Scheduling of flexible manufacturing systems: an ant colony optimization approach

R Kumar[1], **M K Tiwari**[1] and **R Shankar**[2]*
[1]Department of Manufacturing Technology, National Institute of Foundry and Forge Technology, Ranchi, India
[2]Department of Management Studies, Indian Institute of Technology Delhi, New Delhi, India

Abstract: The scheduling problem for flexible manufacturing systems (FMSs) has been attempted in this paper using the ant colony optimization (ACO) technique. Since the operation of a job in FMSs can be performed on more than one machine, the scheduling of the FMS is considered as a computationally hard problem. Ant algorithms are based on the foraging behaviour of real ants. The article deals with the ant algorithm with certain modifications that make it suitable for application to the required problem. The proposed solution procedure applies a graph-based representation technique with nodes and arcs representing operation and transfer from one stage of processing to the other. Individual ants move from the initial node to the final node through all nodes desired to be visited. The solution of the algorithm is a collective outcome of the solution found by all the ants. The pheromone trail is updated after all the ants have found out their respective solutions. Various features like stagnation avoidance and prevention from quick convergence have been incorporated in the proposed algorithm so that the near-optimal solution is obtained for the FMS scheduling problem, which is considered as a non-polynomial (NP)-hard problem. The algorithm stabilizes to the solution in considerably lesser computational effort. Extensive computational experiments have been carried out to study the influence of various parameters on the system performance.

Keywords: flexible manufacturing systems, ant colony optimization, scheduling

NOTATION

c	small positive constant
G	set of arcs connecting all possible combinations of nodes
G_k	set of all nodes still to be visited by ant k
j	a job
j'	maximum number of jobs available from time 0 onwards
J	set of jobs
k	an ant
m	a machine
m'	maximum number of machines
M	set of machines
n	a node
N	total number of operations
NC	counter for number of iterations
$N\_\max$	maximum number of iterations
o	an operation
o_j	an operation of job j
o'_j	maximum number of operations for job j
O_j	operation set of job j
O'	set of all operations
p	randomly generated quantity
p^+_{best}	best makespan
p^+_{iter}	optimal makespan for an iteration iter
$p_{j,o,m}$	processing time of operation o of job j on machine m
p_0	parameter used to attain quick convergence of the algorithm
P_k	makespan by the kth ant
$P^k_{il}(t)$	transition probability of moving from node i to node l for ant k
q	random number
Q	positive constant
S_k	set of nodes allowed at the next step by ant k
t	time
tabu^k	list of nodes travelled by ant k
T_k	tabu list of ant k
α	factor that controls the importance of the trail
β	factor that controls the importance of visibility
η_{kl}	visibility from node k to node l

The MS was received on 25 November 2002 and was accepted after revision for publication on 26 June 2003.
**Corresponding author: Department of Management Studies, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110 016, India.*

ρ	coefficient such that $(1 \cdot \rho)$ represents the evaporation of the trail between time t and $t+$ number of iterations till this instant
τ_{kl}	trail level from node k to node l
$\tau_{il}(0)$	initial pheromone trail on edge il
$\Delta\tau_{il}^{k}$	quantity per unit time of pheromone trail laid on the edge (i, l) by the kth ant between time t and $t+$ number of iterations till this instant
$[\varphi]$	null set

1 INTRODUCTION

A flexible manufacturing system (FMS) consists of a collection of numerically controlled machines with multifunction ability, an automatic material handling system and an online computer network. This network is capable of controlling and directing the whole system. An FMS combines the advantages of a traditional flow line and job-shop systems to meet the changing demands [**1**–**3**]. Thus, it involves many problems, which can be divided into four stages: (a) design, (b) system set-up, (c) scheduling and (d) control [**4**].

The scheduling problem of an FMS can be overviewed as follows. Due to flexibility of the FMS, a given operation can usually be performed on a number of machines. The flexibility of the FMS allows many alternative routeings. Among these multiple routeings it is important to select the best available machine, which can perform the operation. This makes even a two-machine scheduling problem non-polynomial (NP)-hard [**5**, **6**]. Since the operations are computer controlled, set-ups between the consecutive operations are automated, and as most of the operations are processed by NC machines, it can be assumed that processing times are nearly deterministic [**7**].

To solve this combinatorial optimization problem, several approaches have been proposed by researchers. Yih and Thesen [**8**] provided a semi-Markov decision model for a real-time scheduling problem to determine the sequence of parts within an FMS. Thesen and Lie [**9**], Shaw [**10**], Liang *et al.* [**11**] and Chryssolouris *et al.* [**12**] used simulation to acquire knowledge related to relationships between system attributes and performance of dispatching rules, while Talavage and Shodham [**13**] proposed a COMMAND framework, which combines capabilities of simulation and knowledge engineering to search for an efficient design and control strategy. Bengu [**14**], Kim and Kim [**15**] and Duffie and Prabhu [**16**] presented a simulation-based approach. An appropriate combination of dispatching rules for selecting production orders is suggested by Muraki and Ishii [**17**]. Several researchers have applied heuristic search to find the near-optimal schedule. The Petri-net (PN) model can also prove to be a promising tool, as shown by Marie [**18**], Naiqi [**19**] and Kim *et al.* [**20**] for FMS scheduling.

To solve the scheduling problem, the ant colony optimization technique is utilized. Ant algorithms were proposed by Dorigo *et al.* [**21**] as a multi-agent approach to different combinatorial optimization problems. Since then ant algorithms have been modified time and again and have been applied to problems like the travelling salesperson problem (TSP) [**22**–**29**], the quadratic assignment problem (QAP) [**30**–**33**], job-shop scheduling [**34**], the vehicle routeing problem [**35**], etc.

In this paper, a graph-based approach has been applied in conjunction with a modified version of the algorithms to solve the FMS scheduling problem. This shows promising results by converging to the near-optimal solution in lesser computation times and hence saves central processing unit (CPU) time. The proposed methodology takes care of all the parameters of the ant colony optimization (ACO) algorithms and also incorporates preventive measures, such as stagnation avoidance and prevention of early convergence of the solution, to overcome the difficulties in using the ACO algorithms. For a comprehensive survey of ACO the reader is referred to Dorigo *et al.* [**21**].

2 PROBLEM FORMULATION

In this section, the problem is described. In the problem, a job $j \in J$ (where J is a set of jobs, such that $J = \{1, 2, \ldots, j, \ldots, j'\}$, j' being the maximum number of jobs available from time zero onwards) is to be assigned to a machine $m \in M$ (where M is the set of machines such that $M = \{1, 2, \ldots, m, \ldots, m'\}$, m' being the number of machines), such that the makespan is minimum. The job constitutes a sequence of operations and an operation $o_j \in O_j$ (where $O_j = \{o_1, o_2, \ldots, o_j, \ldots, o_j'\}$, o_j' being the maximum number of operations for job j) can have some flexibility depending on the feasibility of an operation to be performed on a machine. For a feasible operation o of job j on machine m, the processing time can be denoted by $p_{j,o,m}$. Here, it should be noted that $O_j \in O'$ where O_j gives the set of operations for job j and O' is the set of all operations on all jobs. Thus, the objective of the problem is to schedule operations of all the jobs on machines such that the overall makespan is minimized. Keeping this objective in mind, the next section proposes an ACO-based scheduling approach.

3 SOLUTION APPROACH

The solution approach applied in the paper is a weighted graph-based approach with its optimization being done by the ACO techniques. The graph-based representation of the FMS scheduling problem is explained below.

3.1 Graph-based representation

For a given set of jobs J and set of machines M with corresponding operations of job j as o_j, a directed graph is constructed. Let $G = (O', A)$ be the set of arcs

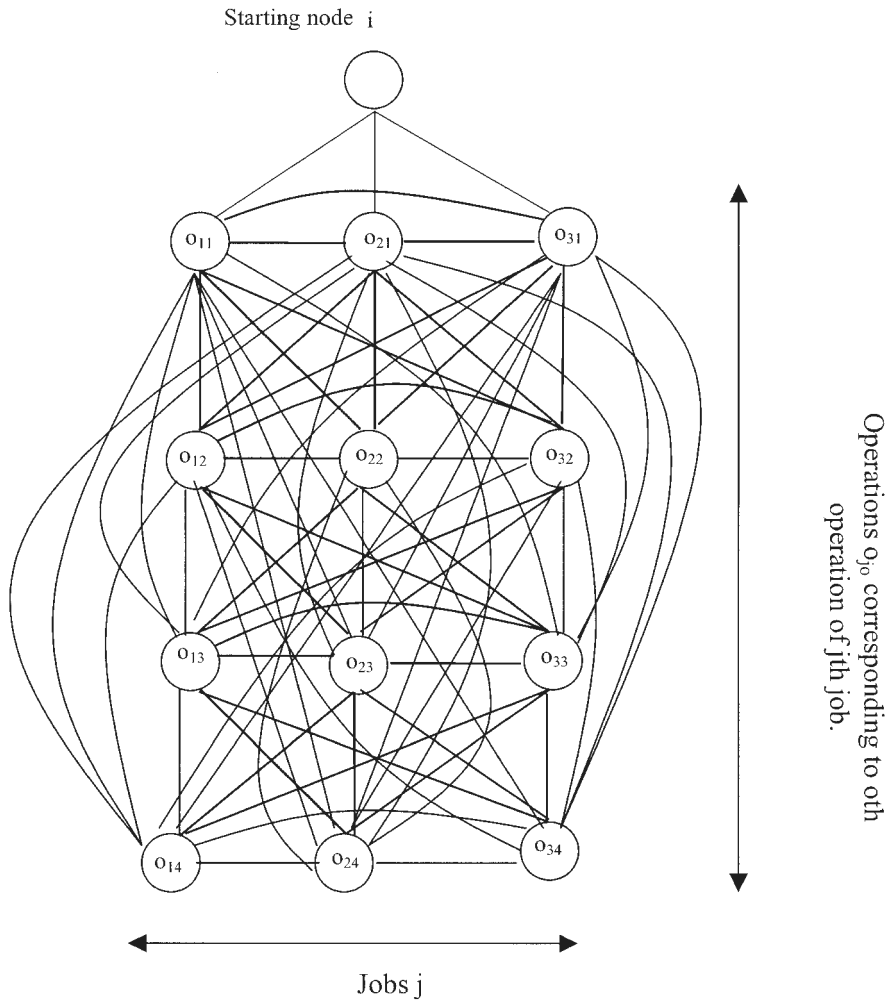

Fig. 1 Graph-based representation of the FMS scheduling problem

connecting all possible combination of nodes. The initial node i in Fig. 1 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 $N(N\cdot\ 1)/2+J$ arcs can be obtained, where all nodes are pairwise 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. Each arc is weighted by a pair of numbers $\{\tau_{kl}, \eta_{kl}\}$, where τ_{kl} is the trail level and η_{kl} is the visibility, computed initially according to a desirability measure derived from a greedy problem-specific heuristic like LPT (longest processing time) or LRPT (largest remaining processing time). These factors will be dealt with later in detail. An overview of the ant systems is presented next.

3.2 Ant systems

The ant systems were initially proposed by Dorigo *et al.* [**21**] as a general-purpose metaheuristic approach for combinatorial optimization problems. The underlying idea was to parallelize search over several constructive computational threads, based on a dynamic memory structure incorporating information on the effectiveness of previously obtained results. More precisely, an artificial ant is a simple computational agent, which iteratively constructs a solution for the problem. This is done by emulating the behaviour of real ants. Ants deposit a substance called *pheromone* on the path that they have traversed from the source to the destination nest and the ants coming at a later stage apply a probabilistic approach in selecting the node with the highest pheromone trail on the paths. Thus the ants move in an *autocatalytic* process (positive feedback), favouring the path along which more ants have travelled and by and by traverse all the nodes.

The approach of the artificial ants applied to the FMS scheduling is slightly different. Each node here signifies an operation of a job on a machine. Due to flexibility of the system there can be more than one possible node for the same operation of the same job. This is possible owing to the capability of more than one machine to do the same job. Thus, whenever an ant chooses a

node for a particular operation, all other nodes belonging to the same operation but on a different machine are to be removed from the set of nodes still to be visited by the ant. This can be understood in the following manner. For each ant, say k, let G_k be the set of all nodes still to be visited and S_k be the set of nodes allowed at the next step. For example, suppose that S_k consists of a node signifying operation o of job j. This operation can be done on machine $m = (1, 2, 3, \ldots, m')$. Thus, on the selection of node $n_{j,o,m}$ all nodes with parameters $n_{o,j}$ are to be eliminated from G_k as well as S_k. The selected node $n_{j,o,m}$ is added to the tabu list T_k for ant k and if the chosen node is not the last in its job then nodes related to its immediate successor are added to S_k. The process is iterated till $G_k = \{\varphi\}$. At the end, the sequence of the nodes visited by the ant given by the tabu list specifies the solution proposed by ant k. The node selection procedure is purely probabilistic. Let τ_{il} be the intensity of the pheromone trail on the edge (i, l) at the time t. Each ant at time t chooses the next node, where it will be at time $t + 1$. Therefore, if each ant similarly chooses its next node, then k' ants (total number of ants) will choose the next node to move in this interval, called an iteration of the ACO algorithm, in time $(t, t + 1)$. Then, after every T iterations $(T = \sum_{j=1}^{j'} o'_j)$ each ant has completed a tour. At this point the trail intensity is updated according to the following rule:

$$\tau_{il}(t + n) = \rho\tau_{il}(t) + \Delta\tau_{il} \tag{1}$$

where ρ is a coefficient such that $(1 \cdot \ \rho)$ represents the evaporation of the trail between time t and $t + n$. Also, in order to avoid unlimited accumulation of the trail, the value of ρ should be $0 < \rho < 1$. Here,

$$\Delta\tau_{il} = \sum_{k=1}^{K'} \Delta\tau_{il}^k \tag{2}$$

where K' is the total number of ants and $\Delta\tau_{il}^k$ is the quantity per unit time of the pheromone trail laid on the edge (i, l) by the kth ant between times t and $t + n$. Also,

$$\Delta\tau_{il}^k = \begin{cases} \dfrac{Q}{P_k} & \text{if the } k\text{th ant uses the edge } (i, l) \text{ in its tour (between time } t \text{ and } t + n) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where Q is a positive constant and P_k is the makespan by the kth ant. The transition probability of moving from node i to node l for the given ant can be given as

$$p_{il}^k(t) = \begin{cases} \dfrac{[\tau_{il}(t)]^\alpha [\eta_{il}]^\beta}{\sum_{k \in S_k} [\tau_{il}(t)]^\alpha [\eta_{il}]^\beta} & \text{if } l \in S_k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where S_k represents the set of nodes to which ant k can move from the present state and η_{il}, the visibility from node i to node l, is a heuristic value obtained from a greedy heuristic. This remains constant throughout. Parameters α and β control the relative importance of the trail versus visibility.

The above relationship among the parameters for an ACO heuristic are adopted from Dorigo *et al.* [**22**]. After the ant has completed its tour, the tabu list is emptied and the ant is free again to choose its path. Thus, this basic procedure is implemented in the improved algorithm to fulfil the objective of solving the FMS scheduling problem.

3.3 Ant algorithm

Several improvements and modifications have been suggested to improve the performance of the original ANT algorithm. A few important ones are mentioned here.

3.3.1 Prevention of a very quick convergence of the algorithm

This can be achieved by incorporating the parameter p_0 in the algorithm:

$$p_0 = \frac{\log_e(\mathrm{NC})}{\log_e(N\_\max)} \tag{5}$$

where NC is the counter for the number of iterations and N_max is the maximum number of iterations. It can be easily seen that $0 \leqslant p_0 \leqslant 1$. For the lower values of NC, the quantity p_0 remains close to 0 and progressively increases as the value of NC increases and approaches unity at the end. Using this quantity p_0 to represent the probability of avoiding newer solutions, p_0 is compared to a randomly generated quantity p. As the value of NC increases, the probability of the event $p > p_0$ decreases. It is ostensibly clear from the above fact that at the lower values of NC, the probability of searching newer paths by the ants is on the higher side, whereas, as NC is gradually increased, the probability to generate new paths considerably decreases. In this way, a very quick convergence of the algorithm to a locally optimized solution is prevented.

3.3.2 Stagnation avoidance

Stagnation denotes the undesirable situation in which all ants construct the same solution over and over again, making further exploration of newer paths almost impossible. This derives from excessive trail levels on the edges of one solution, and can be observed in advanced phases of the search process. It happens when the ACO parameters (α, β, ρ) are not well tuned for undertaking the problem. In particular, stagnation derives from a wrong value of parameter ρ in the original algorithm. If it is too high, stagnation might take place,

while if it is too low, little information is conveyed from previous solutions and the algorithm becomes a randomized greedy search procedure. Thus, ρ should be chosen very carefully. To prevent stagnation, a random number q is generated and compared to the trail on the edge. For a very high pheromone trail (τ_{il}), q will be less and for a low value of the same, q is high. Thus, if the trail is not too high, the algorithm may overlook the best path at the selection point and take up an alternative edge. However, if the trail is very high, the algorithm has a tendency to stick to this edge. Therefore, it can be concluded from the above discussion that the parameter q prevents the system stagnating at points and thus avoids locally optimized solutions.

With these preventive measures, the proposed algorithm is presented as follows.

Step 1: initialization

1. Represent the problem using a weighted directed graph.
2. Randomly distribute ants on the nodes.
3. Set $t = 0$; /*{time counter}*/
4. Set NC = 1; /*(number of iterations/number of cycles counter)*/
5. Set $\tau_{il}(0) = c$; on each node /*$\tau_{il}(0)$ is the initial pheromone trail on the edge il and c is a small positive constant*/
6. Set $\Delta\tau_{il} = 0$; on each node /*$\Delta\tau_{il}$ is the increase in the trail level on edge $i \cdot l$*/
7. Set $\text{tabu}^k = 0$; /*tabu^k gives the list of nodes traversed by ant k*/
8. Set $p_0 = 0$;

Step 2. If NC > N_max go to Step 8 else go to Step 3 /*N_max is the maximum number of iterations*/

Step 3. If $m > m$_max go to Step 7 else go to Step 4 /*m gives the ant number and m_max stands for the maximum number of ants*/

Step 4. If $\text{tabu}^k >= \text{tabu}^k_{\max}$ go to Step 6 else go to Step 5 /*$\text{tabu}^k_{\max}$ gives the maximum number of nodes to be visited by ant k*/

Step 5: node selection

1. Generate random number p ($0 \leqslant p \leqslant 1$).
2. If $p \geqslant p_0$ then go to Step 5(3) else go to Step 5(4).
3. Generate random number q ($q \in S_k$), $select = q$, go to Step 5(10).
4. Compare probabilities of possible outgoing nodes.
5. Choose the node having the highest probability (p^k_{il}).
6. Generate a random number q ($0 \leqslant q \leqslant 1$).
7. If $q \geqslant p^k_{il}$ then go to Step 5(8) else go to Step 5(9).
8. Generate a random number q ($q \in S_k$), $select = q$, go to Step 5(10).
9. Choose the node with the highest p^k_{il} value, $select = 1$.
10. Choose the node $select$ as the next node to move to.
11. Add $select$ to tabu^k, delete it from G_k and S_k and go to Step 4.

Step 6. $m = m + 1$, go to Step 3.

Step 7: updating

1. Find p^+_{iter}; /*p^+_{iter} is the optimal makespan for iteration*/
2. If $p^+_{\text{iter}} < p^+_{\text{best}}$ then $p^+_{\text{best}} = p^+_{\text{iter}}$; /*$p^+_{\text{best}}$ is the best makespan*/
3. Update $\tau_{il}(t)$.
4. Empty all tabu lists (i.e. tabu^k).
5. NC = NC + 1.
6. $p_0 = \log_e(\text{NC})/\log_e(N\_\max)$, go to Step 2.

Step 8: output

p^+_{best}

In the next section, numerical illustrations have been presented to show the effectiveness of the proposed method.

4 NUMERICAL ILLUSTRATION

For the sake of simplicity, first a simple example is considered where the set-up time, etc., has been ignored.

4.1 Example 1

The example of five jobs and three machines of Lee and Dicesare [**36**] is taken. Each job has four operations to be performed. Each operation can be done on one or more machines. The detailed description of the operation of a job and its corresponding machines (if the machine is capable of performing the operation) with its respective processing time is given in Table 1.

The system adopted to represent the time is $p_{j,o,m}$, i.e. the first index refers to the job number j, the second refers to the operation number o on job j and the third refers to machine number m on which the operation o could be performed. The lot size for the jobs is 10. Here set-up and other factors leading to the wastage of time are neglected. The program for the proposed algorithm is coded in C++ programming language. The value of the makespan for the problem with 400 iterations comes out to be 439 by the method proposed by Lee and Dicesari [**36**] and Yim and Lee [**37**], whereas for the same problem the present method gives 420, which is lower than the value obtained by Lee and Dicesari [**36**]. Thus the superiority of the present method is proved over the PN-based method proposed by Lee and Dicesari [**36**].

Table 1 Processing time for Example 1

Job number	Operation number	Machine number	Processing time (s)
1	1	1	7
1	1	3	4
1	2	2	3
1	3	1	3
1	3	3	6
1	4	1	2
1	4	2	4
2	1	1	8
2	1	2	12
2	2	3	4
2	3	1	7
2	3	2	14
2	4	1	8
2	4	3	4
3	1	1	10
3	1	2	15
3	1	3	8
3	2	2	2
3	2	3	6
3	3	1	2
3	3	3	4
3	4	1	6
3	4	2	3
4	1	2	9
4	1	3	5
4	2	1	6
4	2	3	2
4	3	2	7
4	3	3	12
4	4	1	9
4	4	2	6
4	4	3	3
5	1	1	10
5	1	3	15
5	2	2	7
5	2	3	14
5	3	1	5
5	3	2	8
5	4	1	4
5	4	2	6
5	4	3	8

4.2 Example 2

Now the example of scheduling an FMS is taken where set-up times are involved. The system consists of two jobs and two machines with two operations on each job. Here an adequate tool is required for the operation to be performed on a machine. As per the convenience of a tool to be fitted to a machine, the processing time needed for a machine for an operation of a job varies with the tool attached to the machine. For the concerned problem four tools are employed. Moreover, attachment and detachment times of the tool on a machine are given in Table 2 and the respective processing times are provided in Table 3. The representation method for the processing time adopted here is $p_{j,o,m,t}$, where p refers to the processing time of operation o of job j on machine m with tool t.

After carrying out extensive computational experiments the value of the system parameters, such as α, β and ρ, have been tuned. The procedure to determine these parameters is given in detail in the next subsection. The final outcome in the form of makespan at different values of iterations is plotted and is given later in Fig. 6. The finding obtained here utilizes the same settings as that used by Yim and Lee [**37**], where they have used a timed transition PN model for the FMS scheduling problem. The next subsection deals with the system performance using the system parameters.

4.3 Study of system performance

1. The first parameter to be studied is Q. Various values of Q were tested in the range 25–2000. It can be seen from repeated results that the value of Q does not affect the ability to find the near-optimal solution for FMS scheduling. However, for all Q values, as the number of iterations increases, the number of ants selecting the best possible result also increases.
2. The next parameter to be studied is α. It should be noted that α is the parameter related to the τ_{il} (pheromone trail) value, which shows the dependency on the previous pheromone trail. The effect of α values, ranging from 0.25 to 10, was tested on the CPU time. It can be seen from Table 4 that as the number of iterations increases, the CPU time considerably increases. However, for increasing

Table 2 Attachment and detachment times (in s) for operations of jobs on machines with particular tools of Example 2

	Job							
	J1				J2			
	M1		M3		M1			M2
	T2	T4	T1	T3	T1	T2	T4	T4
First operation	8	4	10	15	15	6	9	3
	M1		M3		M2		M3	
	T1	T4	T2	T3	T2	T4	T1	T3
Second operation	7	5	10	12	4	12	8	5

Table 3 Processing time (in s) for Example 2

Machine	Tool			
	T1	T2	T3	T4
M1	7–3	10–6	–	5–9
M2	–	12–6	–	8–10
M3	3–7	6–9	10–5	–

Table 4 Variation of CPU time (in ms) with varying values of α for various values of the number of iterations

Number of iterations	α						
	0.25	0.5	0.75	1.0	2	5	10
50	3	4	4	3	2	2	2
100	5	6	7	5	4	4	4
200	11	12	13	8	8	8	8
500	14	29	30	18	18	20	22
1000	20	42	44	40	39	39	40

values of α, the CPU time increases for α values in the range 0.25–1, after which it shows a stagnated reading. Thus, for higher α values it can be said that the previous trail becomes constant.

3. Now the combined effect of α and β on the system performance is considered. It can be seen from equation (4) that β is the exponent of η_{il} and thus the combined influence of η_{il}^{β} is very important in analysing the role played by it. Let $\rho = 0.5$, $Q = 100$ and the number of iterations be 1000. From Table 5 it can be inferred that, for constant β values, the CPU time increases from 0 to 0.5 and then decreases continuously with almost a constant value. Therefore, it can be concluded that as the dependence on the heuristic value η_{il} increases, the CPU time decreases from $\beta = 0.5$ to $\beta = 1$ and thereafter has an almost constant value. Hence it can be said that system performance is optimal for α in the range 1–2 and for β in the range of 3–5. Thus, dependency on the node selection is more on the heuristic value, as seen in the application of this case.

Table 5 Variation of CPU time (in ms) for corresponding variations in values of α and β

α	β				
	0.5	1	2	3	5
0	10	8	8	7	7
0.5	16	13	12	12	12
1	12	9	9	8	9
2	11	9	9	8	9
3	11	9	9	9	9
5	11	9	9	9	10

4. One of the most important system parameters is ρ. It shows the evaporation rate of the pheromone trail. From Fig. 2 it can be seen that as the value of ρ increases from 0 to 0.5 the CPU time increases and then decreases for ρ values in the range 0.5–1. Thus, it can be inferred that as the value of ρ increases, the CPU time also varies, as shown in Fig. 2.
5. The next observation is for the number of iterations with the mean makespan and CPU time. Initially the value of the mean makespan is very high but progressively stabilizes to a constant value (Fig. 3). It is observed that the mean makespan is quite high for a lower number of iterations, but stabilizes to a constant value as the number of iterations increases.

It can also be observed that as the number of iterations increases, the CPU time for obtaining the near-optimal schedule also increases (Fig. 4). This difference is high when the number of iterations is large. Thus an optimal value of the number of iterations should be chosen such that neither the makespan found is stabilized nor the CPU time is substantially high. It was observed from repeated runs that the number of ants choosing the best path considerably increases when the number of iterations is less, but for an increased number of iterations the fraction of ants choosing the best path stabilizes to about 0.60 (Fig. 5).

The way the algorithm converges to its near-optimal solution is an interesting observation. Referring to Fig. 6, the ants finding solutions towards the

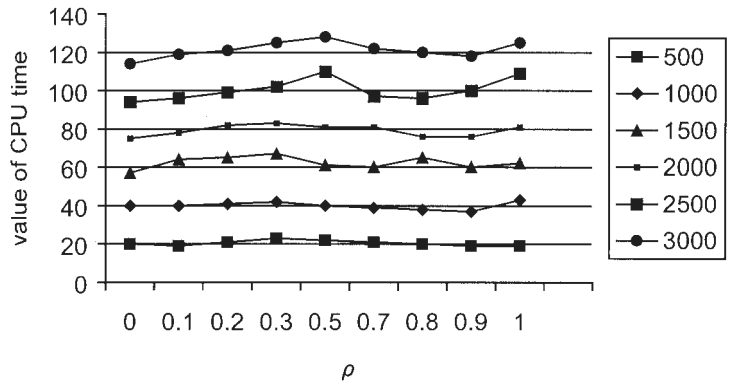

Fig. 2 Variation of CPU time (in ms) with varying iterations and for corresponding values of ρ

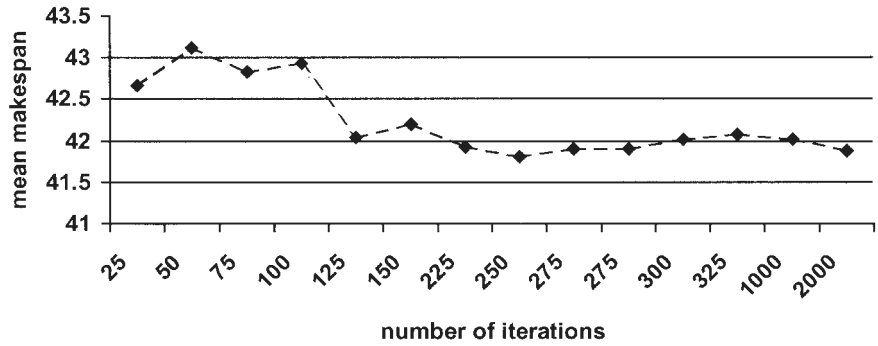

Fig. 3 Mean makespan values for the corresponding number of iterations

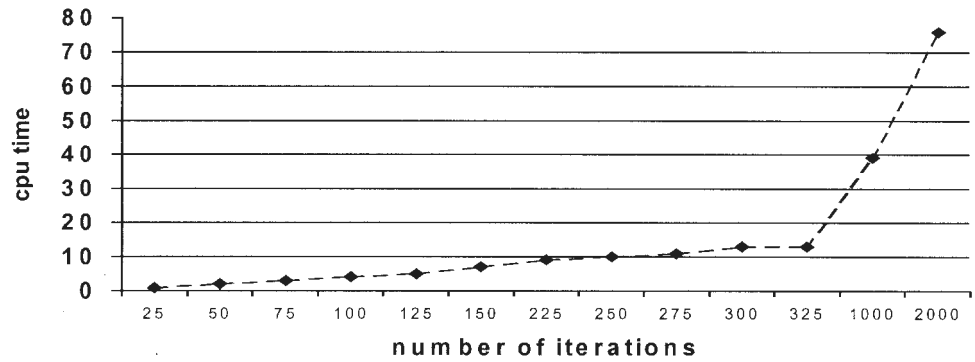

Fig. 4 Variation of CPU time (in ms) with the number of iterations

beginning of the procedure are the ones with more makespan, but gradually the ants try to follow the shortest path. Thus, the solution progressively moves towards the near-optimal solution. After about 500 iterations the ants find almost similar paths, but still the chances of finding newer paths is not nullified.

6. Next the system behaviour with node branching is considered. It can be seen that as the number of iterations increases the solution comes closer to better results. The node branching decreases with the number of iterations (Fig. 7). Also, the makespan values obtained for particular iterations and the corresponding CPU times are shown in Fig. 8. From the figure it can be seen that as the CPU time increases, it is possible to obtain better results with still lower makespan values.

5 CONCLUSION

The paper presents an effective method of scheduling jobs in a flexible manufacturing system through the use of an ant colony optimization technique. It is based on an imitation of the foraging behaviour of real ants. A substance, pheromone, is deposited by natural ants

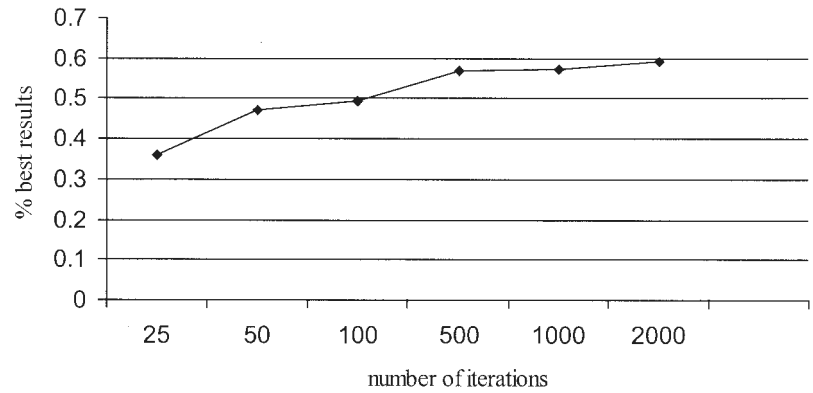

Fig. 5 Percentage of best results obtained with the number of iterations

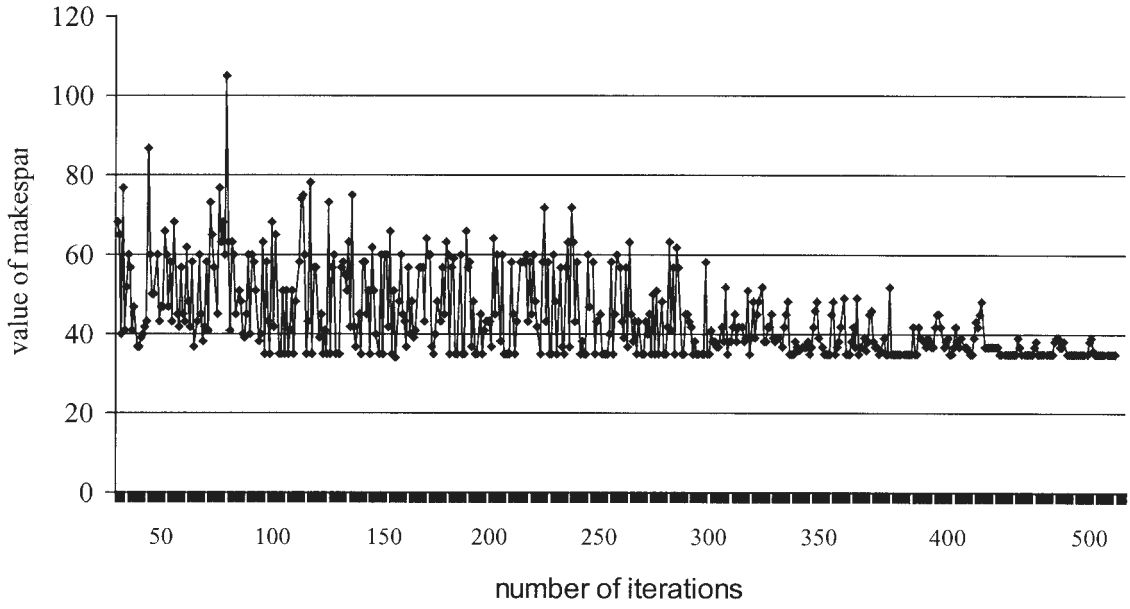

Fig. 6 Typical values of makespan for the FMS scheduling problem with the proposed ant algorithm

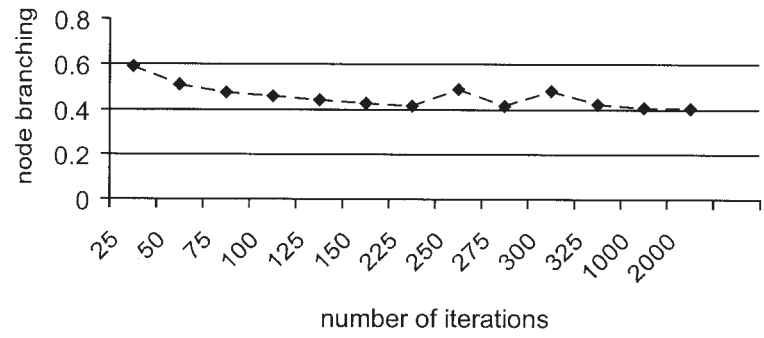

Fig. 7 Percentage of node branching with the number of iterations

after moving through a path. A similar pheromone trail is developed for the scheduling procedure, wherein a directed graph-based approach is used to represent the whole process. Ants move from one node to another node and progressively move towards the final node. Two illustrative problems with varying computational complexities have been considered for analysis and it is observed that the proposed algorithm is effective and robust in dealing with the FMS scheduling problem. For further exploration, a non-graph-based approach can be attempted through the proposed algorithm. Using different rules pertaining to the colonial relationship, pheromone updating can be explored to enhance the capability of the proposed algorithm.

There are several instances in the literature where PN-based models are accepted for solving the scheduling

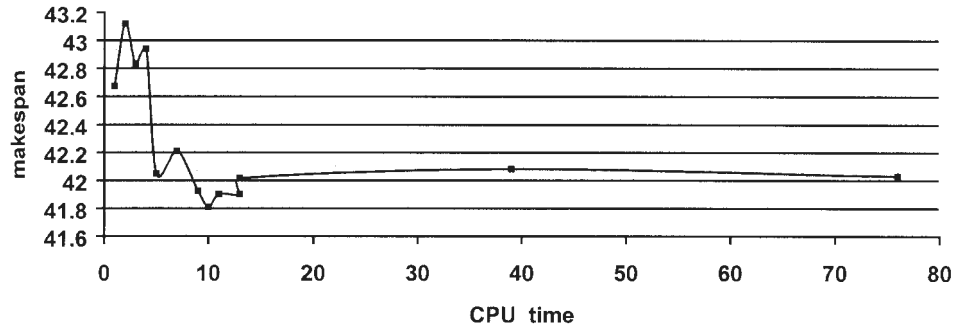

Fig. 8 Obtained CPU time (in ms) for corresponding values of makespan

problem of FMS. Similarly, there is an abundance of literature where several techniques like the branch and bound, tabu search, genetic algorithm, etc., are employed to resolve practical scheduling problems of the FMS. It is suggested that with a few modifications to suit the problem environment, the proposed algorithm can be used to solve a wide range of problems.

ACKNOWLEDGEMENTS

The authors are grateful to the anonymous referees for their constructive comments, which have helped to improve the paper over its earlier version.

REFERENCES

1 **Li, D. C.** and **She, I. S.** Using unsupervised learning technologies to induce scheduling knowledge for FMSs. *Int. J. Prod. Res.*, 1994, **32**(9), 2187–2199.
2 **Shinich, N.** and **Taketoshi, Y.** Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool. *Int. J. Prod. Res.*, 1992, **30**, 411–413.
3 **Haddock, J.** and **O'Keefe, R. M.** Using artificial intelligence to facilitate manufacturing systems simulation. *Computers and Ind. Engng*, 1990, **18**(3), 275–283.
4 **Stecke, K. E.** Design, scheduling and control problems of flexible manufacturing systems. *Ann. Op. Res.*, 1995, **3**, 3–12.
5 **Andreatta, G., Deserti, I.** and **Giraldo, L. N.** Scheduling algorithm for a two machine flexible manufacturing system. *Int. J. Flexible Mfg Systems*, 1995, **7**, 207–227.
6 **Saruncuoglu, L.** and **Karabuk, S.** A beam search-based algorithm and evaluation of scheduling approaches for flexible manufacturing systems. *IIE Trans.*, 1999, **30**, 179–191.
7 **Jeong, K. C.** and **Kim, Y. D.** A real time scheduling mechanism for a flexible manufacturing system: using simulation and dispatch rules. *Int. J. Prod. Res.*, 1998, **36**(9), 2609–2626.
8 **Yih Y.** and **Thesen, A.** Semi-Markov decision models for real-time scheduling. *Int. J. Prod. Res.*, 1991, **29**, 2331–2346.
9 **Thesen, A.** and **Lei, L.** An expert scheduling system for material handling hoists. *J. Mfg Systems*, 1990, **9**, 247–252.
10 **Shaw, M. J.** A pattern-directed approach to flexible manufacturing: a framework for intelligent scheduling, learning, and control. *Int. J. Flexible Mfg Systems*, 1989, **2**, 121–144.
11 **Liang, T. P., Moskowitz, H.** and **Yih, Y.** Intelligent neural networks and semi-Markov process for automated knowledge acquisition: a real time scheduling. *Decision Sci.*, 1992, **23**, 1297–1341.
12 **Chryssolouris, G., Lee, M.** and **Domroese, M.** The use of neural networks in determining operational policies for manufacturing systems. *J. Mfg Systems*, 1991, **10**, 166–175.
13 **Talvage, J. J.** and **Shodhan, R.** Automated development of design and control strategy for FMS. *Int. J. Computer Integrated Mfg*, 1992, **5**, 335–348.
14 **Bengu, G.** A simulation-based scheduler for flexible flow lines. *Int. J. Prod. Res.*, 1994, **32**, 321–344.
15 **Kim, M. H.** and **Kim, Y. D.** Simulation based real time scheduling mechanism in a flexible manufacturing system. *J. Mfg Systems*, 1994, **13**, 85–93.
16 **Duffie, N. A.** and **Prabhu, V. V.** Real time distributed scheduling of heterarchical manufacturing systems. *J. Mfg Systems*, 1994, **13**, 94–107.
17 **Ishii, N.** and **Muraki, M.** An extended dispatching rule approach in an online scheduling framework for the batch process management. *Int. J. Prod. Res.*, 1996, **34**, 329–348.
18 **Marie, P. J.** A class of Petri nets for manufacturing system integration. *IEEE Trans. Robotics and Automn*, 1997, **13**, 317–326.
19 **Naiqi, W.** Necessary and sufficient conditions for deadlock free operation in flexible manufacturing systems using a colored Petri net model. *IEEE Trans. Systems, Man and Cybernetics—Part C: Applications and Reviews*, 1999, **29**(2), 192–204.
20 **Kim, Y. W., Inaba, A., Suzuki, T.** and **Okuma, S.** FMS scheduling based on timed Petri net model and RTA algorithm. In Proceedings of the 2001 IEEE International Conference on *Robotics and Automation*, Seoul, Korea, 21–26 May 2001, pp. 848–853.
21 **Dorigo, M., Maniezzo, V.** and **Colorni, A.** Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politechnico di Milano, Italy, 1991.
22 **Dorigo, M., Maniezzo, V.** and **Colorni, A.** The ant systems: optimization by a colony of cooperative agents. *IEEE Trans. Man, Machine and Cybernetics—Part B*, 1996, **26**(1), 29–41.
23 **Gambardella, L. M.** and **Dorigo, M.** Ant-Q: a reinforcement learning approach to the travelling salesman problem. In Proceedings of the Twelfth International Conference on *Machine Learning*, ML-95, 1995, pp. 252–260.
24 **Dorigo, M.** and **Gambardella, L. M.** Ant colonies for the travelling salesman problem. *Bio Systems*, 1997, **43**, 73–81.
25 **Dorigo, M.** and **Gambardella, L. M.** Ant colony systems: a co-operative learning approach to the travelling salesman problem. *IEEE Trans. Evolutionary Computation*, 1997, **1**(1), 53–66.
26 **Gambardella, L. M.** and **Dorigo, M.** Solving symmetric and asymmetric TSPs by ant colonies. In Proceedings of the IEEE Conference on *The Evolutionary Computation*, ICE96, 1996, pp. 622–627.
27 **Stutzle, T.** and **Hoos, H.** The MAX–MIN ant systems and local search for the travelling salesman problem. In Proceedings of IEEE–ICEC–EPS, IEEE International Conference on *Evolutionary Computation and Evolutionary Programming*, 1997, pp. 309–314.
28 **Stutzle, T.** and **Hoos, H.** Improvements on the ant system: introducing MAX–MIN ant system. In Proceedings of the International Conference on *Artificial Neural Networks and Genetic Algorithms*, 1997, pp. 245–249.
29 **Bullnheimer, B., Hartl, R. F.** and **Strauss, C.** A new rank-based version of the ant system: a computational study. Technical Report POM-03/97, Institute of Management Science, University of Vienna. In *Central European Journal for Operation Research and Economics*, 1997.
30 **Gambardella, L. M., Taillard, E. D.** and **Dorigo, M.** Ant colonies for the QAP. *J. Opl Res. Soc.*, **50**, 167–176.
31 **Stutzle, T.** and **Hoos, H.** MAX–MIN ant systems and local search for the combinatorial optimization problems.

In *Advances and Trends in the Local Search Paradigms for the Optimization* (Eds I. H. Osama, S. Martello and C. Roucariol), 1998, 137–154.

32 **Maniezzo, V.** and **Colorni, A.** The ant systems applied to the quadratic assignment problem. *IEEE Trans. Knowledge and Data Engng*, 1999, **11**(50), 769–778.

33 **Maniezzo, V., Colorni, A.** and **Dorigo, M.** The ant systems applied to the quadratic assignment problem. Technical Report IRIDIA, University Libre de Bruxelles, Belgium, 1994, pp. 28–94.

34 **Colorni, A., Dorigo, M., Maniezzo, V.** and **Trubian, M.** Ant system for job-shop scheduling. *Belgian J. Op. Res., Statistics and Computer Sci.*, 1994, **34**, 39–53.

35 **Bullnheimer, B., Hartl, R. F.** and **Strauss, C.** Applying the ant systems to the vehicle routing problem. In *Advances and Trends in the Local Search Paradigms for Optimization* (Eds I. H. Osama, S. Martello and C. Roucariol), 1998, pp. 109–120.

36 **Lee, D. Y.** and **Dicesare, F.** FMS scheduling using Petri nets and heuristic search. *IEEE Trans. Robotics and Automn*, 1994, **10**(2), 123–132.

37 **Yim, S. J.** and **Lee, D. Y.** Scheduling method with the consideration of machine setup in flexible manufacturing systems. In Proceedings of IEEE International Conference on *Robotics and Automation*, Albuquerque, New Mexico, April 1997, pp. 2735–2740.