Chapter VI
6.1 Introduction

Job shop scheduling has been the area of research for the past two decades. Job shop scheduling problems have been solved using various techniques applying different scheduling rules, heuristics, Neural network, Genetic algorithm and methods like Simulated annealing and Tabu search. An attempt has been made to solve the JSSP with three different representations through Simulated Annealing. The random insertion perturbation scheme has been used to generate neighborhoods. A set of benchmark problems has been tested for the makespan objective. The results obtained through GA have also been compared with the results given by simulated annealing (SA) algorithm.

6.2 Job-shop problems

Manufacturing industries should do effective scheduling to produce the products in required quantity with specified quality at predetermined cost and pre-established time. The job shop model is one type of combinatorial optimization problem. The following four modern heuristic techniques are commonly used for solving these types of combinatorial optimization problems.

1. Simulated annealing algorithm
2. Tabu search algorithm
3. Genetic algorithm
4. Neural network

These techniques are capable of providing high-quality solution with reasonable computational effort.
The job shop scheduling problem consists of \( n \) jobs and \( m \) machines. Each job consists of \( m \) tasks, each of fixed duration. Each task must be processed on a single specified machine and each job visits each machine exactly once. There is a predefined ordering of the tasks within a job. A machine can process only one task at a time. There are no set-up times, no release dates and no due dates. The makespan is the time from the beginning of the first task to start to the end of the last task to finish. The aim is to find start times for each task such that the makespan is minimized.

6.3. Simulated annealing algorithms

Kirkpatrick developed the Simulated Annealing (SA) algorithm in 1983. It is a relatively new approach that has been used to solve many combinatorial problems such as Traveling salesman problem, flow shop-scheduling problems, JSSP and design optimization problems.

SA algorithm is based on the idea of annealing (Slow cooling of metals) in physics. This algorithm gets final global optimum solution (Kalyanmoy Deb, 2000) by gradually going from one solution to next solution. Simulated annealing belongs to a class of local search algorithms that are known as threshold algorithms. Local search algorithms start with a complete solution, from which a series of better solutions is generated.

This algorithm starts out at a high temperature with an initial solution, and neighbors of that solution are randomly visited. At each iteration of the algorithm, neighbors, which make the current solution better, are always accepted. Those, which are worse than the current solution, are accepted with a probability of \( \exp (-\Delta E/T) \) where \( \Delta E \) is the difference in energy or objective function value and \( T \) the temperature. In this way, SA searches for solutions in uphill direction also. This is how it attempts to back out of a local optimum and search for better solutions in neighboring regions. In order to simulate the thermal
equilibrium at every temperature, a number of points are usually tested at a particular temperature, before reducing the temperature. The algorithm is terminated when a sufficiently small temperature is obtained. This technique can also be used to solve many combinatorial optimization problems for finding global optimum solution as it takes large area for searching.

6.3.1. Steps in simulated annealing

Step1: Choose an initial point or solution \( x_t \) and a termination criterion \( \varepsilon \). The termination criterion may be absence of improvement in solution for certain number of cycles, difference between adjacent solutions etc. Set initial temperature \( T \) at a sufficiently high value. Set the number of iterations to be performed at a given temperature as \( n \) and set \( t=0 \).

Step2: Generate a neighborhood point or solution \( x_{t+1} \) using some disturbance method suitable to the given problem.

Step3: If \( \Delta E = E(x_{t+1}) - E(x_t) < 0 \) accept the new neighborhood point or solution and set iteration number \( t=t+1 \) (move down hill);
Else create a random number 'r' in the range (0 to 1). If \( r \leq \exp(-\Delta E/T) \) accept the new solution or point and set iteration no \( t=t+1 \) (move up hill);
Else go to step 2.

Step4: If the termination criterion is met or temperature \( T \) is small (say 20), then terminate the search;
Else, if \( (t \mod n) = 0 \) then lower \( T \) according to the cooling schedule as \( T=TxC \) where \( C \) is the cooling rate and go to step 2;
Else go to step 2.
The following aspects of the algorithm are important. The term $\exp(-\Delta E/T)$ is called the acceptance function. The acceptance function is such that solutions with slightly higher objective function value are more likely to be accepted than those with Objective function values much greater than the current solution's objective function value. The value of temperature parameter $T$ should be set so that more solutions will be accepted when $T$ is high and fewer when it is low. As the value of $T$ is gradually reduced it is implied that one wants worse solutions accepted in the beginning and fewer at the end. The rationale for this is that one wants to avoid entrapment in a local optimum in the early part of the search. The cooling schedule used in the algorithm is a simple yet powerful way of ensuring that the temperature is gradually reduced after a frozen state is reached at certain state or temperature. Although a cooling factor of 0.9 is generally chosen, any value from 0.8 to 0.99 could be used. Choosing a higher value for the cooling factor “C” will mean that the jump from one state to another is slow and gradual. A value close to 0.8 will mean that the drop in temperature is more significant, sometimes causing simulated annealing to terminate quickly.

6.3.2 Four ingredients of simulated annealing

Four basic ingredients are needed to apply simulated annealing in practice:

- a concise problem representation,
- a neighborhood function,
- a transition mechanism and a cooling schedule.

As for the choice of the cooling schedule, there exist some general guidelines. However, no general rules are known that guide the choice of the other ingredients. The way they are handled is still a matter of experience, taste, and a skill left to the annealing practitioner.
Problem Representation: A non-preemptive single-machine schedule can be specified by a simple permutation of the n jobs. A non-preemptive job shop schedule can be specified with m consecutive strings, each one representing a permutation of n operations on a specific machine (Pinedo et al., 1999).

Neighborhood Function: The choice of neighborhood can greatly influence algorithm performance. While choosing a rich neighborhood containing a large number of candidate solutions will increase the likelihood of finding good solutions, the computation time required to select from the available neighbors will also increase.

Some examples are:

Random-2-shifts is a strategy of generating a neighbor solution by shifting two jobs randomly.

Systematic-2-shift is to evaluate the potential pair-shift in the order

(1, 2), (1, 3)... (1, n), (2, 3)... (2, n), (3, 4) ........and so on.

Transition Mechanism: The search process within a neighborhood can be done in a number of ways. A simple way is to select schedules in the neighborhood at random, evaluate these schedules, and decide which one to accept. Another way is to select first schedules that appear promising, then swapping the jobs that affect the objective most (Pinedo et al., 1999).

Cooling Schedule: The value of T may be reduced as T = T* C where C is a positive constant smaller than but close to 1. Typical values lie between 0.8 and 0.99.

6.3.3 Random insertion perturbation scheme

The perturbation scheme adopted in this study (Parthasarathy and Rajendran, 1998) can be best explained by a simple numerical illustration. Considering the mt06 problem, where the no of jobs and no of machines are 6, an initial sequence is randomly generated as 1, 2, 3, 4, 5, 6. This initial sequence is also known as seed sequence and it is represented by S.
The job in the first position can be inserted at any position to its right. Hence, the job in the first position is inserted at any position between 2 and n (here n=6) in the right side and only one random number between 2 and n is used to select the insertion position. When the selected position is 5, job 1 is inserted in position 5, yielding a new sequence $S_1 = \{2, 3, 4, 5, 1, 6\}$.

Similarly, the job in the last position can be inserted at any position to its left. Hence, the job in the last position is inserted at any position between 1 and n-1 (6-1=5) in the left side and only one random number generation between 1 and n-1 is used to select the job insertion position. When the selected position is 2, job 6 is inserted in position 2, yielding a new sequence $S_2 = \{1, 6, 2, 3, 4, 5\}$.

However, jobs in the middle, i.e. 2,3,4,5, are inserted as follows. Considering the job in the second position of sequence S, two positions can be chosen randomly on each side for its insertion. This job can be inserted in at any position between $(2+1)$ and n (i.e. a position to its right) and any position between 1 and $(2-1)$ (i.e. a position to its left). When position 1 is selected at the left and position 4 is selected at the right of job 2, the new sequences thus generated are $S_3 = \{2,1,3,4,5,6\}$ and $S_4 = \{1,3,4,2,5,6\}$. Similarly, for the jobs in position 3, 4, and 5 one can select two positions randomly, one to the right and one to the left, and obtain the resulting sequences as the ones shown below:

- $S_5 = \{1,3,2,4,5,6\}$, $S_6 = \{1,2,4,5,3,6\}$, $S_7 = \{1,2,4,3,5,6\}$, $S_8 = \{1,2,3,5,6,4\}$,
- $S_9 = \{1,2,3,5,4,6\}$ and $S_{10} = \{1,2,3,4,6,5\}$.

Thus the Random insertion perturbation scheme generates $2 \times (n-1)$ new neighborhood sequences for all sizes of job shop scheduling problems.
6.3.4 Parameters in simulated annealing algorithm

The set of parameters controlling the performance of simulated annealing are discussed in the following sections.

6.3.4.1 Initial sequence (seed sequence)

In simulated annealing the initial or seed, sequence is very important. The initial sequence may be given randomly. The sequence may be obtained through various dispatching rules or from other algorithms such as GA. The sequence can be based on any representation. When the initial sequence is good, the convergence towards global optimum solution tends to be faster or otherwise computational effort for finding final good global optimum solution is more.

6.3.4.2 Initial temperature

At starting of SA, most of the solutions (though “not good”) are accepted. As the algorithm progresses, cooling process takes place and the probability of accepting non-improving solution is decreased. It happens due to the following reasons. For example at starting of SA taking the initial temperature $T$ as 475 and the difference of two successive neighborhoods $\Delta$ as 30, $\exp(-\Delta/T) = \exp(-30/475) = 0.93$. At the end of SA taking the terminating temperature $(T) = 30^\circ C$ and the difference $\Delta$ as 30, $\exp(-30/30) = 0.36$. From this example at starting of SA, with higher temperature, one accepts the inferior solution, as probability is 0.93. However, at the end SA rejects the inferior solution, as probability is 0.36 only. Thus SA accepts more “not good” solution at starting.

The initial temperature has been fixed at 475, as this value of initial temperature will lead to the acceptance of an inferior solution (inferior by 50 % relative to the original solution) with a probability of 0.9, when the SA algorithm starts.(Parthasarathy and Rajendran, 1998)
6.3.4.3 Iterations at each temperature

Iteration at each temperature depends upon whether it reaches “frozen” state or not. Frozen state is a state at which probability of finding new better solution is relatively very small. If the iteration reaches the frozen state, then the temperature is reduced. Otherwise, some counts for iterations (usually n= 20 to 100) may be set for all the temperatures till the end.

6.3.4.4 Cooling factor

The temperature is reduced gradually (not rapid) for reducing the search space by using cooling factor “C”. The value for “C” ranges from 0 to 1(closer to 1). For obtaining good result “C” value is taken from 0.9 to 0.99. If the cooling factor is high (C=0.99), the jump from one state to another state is slow and gradual. If the cooling factor is small (C=0.8 or 0.5), drop in temperature is more significant, some times causing SA to terminate quickly.

6.3.4.5 Termination

If the final temperature of the SA reaches the temperature, say 20 or 30, the simulated annealing process may be terminated.

6.4 Application of simulated annealing algorithm

The three schemes discussed in Chapter 5 namely Job based representation, operation based representation and a proposed representation (Amirthagadeswaran and Arunachalam, 2006) have been adopted in solving a set of benchmark problems.

6.4.1 Objective function

The objective of the problem is to find the job sequence for which the makespan $C_{max}$ is minimum.

Objective function = $f(x)$
6.4.2 Input

The input data are: Job numbers, operation sequences, operation time, starting temperature, number of iterations required at a temperature, cooling factor and final temperature for termination.

6.4.3 Neighborhood generation

Random insertion perturbation scheme has been adopted to promote neighboring solutions.

6.4.4 Finding solutions

Accepting/rejecting the solutions considering the objective function value and the probability for acceptance.

6.4.5 Output

The solution, which gets improved during each iteration/temperature stage, is available as the result.

6.5. Validation

To validate the proposed heuristic, two instances (mto6, mt10) from the suite of Fisher and Thompson (1963), two instances (abz5, abz6) set by Adams et al. (1988), two instances (yn1, yn2) set by Yamada and Nakano (1991) and one instance (la36) set by Lawrence (1984) have been selected. The problems are available in OR - Library (http://ms.ic.ac.uk/info.html).

The following parameter setting has been used for the simulated annealing algorithm.

Initial temperature = 475
No of iterations at a particular temperature = 60/10
Cooling rate = 0.9/0.8
Final temperature = 20
The values for various parameters have been chosen such that the genetic algorithm and simulated annealing algorithm take about the same CPU time for processing. All the three representation schemes have been coded and the algorithm implemented in the C language. The system used is an Intel Pentium IV CPU at 1.8 GHz, 128 MB RAM processor.

6.6. Results and discussions

The variation in the solutions and the best solution for about 200 iterations have been recorded and presented in figure 6.1. The details about the name of the problem, size of the problem, the makespan found through the algorithms for the three methods of representation and deduction of schedule are presented in table 6.1. The results are also presented in the form of charts in figures 6.1, 6.2 and 6.3.

![Figure 6.1 A typical online performance of SA on the abz5 benchmark problem](image)

<table>
<thead>
<tr>
<th>Problem No</th>
<th>Problem Instance</th>
<th>Size</th>
<th>Genetic algorithm</th>
<th>Simulated annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Job based</td>
<td>Operation based</td>
</tr>
<tr>
<td>1</td>
<td>abz5</td>
<td>10 x 10</td>
<td>1424</td>
<td>1258</td>
</tr>
<tr>
<td>2</td>
<td>abz6</td>
<td>10 x 10</td>
<td>1063</td>
<td>966</td>
</tr>
<tr>
<td>3</td>
<td>mt 06</td>
<td>6 x 6</td>
<td>61</td>
<td>55</td>
</tr>
<tr>
<td>4</td>
<td>mt 10</td>
<td>10 x 10</td>
<td>1142</td>
<td>993</td>
</tr>
<tr>
<td>5</td>
<td>yn 1</td>
<td>20 x 20</td>
<td>1180</td>
<td>1015</td>
</tr>
<tr>
<td>6</td>
<td>yn 2</td>
<td>20 x 20</td>
<td>1241</td>
<td>1019</td>
</tr>
<tr>
<td>7</td>
<td>la 36</td>
<td>15 x 15</td>
<td>1621</td>
<td>1366</td>
</tr>
</tbody>
</table>

Table 6.1 Results with GA and SA for a set of problem instances
Proposed scheme

Figure 6.2 Comparison of GA and SA for Job based scheme

Operation based scheme

Figure 6.3 Comparison of GA and SA for Operation based scheme

Proposed scheme

Figure 6.4 Comparison of GA and SA for Proposed scheme
The parameter settings applied for the various representation schemes for GA and simulated annealing are given table 6.2. The parameter settings are applied such that both the algorithms take about the same computational time. However, the parameters may be fine tuned for better results.

**Table 6.2 Parameter settings for the algorithms**

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Genetic algorithm</th>
<th>Simulated annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Popn. size</td>
<td>Prob. of crossover</td>
</tr>
<tr>
<td>Job based scheme</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>Operation based scheme</td>
<td>20</td>
<td>80</td>
</tr>
<tr>
<td>Proposed scheme</td>
<td>20</td>
<td>80</td>
</tr>
</tbody>
</table>

6.7 Conclusion

The JSSP has been approached with simulated annealing. The method has been applied with three representation schemes. The results with the three representation schemes have been presented. The performance of simulated annealing algorithm has been compared with GA. It is concluded that the performance of simulated annealing algorithm is comparable to GA in solving job shop scheduling problems. The algorithms have been coded using C language. With suitable tuning of parameters of the algorithm, the performance of the algorithm may be enhanced.