Scheduling of Flexible Manufacturing System using Tabu Search Technique

by

P.Senthil Velmurugan
profsvm@yahoo.co.in
Department of Mechanical Engineering
Kongu Engineering College
Erode – 638 052, India

V.P.Eswaramurthy
eswar_murthy@yahoo.com
Department of Computer Applications
Kongu Engineering College
Erode – 638 052, India

V.Selladurai
profvsdcit@yahoo.com
Department of Mechanical Engineering
Coimbatore Institute of Technology
Coimbatore-641 014, India

A.Tamilarasi
angamuthu_tamilarasi@yahoo.co.in
Department of Computer Applications
Kongu Engineering College
Erode – 638 052, India

Abstract: *The general Flexible Manufacturing System (FMS) scheduling problem is combinatorial in nature and extremely difficult to find optimal solutions conventionally. In recent years, much attention has been made to general heuristics such as Genetic Algorithm, Ant Colony Optimization, Tabu Search and Simulated Annealing to solve these types of problems. This paper presents a Tabu Search (TS) approach to minimize makespan for the FMS scheduling problem. TS is deterministic oriented Search procedure that constrains searching and seeks to transcend local optimality by storing the Search history in its memory. The proposed method uses dispatching rules to obtain initial solution and searches for new solutions in a neighborhood based on the critical paths of the jobs. Several benchmark problems have been tested using this algorithm for the best makespan. The results indicate that the new approach produces better makespan value compared to earlier results.*

I Introduction to Scheduling

Scheduling is considered to be a major task for shop floor productivity improvement. It is an allocation of resources applying limiting factors of time and cost to perform a collection of tasks. Scheduling problems are very different from others. The broad families of scheduling problems are machine shop scheduling, flow shop scheduling, job shop scheduling and Flexible Manufacturing System (FMS) scheduling. These scheduling problems can be distinguished depending on the degrees of freedom in positioning resource supply and resource demand intervals in time. The utilization of FMS in a job shop environment can basically be characterized by three kinds of flexibility such as process flexibility, routing flexibility and machine flexibility (Hansmann, 1997), which are closely related to one another.

II FMS Scheduling

FMS consists of more than one complementary or supplementary machine, many of which is of the CNC type, served by compatible, common and preferably automatic tool and work piece supply systems under the supervision of an integrated control (Rao et al., 1993). The processing time in FMS is assumed to be nearly deterministic. FMS combines the advantages of a traditional flow lines and job shop systems to meet the changing demands. In FMS, the operation of a particular job is not dependent on other operation of the same job. Hence, the operation sequence of a particular job can be modified according to the availability of a machine and a job.

Due to flexibility in the FMS, a given operation can usually be performed on a number of machines and thus it allows many alternative routings. Among these multiple routings it is important to select the best available machine which can perform the operation. This makes even a two-machine problem non-polynomial (NP)-difficult. The job constitutes a sequence of operations and each operation can have some flexibility depending on the feasibility of an operation to be performed on a machine. The main objective of FMS scheduling problem is to minimize the makespan i.e., total completion time of all the operations. This paper presents a Tabu Search (TS) approach to minimize the makespan C_{max} (throughput time) for the FMS scheduling problem.

II.1 Representation Models

A schedule is defined as a complete and feasible ordering of operations that is processed on each machine and there are two ways of graphically representing such an ordering:

- Gantt chart
- Disjunctive graph

II.1.1 Gantt chart

Gantt chart is a graphical representation of position of job and operations on the respective machines. In Gantt chart, one should try to move various operation blocks to the left as much as possible on each machine. This will help to have a compact schedule which will generally minimize the makespan measure. Figure 1 shows the sample Gantt chart and the hatched area in the Figure indicates the idle time. Due to precedence constraints idle times arise with the machine waiting to process an operation**.**

Table 1. Sample Problem (2 X 3)

Job	Operation (Processing Time)		
	1	2	3

1	1(3)
	2(4)
		2(5)	1(6)

2	2(4)	1(5)	1(6)
			2(7)

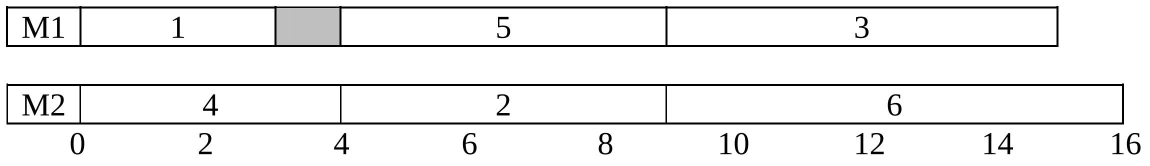

Figure 1. Sample Gantt chart

The assumptions that have been made in the scheduling problem are as follows:

1. Each machine can perform only one operation at a time on any job.
2. A job, once taken up for processing, should be completed before another job can be taken up, i.e. job pre-emption is not allowed.
3. No two successive operations of a job can be performed on the same machine.
4. There are no interruptions in the shop floor, i.e. no machines breakdown
5. The jobs are independent of each other; no assembly is involved.

II.1.2 Disjunctive graph

FMS scheduling problem can be represented by a disjunctive graph (Van Laarhoven et al., 1992). In this disjunctive graph a vertex represents an operation. The conjunctive arcs which are directed lines connect two consecutive operations of the same job. The disjunctive arcs which are pairs of opposite directed lines connect a pair of operations belonging to different jobs to be processed on the same machine. The detailed explanation is given in section IV.

III Literature Review

Kusiak.A (1986) discussed the FMS and flexibilities in the system. In this work, classification of the FMS and flexibilities is presented. The tools and techniques applicable for solving the FMS problems are listed. The design problems and the operational problems in the FMS are discussed. Soleymanpour, D.M et al. (2004) dealt with the single row machine layout problem of FMS in which the size of machines and the clearance between the machines are assumed to be different. It is shown that the formulated 0-1 non-linear model is more intractable than the traditional QAP formulation of facility layout problem.

Kumar. R et al. (2003) presented an ant colony optimization approach for the scheduling of the FMS. This work dealt with the ant algorithm could be applied for the required problem with certain modifications. The proposed solution procedure applied a graph – based representation technique with nodes and arcs representing operations and transferred from one stage of processing to the other. This algorithm stabilizes to the solution considerably in lesser computational efforts. Extensive computational experiments have been carried out to study the influence of various parameters on the system performance.
Glover. F (1989) (1990) presented the fundamental principles of TS as a strategy for combinatorial optimization problems. Part I of this study indicated the basic principles, ranging from the short-term memory process at the core of the Search to the intermediate and long term memory process for intensifying and diversifying the Search. Part II introduced new dynamic strategies for managing *Tabu lists*, allowing complete exploitation of underlying evaluation functions. Ponnambalam S.G et al. (2002) presented the scheduling procedure to generate the optimum schedule using TS method. In this work a scheduling procedure was developed for a specific FMS to maintain its flexibility and thereby the intended performance was measured.

The schedule obtained by TS was compared with the solutions obtained by different scheduling rules.

IV Problem Formulation

The Figure 2 shows (Kumar et al., 2003) the graph-based representation of FMS scheduling problem. For a given set of jobs *j* and set of machines *m* with corresponding operations of job *j* as o_j, directed graph is constructed. Let G = (O', A) be the set of arcs connecting all possible combination of nodes. The initial node *i* is necessary in order to specify the first scheduled job when several jobs have their first operations on the same machine. Considering *n* as the total number of operations, (*n*+1) nodes and *n* (*n*-1)/2 + *j* arcs can be obtained. All nodes are pair-wise connected except node *i* which is connected only to the first operation of each job. Thus, each node corresponds to an operation of a particular job on a particular machine.

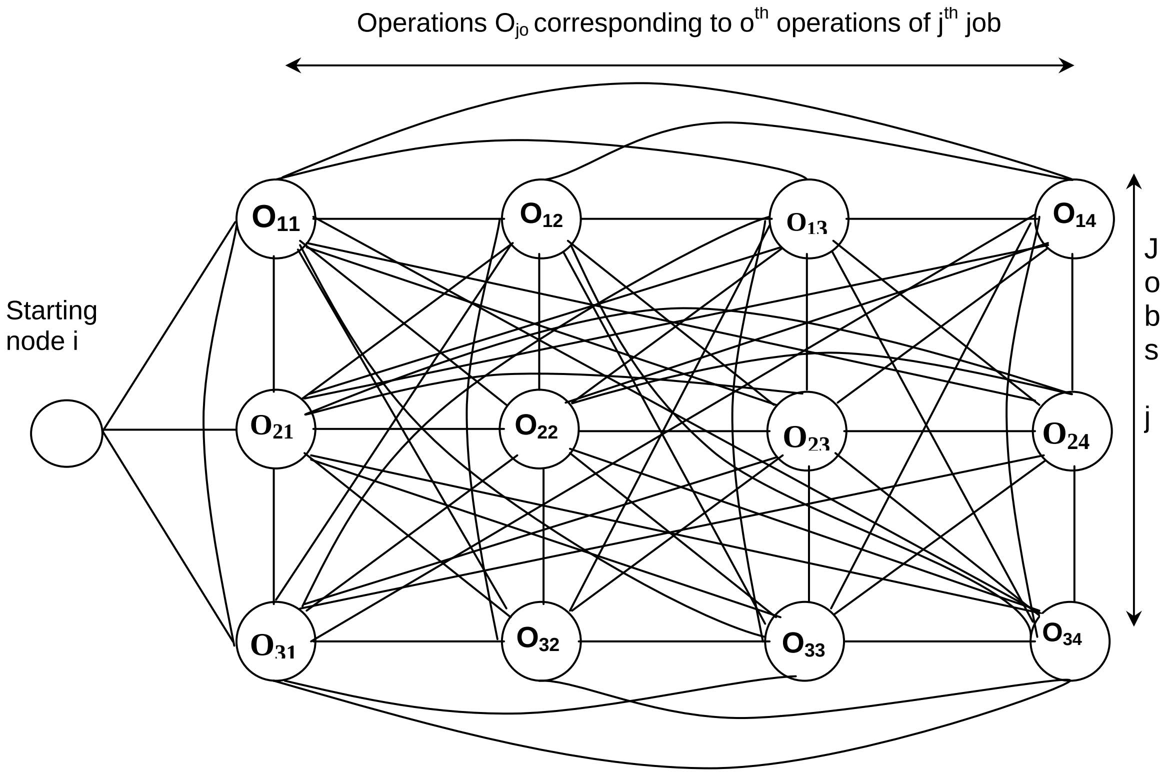

Figure 2. FMS Scheduling problem

IV.1 Problem solving methodology

The first step in TS is to create an initial solution. At this step a schedule is created and the makespan value is found. The neighborhood is identified and swapped. Then by changing the neighborhood structure the new schedule is created and the makespan value is found. If the makespan value for new schedule is less than the existing makespan value, then the new schedule is considered to be the best schedule. In this way, entire schedule is generated and the best result is found.

V Tabu Search

TS (Watson et al., 2003) is a local Search algorithm that requires a starting solution and a neighborhood structure. It proceeds by transiting from solution to solution using transitions (or moves) defined by the neighborhood structure in a systematic way until some stopping criterion is met. TS examine all the neighbors of the current solution and select the best admissible move (note that this move might be a degrading one). An admissible move is a move which is not on the *Tabu list*. The *Tabu list* is a list containing forbidden moves. Here, an initial solution is obtained by selecting the best solution (i.e., the one with minimum makespan) from any priority dispatching schedules. The disjunctive graph representation of the scheduling problem gives rise to two important observations, namely 1) given a feasible solution, the exchange of two adjacent critical operations (on the same machine) cannot yield an infeasible solution and 2) the makespan can be improved only by performing such exchanges. These insights are used to define a Search neighborhood. At each move, the effect of each possible exchange is estimated. In the event that no admissible moves are available, all Tabu restrictions are released and the Search continues from the current solution.

V.1 Elements of TS Algorithms:

The elements of the TS algorithm (Geyik and Cedimoglu, 2004) in connection with job shop problem are defined as follows:

Initial solution: A feasible initial solution is obtained by selecting from any one priority dispatching solutions. Initial solution affects the scheduling solution quality. In this paper, the Shortest Processing Time (SPT) rule is used as an initial solution method.

Neighborhood Structure: A neighborhood structure is a mechanism which contains a new set of neighbor solutions by applying simple modifications to given solutions. Each neighbor solution is reached from a given solution by move. A sequencing move can be defined as the identification of certain adjacent critical operation pairs and then the exchange of every adjacent critical operation pair on every machine. Critical operation pairs are obtained from Chu et al. (1998) algorithm by using the pair wise exchange in the disjunctive graph. Neighborhood structure is directly effective on the efficiency of TS because TS proceeds iteratively from one neighbor to another in problem solution space. Therefore a neighborhood structure must eliminate unnecessary and infeasible moves if possible.

Move: The best neighbor which is not in Tabu or satisfies a given aspiration criterion is selected as a new seed solution. The best neighbor is one whose objective function C_{max} is minimum. If the entire neighbors are in Tabu or no neighbor satisfies the aspiration criterion then the oldest neighbor entering the *Tabu list* at first will be selected as new seed solution.

Tabu list: The purpose of using *Tabu list* is to prevent the Search from degenerating by starting to cycle between the same solutions. The elements added on the *Tabu list* are attributive. The main aim of using an attributive representation is to save computer memory. The attribute that represents an element in sequencing is two operations that have interchanged in recent move. It is the arc (j, i) which is obtained by swapping a pair of operations (i, j). In order to identify Tabu status of the neighbor (j, i), it is checked whether it is on the *Tabu list* or not; if so, it is labeled as "Tabu". *Tabu list* is updated after each move in *so far* as the strategic forgetting occurs.

Aspiration criterion: The aim of using aspiration criterion, when it is necessary, is to override the Tabu status of a neighbor. The aspiration criterion is used as follows: if the move yields a solution better than the best obtained *so far* then the move is performed even it is in Tabu.

Termination criterion: The TS algorithm is terminated when the number of disimproving moves is reached to a maximum set value of 900 or no neighbor is generated or an infeasible solution is encountered.

V.2 TS Algorithm

TS relies on the systematic use of memory to guide the Search process. It uses a local Search that every step makes the best possible move from current solution to a neighbor solution even if the new solution is worse than the current one. To prevent local Search from immediately returning to previously visited solution and generally to avoid cycling, TS can explicitly memorize recently visited solutions and forbid moving back to them. Commonly, TS forbids reversing the effect of recently applied moves by declaring Tabu, those solution attributes that change in the local Search.

A general TS algorithm is as follows:

```
Step 1: Start the initial solution;  store it as current seed and the best solution.
Step 2: Generate neighbors for the current seed solutions by a neighborhood
        structure.
        Select a neighbor which is not Tabu or satisfies a given aspiration
        criterion and move it as new seed solution.
        Update the Tabu list.
        Store it as the new best solution, if the new seed solution is better than the
        earlier solution
Step 3: Repeat step 2 until a termination criterion is satisfied.
```

By following these steps the problem can be solved by the TS method.

V.3.TS algorithm for the FMS Problem

```
STEP 1: Initialization
    1. Represent the problem using a disjunctive graph
    2. Set Cycle = 1        // initialize the cycle counter
    3. Set Tabu cycle = 0         //Initialize the Tabu list
STEP 2:
    1. Generate the initial solution
    2. Store the makespan and job sequence in each machine as a Tabu and also as a current
       job order

STEP 3:
        If Cycle > maxcycle go to step 8 otherwise go to step 4
        // maxcycle is maximum number of cycle
STEP 4: Neighbor
    1. Find the Critical path for the job order
    2. Find the neighbors for the above critical path
```

3. Find the makespan for each neighbor and store it in neighbor's list
4. Sort the makespan with neighbors in neighbor's list in ascending order
5. Store the neighbor, which has minimum makespan as the best
6. Store the job sequence in each machine as a current job order

STEP 5: Move

1. Check whether the best neighbor is in *Tabu list*, if not, go to step 6 or go to step 5[2]
2. Check whether the next neighbor is available in neighbor's list, if so, store the next neighbor in the list as best and go to step 5 [1] or go to step 5[3]
3. Find the neighbors with minimum cycle in *to visit's* list and delete the neighbor in *to visit* list
4. Store the above neighbor as the best
5. Restore the job order in *to visit's* list as current job order and go to step 5 [1]

STEP 6:

1. Store the best neighbor, its makespan and its job order in *Tabu list*
2. Store neighbors with its job order in neighbor's list which is not in *Tabu list* as in *to visit's* list

STEP 7:

Cycle = cycle + 1; and go to step 3.

STEP 8:

1. Find the minimum makespan in *Tabu list*
2. Store its makespan and job order as the best

STEP 9: Output

1. Best makespan
2. Best job order

VI Implementation

Let us consider an example problem in Table 1. The algorithm presented in previous section is explained in this example. An initial solution for the problem is obtained by SPT rule. A feasible schedule of this problem is represented in Figure 3. Gantt chart of this schedule is seen in Figure 1.

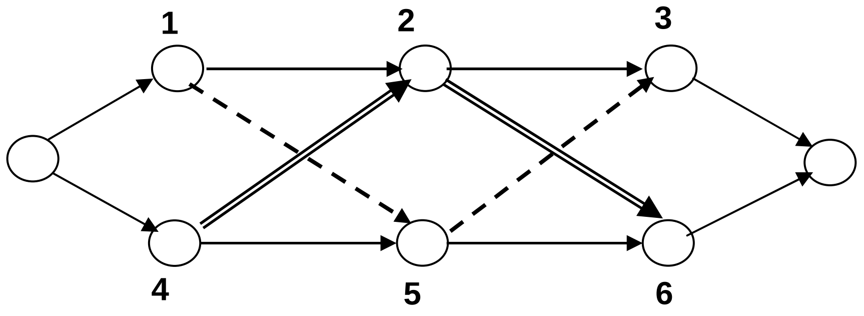

Figure 3. A Feasible Solution for the Disjunctive Graph

The sequence of operations on machines is M_1 = { $O_1 \rightarrow O_5 \rightarrow O_3$ }, M_2 = { $O_4 \rightarrow O_2 \rightarrow O_6$}. In this case the makespan for initial solution is 16. The obtained solution is only one of the possible ones and its optimality for makespan is not guaranteed. Here, the obtained solution is stored as the current seed and the best solution. Using pair wise exchanges the neighborhood is generated for this schedule. For each schedule, makespan time is calculated and the schedule which has the minimum makespan time is taken as the initial schedule for the next iteration. Neighbors found are $0_5 \rightarrow 0_6$, $O_4 \rightarrow O_2$ and $O_2 \rightarrow O_6$. After reaching neighbor solution, sequencing move is done by swapping the operation pairs. The completion time of each neighbor is calculated. Figure 4 shows the representation of disjunctive graph by swapping the neighbors $O_5 \rightarrow O_3$, $O_4 \rightarrow O_2$ and $O_2 \rightarrow O_6$**.** The obtained makespan for the neighbor $O_5 \rightarrow O_3$, $O_4 \rightarrow O_2$ and $O_2 \rightarrow O_6$ is 27, 23 and 25 respectively. The move selects the neighbor 2 ($O_4 \rightarrow O_2$) because its objective function is minimum. This neighbor satisfies the aspiration criterion, that is, the neighbor 2 is not in the *Tabu list*. So, the neighbor 2 ($O_4 \rightarrow O_2$) is added to the *Tabu list*. It is the new seed solution. The schedule for the neighbor 2 is taken as the initial schedule for the next iteration.

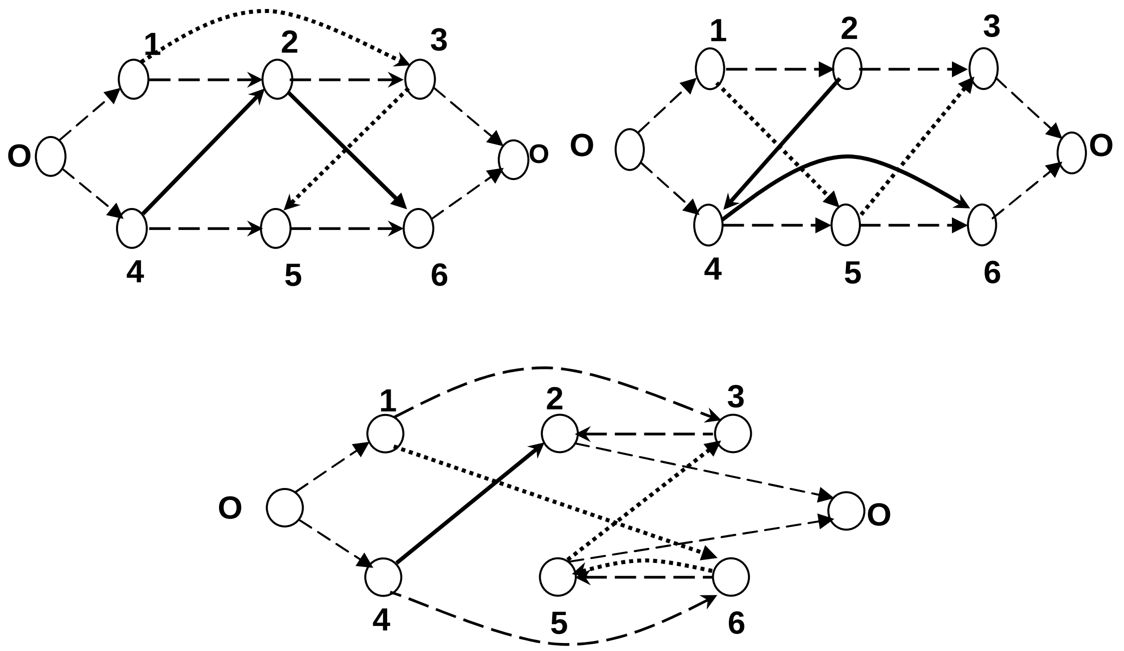

Figure 4. Disjunctive Graph for swapping the neighbors

VI.1 Flow Chart

The flow chart for the TS algorithm is shown in Figure 5.

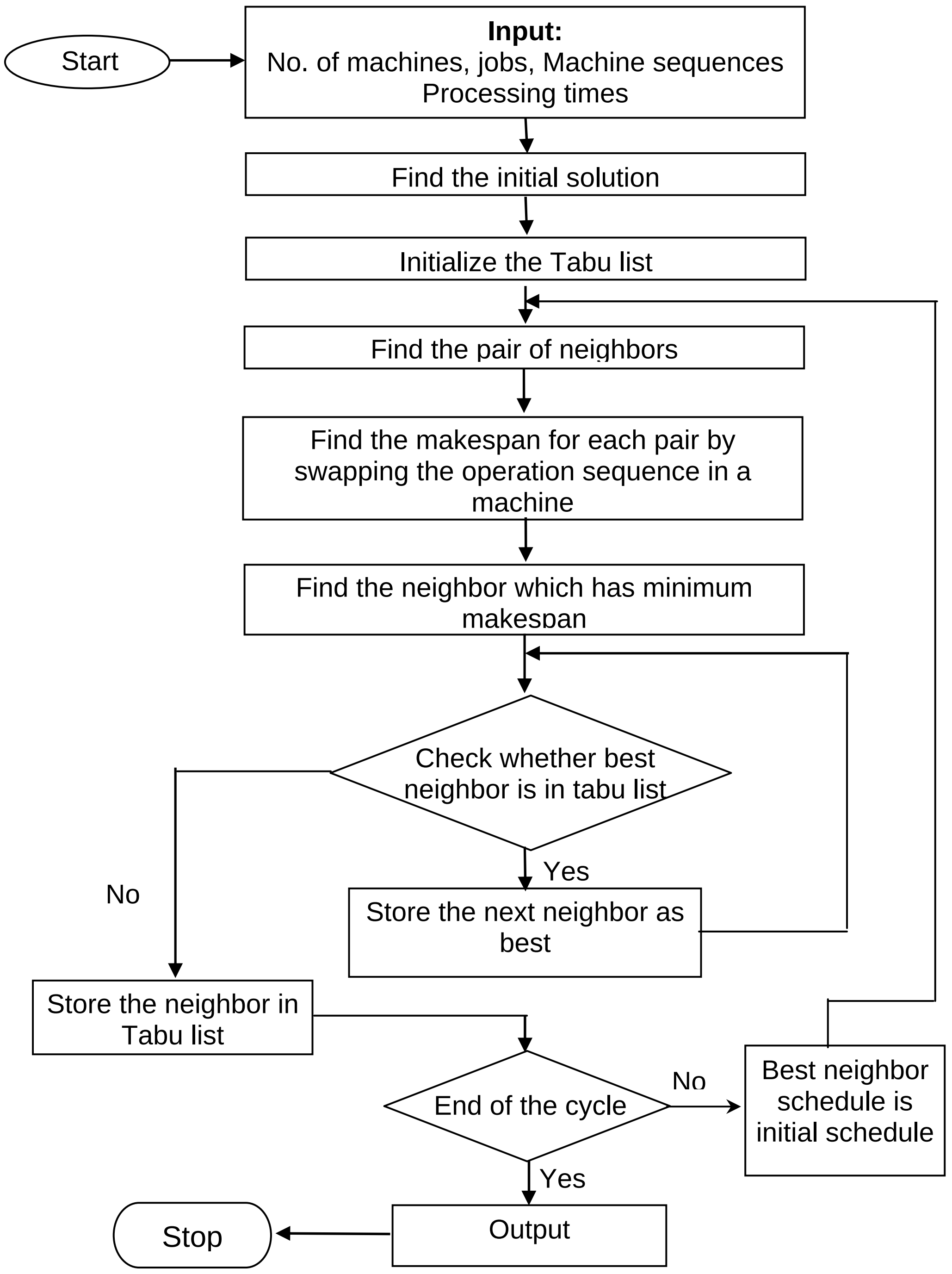
Start
Input:
No. of machines, jobs, Machine sequences
Processing times
Find the initial solution
Initialize the Tabu list
Find the pair of neighbors
Find the makespan for each pair by swapping the operation sequence in a machine
Find the neighbor which has minimum makespan
Check whether best neighbor is in tabu list
Yes
Store the next neighbor as best
No
Store the neighbor in Tabu list
End of the cycle
No
Best neighbor schedule is initial schedule
Yes
Output
Stop

Figure 5. Flow chart for Tabu Search

VI.2 TS Procedure

The first step in the TS is to create an initial solution and the *Tabu list* is started with empty. In this step a schedule is created and the makespan value is found and stored as the best. The neighbors are identified for this schedule and makespan is obtained for each pair by swapping its neighbor. Neighbors which have minimum makespan are stored as best which satisfy the aspiration criterion or not in *Tabu list*. This new schedule acts as the initial schedule for the next cycle. If the makespan value for new schedule is less than the existing makespan value, then the new schedule is the best schedule and the neighbor is stored in the *Tabu list*. Repeat the above steps till the end of the cycle or meets the termination criterion. By this way, the entire schedule is generated and the best result is found. Good results are obtained because of the versatility of the C++ language and its ability to process the codes.

VII Results and Discussion

The developed software for the TS algorithm is coded in C++ language in Linux environment and tested against the benchmark problems proposed by Lee and DiCesare (1994), Kumar et al. (2003), and Kazerooni et al. (1997). The problems are given in Appendix I

Table 2. Computational results

Test	Makespan		
	Value obtained by Lee and DiCesare (LD)	Value obtained by Kumar (LDK)	Value obtained based on proposed method
LD	439	420	410
AK01	NA	5950	5436
AK02	NA	NA	4700

VII.1 Comparison for the best Makespan

Figure 6 shows the values of the best makespan for the benchmark problems and the results obtained by using TS algorithms. The makespan value for the method proposed by Lee and DiCesare (LD) is 439 and for the same problem Kumar et al. has obtained 420 whereas the present method gives 410 which is lower than the earlier values. In the same way the obtained makespan value for the problem proposed by Kazerooni (AK01) is less than the earlier value.

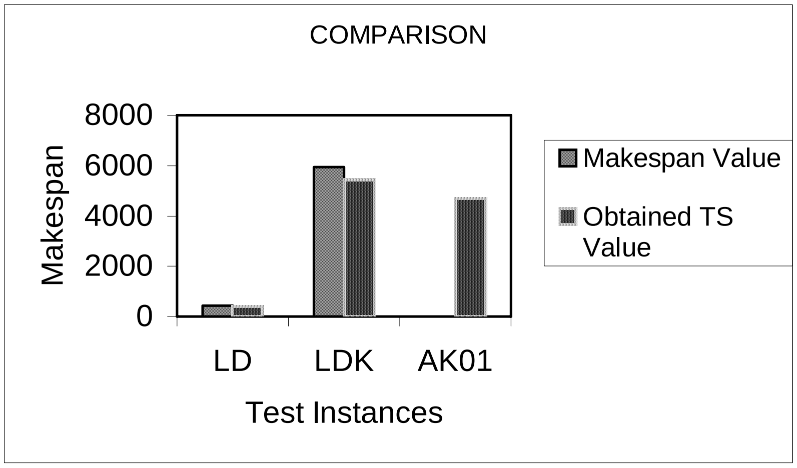

Figure 6. Comparison of obtained best makespan with benchmark values

VII.2 Makespan Vs Number of Cycles

The graphs plotted against the number of cycles and the makespan values for different problems are shown in Figures 7 – 9. It is learned that initially the makespan value decreases as the number of cycles increases and after certain number of cycles the makespan value is almost constant. But after certain number of cycles the makespan value starts increasing and it is mainly due to the neighborhood cyclic behavior of the TS algorithm.

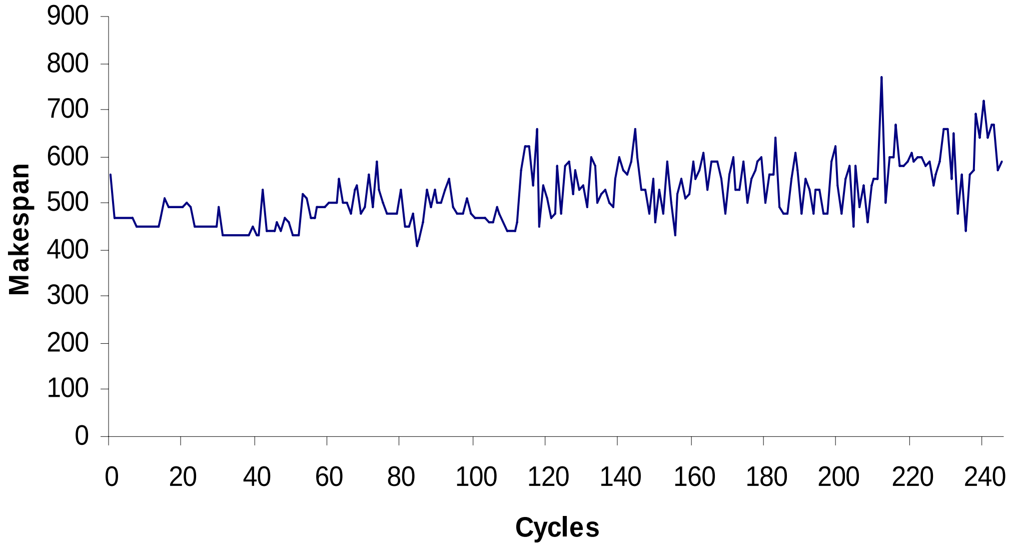

Figure 7. Typical value of makespan for the problem LD (5X3)

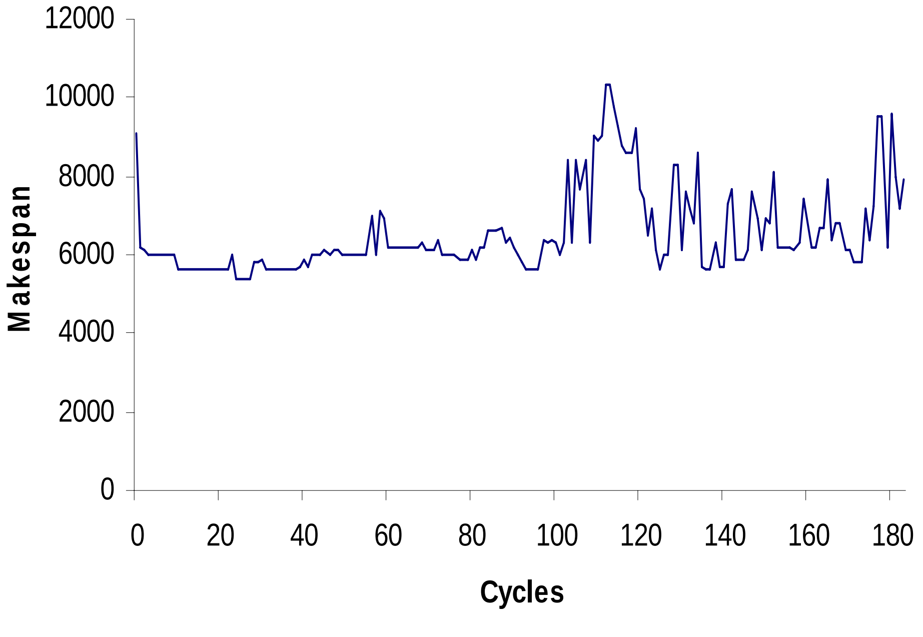

Figure 8. Typical value of makespan for the problem AK01 (7X6)

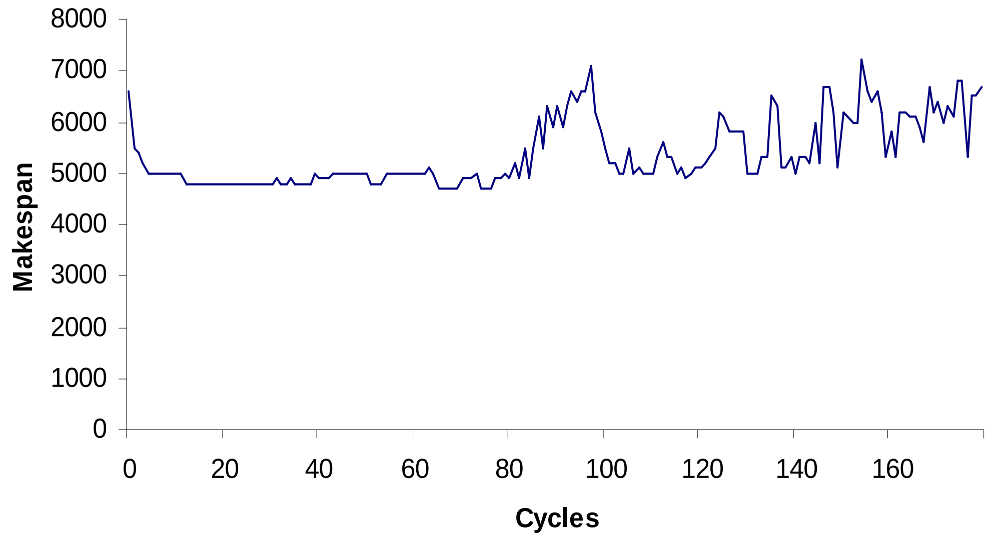

Figure 9. Typical value of makespan for the problem AK02 (7X6)

VII.3 Gantt chart for LD Problem

The Figure 10 shows the Gantt chart for LD problem using TS method. From this the makespan value is found to be 410 for the optimal path.

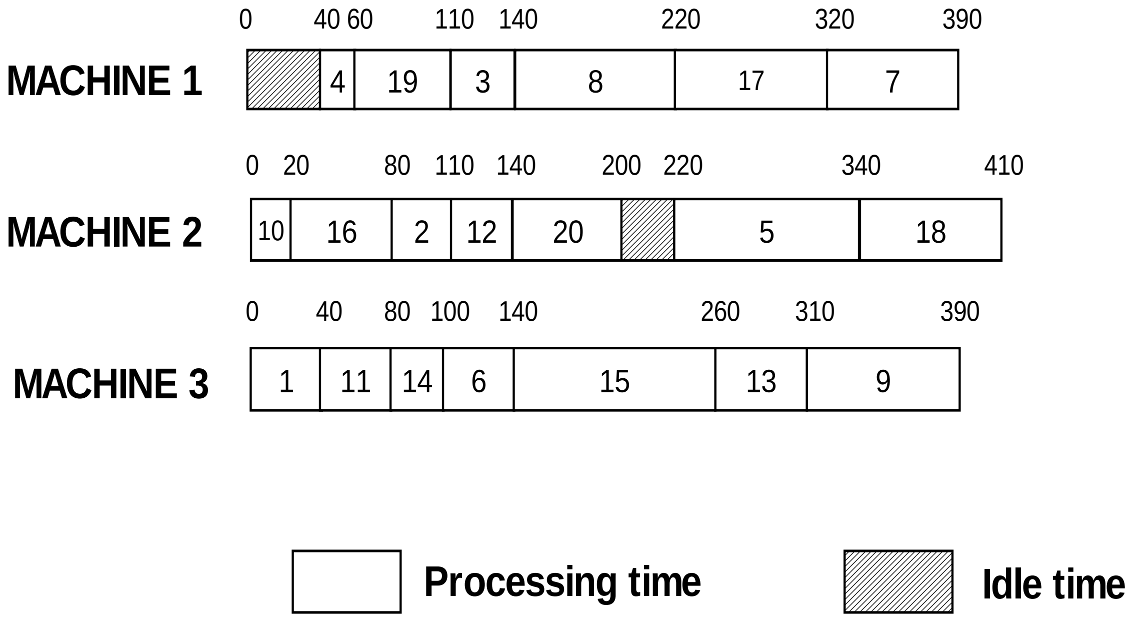

Figure 10. Gantt chart for LD problem

The Figure 11 shows the disjunctive graph for LD problem using TS method.

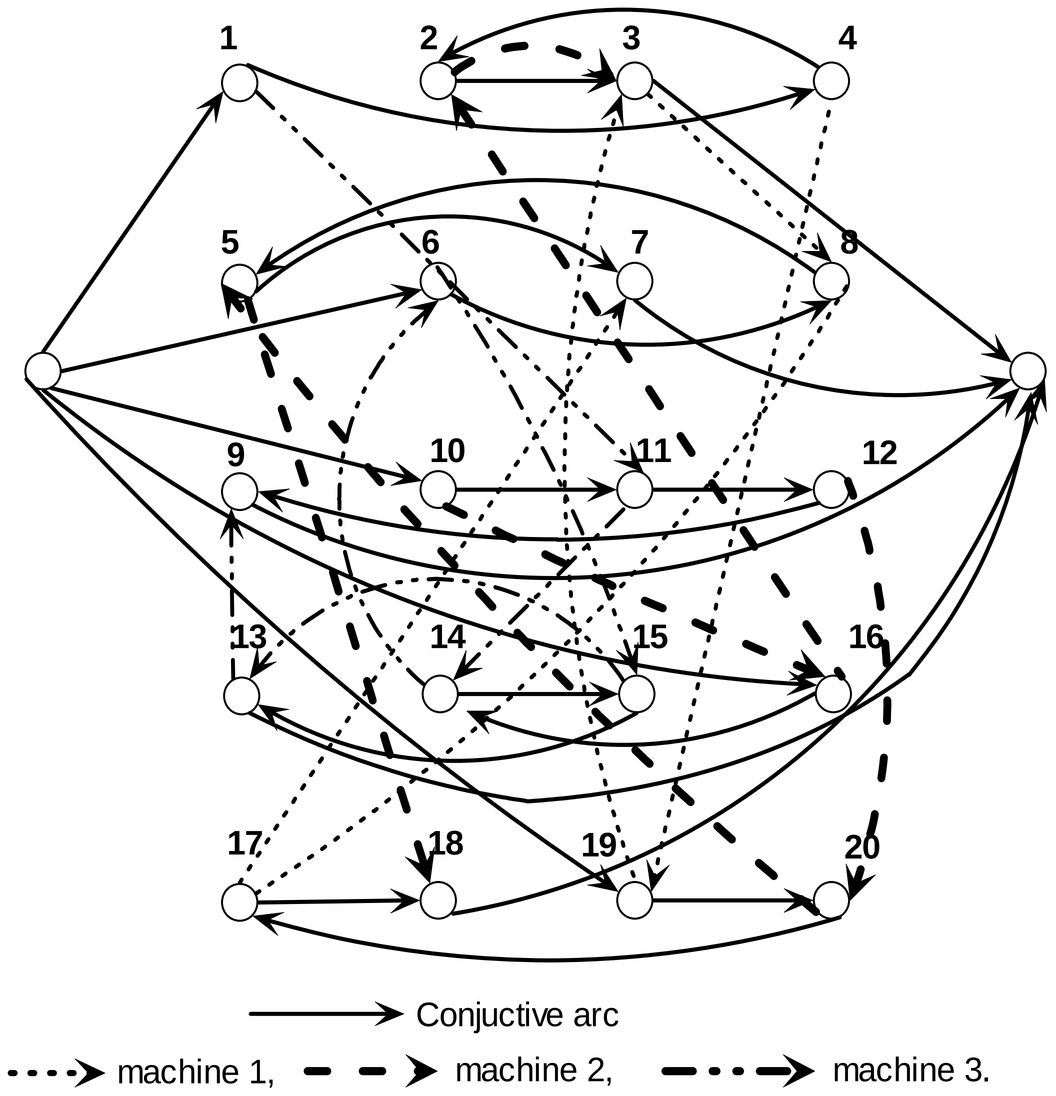

Figure 11. Disjunctive graph for LD problem

VII.4 Implementation of TS algorithm

The Table 3 exhibits the details about Tabu implementation for LD problem.

Neighbors	Makespan	Move	Tabu list
(5, 17), (20, 7), (7, 18), (8, 3), (15, 13), (2, 12), (14, 10), (13, 9), (16, 2), (11, 14), (12, 20), (4, 8), (6, 15),	470, 530, 530, 530, 560, 560, 560, 560, 570, 570, 610, 620, 620,	(5, 17)	(5, 17)
(2, 12), (20, 7), (19, 17), (13, 9), (14, 10), (11, 14), (8, 3), (17, 5), (12, 20), (6, 15), (15, 13), (4, 8), (16, 2),	470, 470, 470, 470, 500, 510, 550, 560, 570, 580, 580, 590, 590,	(2, 12)	(2, 12) , (5, 17)
(11, 14), (12, 2), (19, 17), (13, 9), (20, 7), (2, 20), (16, 12), (17, 5), (14, 10), (6, 15), (15, 13), (4, 8)	470, 470, 470, 470, 480, 520, 560, 560, 560, 580, 580, 590	(11, 14)	(11, 14), (2, 12), (5, 17)
(14, 11), (19, 17), (13, 9), (20, 7), (4, 8), (11,10), (12, 2), (2, 20), (17, 5), (6, 15), (15, 13),	470, 470, 470, 480, 490, 490, 510, 520, 560, 580, 580,	(14, 11)	(14, 11), (11, 14), (2, 12), (5, 17)
:	:	:	:

Table 3. Generation of neighbors in each iteration and movement of best neighbor

VIII Conclusion

In this paper, the method of solving the FMS scheduling using TS technique with an objective of minimising the makespan is presented. The disjunctive graph is modified to represent the operation-based FMS schedule. The elements of TS used for solving the scheduling problem are explained.

The TS has been modified for solving the FMS scheduling problem and the algorithm is explained with flow chart. In the paper, the method of implementation of TS for FMS problem is explained with simple numerical example. SPT dispatching rule is used to get the initial solution for the proposed TS algorithm.

Software has been developed in C under Linux environment for the TS algorithm and tested on several FMS problem instances. The obtained results are compared with the results obtained by different authors. It is proved that for all the problem instances tested better makespan value has been obtained from this proposed study when compared with other authors' results.

Although there is a huge amount of literature on classical TS, the TS algorithm for solving real world FMS problem does not have a rich literature. So there is a need to develop effective algorithm for this complex problem. An effort is made in this paper to model and solve the different benchmark FMS problems. With the inclusion of tooling and machine constraints this developed TS algorithm could be used to solve real world problems.

References

1. Chu, C., Proth, J. M., and Wang, C. (1998). Improving Job-Shop Schedules Through Critical Pairwise Exchanges, *International journal of production Research*, Vol. 36, No. 3, pp. 683- 694.
2. Geyik, F., and Cedimoglu, I. H. (2004). The Strategies and Parameters of Tabu Search for Job–Shop Scheduling, *Journal of Intelligent Manufacturing*, Vol. 15, pp. 439 – 448.
3. Glover, F. (1989). Tabu Search – Part I, *Operations Research Society of America, Journal on Computing,* Vol. 1, No.3, pp. 190-206.
4. Glover, F. (1990). Tabu Search – Part II, *Operations Research Society of America, Journal on Computing*, Vol. 2, No.1, pp. 4-32.
5. Hansmann, K.W., and Michael Hoeck, M. (1997). Production Control of a Flexible Manufacturing System in a Job Shop Environment, *Int. Trans. Operational Research*, Vol. 4, No. 516, pp. 341-351.
6. Kazerooni, A., Chan, F. T. S., and Abhary, K. (1997). A Fuzzy Integrated Decision-Making Support System for Scheduling of FMS using Simulation, *Computer Integrated Manufacturing System*, Vol. 10, No. 1, pp. 27 – 34.
7. Kumar, R., Tiwari, M. K., and Shankar, R. (2003). Scheduling of Flexible Manufacturing Systems: An Ant Colony Optimization Approach, *Proc. Institution of Mechanical Engineers, Part B: J. Engineering Manufacture*, Vol. 217, pp. 1443-1453.
8. Kusiak, A. (1986). Application of Operational Research Models and Techniques in Flexible Manufacturing Systems, *European Journal of Operational Research*, Vol. 24, pp. 336-345.
9. Lee, D.Y., and DiCesare, F. (1994). Scheduling Flexible Manufacturing Systems using Petri nets and Heuristic Search, *IEEE Trans. on Robotics and Automation*, Vol. 10, No. 2, pp. 123-132.
10. Ponnambalam, S. G., Ganapathy, V., Saravana Sankar, S., and Karthikeyan,R. (2002). Scheduling Flexible Manufacturing System using Tabu Search Method, *IEEE ICIT '02*, Bangkok, Thailand, pp. 1043-1048.
11. Rao, P. N., Tewari, N. K., and Kundra, T. K. (1993). Computer Aided Manufacturing, *Tata McGraw-Hill Publishing Company Limited*, New Delhi, pp. 351.
12. Soleymanpour, D. M., Vrat.P., and Shankar R. (2004). An Ant Algorithm for the Single Row Layout Problem in Flexible Manufacturing Systems, *Computers and Operations Research,* Vol. 32, No. 3, pp. 583-598.
13. Van Laarhoven, P. J. M., Aarts, E. H. L., and Lenstra, J. K. (1992). Job Shop Scheduling Problem by Simulated Annealing, *Operations Research*, Vol. 40, pp. 113-125.
14. Watson, J. P., Christopher Beck, J., Adele E. Howe, and Darrell Whitley, L. (2003). Problem Difficulty for Tabu Search in Job-Shop Scheduling, *Artificial Intelligence*, Vol. 143, pp. 189–217.

APPENDIX I

TEST INSTANCES

Table AI.1 FMS Scheduling data for Lee and Dicesare (LD).

Job	O_1	O_2	O_3	O_4
1	1(7) 3(4)	2(3)	1(3) 3(6)	1(2) 2(4)
2	1(8) 2(12)	3(4)	1(7) 2(14)	1(8) 3(4)
3	1(10) 2(15) 3(8)	2(2) 3(6)	1(2) 3(4)	1(6) 2(3)
4	2(9) 3(5)	1(6) 3(2)	2(7) 3(12)	1(9) 2(6) 3(3)
5	1(10) 3(15)	2(7) 3(14)	1(5) 2(8)	1(4) 2(6) 3(8)

Table AI.2 FMS Scheduling data for A.Kazerooni (AK01)

Job	O_1	O_2	O_3	O_4	O_5
1	4(30)	1(50) 2(50)	4(31)	1(30)	6(10)
2	3(90)	1(70)	2(120)	3(40)	6(10)
3	5(80)	4(110) 1(110)	5(140) 2(140)	4(50)	6(10)
4	2(50)	3(150) 4(150)	2(70) 1(70)	5(60)	6(10)
5	3(40)	1(70)	4(20)	2(70)	6(10)
6	2 (20) 5 (20)	1 (50) 3 (50)	4 (240) 5 (240)	2 (80)	6(10)
7	3 (60) 1 (60)	4 (60) 5 (60)	3 (70)	4 (90)	6(10)

Table AI.3 FMS Scheduling data for A.Kazerooni (AK02)

Job	O_1	O_2	O_3	O_4	O_5
1	4(30)	1(50) 2(50)	4(20)	1(10)	6(30)
2	3(90)	1(40)	2(120)	3(20)	6(60)
3	5(80)	4(30) 1(30)	5(70) 2(70)	4(20)	6(30)
4	2(50)	3(10) 4(10)	1(90) 2(90)	5(10)	6(30)
5	3(40)	1(50)	4(20)	2(70)	6(30)
6	2 (20) 5 (20)	1 (50) 3 (50)	4 (150) 5 (150)	2 (80)	6 (30)
7	1 (80) 3 (80)	4 (60) 5 (60)	3 (30)	4 (30)	6 (30)