International Journal of Hybrid Information Technology
Vol.8, No.11 (2015), pp.273-284
http://dx.doi.org/10.14257/ijhit.2015.8.11.23

An Efficient Approach to Job Shop Scheduling Problem using Simulated Annealing

Shouvik Chakraborty[1*] and Sandeep Bhowmik[2]

[1]*P.G. Student, Department of Computer Science & Engineering, University of Kalyani, West Bengal, India*
[2]*Assistant Professor, Department of Computer Science & Engineering, Hooghly Engineering & Technology College, West Bengal, India*
[1]*shouvikchakraborty51@gmail.com*, [2]*bhowmik.sandeep@gmail.com*

Abstract

The Job-Shop Scheduling Problem (JSSP) is a well-known and one of the challenging combinatorial optimization problems and falls in the NP-complete problem class. This paper presents an algorithm based on integrating Genetic Algorithms and Simulated Annealing methods to solve the Job Shop Scheduling problem. The procedure is an approximation algorithm for the optimization problem i.e. obtaining the minimum makespan in a job shop. The proposed algorithm is based on Genetic algorithm and simulated annealing. SA is an iterative well known improvement to combinatorial optimization problems. The procedure considers the acceptance of cost-increasing solutions with a nonzero probability to overcome the local minima. The problem studied in this research paper moves around the allocation of different operation to the machine and sequencing of those operations under some specific sequence constraint.

Keywords: *Job-Shop Scheduling, Makespan, Simulated Annealing, Genetic Algorithm*

1. Introduction

In job-shop scheduling problem (JSSP) environment, there are n jobs to be processed using m machines with a certain objective function (*i.e.*, makespan) to be minimized. The problem has been explored by many scientists and different algorithms have been proposed by them. Some very special cases of the JSSP can be solved in polynomial time, but their immediate generalizations are considered as NP-hard [1]. In different situations, the combination of goals and available resources exponentially increases the total search space. Hence, the production of consistently good scheduling is practically difficult, because we have to deal with a huge combinatorial search space and some constraints between operations. Approaches to solve JSSP can be divided in two categories: 1) Exact Method, like branch and bound, linear programming etc. It works well and guarantees convergence for small problems. But to solve large problems we need some Approximation algorithms which include priority search, genetic algorithm *etc*. In this paper Simulated Annealing (SA) (Kirkpatrick, Gelatt and Vecchi 1983, and Cerny 1985) has been applied with some modifications to deal with problem of job shop scheduling. The main problem is how to cope with local minima within a reasonable time. The problem is how to find the most effective types and order of operators to evolve the solutions space, where one operator may be successful for one type of problem with a certain order and may not be so preferable for some other problems. A schedule is the assignment of the operations on the machines. The main goal is to obtain a feasible schedule of smallest length. The static job shop-scheduling problem makes assumption that all jobs are available at the very beginning and the processing and set up times are

* Corresponding Author

ISSN: 1738-9968 IJHIT

deterministic in nature. Dynamic scheduling allows new job arrival over time [2]. This static JSSP can be solved using priority rules that used to select a job to be assigned on a machine from queue. There are many rules for big problems, *e.g.*, SPT (Shortest Processing Time), RPT (Remaining Processing Time), DS (Dynamic Slack), TPT (Total Processing Time) *etc*. But not a single dispatching rule ensures good result in various job shop instances. Because of this difficulty, heuristic procedures are an very good alternative. The JSSP is well known as one of the most difficult NP-complete problems [3] which are also popular for its practical applications in manufacturing industry. A good number of algorithms have been proposed for solving JSSPs [4]. However, no single algorithm is capable of solving JSSPs of all kind optimally (or near optimally) within a moderate time span. Thus, there is a provision to analyze the difficulties of JSSPs and to design algorithms that may be capable to solve most of the standard problems. The paper is organized as follows. After this introduction, a brief outline of the standard job-shop scheduling problem and formulation of solution as chromosome are described in Section 2. Simulated Annealing approach is described in Section 3. The SA approach for formulation of the solution chromosome to solve JSSPs is provided in Section 4. Section 5 shows the experimental results and necessary statistical analysis to measure the performance. Finally, conclusions and future research direction are presented.

2. Problem Description

The problem studied in the paper is a deterministic and static n-job, m-machine JSSP. In this problem, n jobs are to be processed by m machines. Each job consists of a predetermined sequence of task processes, each of which needs to be processed without preemption for a given period of time on a given machine. Explaining the problem more specifically, let J={1, 2,..., n} denote the set of jobs, M={1,2,…,m} represent the set of machines and n×m operations to be scheduled, the operations are interrelated by the precedence constraints, which force each operation j to be scheduled after all predecessor operations are completed. Moreover, operation j can only be scheduled if the required machine is idle. The standard JSSP makes the following assumptions:

- The processing time for each job in a particular machine is defined earlier.
- There must be a pre-defined sequence of machine requirements for each job.
- A machine can handle only one particular job at a time.
- Each job visits each machine only once.
- Once a machine starts to process a job, no interruption is allowed.
- Each and every machine has full efficiency.

Job-shop scheduling problems can be classified as follows [13]

1. N jobs, 1 machine
2. N jobs, 2 machines (flow shop)
3. N jobs, 2 machines (any order)
4. N jobs, 3 machines (flow shop)
5. N jobs, M machines

An operation sequence on a machine called job sequence is unknown. The full set of job sequences is called a symbolic representation and a feasible symbolic representation is called a schedule. The main goal of the problem is to minimize the maximum time taken to complete every job while satisfying the machining constraints. That is to find the schedule that minimizes the total elapsed time. The objective function of the above model can be represented as min ($F_{n\times m}$) where, the aim is to minimizes the finish time of the last operation, namely, the makespan [5]. The fundamental concept of permutations is widely used in the representational issue of combinational optimization problems [6]. A solution of assignment

problems can be represented by permutation schemas. In Table 1, the machine sequence for one small JSSP is given.

Table 1. Machine Sequence for 4X3 Problem

Job	Machine Sequence		
J1	M3	M1	M2
J2	M2	M3	M1
J3	M1	M2	M3
J4	M1	M2	M3

As stated in the Table above, job j3 requires its first, second and third operation to be processed on machine m1, m2 and m3 respectively, and job j1 requires machine m3, m1 and m2 respectively. An individual solution (chromosome) indicates the priority sequence of jobs. As the priority sequence of jobs change the makespan also varies. So the problem is to find the optimal chromosome having the job sequence that will minimize makespan. Here a small example has been shown with only three jobs to be scheduled in four machines. This algorithm is not computationally cheaper for large scale problems because of its complexity. Figure 1 shows the Gantt chart to calculate the makespan for the problem considered in Table 1, where the job priority sequence has been considered as {j2, j1, j3, j4}. The representation indicates that job j2 has higher priority than j1 and so on.

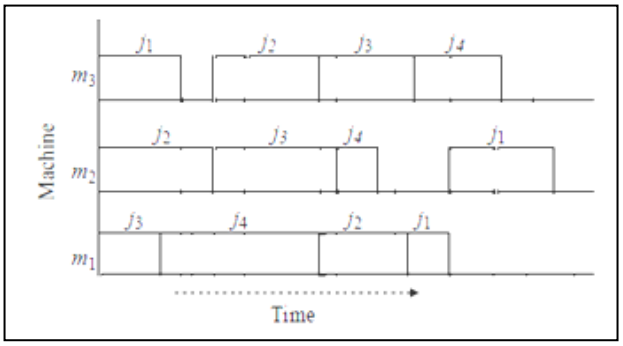

Figure 1. Gantt Chart while Building the Schedule for a 4×3 job-shop Scheduling Problem. The 'X' Axis Represents Execution time and 'Y' axis Represents Machine. The Priority Sequence Considered here is {j2, j1, j3, j4}

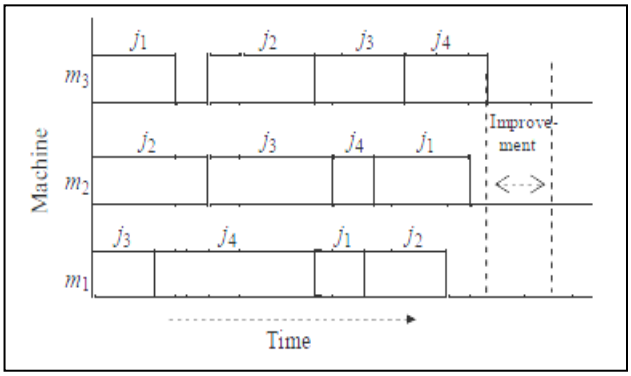

Figure 2. Gantt Chart while Building the Schedule for a 4×3 job-shop Scheduling Problem. The 'X' Axis Represents Execution time and 'Y' axis Represents Machine. The Priority Sequence Considered here is { j1, j2, j3, j4 }

A simple example of the re-ordering process is shown in Figure 2, where the symbolic representation has been considered as {j1, j2, j3, j4}. Since j1 has higher priority than j2, the job allocation order in m1 changes where j1 is scheduled before j2 and thus the makaspan improves [13]. So the applied chromosome (the symbolic representation) is able to generate better makespan. The improvement of the makespan is marked by the dotted line. Researchers have proposed different rules of priority for obtaining improved solutions for job shop scheduling problems.

3. Simulated Annealing

An SA algorithm is an AI technique depending on the behavior of cooling metal in metallurgy. It can be used to obtain solutions to difficult or sometimes impossible combinatorial optimization problems. From its introduction, independently by Kirkpatrick, Gelatt and Vecchi (1983) and Cerny (1985), SA has been applied to solve a good number of combinatorial optimization. For a literature study, the reader is referred to Van Laarhoven and arts(1987). SA is a popular random search technique and takes its analogy from physical annealing process of solids [7]. Annealing is a process in metallurgy where object (*i.e.*, metal) is heated up very strongly. After that a slow cooling process is performed to get a very pure structure of crystal with a lowest possible energy so that the number of fractures and irregularities reduces. First the high temperature accelerates the irregular movement of the particles. They can find an optimal position within the crystal structure. Simulated annealing (also known as monte carlo annealing or probabilistic hill-climber) is a relatively simple and straight forward algorithm which involves metropolis Monte Carlo method.

The metropolis Monte Carlo algorithm is applicable in simulated annealing because only energetically feasible states will be considered at a particular temperature. The SA process is therefore a metropolis Monte Carlo simulation. The temperature is slowly reduced to reduce the search space, and when the temperature is low enough the system will reach favorable state. Simulated Annealing can also be used to find the optimum solution of the problems by efficiently determining the initial (high) and final (low) temperatures which are used in place of kT (where k is a Boltzmann's constant) in the acceptance checking. The Simulated Annealing procedure can be describes as follows.

```
Start with the system at known energy E.
T = temperature = hot (high);
frozen = false;
While (! frozen) {
        repeat {
                Perturb the system slightly by moving particles
                Compute E, change in energy due to perturbation
                If (ΔE < 0)
                        Then accept this perturbation, this is the new system
                configuration.
                Else
                        accept maybe, with probability = exp (-ΔE/kT)
        } until (the system is in thermal equilibrium at this T)
        If (ΔE still decreasing over the last few temperatures)
                Then T=0.9T //Reduce the temperature and do more perturbations
        Else frozen=true
}
return (final configuration as low-energy solution)
```

Simulated annealing process treats the structure of the substance as the codified solution, and perturbation and acceptance is based on the temperature [8]. The algorithm consists of three steps: 1) perturb the solution, 2) evaluate the quality of the solution, 3)

and accept the solution if it is better than the new one. The method of simulated annealing can be easily understood by observing Figure 3, which shows the internal uneven surface (with peaks and valleys).

Several works recently appeared in the literature show the interest of using SA, specially embedding it into population based approaches as ACO, PSO and GA. Examples of PSO hybridized by incorporating SA intensification can be found in [9], where the proposed hybrid PSO, which includes a probabilistically applied local search and a learning-guided multi-neighborhood SA, is applied to makespan minimization in a permutation flow shop scheduling problem with the limited buffers between consecutive machines.

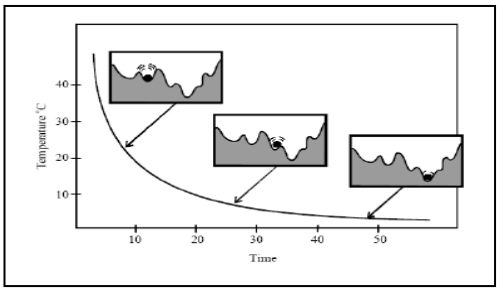

Figure 3. The Method of Simulated Annealing

4. Problem Description using SA

In solving JSSPs using SAs, the chromosome of each individual usually builds the schedule. Chromosomes can be represented by integer, binary or real numbers. Some well known representations for solving JSSP are: operation based, preference-list based, priority-rule based, and job pair-relation based representations [10]. Here the job priority relation sequence based representation as been selected due to the flexibility of applying SA operators. In this representation, a chromosome is represented by an integer string, where each integer stands for a job identity number.

The objective or fitness function takes the input as the number of jobs, and machine sequence of the corresponding job. Each chromosome genes are assigned by an integer number, and then operations are performed to get the corresponding job sequence in a chromosome. The fitness function produces the output as a makespan value for the corresponding operation sequence. The use of Simulated Annealing for generation of an optimal chromosome in job shop scheduling has been discussed in the following section. To accomplish the task, initially a random sequence of jobs is generated and then the processing follow the Simulated Annealing approach as described below. Here the temperature (T) tuning parameter (r) is initialized with the value 0.99.

Let, r = 0.99
Step 1: Get an initial solution of N chromosome.
Step 2: Set an initial temperature T >0.
Step 3: While not frozen do the following.
Step 3.1: Select N/2 from N chromosomes.
Step 3.2: Perturb each chromosome slightly (by moving a particle).
Step 3.3: Find the makespan values for each newly generated chromosome (i.e. S').
Step 3.3.1: Let Δ = cost(S') – cost(S).
Step 3.3.2: If ($\Delta < 0$) (downhill move), set S = S'.
Step 3.3.3: If ($\Delta > 0$) (uphill move), set S = S' with Probability = exp (-Δ/T).
Step 3.4: Choose the N best chromosomes which have the minimum makespan values from the newly generated and also from old chromosomes.
Step 3.5: Set T = r * T (Reduce temperature).
System architecture of our method is shown in Figure 4.

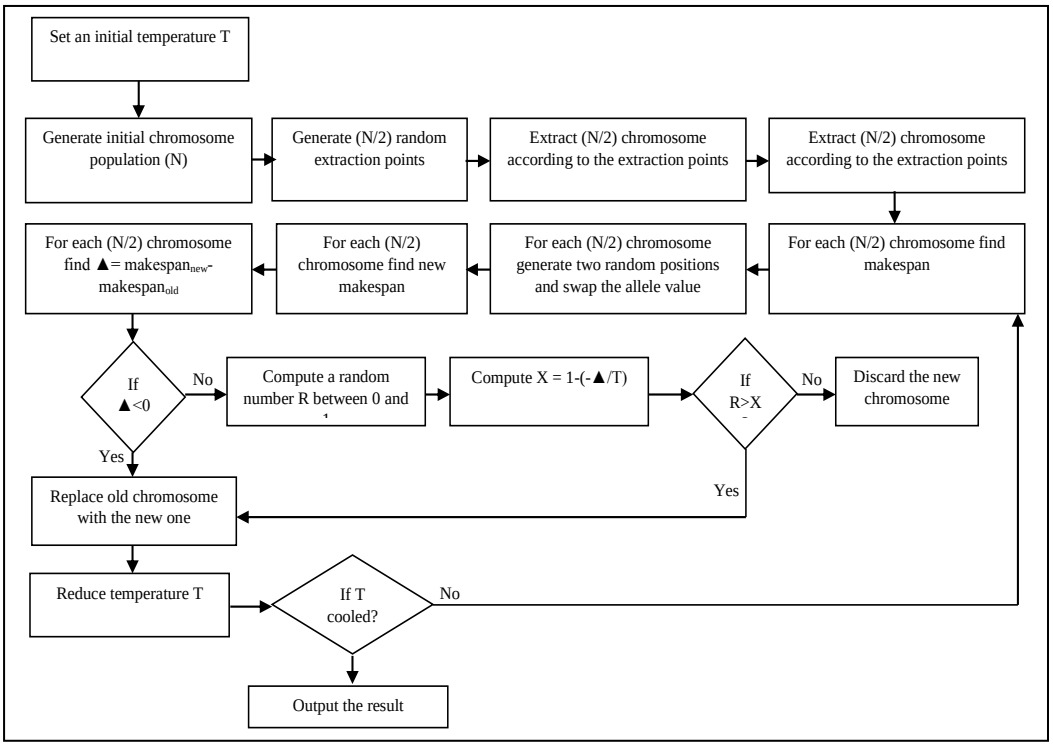

Figure 4. System Architecture

To illustrate the effectiveness and test the performance of the described algorithm, it has been used to solve some the well-known benchmark problems designed by Lawrence [11].

5. Results and Discussions

In the initial temperature T in the simulated annealing process is generally a high temperature [12] and was set it as 200. The procedure takes the input as the number of jobs, operation time of those jobs and machine Sequence of the corresponding job. The procedure has been tested on many test cases. The result of three test cases with different size is given in this paper. Table 2 shows the LA05 problem with 10 jobs to be scheduled in 5 machines. Table 3 shows the LA10 problem with 15 jobs to be scheduled in 5 machines. Table 4 shows the LA12 problem with 20 jobs to be scheduled in 5 machines. The process of generation of the sequence string will take these matrix as input and will provide an optimal sequence of jobs as output.

Table 2. The LA05 Problem with 10 Jobs and 5 Machines

Job	Machine (Makespan)				
0	1(72)	0(87)	4(95)	2(66)	3(60)
1	4(5)	3(35)	0(48)	2(39)	1(54)
2	1(46)	3(20)	2(21)	0(97)	4(55)
3	0(59)	3(19)	4(46)	1(34)	2(37)
4	4(23)	2(73)	3(25)	1(24)	0(28)
5	3(28)	0(45)	4(5)	1(78)	2(83)
6	0(53)	3(71)	1(37)	4(29)	2(12)
7	4(12)	2(87)	3(33)	1(55)	0(38)
8	2(49)	3(83)	1(40)	0(48)	4(7)
9	2(65)	3(17)	0(90)	4(27)	1(23)

Table 3. The LA10 Problem with 15 Jobs and 5 Machines

Job	Machine (Makespan)				
0	1(58)	2(44)	3(5)	0(9)	4(58)
1	1(89)	0(97)	4(96)	3(77)	2(84)
2	0(77)	1(87)	2(81)	4(39)	3(85)
3	3(57)	1(21)	2(31)	0(15)	4(73)
4	2(48)	0(40)	1(49)	3(70)	4(71)
5	3(34)	4(82)	2(80)	0(10)	1(22)
6	1(91)	4(75)	0(55)	2(17)	3(7)
7	2(62)	3(47)	1(72)	4(35)	0(11)
8	0(64)	3(75)	4(50)	1(90)	2(94)
9	2(67)	4(20)	3(15)	0(12)	1(71)
10	0(52)	4(93)	3(68)	2(29)	1(57)
11	2(70)	0(58)	1(93)	4(7)	3(77)
12	3(27)	2(82)	1(63)	4(6)	0(95)
13	1(87)	2(56)	4(36)	0(26)	3(48)
14	3(76)	2(36)	0(36)	4(15)	1(8)

Table 4. The LA12 Problem with 20 Jobs and 5 Machines

Job	Machines (Makespan)				
0	1(23)	0(82)	4(84)	2(45)	3(38)
1	3(50)	4(41)	1(29)	0(18)	2(21)
2	4(16)	3(54)	1(52)	2(38)	0(52)
3	1(62)	3(57)	4(37)	2(74)	0(54)
4	3(68)	1(61)	2(30)	0(81)	4(57)
5	1(89)	2(89)	3(11)	0(79)	4(81)
6	1(66)	0(91)	3(33)	4(20)	2(20)
7	3(8)	4(24)	2(55)	0(32)	1(84)
8	0(7)	2(64)	1(39)	4(56)	3(54)
9	0(19)	4(40)	3(7)	2(8)	1(83)
10	0(63)	2(64)	3(91)	4(40)	1(6)
11	1(42)	3(61)	4(15)	2(98)	0(74)
12	1(80)	0(26)	3(75)	4(6)	2(87)
13	2(39)	4(22)	0(75)	3(24)	1(44)
14	1(15)	3(79)	4(8)	0(12)	2(20)
15	3(26)	2(43)	0(80)	4(22)	1(61)
16	2(62)	1(36)	0(63)	3(96)	4(40)
17	1(33)	3(18)	0(22)	4(5)	2(10)
18	2(64)	4(64)	0(89)	1(96)	3(95)
19	2(18)	4(23)	3(15)	1(38)	0(8)

The program had been executed to produce 3000 generations of the solution. Initial population contained randomly generated sequence strings. Trial runs were made with single solutions to observe the pattern of development. The result for different job sequences generated in those trials is shown in Table 5 (for LA05), Table 6 (for LA10) and Table 7 (for LA12). The optimum makespan achieved in each of the trials varied as an expected characteristic of the evolutionary process.

Table 5. Job Sequences and Achieved Makespan for LA05

Sequence Number	Chromosome (Sequence Strings)	Makespan
1	6 4 7 1 8 9 0 3 5 2	648
2	2 6 5 3 1 7 9 0 4 8	644
3	0 8 3 7 2 4 6 5 1 9	643
4	6 7 9 1 5 8 4 2 3 0	641
5	5 7 2 1 6 3 0 8 9 4	640
6	0 5 8 6 7 9 1 4 2 3	635
7	4 3 9 1 7 6 8 2 5 0	631
8	5 9 1 6 3 7 8 0 4 2	625
9	6 3 8 7 9 2 4 0 5 1	619
10	8 3 0 2 6 9 7 1 5 4	615
11	9 8 1 7 4 3 5 6 2 0	613
12	5 6 4 9 8 1 7 2 0 3	610
13	0 1 6 5 8 3 2 9 7 4	609
14	0 2 9 1 3 5 6 7 8 4	607
15	5 4 6 8 2 9 3 1 0 7	605
16	9 7 5 6 3 8 1 0 2 4	603
17	4 1 6 3 2 5 9 7 0 8	600
18	1 3 6 5 0 7 9 2 8 4	593
19	9 8 6 3 2 0 5 1 4 7	593

Table 6. Job Sequences and Achieved Makespan for LA10

Sequence Number	Chromosome (Sequence Strings)	Makespan
1	12 11 0 7 10 2 5 1 6 14 8 13 3 4 9	1028
2	12 10 8 4 0 7 3 9 2 1 11 6 14 5 13	1025
3	5 1 11 12 10 2 4 14 3 13 6 8 9 7 0	1023
4	9 10 4 13 12 2 6 3 8 7 11 1 0 5 14	1022
5	3 12 8 1 10 5 6 11 14 0 7 13 2 4 9	1020
6	3 2 9 1 4 6 11 13 8 5 7 10 12 14 0	1017
7	2 10 0 5 12 4 13 8 3 6 1 11 7 14 9	1016
8	3 1 12 8 4 7 13 0 5 2 9 6 10 11 14	1012
9	1 14 4 7 13 8 6 0 2 5 9 10 3 11 12	1009
10	14 8 5 6 13 1 12 3 11 2 0 9 4 7 10	1008
11	0 4 8 7 5 3 12 1 2 11 10 14 6 13 9	1007
12	3 12 10 5 0 14 4 7 9 8 6 2 1 11 13	1006
13	10 6 8 12 4 1 11 13 14 3 9 5 2 0 7	1005
14	14 1 6 8 12 5 10 4 3 2 0 9 13 11 7	1004
15	11 12 10 13 5 14 1 2 4 8 0 7 9 3 6	1003
16	5 12 1 14 8 9 4 11 0 13 3 2 10 7 6	1001
17	4 2 1 6 9 8 10 5 13 7 3 12 14 0 11	993
18	8 2 3 5 6 0 7 9 10 1 13 4 14 12 11	990
19	1 13 14 0 2 10 7 8 3 5 4 11 6 9 12	988
20	5 13 10 3 0 4 2 6 1 7 9 14 8 11 12	983
21	14 13 4 8 3 1 5 11 12 6 10 9 2 0 7	981
22	10 1 7 6 0 3 4 11 2 12 5 8 14 13 9	973
23	3 1 12 6 13 11 4 5 14 8 7 10 2 0 9	966
24	1 4 3 2 11 9 12 7 10 6 13 0 8 5 14	965
25	1 8 14 2 5 13 6 12 4 10 0 9 3 11 7	962
26	5 13 2 0 12 4 11 1 6 9 3 10 7 8 14	958
27	3 2 9 1 4 6 11 13 8 5 0 10 12 14 7	958

Table 7. Job Sequences and Achieved Makespan for LA12

Sequence Number	Chromosome (Sequence Strings)	Makespan
1	9 4 17 5 0 15 12 6 1 3 14 11 8 19 18 10 2 13 16 7	1136
2	3 2 5 8 19 13 17 4 14 11 18 15 10 9 6 0 12 16 1 7	1134
3	7 19 8 10 14 17 4 15 16 2 6 12 5 18 0 3 1 11 13 9	1132
4	7 19 9 17 16 10 4 15 18 13 11 0 8 1 5 6 12 14 3 2	1131
5	18 1 9 7 17 8 15 4 0 3 16 13 5 10 12 19 14 11 2 6	1129
6	18 14 8 10 13 6 19 16 0 9 1 15 11 2 12 3 7 5 4 17	1126
7	5 13 2 17 15 14 18 0 1 19 9 8 3 16 10 4 12 11 6 7	1124
8	0 15 2 13 12 1 5 9 14 4 10 3 7 16 6 18 17 11 8 19	1121
9	19 17 9 5 15 3 2 10 14 7 18 16 8 0 12 11 1 6 4 13	1120
10	12 7 11 6 14 18 0 5 3 4 17 19 15 2 9 10 8 16 1 13	1115
11	10 19 3 18 5 1 0 14 4 15 12 13 2 11 6 9 8 17 16 7	1115
12	19 10 17 15 5 9 8 18 7 1 11 16 6 0 12 3 2 14 13 4	1112
13	19 5 3 4 10 16 11 6 12 17 0 18 1 13 15 9 14 7 8 2	1106
14	12 13 7 17 16 4 3 5 8 6 18 0 19 1 14 11 9 10 2 15	1101
15	2 10 5 16 7 4 18 6 15 0 14 12 8 17 13 1 19 3 11 9	1097
16	12 15 1 6 0 11 14 9 3 7 8 18 16 17 4 13 5 10 2 19	1095
17	0 3 6 18 5 12 11 7 4 9 19 14 10 8 15 2 16 13 1 17	1092
18	12 8 6 11 16 13 0 9 17 2 14 18 19 10 3 1 5 4 7 15	1087
19	16 9 11 6 0 3 13 19 14 5 15 18 10 7 17 8 4 12 1 2	1082
20	12 14 3 0 4 19 16 15 2 9 10 6 8 7 5 18 13 1 11 17	1081
21	12 19 13 15 11 10 0 5 16 4 9 8 3 14 18 7 2 6 17 1	1072
22	6 14 7 0 10 4 11 18 17 5 3 9 1 16 12 13 19 8 2 15	1065
23	11 1 16 6 7 9 10 4 3 12 0 19 14 8 18 17 5 2 15 13	1064
24	0 2 6 1 8 13 5 11 15 18 14 19 10 16 12 4 9 17 7 3	1060
25	3 9 12 6 0 16 1 19 4 5 15 18 10 7 17 8 11 14 13 2	1054
26	15 8 6 11 16 13 0 9 17 12 14 18 19 10 3 1 5 4 7 2	1051
27	5 12 9 17 3 1 19 18 7 4 11 10 13 16 15 8 0 14 6 2	1050
28	6 1 13 0 19 16 2 12 18 5 8 10 15 11 7 17 14 3 9 4	1049
29	6 4 1 17 0 10 14 9 3 7 8 18 16 12 11 13 5 15 2 19	1048
30	0 3 6 18 15 12 11 7 13 9 19 17 10 8 5 2 16 4 14 1	1047
31	19 5 3 4 9 16 6 7 12 10 0 18 1 13 15 17 14 11 8 2	1046
32	10 9 1 6 8 12 15 16 0 14 18 4 3 13 11 19 17 5 7 2	1045
33	19 9 12 6 8 10 13 14 4 11 17 5 16 0 1 2 3 18 15 7	1039
34	4 8 5 3 12 18 11 7 16 0 6 14 17 9 1 13 2 19 10 15	1039

Figure 5 shows the graphical result of the experiment.

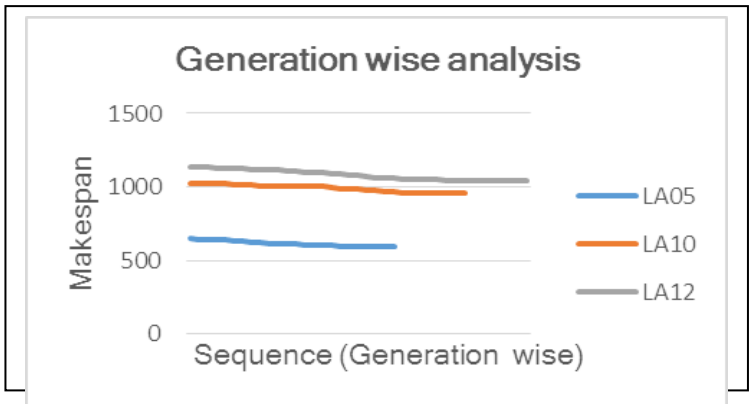

Figure 5. Graphical Result of the Experiment

In the Figure 5 x-axis represents the Sequences (according to the No. of generations) and y axis represents the makespan value. The Figure plots the representative curve finding the best solution. It can be observed that after a certain time the makespan value becomes fixed at a particular value which is the optimum one.

6. Conclusion

The JSSP is a very well known member of the combinatorial optimization problem class. A considerable amount of work has already been done to improve algorithms to give a steady and optimal output. Some algorithms are eminent for special case problems. But still no algorithm guarantees optimality. The problem studied in this research paper revolves around the assignment of jobs to the machine and sequencing of jobs on machine under pre-specified sequence constraint. This research focuses on the permutation representation of the combinational problem. In the proposed algorithm, the problem representation is simple and easy when compared with other heuristic methods. To solve larger-size problems effectively, it is crucial to incorporate local search methods that use domain specific knowledge.

References

[1] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, "Sequencing and scheduling: Algorithms and complexity", S.C. Graves, A. H. G. Rinnoy Kan and P. Zipkin, editors, Handbooks in Operations Research and Management Science, vol. 4, **(1993)**, North-Holland.
[2] A. M. Iarijani, K. Sabri-laghaie and M. Heydari, "Solving Flexible Job shop Scheduling with Multi Objective Approach", International Journal of Engineering, Science and Technology, vol. 2, no. 1, **(2010)**, pp. 144-151.
[3] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update", European Journal of Operational Research, vol. 174, no. 1, **(2006)**, pp. 23–37.
[4] L. Sun, X. Cheng and Liang, "Solving Job Shop Scheduling Problem Using Genetic Algorithm with Penalty Function", International Journal of Intelligent Information Processing, vol. 1, no. 2, **(2010)**.
[5] J. F. Goncalves, J. J. D. M. Mendes and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem", Europian Jurnal Opertion Research, vol. 167, **(2005)**, pp.77-95.
[6] C. Bierwirth, D. C. Mattfeld and H. Kopfe, "On Permutation Representations for Scheduling Problems; Parallel Problem Solving from Nature IV", Springer, **(1996)**, pp. 310-318.
[7] K. Thanushkodi and K. Deeba, "On performance analysis of hybrid algorithm (improved PSO with simulated annealing) with GA, PSO for multiprocessor job scheduling", WSEAS Transactions on Computers, ACM, vol. 10, Issue 9, **(2011)**, pp. 287-300.
[8] X. Wang and J. Li, "Hybrid particle swarm optimization with simulated annealing, International Conference on Machine Learning and Cybernetics", IEEE, vol. 4, **(2004)**, pp. 2402-2405.
[9] B. Liu, L. Wang and Y.-H. Jin, "An effective hybrid PSO-based algorithm for flow shop scheduling with limited buffers", Computers & Operations Research, ACM, vol. 35, Issue 9, **(2008)**, pp. 2791-2806.
[10] N. Zribi, I. Kacem, A. EL Kamel and P. Borne, "Optimization by Phases for Flexible Job Shop Scheduling Problem", 5th Asian Control Conference, **(2004)**, pp. 1889 -1895.
[11] S. Lawrence, "Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques", Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, **(1984)**.
[12] Y. Peng, D. Zhang and F. Y. L Chin, "A hybrid simulated annealing algorithm for container loading problem", Proceedings of the first ACM/Sigevo Summit On Genetic And Evolutionary Computation, **(2009)**, pp. 919-922.
[13] S. Chakraborty and S. Bhowmik, "Job Shop Scheduling using Simulated Annealing", proceedings of the 1st International Conference IC3A, McGrawHill Publication, JIS College of Engineering, Kalyani, India, ISBN: 978-1-25-906393-0, **(2013)** January, pp. 69-73.

Authors

Shouvik Chakraborty, He is pursuing M.Tech in Computer Science and Engineering from University of Kalyani, West Bengal, India. He received his B.Tech from Hooghly Engineering & Technology College, West Bengal under West Bengal University of Technology, West Bengal, India. His research interests include soft and evolutionary computing, bioinformatics, digital image processing and cloud computing.

Sandeep Bhowmik, He is currently working as Assistant Professor in the Computer Science & Engineering Department at Hooghly Engineering & Technology College, Hooghly, West Bengal, India. He obtained M.Sc. in Computer Technology from University of Burdwan and later received M.Tech. in Software Engineering from West Bengal University of Technology. His area of interest in research is Combinatorial Optimization and Information Processing.

