
| | $O_{5,4}$ | – | 3 | – | – | 4 |
| $J_6$ | $O_{6,1}$ | – | 2 | 6 | – | – |
| | $O_{6,2}$ | – | - | 8 | – | – |
| $J_7$ | $O_{7,1}$ | – | – | 3 | 8 | – |
| | $O_{7,2}$ | – | – | – | 7 | 4 |
| $J_8$ | $O_{8,1}$ | 3 | – | 5 | – | – |
| | $O_{8,2}$ | – | – | - | 9 | 6 |
| | $O_{8,3}$ | – | – | 7 | – | – |
| | $O_{8,4}$ | – | 3 | – | – | – |

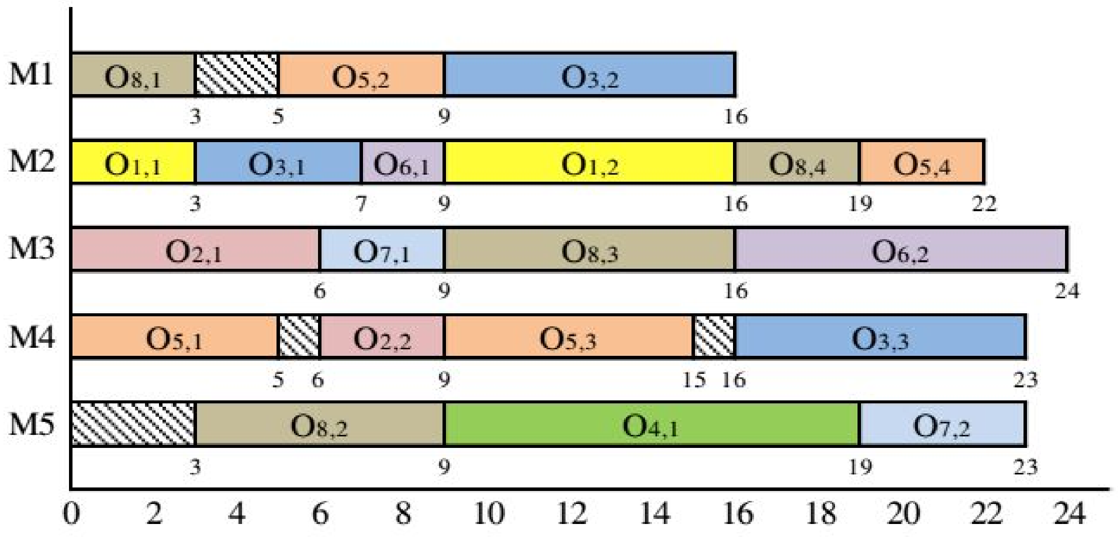
Problem 8×5 with 20 operations

**Computational results of SJF:**



Gantt chart of problem 8×5 with 20 operations (Cm=35, Wm=25, Wt=97)

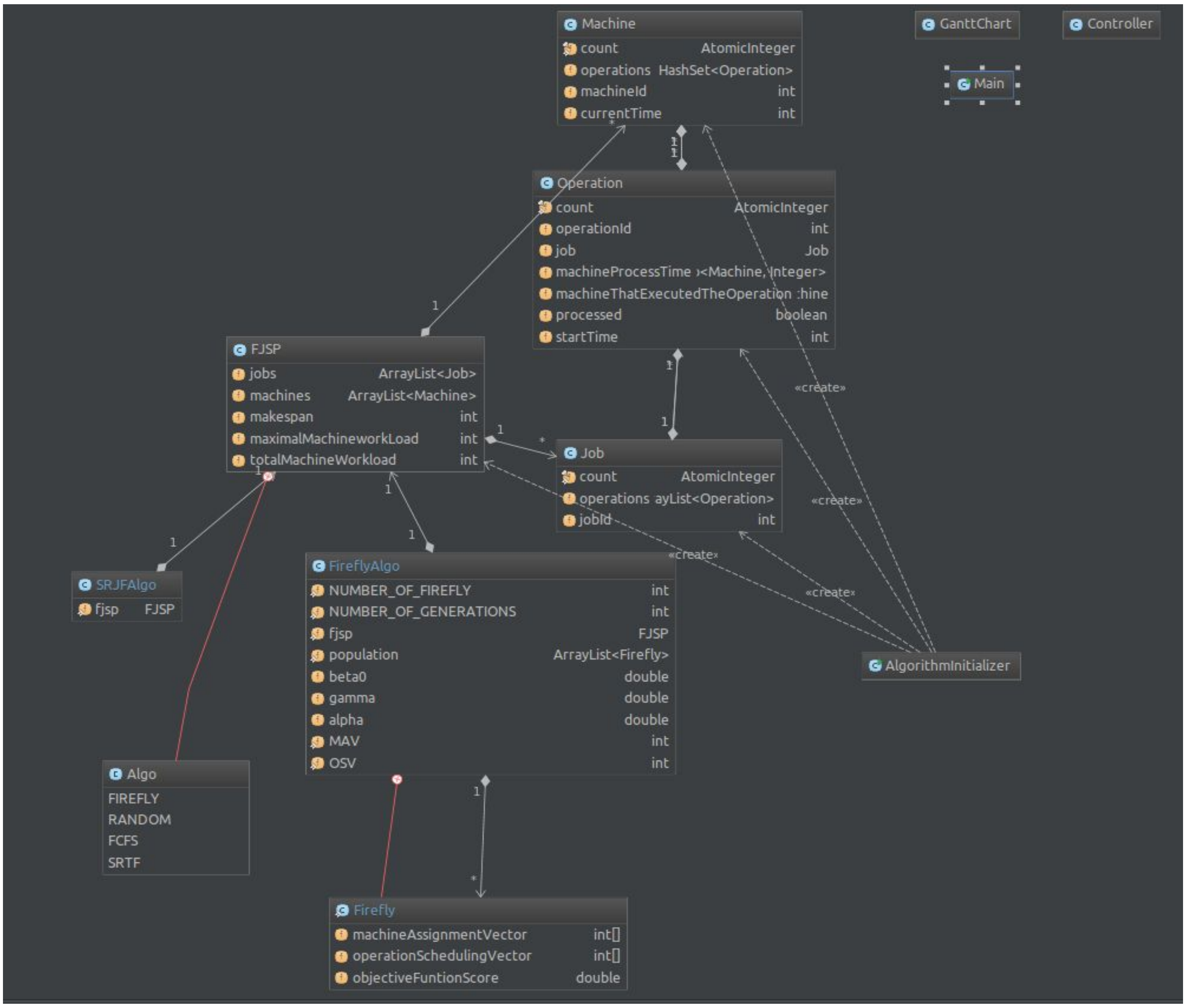
**Computational results of Firefly:**



Gantt chart of problem 8×5 with 20 operations (Cm=24, Wm=24, Wt=101)

Here we observe the makespan, and maximum machine workload to reduce by a great extend and the total workload of machines to increase, all these factors of the Firefly algorithm beat the SJF algorithm to solve optimize the Job Shop Scheduling Problem.

## Code Design



https://github.com/suragys/Firefly-Algorithm-for-Job-Shop-Scheduling

## Conclusion

An effective discrete firefly algorithm is proposed for multiobjective flexible job shop scheduling with limited resource constraints. The objective function considered is minimisation of makespan, maximal workload and total workload of machines. Instead of applying the standard firefly algorithm, we proposed the discrete version of the continuous function such as distance, attractiveness and movement to update a firefly position. A combination of rules is utilised for generating the initial population. In addition, two neighbourhood structures in relation to machine assignment and operation sequence were used in the algorithm to direct the local search to the more promising search space. The performance of the presented approach is evaluated in comparison with the results obtained from other algorithm. The obtained computational results and time demonstrated the effectiveness of the proposed approach. The future work is to enhance the convergence capability of the algorithm and to generalise the application of the proposed discrete firefly algorithm for other combinatorial optimisation problems.

## References

1. S. Karthikeyan & P. Asokan & S. Nickolas, *‘**A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints**’*, Springer-Verlag London (2014).
2. Aphirak Khadwilard, Sirikarn Chansombat, Thatchai Thepphakorn, Peeraya Thapatsuwan, Warattapop Chainate and Pupong Pongcharoen, ***‘Application of Firefly Algorithm and Its Parameter Setting for Job Shop Scheduling***’, The Journal of Industrial Technology, Vol. 8, No. (1 January – April 2012).
3. Yang XS (2013) ‘***Multiobjective firefly algorithm for continuous optimization’***. Eng Comput 29(2):175–184