International Journal of Innovative Research and Advanced Studies (IJIRAS)
Volume 3 Issue 13, December 2016

ISSN: 2394-4404

Improvement Of Ant Colony System By Using Pheromone Updation

B. Sasikala

Assistant Professor, Dept of Computer Science,
Bharathidasan University Constituent College (W),
Orathanadu, Thanjavur District

Dr. V. P. Eswaramurthy

Assistant Professor, Dept of Computer Applications,
Government Arts and Science College, Komarapalayam,
Namakkal District

Abstract: ***Scheduling theory is one of the significant research areas in operations research. It has been the subject of research with techniques ranging from simple dispatching rules to sophisticated learning algorithms. Effective scheduling can improve production systems and the utilization of resources. The job shop problem is under this category and is combinatorial in nature. Research on optimization of job shop problem is one of the most significant and promising areas of optimization. The nature has inspired several metaheuristics, outstanding among these is ant colony optimization (ACO), which have proved to be very effective and efficient in problems of high complexity (Np-hard) in combinatorial optimization. This paper presents an application of the ant colony optimization metaheuristic to job shop problem. The inspiring source of ant colony optimization is pheromone trail laying and following behavior of real ant. The methods of updating the pheromone have more influence in solving instances of job shop problem. An algorithm is introduced to improve the basic ant colony system by using a pheromone updating strategy.***

Keywords: Job Shop Scheduling, Ant Colony Optimization, Meta-Heuristic, Combinatorial Optimization, Makespan

I. INTRODUCTION

General job shop problem is the probably most studied one by academic research during the last three decades and is notoriously difficult problem to solve. The JSSP is an NP (Nondeterministic Polynomial) hard problem and among those optimization problems, it is one of the least tractable known problem [1]. It is purely deterministic, since processing time and constraints are fixed and no stochastic events occur. Each operation has its own processing time and has to be processed on a dedicated machine. Given a nuber of ‘n’ jobs, the jobs have to be processed on ‘m’ machines. Each job has its own machine order and no relation exists between machine orders of any two jobs. For each job, the machine order of operations is prescribed and is known as technological production recipe or technological constraints, which are static to a problem instance. A feasible schedule is a sequence of operations on a machine ensuring the job shop constraints. A makespan is defined as the maximum completion time of the jobs. The objective of the JSSP is to find a feasible schedule with minimum makespan.

A recent adaptive algorithm, named Ant System, is introduced and used to solve the problem of jobshop scheduling. The algorithm was first introduced by Dorigo, Maniezzo and Colorni in 1991[2] and is derived from the foraging and recruiting behaviour observed in an ant colony. It can be applied to combinatorial optimisation problems. The study from biology of an ant colony shows that its behaviour is highly structured. Knowing that a single ant has limited capacities (i.e., a single ant is not capable of communicating directly with other ants about past experiences), it is curious to know how the ants co-operate so as to achieve such a complex and organised behaviour of the whole colony. Maybe one of the most studied cooperation phenomena among ants is the so-called foraging and recruiting behaviour [3][4][5][6]. This behaviour describes how ants explore the world in search of food sources, then find their way back to the nest and indicate the food source to the other ants of the colony. To do so, ants use an indirect way to communicate through tracks of pheromone, a chemical substance that they can deposit and are attracted to. Each ant, upon finding a food source, deposit's fractions of pheromone on the way back to the nest so as to indicate the source to the others. The accumulated pheromone serves as a distributed memory, shared by all the other ants and marks in terms of probability the most visited paths between the nest and the food source. The ACO algorithm lies in the loop, where in every iteration, each ant moves from a state to another one corresponding to a more complete partial solution.

The goal of this work is to develop an algorithm which uses modified pheromone update rule called stair pheromone update rule so that good quality solutions will be obtained. This paper is structured as follows. In Section 2, JSSP is explained and is formally described. In Section 3, ACO algorithm is described. In Section 4, literature review is given. In Section 5, Implementation of ACO to JSSP is given with the algorithm which uses proposed updating strategy. In Section 6 experimental results and discussion of variants of the proposed methods are given and finally in Section 7, the conclusion is given.

II. JOB SHOP SCHEDULING

A job-shop is the set of all the machines that are identified with a particular set of operations. The process consists of the machine, the jobs (operations) and a statement of the disciplines that restrict the manner in which operations can be assigned to specific points on the time scale of the appropriate machine. The job shop scheduling problem consists of set of jobs $J = \{1 \ldots n\}$, a set of machines $M = \{1 \ldots m\}$ and a set of operations $V=\{O_{ij} \mid i\in[1,n], j\in[1,m]\}$ where n and m are total number of jobs and machines respectively in a job shop. On these machines $M_1, M_2, \ldots M_m$, the jobs $J_1, J_2 \ldots J_n$ are to be scheduled. Each job J_i has set of operations O_{i1}, O_{i2}, $O_{i3} \ldots O_{ik}$, where k is a total number of operations in the job J_i. The operations precedence constraints are associated with each job and ensure that the operation O_{ij} will be processed only after the processing of operation O_{ij-1} (for $j > 0$) in a particular job i. Generally the standard model of n jobs and m machines job shop is denoted by $n/m/G/C_{max}$. The parameter G is the technological matrix denoting the processing order of machines for different jobs. An example of the technological matrix G can be represented as follows:

$$G = \begin{bmatrix} M_2 & M_3 & M_1 \\ M_1 & M_2 & M_3 \\ M_3 & M_1 & M_2 \end{bmatrix}$$

Each row represents a job. For example for the first job, first operation is performed on machine M_2, the second operation is performed on machine M_3 and third operation is performed on machine M_1. Like this, other jobs are executed on different machines as denoted in G. The processing time of the operations is represented as follows:

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix}$$

Hence P_{ij} represents the time of j^{th} operation of i^{th} job. C_{max} is the makespan representing total completion time of all the operations. The processing of operation i on machine j is denoted by O_{ij} and the completion time of O_{ij} in the relation $o_{ik} \rightarrow o_{ij}$. The completion time of all C_{ij} will be found. The main aim is to minimize C_{max} value. Then C_{max} is calculated as follows:

$$C_{max} = \max_{\text{all } o_{ij} \in V} (C_{ij}) \qquad (1)$$

III. ANT COLONY OPTIMIZATION

Ant colony optimization (ACO) takes inspiration from the foraging behavior of some ant species. These ants deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. Ant colony optimization exploits a similar mechanism for solving optimization problems.

A. STATE TRANSITION RULE

The ant algorithm can be represented into a graph, in which the ants move along every branch from one node to another node and so construct paths representing solutions. Starting in an initial node, every ant chooses the next node in its path according to the state transition rule by using probability of transition. Let S be the set of nodes at decision point i. The transition probability for choosing the edge from node i to node j by the ant k at the time t is calculated as in equation (2). τ(i. j) is the quantity of the pheromone on the edge between node i and node j. η(j) is the inverse of the operation time of the node j. α and β tune the relative importance in probability of the amount of the pheromone versus the heuristic distance.

$$P_k(i, j) = \begin{cases} \dfrac{[\tau(i, j)]^{\alpha} . [\eta(j)]^{\beta}}{\sum_{j\in S} [\tau(i,j)]^{\alpha} . [\eta(j)]^{\beta}} & \text{if } j\in S \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

According to this rule, the artificial ants move to the node that has a higher amount of pheromone and lesser value of operation time and will have a higher probability to be scheduled in the practical solution. The priority rule is set in the form of the heuristic distance. The heuristic distance is equal to the processing time of the particular operation. In this paper, the shortest processing time priority rule is selected and the sequence of the job can be found based on the ant's sequence of visited nodes.

B. PHEROMONE UPDATING RULE

A certain amount of pheromone is dropped when an ant goes by. The pheromone values of all the edges are modified during the construction of a solution. A certain amount of pheromone is evaporated or deposited, when an ant travel through the edges. i.e. pheromone on each edge belonging to

ISSN: 2394-4404

one of the computed solutions is modified by an amount of pheromone proportional to its solution value. There are two kinds of pheromone update strategies called local updating rule and the global updating rule.

- ✓ *LOCAL UPDATING RULE:* Local updating rule is indented to avoid a strong edge being chosen by all the ants. While constructing its path, an ant will modify the amount pheromone on the passed edge (i, j) by applying the local updating rule.

$$\tau(i, j) = (1- \rho).\tau(i, j) + \rho.\ \tau_0/TL \qquad (3)$$

where ρ is the coefficient representing pheromone evaporation (Note: $0 < \rho < 1$). τ_0 is the initial pheromone value. TL is the length by the current ant's path. This local updating rule is used to modify the pheromone trail on the edges at each time the ant travels through these edges according to equation (3). The local updating rule is equivalent to the trail evaporation in real ants.

- ✓ *GLOBAL UPDATING RULE:* Global updating is indented to reward edges belonging to shortest path. Once all ants have arrived at their destination, the amount of pheromone on the edge (i, j) belonging to the shortest path is modified again by applying the global updating rule given in equation (4).

$$\tau(i, j) = \tau(i, j) + (1-\rho).q/L_k \qquad (4)$$

q is a value chosen randomly with uniform probability in [0,1]. L_k is the minimum tour length. The amount of pheromone deposited on each edge is inversely proportional to the length of the path so as to enable the shorter path to get higher amount of pheromone deposited on edges.

IV. LITERATURE REVIEW

Among the modern metaheuristic-based algorithms, a significant interest has been arisen in applying ACO and AIS algorithms to solve the complex combinatorial optimization problems. ACO is a promising metaheuristic algorithm that a pioneering work relevant to it was done by Dorigo et al. [7]. In the literature, there are only few reviews dealing with ACO algorithm in the scheduling problems. Since the development of ACO, researchers have applied it to a number of combinatorial optimization problems particularly to scheduling. Some of these include: the sequential ordering, job shop scheduling, flow shop scheduling, vehicle routing, bus driver scheduling and tardiness scheduling problems. In most of these cases, the scheduling problem at hand is reduced to a TSP like problem in which, the problem is to find some optimal path through a graph of some sort.

Alberto Colorni et al. [8] presented how ant system works for Job Shop Scheduling problem. Different priority rules are discussed and the longest remaining processing time (LRT) is applied in the problem. This work deals how the ant passes through all the nodes based on the pheromone values and the experiments on different problem instances are carried out. Sjoerd van der Zwaan et al. [9] presented how the Ant Colony Optimization is used to solve the problem of Job Shop Scheduling. Statistic analysis for parameter tuning is also presented and the obtained solutions are compared with benchmark problems in job shop scheduling. Christian Blum et al [10] proposed MAX-MIN ant system in the hyper-cube framework approach to tackle the broad class of group shop scheduling problem instances. It probabilistically constructs solutions using the non delay guidance which employs black-box local search procedures for improving the constructed solutions. Ventresca et al. [11] presented an ant colony optimization algorithm approach to the job shop-scheduling problem. The main goal was to examine pheromone-updating techniques and their effects on solution space exploration. A rudimentary ant system was examined along with a Max-Min ant system approach. A new technique called foot stepping was proposed that improves the solution quality of the ant system. The neighborhood is an adaptation of the successful neighborhood derived by Nowicki and Smutnicki for the JSS problem. Jun Zhang et al [12] implanted an ant colony optimization technique for job scheduling problem which compares iterations and scheduling time by setting different values for parameter influencing importance of the pheromone trial and heuristic distance. The results have shown that the performance of the ACS for the JSSP largely depends on the parameter values and the number of the ants. Adjusting these parameter values takes a great deal of time, the optimal parameter values depend on the problem to be solved and is difficult to find an all purpose setting of parameters for all problems.

V. IMPLEMENTATION

A. STAIR PHEROMONE UPDATION

The aim of the pheromone update is to increase the pheromone values associated with good solutions. There are many number of rules to implement the updating of the pheromone trails. One variant of this rule called "stair pheromone updating rule" is introduced and this proposed work reduces the computational effort and difficulty of the algorithm by removing the application of variation parameter method used in various research works. There are three ways for applying this strategy. They are

- ✓ Stair Local Pheromone Updating Rule (SLPUR)
- ✓ Stair Global Pheromone Updating Rule (SGPUR)
- ✓ Stair Series Pheromone Updating Rule (SSPUR)
- ✓ Stair Concurrent Pheromone Updating Rule (SCPUR)

In stair pheromone updating rule, interval value is given and this value is used to make a decision whether the pheromone updating rule is applied during the execution of a cycle. The local and global pheromones are updated as denoted in equation (3) and (4) respectively. In SLPUR, for the first interval number of cycles, the local pheromone updating is applied and the next interval number of cycles, the local pheromone updating is not applied and vice versa. Similarly in SGPUR, for the first interval number of cycles, the global pheromone updating is applied and the next interval number of cycles, those updating is not applied and vice versa. In both cases this will be repeated interchangeably till the termination condition is met. In SSPUR, for the first interval local updating is applied and for the next interval global updating is applied and interchangeably these two updating rules are applied. In SCPUR, for the the first interval both local updating and global updating rules are applied and for the next interval both updating rules are not applied. The interval value affects the quality of the solution. This value can be tuned like other parameters α, β and ρ. Figure 1 shows the pseudo code for stair pheromone updating strategy.

Algorithm: StairUpdate( )

```
    Parameter: τ0, τ(i ,j), ρ, Lk, TL, q, slpur, sgpur, sspur,
scpur, stairinterval, stair
Step 1: Turnover        // Turn over the stair value//
            If cycno % stairinterval = 0
                If stair = true Then
                    stair = false
                Else
                    stair = true
                End If
            End If
Step 2: Local           // Local pheromone updating//
          If slpur = true Then
            If stair = true Then
              τ(i, j) = (1- ρ)τ(i, j) + ρ. τ0/TL       //Apply
local pheromone updating rule
            End If
          End If
Step 3: Global          // Global pheromone updating//
          If sgpur = true Then
            If stair = true Then
              τ(i, j) = τ(i, j) + (1-ρ).q/Lk           //Apply
global pheromone updating rule
            End If
          End If
Step 4: Series          // Series pheromone updating//
          If sspur = true Then
            If stair = true Then
              τ(i, j) = (1- ρ)τ(i, j) + ρ. τ0/TL       //Apply
local pheromone updating rule
            Else
              τ(i, j) = τ(i, j) + (1-ρ).q/Lk           //Apply
global pheromone updating rule
            End If
          End If
Step 5: Concurrent      // Concurrent pheromone updating//
          If scpur = true Then
            If stair = true Then
              τ(i, j) = (1- ρ)τ(i, j) + ρ. τ0/TL       //Apply
local pheromone updating rule
              T(i, j) = τ(i, j) + (1-ρ).q/Lk           //Apply
global pheromone updating rule
            End If
          End If
```

Figure 1: Pseudo algorithm for the stair pheromone updating rule

The algorithm uses initial pheromone value τ_0, the pheromone $\tau(i,j)$ for the edge (i, j), evaporation coefficient ρ, the best travel around length of overall ants L_k, the travel around length of the current ant's path TL, a value q chosen randomly with uniform probability in [0,1], boolean variables slpur, sgpur, sspur, scpur for pheromone updating rules, interval value stairinterval to decide the updation and a boolean variable stair to flip the updation.

B. CONSTRUCTING THE SOLUTION

To construct the solution, the parameters α, β and ρ values are assigned by some input values. These parameters can be tuned to produce the better results. All nodes are represented by using the following matrix Z:

$$Z = \begin{pmatrix} z_{11}, z_{12} \ \dots \ z_{1m} \\ z_{21}, z_{22}, \dots \ z_{2m} \\ \dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots \\ z_{n1}, z_{n2} \ \dots \ z_{nm} \end{pmatrix}$$

The matrix *Z* has one to one correspondence with technological matrix *G* and operation time matrix P. Every node in the disjunctive graph is also represented by using *Z*.

The initial pheromone value τ_0 is calculated as

$$\tau_0 = \frac{1}{n.L} \qquad (5)$$

n is the number of jobs. L is the average of minimum and maximum operation time. The pheromone for all edges ($\tau(i, j)$ for all i and j) are assigned by the value τ_0. Let *S* be the set of allowed nodes at particular decision point i. Initially *S* is calculated as

$$S = \{ z(i, j)/z(i, j) \in Z, i \in (1..n), j=1 \} \qquad (6)$$

Hence *S* has nodes corresponding to the first operation of all the *n* jobs. Let T be the set of visited nodes and initially T is empty. α and β represent the relative importance in probability of the amount of the pheromone and the operation time respectively. The probability for choosing the edge from decision point i to node j by the k^th ant at the time t(initially t =0) is calculated by the following state transition rule.

$$P_{ij}^{k}(t=0) = \begin{cases} \dfrac{(\tau_0)^{\alpha} (\eta(i, j))^{\beta}}{\sum_{z(i,j) \in S} (\tau_0)^{\alpha} (\eta(i, j))^{\beta}} & \text{If } j \in S \\ 0 & \text{Otherwiswe} \end{cases} \qquad (7)$$

Hence this probability is the function of pheromone on the edges and operation time. As already noted in equation (2), $\eta(i, j)$ is inverse of the processing time of the operation denoted by the node z(i, j) in S. Using this probability value Monte Carlo probability is calculated and a random number r is generated between 0 and 1. According to the values of Monte Carlo probability and the random number, a particular node z(i, j) is visited and this node is added to the set T which contains the visited nodes and this node n(i, j) is also removed from the set S. Now the new node z(i, j+1) is added to the set S. The node z(i, j+1) is produced by the conjunctive arc of the graph D. Like this, all the nodes in the set S are being visited one by one and added to the set T. If there cannot be a node produced by the conjunctive arc of the particular job (for z(i, j), z(i, j+1)=null), the new node could not be added to set S. Hence at this time the set S has has the allowed nodes corresponding to the n-1 number of jobs. Generally after completion of q number of jobs in a job shop, S has n-q number of nodes. Finally the set T has all the nodes visited and set S would be empty. The set of nodes in T could form a path from the start to the end. The makespan value is calculated for this k^{th} ($1 \leq k \leq$ total ants) ant and is stored in the memory. If all the ants complete a tour, one cycle is completed. Let L_k be the minimum makespan found. The new pheromone value for all the edges are assigned by

$$\tau(i, j) = 1/(1-\rho)*L_k \quad (8)$$

for all i and j values. Hence all the edges receive this pheromone value. This is the initial solution found.

The constructed solution is improved using the algorithm which uses stair pheromone updating rule. The stair pheromone update strategy rule mentioned in the section 5 is applied on the current solution. i.e. For local updating, the pheromone for all the edges in different paths of the ants is calculated as given in equation (3) and for global updating, the pheromone for all edges in ant's path having lowest makespan is calculated as given in equation (4). These updating are based on the stair interval value. The different edges receive the different pheromone value now. Figure 2 shows the pseudo code for the algorithm using the "Stair pheromone updating rule" which allows to guide the search more into sub-optimal regions of the search space without loosing the algorithm's capability of recovering from dead-ends by imposing to explore other directions in the search space to improve the constructed solution.

The different parameters are given for the algorithm. MAXNODES represents the maximum number of operations in the problem. HIGH is used to initialize the C_{max} which would be improved in each iteration. L represents average of minimum and maximum operation time in the job shop. MAXANTS is used to store the maximum number of ants. T and MinT are used to store the current ant's path and the best ant's path respectively. The new solution is constructed by using the set S, the set T and procedures mentioned in the construction of initial solution. The state transition rule given in equation (2) is used to improve the constructed solution. This process is iterated until a termination criterion is met. The termination criterion is either the concentration of all the ants in the same path or reaching the maximum number of cycles.

Algorithm: ConstructAndImprove( )

Parameters: n, m, α, β, ρ, τ_0 $\tau(i, j)$, $\eta(i, j)$, N, MAXNODES, HIGH, L, MAXANTS, T, MinT, S

Step 1: //Initialization//
    $\tau_0 = 1/(n.L)$, C_{max} = HIGH, CYCNO = 0
    MinT = empty with MAXNODES positions,
Step 2: //Constructing and Improving the Solution//
    For k = 1 to MAXANTS do
        T = null   // T is empty at the starting point of each ant k
        For i = 1 to n do
            Add z[i, j] to S
            j[i]=1
        End For
        FinalVisited = 0
        For i = 0 to MAXNODES+1 do
            b = 0, v = FinalVisited
            For q = 1 to n do
                $a[q] = \tau(v, z[q, j[q]])^{\alpha} . \eta(q, j[q])^{\beta}$ //If CYCNO = 0, $\tau(v, n[q, j[q]]) = \tau_0$
                b = b + a[q]
            End For
            For i = 1 to n do
                P[q, j[q]] = a[q] / b   //Apply state transition rule
            End For
            Choose a node z[q, j[q]] from S with Monte Carlo Probability using
                P[q, j[q]] for q = 1..n and assign this node to FinalVisited
            Add z[q, j[q]] to T
            Delete the node z[q, j[q]] from S
            If z[q, j[q]+1] is successor of z[q, j[q]]
                j[q]=j[q]+1, Add the node z[q, j[q]] to S
            End If
        End For
        MAKESPAN = Time to complete the tour by k^{th} ant
        If C_{max} > MAKESPAN Then
            Copy T into MinT, C_{max} = MAKESPAN
        End If
        If CYCNO > 0 Then
            Goto Step 2 of StairUpdate( )
        End If
    End For
Step 3: //New Pheromone Calculation after the constructing of initial solution//
    If CYCNO = 0 Then   //New pheromone after the initial solution only
        For i = 0 to n do
            For j = 0 to m do
                $\tau(i, j)=1/((1-\rho)*C_{max}$ //New pheromone calculation
            End For
        End For
    End If
Step 4: //Global pheromone updation//
    If CYCNO > 0 Then
        Goto Step 3 of StairUpdate( )
    End If
Step 5: //Checking the end of execution//
    CYCNO = CYCNO + 1
    If Termination_Criterion = true Then
        For i = 1 to MAXNODES do
            Print MinT[i]   // Print the minimum path
        End For
        Print C_{max} // Print the best visit length
    Else
        Go to Step2
    End If

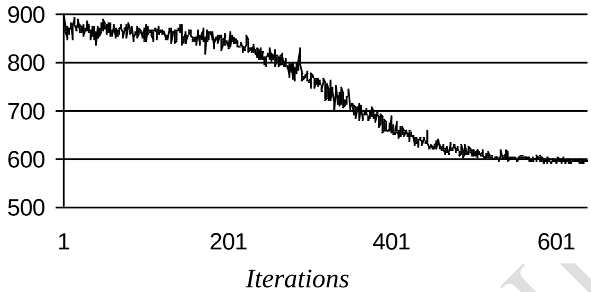
Figure 2: Pseudo algorithm for ant colony optimization to JSSP

VI. EXPERIMENTAL RESULTS AND DISCUSSION

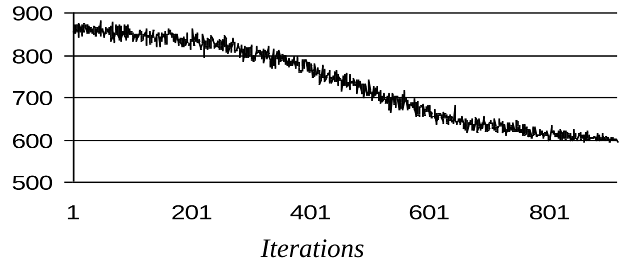
The proposed stair pheromone update strategies are tested using well known JSSP instances with $\alpha = 0.6$, $\beta = 0.8$ and $\rho = 0.01$. The number of ants used for testing is $n \times m$ and the stair interval value is set to 25. The interval value could be changed according to the complexity of the problem so as to produce the best result with lesser number of cycles. Table 1 shows the number of cycles required to reach the optimal makespan values for six instances of Lawrence series LA01, LA02, LA03, LA04, LA05 and LA10. Typical runs of LA05 problem are also illustrated in figures 3(a)-(d) using four different pheromone update strategies. It is clear that pheromone trail update using SLPUR results in more cycles required to reach the optimal solution. By using SGPUR lesser number of cycles are required to get the optimal value. In other two rules, SSPUR and SCPUR, the required cycles to produce the best result are reasonable level between SLPUR and SGPUR.

Problem instance	Best makespan value	Number of cycles to reach the deliberation level at best makespan value			
		SLPUR	SGPUR	SSPUR	SCPUR
LA01	666	2500	925	2140	1599
LA02	655	3647	1219	2761	1908
LA03	597	5232	1683	3125	2287
LA04	590	5297	1928	3573	2684
LA05	593	1654	605	1263	875
LA10	958	7651	2495	4899	3928

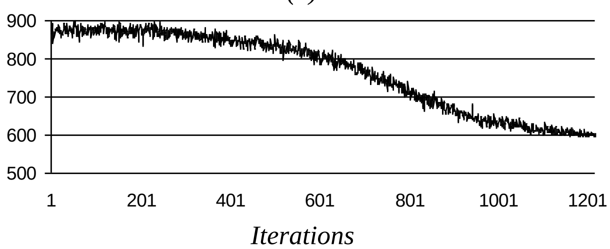
Table 1: Number of cycles necessary to reach the optimal values by using SLPUR, SGPUR, SSPUR and SCPUR for six problems with α = 0.6, β = 0.8 and ρ =0.01



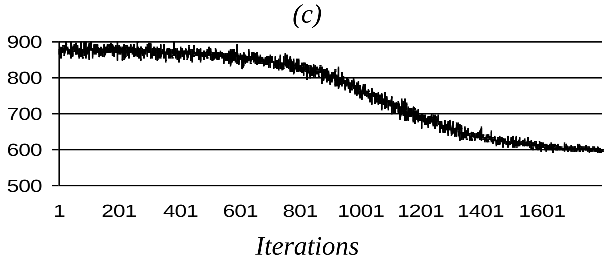
(a)



(b)



(c)



(d)

Figure 3: Comparison of makespan with Iterations

In SLPUR, the global pheromone trail updating, which concentrates on the best path, is not applied so as to explore the ants to different paths and this results in more cycles to produce the optimal value. In SGPUR, only the pheromone trial update on the best path is carried so that the solution is produced in lesser number of cycles. In SSPUR and SCPUR, local and global pheromone updating are carried out so as to get the result in temperate number of cycles. To avoid struck in local minima, exploring the ants in different directions over the search space is implicitly used in the proposed stair pheromone update strategy. Also SCPUR produces better result than SSPUR, because local and global updating rules are applied concurrent in SCPUR. Hence it is clear that sequential application of local and global update results in more cycles than the parallel application of these trial updates.

Figures 3 shows the convergent speed of ACO for the problem LA05. From Figure 3(a), it is clear that the convergent speed is smaller in first 200 cycles and it is increased between the cycles 200 and 400 by using SGPUR. From 400th cycles onwards the convergent speed is again decreased and the optimal value is produced at 609th cycle. After getting the optimal value, the rate of convergence becomes zero. From figures 3(b)-(d), it is clear that the convergent speed is smaller between the cycles 1 and 400, 1 and 600 and 1 and 800 by using the rules SCPUR, SSPUR and SLUPR respectively. As stated for the rule SGPUR, the convergent speed is increased between the cycles 400 and 600, 600 and 900 and 800 and 1200 by the respective application of SCPUR, SSPUR and SLUPR. After these cycles, the rate of convergent is decreased and the optimal value is produced at one level. After producing the optimal result, the convergent speed becomes zero and it is in the same level for further more cycles.

VII. CONCLUSION

This paper presented the application of ant colony optimization system to solve job shop scheduling problems. The goal of the work was to gain some insight into the influence of various pheromone updating strategies which seems to play an important role on the construction of good solutions. A new technique called stair pheromone updating strategy was proposed to improve the constructed solution. Four variations of stair pheromone updating strategies were proposed to determine the convergence rate of the algorithm as well as the quality of the solution by applying trail updating rule with the particular interval of time. This method was used to guide the search more into sub-optimal regions of the search space without losing the algorithm's capability of recovering from dead-ends by exploring other directions in the search space so as to get the optimal solution without getting struck in local optima. The proposed method was divided into four class of updating strategies and the performance tests were carried out on each of these strategies by using well known benchmark problems.

REFERENCES

[1] M. R. Garey, and D. S. Johnson., "Computers and Intractability, A Guide to the Theory of NP-Completeness", W.H. Freeman and Company, 1979.

[2] M. Dorigo, V. Maniezzo, A. Colorni, "The ant system: an autocatalytic optimizing process", Technical Report TR91-016, Politecnico di Milano, 1991.

[3] M. Dorigo, V. Maniezzo, A. Colorni, "Distributed Optimization by Ant Colonies", Proceedings of ECAL91 – European Conference on Artificial Life, Elsevier Publishing, Amsterdam, The Netherlands, pp. 134-142, 1991.

[4] M. Dorigo, "Optimization, Learning and Natural Algorithms", Ph.D Thesis, Dip. Elettronica e Informazione, Politecnico di Milano, Italy, 1992.

[5] M. Dorigo and L. M. Gambardella, "Ant colony System: A Cooperative Learning Approach to the Travelling Salesman Problem", IEEE Trans. On Evolutionary Computation, vol.1, no.1, pp. 53-66, 1997.

[6] M. Dorigo, V. Maniezzo and A. Colorni, "An Investigation of some properties of an Ant Algorithm", Proceedings of the Parallel Problem Solving From Nature Conference (PPSN92), Elsevier Publishing, Brussels, Belgium, pp. 509-520, 1992.

[7] M. Dorigo, V. Maniezzo, A. Colorni, "The Ant System: Optimization by a colony of cooperating agents" IEEE Trans.Systems, Man, Cybernetics, Vol.26, no.2, pp.29-41, 1996.

A. Colorni, M. Dorigo, V. Maniezzo, M. Trubian. "Ant system for Job shop Scheduling"; JORBEL- Belgian Journal of Operations Research, Statistics and Computer Science, Vol.34, pp. 39-53, 1993.

[8] S. Van Der Swaan and C. Marques "Ant Colony optimization for Job shop scheduling", In Proceedings of the third workshop Genetic Algorithms and Artificial Life (GAAL 1999), 1999.

[9] Christian Blum, Michael Samples, "An Ant Colony Optimization Algorithm for Shop Scheduling Problems", Journal of Mathematical Modeling and Algorithms, Kluwer Academic Publishers, Netherlands, Vol. 3, pp. 285- 308, 2004.

[10] M. Ventresca and B. M. Ombuki, "Ant Colony Optimization for Job Shop Scheduling Problem", Technical Report #CS-04-04, Canada, 2004.

[11] Jun Zhang, Xiaomin Hu, X. Tan, J.H. Zhong and Q. Huang, "Implementation of an Ant Colony Optimization technique for job shop scheduling problem", Transactions of the Institute of Measurement and Control, vol. 28, no.1, pp. 93-108, 2006.