Artificial Bee Colony for Job Shop Scheduling Problem

B.Tech Project Report

Submitted in partial fulfillment
of the requirements for the award of the degree
of

BACHELOR OF TECHNOLOGY

Submitted by

Vinay Saini	IIT2009074
Vikash Gupta	IIT2009088
Vinay Kumar	IIT2009168

Under the Guidance of:

Dr. K.P. Singh

IIIT-Allahabad

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
ALLAHABAD – 211 012 (INDIA)

December 4, 2012.

Candidate's Declaration

We hereby declare that the work presented in this project report entitled "**An Artificial Bee Colony for Job Shop Scheduling Problem"**, submitted as a semester Mini project for Seventh semester of B.Tech. (IT) is an authenticated record of our original work carried out from July 2012 to December 2012 under the guidance of Dr. K.P. Singh. Due acknowledgements has been made in the text to all other material used.

Place: Allahabad
Date: December 4, 2012.

Vinay Saini (IIT2009074)
Vikash Gupta (IIT2009088)
Vinay Kumar(IIT2009168)

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Place: Allahabad
Date: December 4, 2012.

Dr. K.P. Singh
…………………………

<u>**Acknowledgements**</u>

We would like to thank **Dr. K.P. Singh** who has been a constant source of inspiration and guidance for us. I also owe a great deal of gratitude project evaluation committee members, for their valuable insights that help me view the project in the new light. Further I take this opportunity to express my gratitude to all my teachers and project guides during the course of my study at IIITA, to help me learn so much in a short span of four years at IIITA – learning memories, enough to last a lifetime.

TABLE OF CONTENTS

ABSTRACT

An artificial bee colony (ABC) is an optimization algorithm based on the intelligent foraging behavior of honey bee swarm. The job shop scheduling problem (JSSP) is the problem of assigning the job in the system in a manner that will optimize the overall performance, while assuring the correctness of the result. ABC algorithm is proposed in this paper for solving the JSSP with the criterion minimize the maximum completion time (makespan).

INTRODUCTION

JSSP is an optimization problem that can be describe in terms of a set of jobs, each with one or more operations. The operations of a job have to be processed in a specified sequence on a specific set of machines. The time required for all operations to complete their processes is called the makespan. The objective of JSSP aims to minimize the makespan value [4].

Many approaches using both mathematical formulations and heuristic method have been developed to solve this problem. In recent years several algorithm employing a heuristic approach such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) have been applied to solve JSSP.

In this paper we are using ABC algorithm for solving job shop scheduling problem. Artificial Bee Colony (ABC) is one of the most recently defined algorithms by Dervis Karaboga in 2005, motivated by the intelligent behavior of honey bees. It is as simple as Particle Swarm Algorithm and differential algorithms (DE), and uses only common control parameters such as colony size and maximum cycle number. ABC as an optimization tool, provides a population based search procedure in which individuals called food positions are modified by the artificial bees with time.

Motivation

The job-shop scheduling problem has attracted many researchers' attention in the past few decades, and many algorithms based on heuristic algorithms genetic algorithms and ABC algorithms have been presented to solve it, respectively. Many comparative studies showed that the performance of the ABC algorithm was competitive to other population based algorithm with advantage of employing fewer control parameters in the continuous space.

Goal

Goal of our project is to minimize the maximum completion time (makespan) of all jobs in JSSP using ABC algorithm.

Literature Survey

The ABC algorithm is relatively new swarm intelligence based optimizer. It mimics the cooperative foraging behavior of a swarm of honey bees. ABC was initially proposed by Karaboga in 2005[6] for optimizing multi-variable and multi-modal continuous functions. The latest research has revealed some good properties of ABC. Especially, the number of control parameters in ABC fewer than of other population based algorithms, which makes it easier to be implemented. Meanwhile the optimization performance of ABC is comparable and sometimes better to the other algorithm. Therefore ABC has aroused much interest and has been successfully applied to different kinds of optimization.

Since 2005, D. Karaboga and his research group have been studying the ABC algorithm and its applications to real world problems. Karaboga and Basturk[1] have investigated the performance of the ABC algorithm on unconstrained numerical optimization problems and its extended version for the constrained optimization problems and Karaboga et al. applied ABC algorithm to neural network training. In 2010, Hadidi[9] et al. employed an Artificial Bee Colony (ABC) Algorithm based approach for structural optimization. In 2011, Zhang[10] et al. employed the ABC for optimal multi-level thresholding, MR brain image classification, cluster analysis, face pose estimation, and 2D protein folding.

Job shop scheduling problem (JSSP) is one of the well-known hardest combinatorial optimization problems. JSSP being amongst the worst member of the class of NP-hard problems (Gary and Johnson, [1979]), there is still a lot of room for improvement in the existing techniques. Because of its large solution space JSSP is considered to be comparatively one of the hardest problem to solve. “If there are *n* jobs and *m* machines the number theoretically possible solutions is equal to $(n!)^m$”. Among these solutions an optimal solution, for a certain measure of performance, can be found after checking all possible alternatives. But checking of all the possible alternatives can only be possible in small size of problem.

Xia and Wu proposed a hierarchical approach by using particle swarm optimization (PSO) to assign operations to machines and by using simulation annealing (SA) to schedule operations on each machine for solving the multi-objective JSSP. An advance Tabu Search (TS) proposed by Eugeniusz Nowicki and Czeslaw Smutnicki[8] to solve JSSP. Cheng studied the problem of n jobs on m identical parallel machines. He determined an optimal schedule so as to minimize the sum of all completion times, earliness and tardiness penalties of each job and showed that this problem is NP-hard, and presented a heuristic algorithm to find near optimal solutions for the problem. Huang used Ant Colony Optimization (ACO) algorithm to solve the job shop scheduling problem and found ACO algorithm performs well in scheduling problem and uses less time to solve the problem.

Proposed Approach

Our project is divided into three phases:-

- Understanding Artificial Bee Colony (ABC) algorithm.
- Understanding the Job Shop Scheduling Problem (JSSP).
- Implementing the ABC to solve the JSSP.

Artificial Bee Colony Algorithm[1][4]

In the ABC algorithm, the artificial bees are classified into three groups: the employed bees, the onlookers and the scouts. A bee that is exploiting a food source is classified as employed. The employed bees share information with the onlooker bees, which are waiting in the hive and watching the dances of employed bees. The onlooker bees will then choose a food source with probability proportional to quality of that food source. Therefore, good food sources attract more bees than the bad ones. Scout bees search for new food sources randomly in the vicinity of the hive. When a scout or onlooker bee finds a food source, it becomes employed. When a food source has been full exploited, all the employed bees associated with it will abandon the position, and may become scouts again. Therefore, scouts bees perform the job of "exploration", whereas employed and onlooker bees perform the job of "exploitation".

In ABC, the first half of the colony consists of employed bees and the other half are onlookers. The number of employed bees is equal to the number of food sources (SN) because it is assumed there is only one employed bee for each food sources. The main procedure of ABC can be described as follows:

Step 1: Initialize the food sources.
Step 2: Each employed bee start to work on a food source.
Step 3: Each onlooker bee selects a food source according to the nectar information shared by the employed.
Step 4: Determine the scout bees, which will search for food sources in a random manner.
Step 5: Test whether the termination condition is met. If not, go back to Step 2.

The detailed description for each step is given below.

(1) *The initialization phase.* The SN initial solutions are randomly generated D-dimensional real vectors. Let $x_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}$ represent the i-th food sources, which is obtained by

$$x_{i,d} = x_d(min) + r \times (x_d(max) - x_d(min)),\ d = 1, \ldots , D \qquad (1)$$

where *r* is uniform random number in the range [0,1], and $x_d(min)$ and $x_d(max)$ are the lower and upper bounds for dimension d, respectively.

(2) *The employed bee phase.* In this stage, each employed bee is associated with a solution. She exerts a random modification on the solution (original food source) for finding a new

solution (new food source). This implements the function of neighborhood search. The new solution v_i is generated from x_i using a differential expression:

$$v_{i,d} = x_{i,d} + r' \times (x_{i,d} - x_{k,d}) \qquad (2)$$

where *d* is randomly selected from {1, … , D}, *k* is randomly selected from {1, … , SN} such that *k != i* and r′ is a uniform random number in the range [-1,1].

Once v_i is obtained, it will be evaluated and compared to x_i. If the fitness of v_i is better than that of x_i (i.e, the nectar amount of the new food source is higher than the old one), the bee will forget the old solution and memorize the new one. Otherwise, she will keep working on x_i.

(3) *The onlooker bee phase.* When all employed bees have finished their local search, they share the nectar information of their food source with the onlookers, each of whom will then select a food source in a probabilistic manner. The probability p_i by which an onlooker bee chooses food source x_i is calculated as follows:

$$p_i = f_i / \sum_{i=1}^{SN} fi$$

where f_i is the fitness value of x_i . Obviously, the onlooker bees tend to choose the food sources with higher nectar amount.

Once the onlooker has selected a food source x_i , she will also conduct a local search on x_i according Eq.(2). As in the previous case, if the modified solution has a better fitness the new solution will replace x_i.

(4) *The scout bee phase.* In ABC, if the quality of a solution cannot be improved after a predetermined number *(limit)* of trials, the food source is assumed to be abandoned, and the corresponding employed bee becomes a scout. The scout will then produce a food source randomly by using Eq.(1).

Applications of ABC algorithms

- An Unconstrained Optimization Problem: Neural Network Training for the XOR problem.
- A Constrained Optimization Problem: Welded Beam Design.
- Scheduling jobs for a production machine.
- Symbolic Regression.

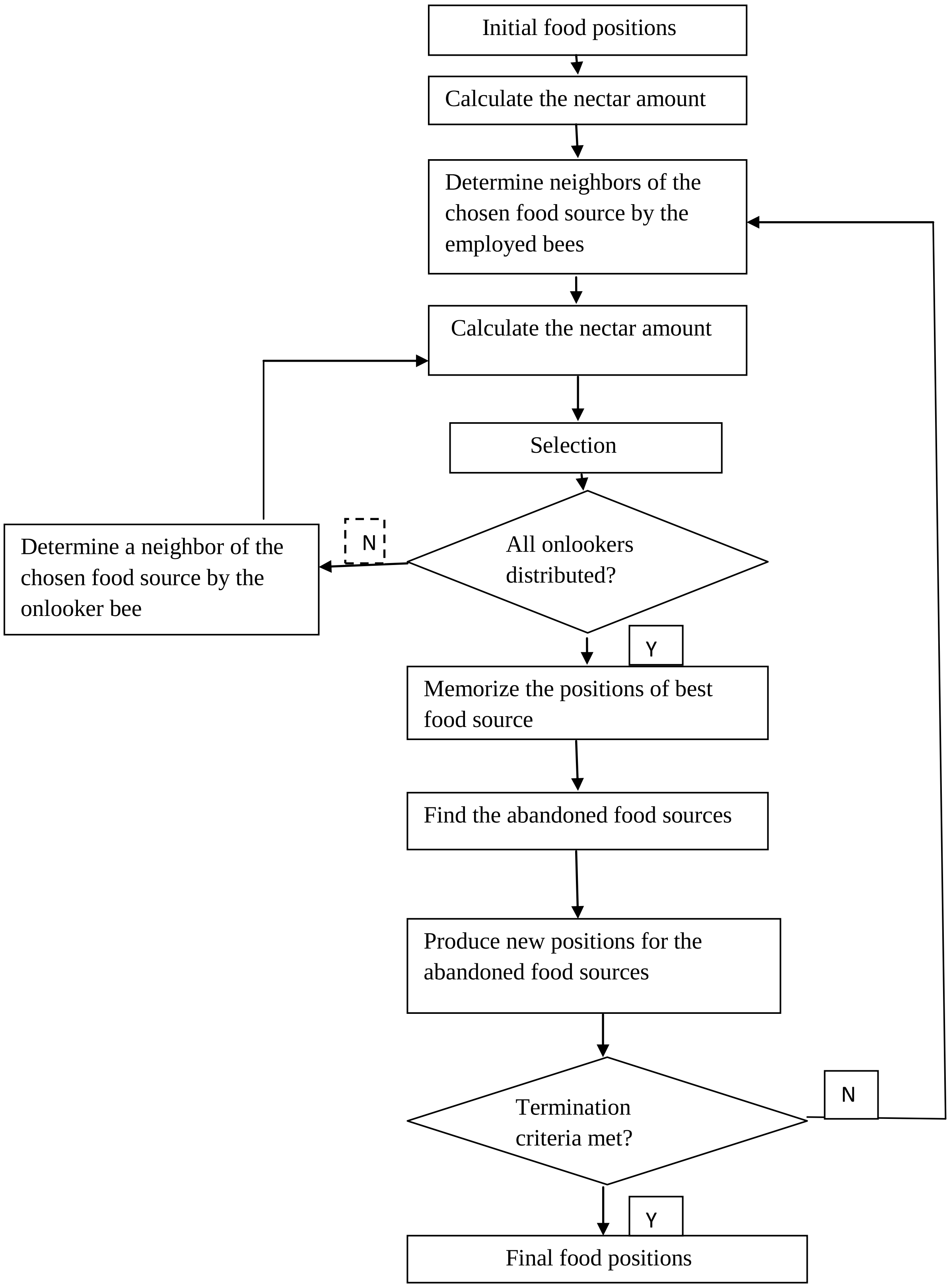

Initial food positions
Calculate the nectar amount
Determine neighbors of the chosen food source by the employed bees
Calculate the nectar amount
Selection
All onlookers distributed?
N
Determine a neighbor of the chosen food source by the onlooker bee
Y
Memorize the positions of best food source
Find the abandoned food sources
Produce new positions for the abandoned food sources
Termination criteria met?
N
Y
Final food positions

Fig. 1 Flow chart of the ABC Algorithm[2]

Job Shop Scheduling Problem[5]

A schedule problem is characterized as jobs sequence and allocation on machines during a time period so that the required fitness function is met.
The schedule problem can be defined by the next sets:
- set of jobs $J = \{J_1, J_2, \ldots, J_n\}$;
- set of machines $M = \{M_1, M_2, \ldots, M_m\}$;
- set of operation $O = \{O_1, O_2, \ldots , O_k\}$.

Job J consists of a sequence of k operations $O_{1j}, O_{2j}, \ldots, O_{ij}, \ldots O_{kj}$, which must be processed in this order, i.e. we have precedence constraints of the form $O_{ij}<O_{(i+1)j}$
($i = 1, 2, \ldots, k$-1). The precedence expresses the time continuity, where operation O_{ij} cannot start sooner than the operation O_{i+1j} finishes. For every operation O_{ij} and
machine M_{ij} a processing time T_{ij} ($i = 1, 2,\ldots,m$, $j = 1,2,\ldots,k$) is defined.

The other technological restrictions of processing operations on machine can be described as follows:-
- an operation can be processed on one machine only;
- an operation is "atomic", i.e. the production process of operation cannot be interrupted by an arrival of other operation;
- it is specified in several processes, that two operations cannot be processed on one machine at simultaneously;
- the processing time of the work must be strictly obeyed .

The target function is to minimize the finish and processing times. The scheduling problem is to find the best operation or jobs sequence on all machines in order to minimize the processing time and other required parameters. In this report an approach to solving job-shop scheduling problem with the support of particle swarm optimization based on real manufacturing system problem is described. A similar scheduling problem has been solved in and other biologically inspired method in .

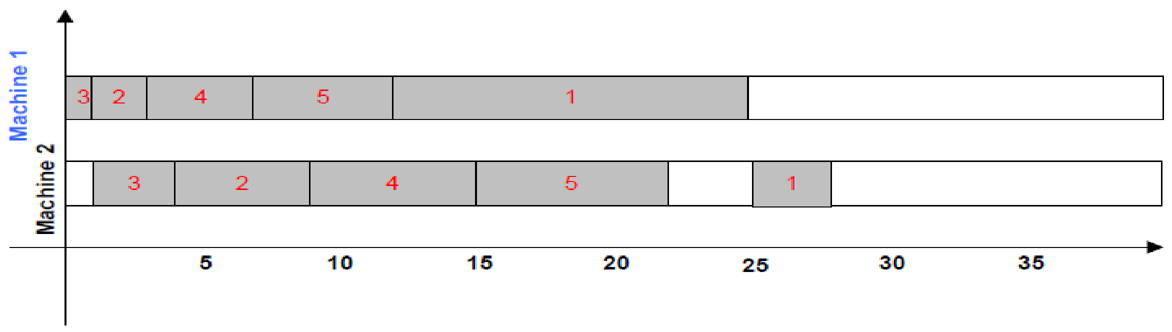
Here is the Gantt Chart for representation of scheduling of jobs on machines:-

Gantt Chart

This does not provide any rules for choosing but simply presents a graphical technique for displaying results (and schedule) and for evaluating results (makespan, idle time, waiting time, machine utilization).
5 jobs, 2 machines, each job must first go to machine 1, and then 2 – without changing order.
Processing times are:

JOBS	Machine 1	Machine 2
1	13	3
2	2	5
3	1	3
4	4	6
5	5	7

Assume order jobs are worked is {3,2,4,5,1}



Here we assume setup time is included in process time.
Makespan = 28
Machine 1 has no idle time except 3 units at end of day
Machine 2 has 3 units of idle time plus 1 unit at beginning of day.

Input

- No. of Jobs and Machines
- Processing time matrix in which the processing time for each job on each machine is stored in matrix form. It is a NxM matrix where N is the number of jobs and M is the number of machines.
- Job sequence matrix is table which store the required sequence of each job in the problem.

Output

The output is expressed as a Gantt chart representing the exact sequence on each machine and the total makespan for input.

Details about problem solution -

The first step (Initialization)-

- The Initial parameters such as number of employed bees, onlooker bees and maximum cycle number are set.
- Next, the job's processing time on each machine and the job's machine sequence will be given at this step.
- In our solution representation, a solution in JSSP is an operation scheduling list, which is represented as a food source(x) in our ABC algorithm.
- Each dimension in a food source represents one operation of a job.
- The fitness of each food source f(x) is determined by the inverse of its makespan value ($C_{max}(x)$) which is calculated during the selection of feasible solutions.

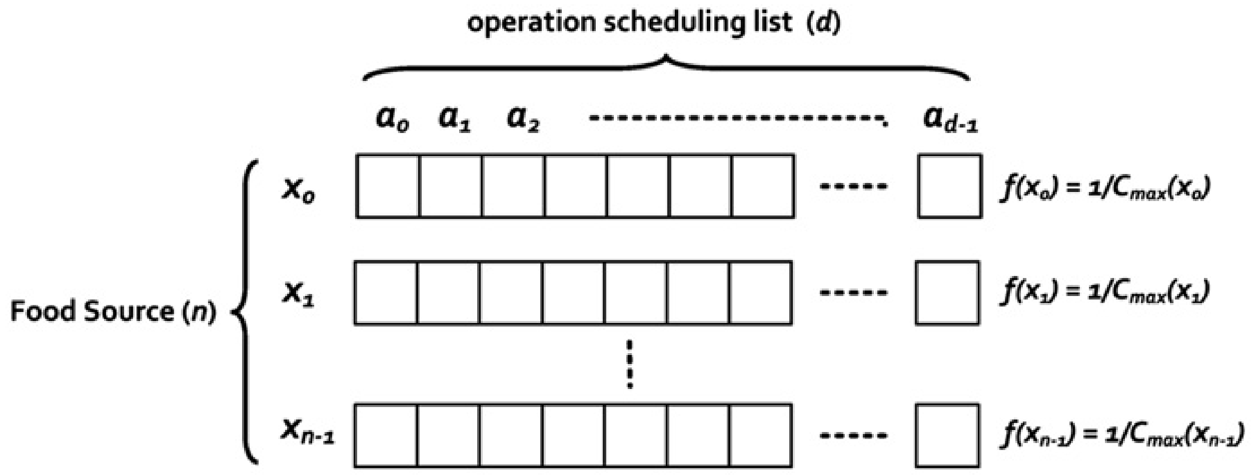

operation scheduling list (d)
a_0 a_1 a_2 a_{d-1}
Food Source (n)
x_0
x_1
x_{n-1}
$f(x_0) = 1/C_{max}(x_0)$
$f(x_1) = 1/C_{max}(x_1)$
$f(x_{n-1}) = 1/C_{max}(x_{n-1})$

Fig – Mapping between the food sources and operation scheduling list where $f(x_i) = 1/C_{max}(x_i)$.

The second step (Employed Bees Phase) –

- The candidate food sources are updated by employed bees.
- Position based crossover (PBX) is the mechanism used to update employed bee's old food sources.
- In this method, a set of job operations from the old food source is selected randomly.
- Each dimension in the operation scheduling list of the old food source is selected to produce the new position with the probability greater than 0.5. This probability value (0.5) has been chosen based on observations.

- The job operations on the neighboring food source will be selected and placed into empty positions from left to right of the operation scheduling list in the new food source.
- The old food source in the employed bee's memory will be replaced by the new candidate food source if the new food position has a better fitness value.

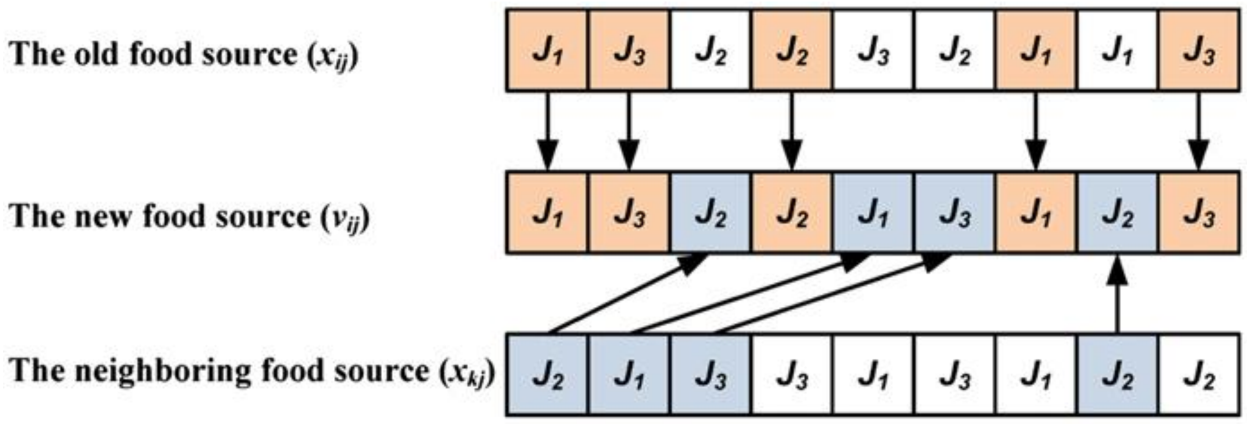

The old food source (x_{ij})
The new food source (v_{ij})
The neighboring food source (x_{kj})
J1 J3 J2 J2 J3 J2 J1 J1 J3
J1 J3 J2 J2 J1 J3 J1 J2 J3
J2 J1 J3 J3 J1 J3 J1 J2 J2

Fig – Exchanging information with a neighboring food source based on PBX method.

The third step (Onlooker bees phase) –

- The employed bees share new solutions that they have found with the onlooker bees, who then select these solutions based on probability (P_i) calculated from an equation below

$$P_i = \frac{f(Vi)}{\sum_{n=1}^{SN} f(Vi)}$$

Where $f(V_i)$ is the fitness value of the food source i and SN is the number of food sources.

- Next, we generate a random probability (p) and compare it with P_i , if P_i is greater than p then we assign ith food source to onlooker bee .
- The PBX method is also used to update the old solution of the onlooker bees to the new solution based on the neighboring food sources.
- The old food source in the onlooker bee's memory will be replaced by the new food source if the new food source has better fitness value.
- Above three steps will continue till each onlooker bee gets a food source.
- After completion of the above steps we memories best food source.

The fourth step (Scout Bees Phase) –

- Now, those food sources that have not been updated for some given number of iterations (*limit*) then employed bees abandon those food sources and these employed bees become scout bees.
- The scout bees ignore the old solution and randomly search for new solution by using random food generation.

Results and Discussions

The Gantt chart drawn as the output is displayed for different input combination.

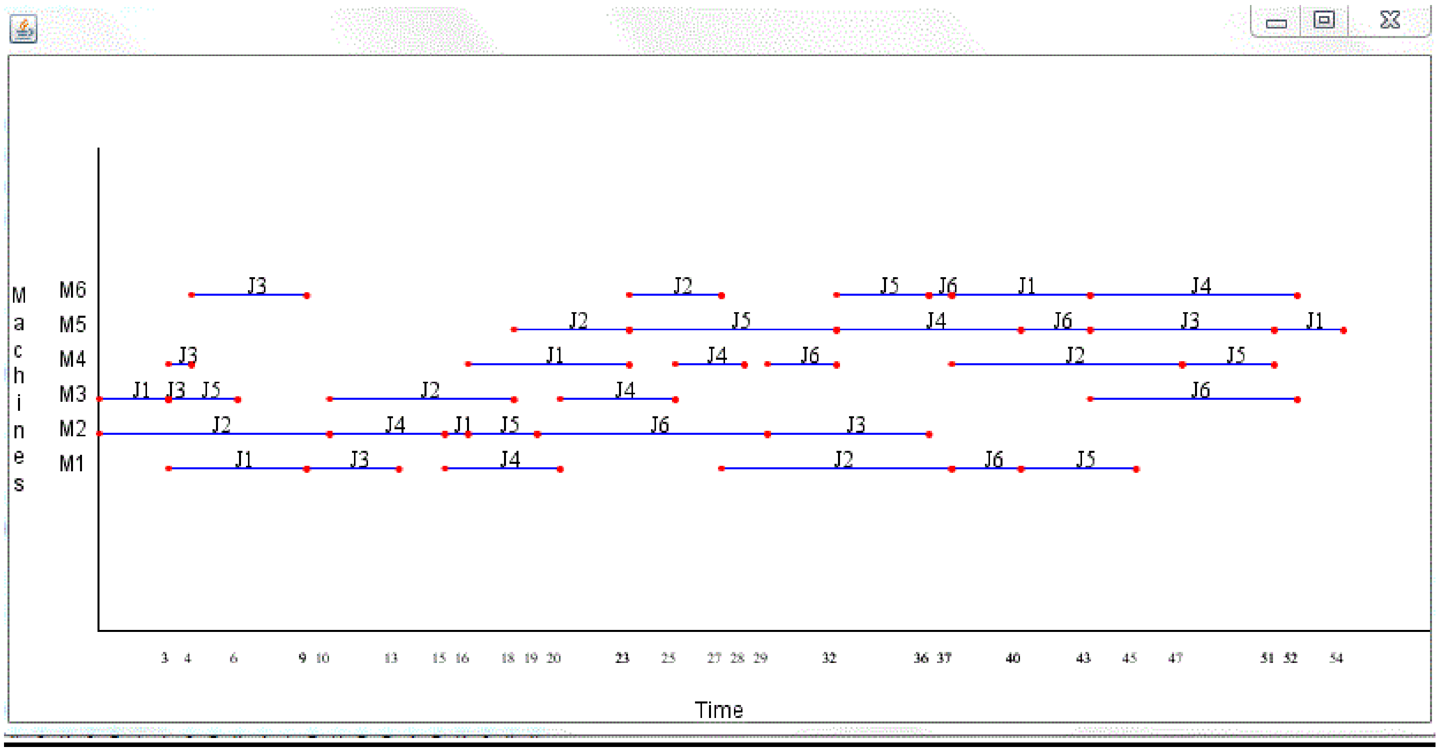


Fig. - Gant chart for 6 machines and 6 jobs (ft06)

J_i – Shows ith job — - Showed processing of jobs on machines.

M_i – Shows ith machine

At time axis we are showing end time (completion time) of jobs on machines.

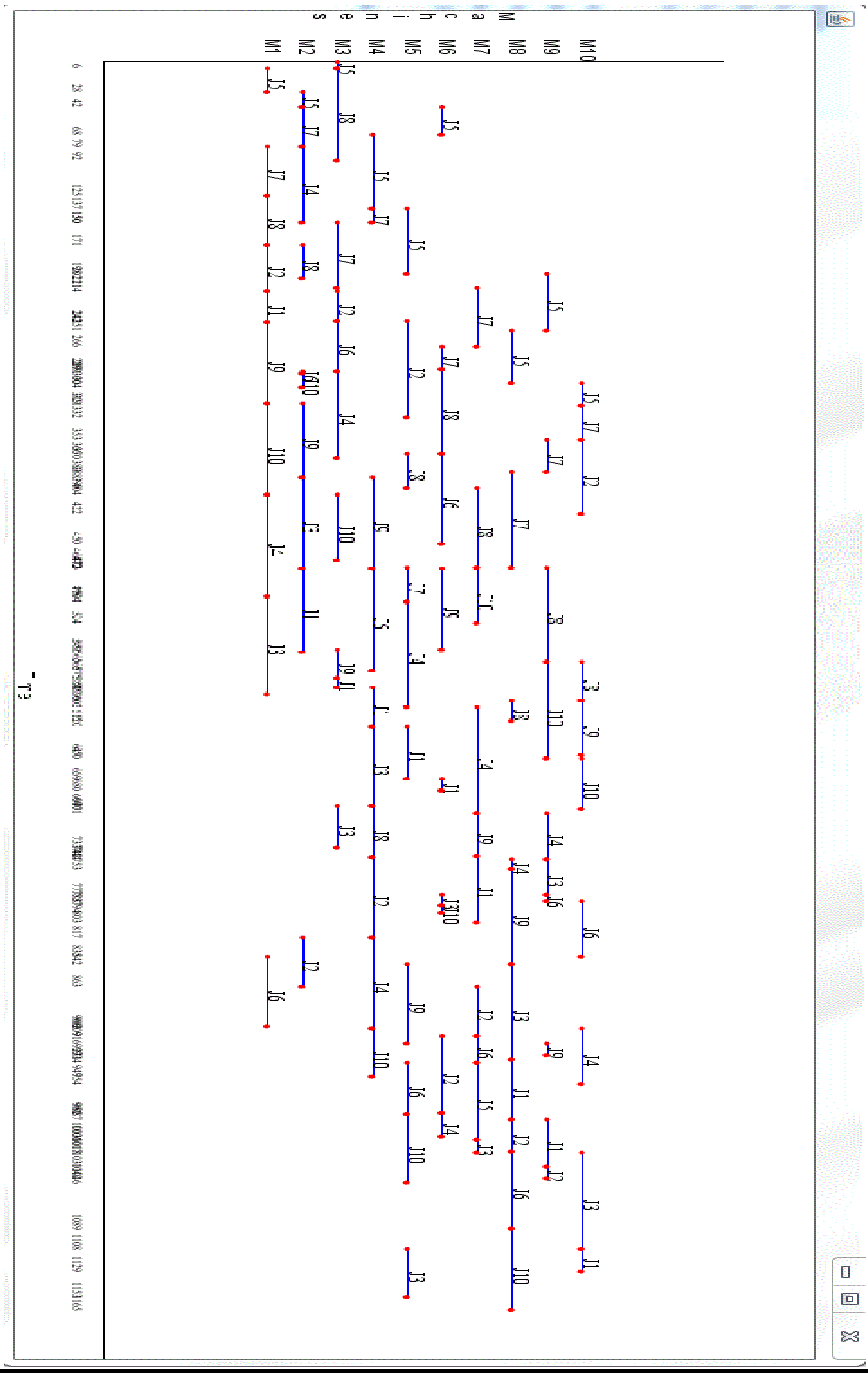

Machines
Time
M1
M2
M3
M4
M5
M6
M7
M8
M9
M10

Fig. - Gant chart for 10 machines and 10 jobs (ft10)

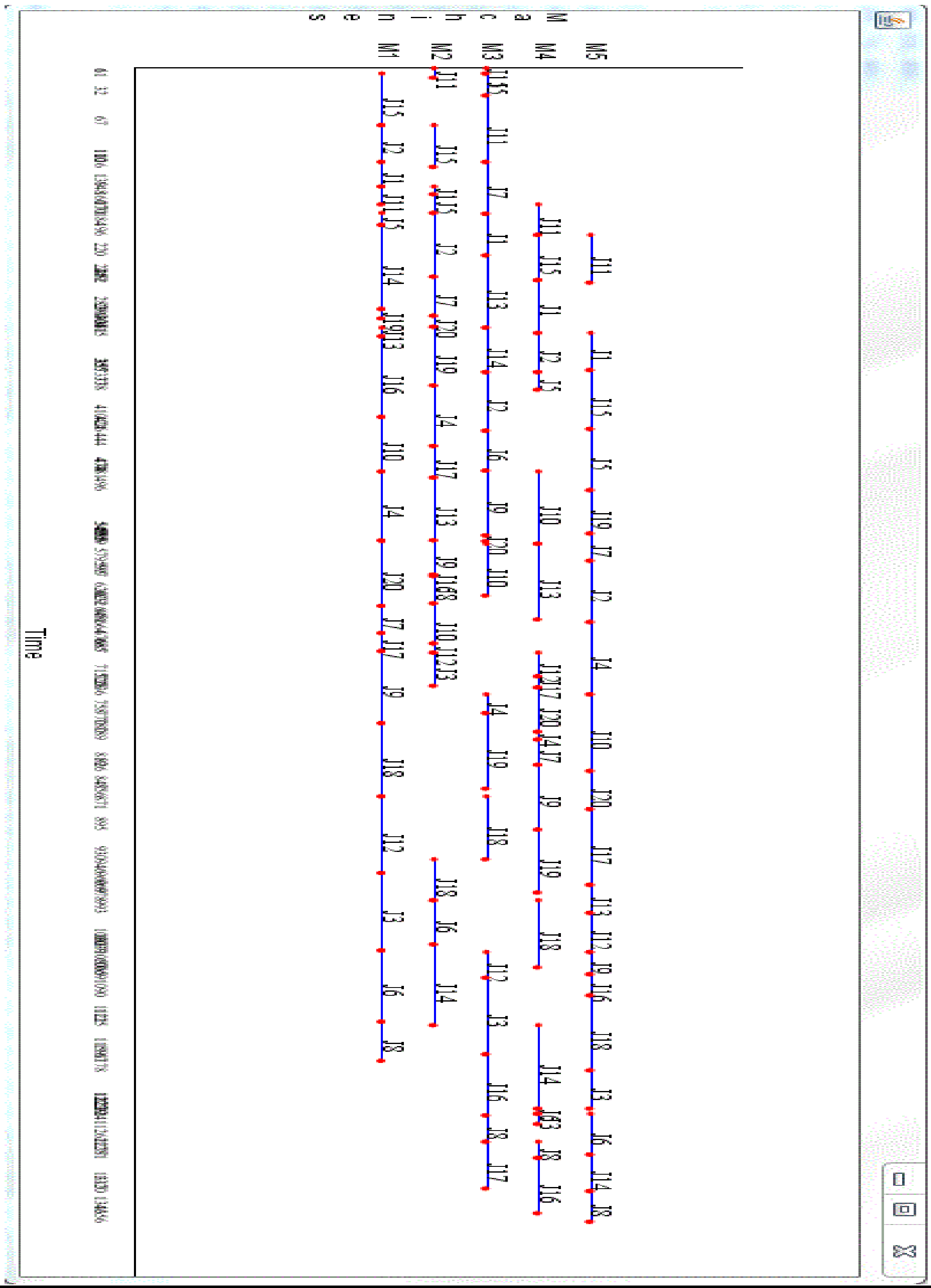


Fig. - Gant chart for 5 machines and 20 jobs (ft20)

Comparison of ABC and GA for solving the JSSP :

Serial	Instance	Size	BKS	ABC			GA		
				Best	Avg.	RPE	Best	Avg.	RPE
1	ft06	6x6	54	54	55.43	0.00	63	72.28	16.66
2	ft10	10x10	930	1118	1216.73	20.21	1343	1504.54	44.40
3	ft20	20x5	1165	1341	1411.27	16.82	1628	1773.92	39.74
4	la01	10x5	666	667	688.9	0.15	805	912.51	20.87
5	la06	15x5	926	926	972.3	0.00	1109	1234.28	19.76
6	la11	20x5	1222	1328	1395.93	8.67	1516	1689.86	24.05
7	la16	10x10	945	1067	1107.43	12.91	1277	1402.15	35.13
8	la21	15x10	1046	1371	1439	31.07	1583	1791.77	51.33
9	la26	20x10	1218	1513	1703.2	24.22	1930	2143.27	58.45
10	la31	30x10	1784	2322	2406.47	30.15	2644	2885.38	48.30
11	la36	15x15	1268	1716	1791.6	35.33	1924	2161.0	51.73
12	abz05	10x10	1234	1464	1517.5	18.13	1744	1930.59	41.32
13	abz07	20x15	656	1006	1052.3	53.35	1095	1257.81	66.92
14	orb01	10x10	1059	1299	1356.03	22.66	1501	1696.10	41.73

- In the table, "Instance" means the problem name, "size" means the problem size of n jobs on m machines, "BKS" means the best known solution for the instance, "Best" and "Avg." means the best and average solution found by each algorithm respectively over 30 runs, and "RPE" means relative percent error with respect to the best known solution which is calculated from the equation given below

$$\text{RPE} = \frac{(\text{Best}-\text{BKS})}{\text{BKS}} \text{ x } 100.$$

- The ABC algorithm is compared with other approach GA with same parameters i.e. the number of populations and number of iterations.
- The performance of our ABC algorithm is evaluated by testing them on the above 14 benchmark problems taken from the Operations Research Library (OR - Library).
    - 3 problems from Fisher and Thompson referred as ft06, ft10, ft20.
    - 8 problems from Lawrence referred as la01, la06, la11, la16, la21, la26, la31, la36.
    - 2 problems from Adams et al. referred as abz01 and abz07.
    - 1 problem from Applegate and cook referred as orb01.
    - 1 problem from Yamada and Nakano referred as yn01.
- The size of these problems instances range from 6 to 20 jobs and 5 to 10 machines.

- The objective is to find the minimum makespan value from these benchmark problems.
- Total population size is set to 50. Both the number of employed bees and number of onlooker bee are set equal i.e. 25.
- The maximum cycle number (MCN) is set to 2000.
- Each of the experiment is repeated 30 times with different random seeds

Conclusions

- The result obtained from our proposed approach show that ABC can find the best known solution more effectively than GA.
- Moreover, ABC algorithm gives better average makspan value.
- Hence, the algorithm can serve as an alternative method in the job shop scheduling domain.

Future Work

In the future work, JSSP can be considered in a case where “Job interrupt” is permitted. In that case, ABC algorithm may split a job into a number of smaller sub-jobs. This will allow each job to be preempted by other jobs with higher priorities. The preempted job can later be resumed when a machine is free. After applying this improvement, ABC will be more practical in most of the Job Shop Scheduling Problems.

TIMELINE

Pre-Mid Semester work **Completed.**

- We have studied Artificial Bee Colony (ABC) algorithm.
- We have studied about JSSP.
- We have studied Genetic Algorithm (GA).
- We have also studied the complexities associated with the JSSP.

Post Mid Semester work **Completed.**

- We have implemented ABC and GA to JSSP.
- Comparison of results obtained from ABC and GA.

REFERENCES

1. B.Basturk, Dervis Karaboga, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,” Journal of global optimization 39, Springer: 459-471, DOI 10.1007/s10898-007-9149-x.
2. Cheng Wu, Rui Zhang, “An Artificial Bee Colony Algorithm fr the Job Shop Scheduling Problem with Random Processing Times,” Entropy 2011, 13, 1708-1729; doi: 10.3390/e 13091708.
3. Manish Gupta, Govind Sharma, “An Efficient Modified Artificial Bee Colony Algorithm for Job Shop Scheduling Problem,” International Journal of Soft Computing and Engineering(IJSCE), ISSN: 2231-2307, Volume-1, Issue-6, January 2012.
4. Ling Wang, Gang Zhou, Ye Xu, Shengyo Wang, Min Liu, “An Effective Artificial Bee Colony Algorithm for the Flexible Job-Shop Scheduling Problem,” International Journal of Advanced Manufacturing Technology (2012) 60:303-315; doi 10.1007/s00170-011-3610-1.
5. Takeshi Yamada, Ryohei Nakano, “Genetic algorithms in engineering systems,” Job-shop scheduling, chapter 7, pp. 134-160, IEE control engineering series 55, 1997.
6. D. Karaboga, “An Idea Based On Honey Bee Swarm for Numerical Optimization,” Technical Report-TR06,Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
7. Dervis Karaboga, Bahriye Akay, “A comperative study of Artificial Bee colony algorithm,” Applied Mathematics and Computation 214, Elsevier, 2009, 108-132.
8. Eugeniusz Nowicki, Czeslaw Smutnicki, “An Advanced Tabu Search Algorithm For The Job Shop Problem,” Journal of Scheduling 8, Springer: 145-159, 2005.
9. Ali Hadidi, Sina Kazemzadeh Azad, Saeid Kazemzadeh Azad, “Structural optimization using artificial bee colony algorithm, 2nd International Conference on Engineering Optimization,” 2010, September 6 – 9, Lisbon, Portugal.
10. Y. Zhang and L. Wu, “Optimal multi-level Thresholding based on Maximum Tsallis Entropy via an Artificial Bee Colony Approach,” Entropy, vol. 13, no. 4, (2011), pp. 841-859.
11. Anan Banharnsakun, Booncharoen Sirinaovakul, Tiranee Achalakul, “Job Shop Scheduling with the Best-so-far ABC,” Engineering Application of Artificial Intelligence 25, Elsevier, 2012, 583-593.

Instruction for running the code

There are 5 files:

1. ABC.java
2. JSSP.java
3. GA.java
4. GanttChart.java
5. JSSPwABC.java

ABC.java:

We set all initial parameter such as population size, etc. for ABC.
Parameters:

SN – No. of food sources,
noj – No. of jobs,
nom – No. of machines

After this we apply algorithm for solving the problem.
Operations are:

Employed bees phase
Onlooker bees phase
Scout bees phase

JSSP.java

We feed input for solving the JSSP problem like job's machine sequence matrix and job's processing time on machines matrix.

jmS[][] – job's machine sequence matrix
jpT[][] – job's processing time on machine matrix

After this we calculate the makespan time using gantt chart method for the operations sequence list.

GA.java

Apply GA algorithm for same problem.
Operations are:

Mutation
Crossover
Neighbor Search

GanttChart.java

Here we make the gantt chart for solution.

JSSPwABC.java

This is main file and you have to only run this file to run the project. All above files are integrated in this file.