Journal of Applied Sciences 8 (22): 4121-4128, 2008
ISSN 1812-5654

Solving Flow Shop Sequencing Problem for Deteriorating Jobs by Using Electro Magnetic Algorithm

H. Davoudpour and M. Hadji Molana
Department of Industrial Engineering, Amirkabir University of Technology (Tehran Polytechnic), Hafez Ave, Tehran, Iran

Abstract: In the classical scheduling problems, processing time was assumed to be constant and takes predefined values. In many realistic environments, such as machine maintaining or crisis event management, processing time on each machine depends on the position of jobs on the machine sequence or their starting time on that machine. We assume that processing time was an increasing linear function of its start time, in other words $p_{ij} = \alpha_{ij}t_{ij}$, in the literature these jobs were called deteriorating jobs. This problem addresses classic n\m\f\$C_{max}$ scheduling problem with a new assumption on the processing time of the jobs and it was surmised in the format of n\m\p,$p_{ij} = \alpha_{ij}t_{ij}$\$C_{max}$. We have used new Electro Magnetic meta-heuristic algorithm to find near optimum solution and compared the results with the results obtained from modified classical algorithms like CDS and Palmer.

Key words: Machine scheduling, meta heuristic, computational complexity, crisis event management, make span

INTRODUCTION

The machine-scheduling model is one of the most recent methods that are used to allocate limited resources among projects. In this model, resources are taken for machines and projects are taken for jobs, whereas different environmental conditions are formulated as production constraints. In most of the machine scheduling problems, the job processing times are assumed as known and constant, which is not applicable in many practical scenarios. However, in recent years some authors have investigated stochastic machine scheduling problems. Pinedo (1995) reviewed comprehensively the stochastic scheduling problems. It is observed that in the last decade there is a growing interest in considering problems involving scheduling with time-dependent processing times, i.e., problems where processing time of the job depends on the starting time on each machine. There are two main categories for these sorts of the problems. If the processing time decreased by postponing the operation on machine it is called jobs with learning effect, for example, in drying process, if the drying operation of the batch is postponed to a later time, natural evaporation leads to decrease in processing time. In the second category processing time increased by postponing the starting time of the process, which is known as deteriorating jobs. Kunnathur and Gupta (1990), Mosheiov (1994) and Sundararaghavan and Kunnathur (1994) have depicted the applications of these problems, some of them are mentioned below:

- In crisis management, like fire fighting or earthquake, where the situation deteriorated by time, the processing time of each reaction increased by postponing the process
- Maintaining and repairing production machines in factories usually depends on the duration of their active work time; if the interval between each repairing audit increases, the time for repairing operation increases as well
- The recovery duration of many diseases depends on the curing start time
- The similar case is for military attacks or defense actions where the condition worsens with time

Gupta and Gupta (1988) were of the earliest researchers who studied these problems in 1987. They assumed that each job processing time on a machine can be expressed by the following function: $P_i = f(t_i)$, in which the t_i is the starting time of *i*th job on the machine. They investigated three different expressions of the function:

$P_i = a_i+b_it_i$
$P_i = a_i+b_it_i+c_it_i^2$
$P_i = a_i+b_it_i+c_it_i^2+\ldots+m_it_i^m$

Corresponding Author: Seyyed Mohammad Hadji Molana, Department of Industrial Engineering, Amirkabir University of Technology (Tehran Polytechnic), Hafez Ave, Tehran, Iran
Tel: +98-21-88323980 Fax: +98-21-66413025

Other researchers followed this idea by concentrating on simple linear function, later other functions such as step-wise functions have also been considered.

Alidaee and Womer (1999) reviewed deteriorating jobs and categorized all possible functions in seven groups. It is an essential reference in selecting the increasing or decreasing function. They also investigated the complexity of the problems. Later in 2002 Cheng *et al.* (2004) considered the complexity of the deteriorating jobs in more details.

However, most of the studies have focused on the single machine problem. In recent years some authors investigated two-machines flow shop with deteriorating jobs via different objective functions. Wang *et al.* (2006) considered minimizing total completion time. They have proposed a heuristic algorithm to overcome the inefficiency of the branch-and-bound algorithm. Wu and Lee (2006) studied minimizing mean flow time, a branch-and-bound algorithm and several heuristic algorithms are provided to search for the optimal solution and the near-optimal solutions. Later Lee *et al.* (2007) provided an extensive review of that problem with make-span function. Because of computational complexity, there have been fewer studies in problems having more than two machines. Mosheiov (1995) considered the case of M-parallel identical machines, with one step operation and multi-step deterioration function. He has introduced a heuristic algorithm for minimizing make-span and has compared the answers with that of integer model solution. Hsieh and Bricker (1997) investigated the M-parallel identical machine problem with $p_i = x_i+\alpha_i t_i$ and $p_i = \alpha_i t_i$ deterioration function and minimizing make-span objective function. They have introduced some heuristics for this problem and have compared their performances. Later on Mosheiov (1998) studied the *M*-parallel identical machines problem having deterioration function $p_i = \alpha_i t_i$ and introduced another heuristic algorithm and a lower bound for minimizing the make-span. Khalil and Samson (2001) investigated the (Hsieh and Bricker, 1997) problem by using Simulated Annealing meta-heuristic algorithm and compared their answers with former ones. Gawiejnowicz *et al.* (2006) considered the M-parallel identical machines problem with $p_i = x_i+\alpha_i t_i$ and $p_i = \alpha_i t_i$ deterioration functions and completion time objective function. They have proposed another heuristic algorithm and reported their computational results. Kang and Ng (2007) studied the NP-hard problem of scheduling n deteriorating jobs on m identical parallel machines to minimize the make-span. They have presented a fully polynomial-time approximation scheme for the problem. Ji and Cheng (2008) considered the parallel-machine scheduling problem in which the objective is to minimize the total completion time. They have gone a Fully Polynomial-Time Approximation Scheme (FPTAS) for the case with m identical machines, where m is fixed. Lee and Wu (2008) investigated a multi-machine scheduling problem in which machines are not always available; each machine is assumed to have a maintenance period which is known in advance. Both the resumable and non-resumable cases are discussed with the objective of minimizing the makespan. A lower bound and a heuristic algorithm are proposed for each case.

In all of the Multi-machine studies, only parallel machines have been investigated. However, Mosheiov (2002) has assumed a simple linear deterioration function $p_{ij} = \alpha_{ij} t_{ij}$ and make-span objective function and provided a complete analysis of the complexity of flow-shops, open-shops and job-shop problems. He has proved that the Johnson algorithm provides optimal answer for two-machine case in flow-shop environment. He has also shown that the computational complexity of problems having more than two machines falls in the category of NP-completeness.

The n\m\f\$C_{max}$ problem is one of the classic machine scheduling problems. In this case m different operations are done by m different machines on n different jobs and jobs follow of the flow shop model, i.e., the operational sequence of all the jobs remains same. If the job sequence on all machines be also the same, it is called permutation scheduling problem which is noted by n\m\p\$C_{max}$.

Johnson (1954) has found optimal algorithm for m = 2. Later on many researchers have proposed many different heuristics and meta-heuristic algorithms for solving problems for m>2. There have been numerous comparisons between the performances of proposed algorithms. Ruiz and Concepcion (2005) have presented a comprehensive review of these comparisons. They also have compared 25 different heuristic and meta-heuristic methods for solving n\m\p\C_{max} problem by using Taillard's (1993) test method- a set of 120 instance of various sizes, having 20, 50, 100, 200 and 500 jobs and 5, 10 or 20 machines, with 10 problem inside each set. It is observed that the CDS and Palmer heuristic algorithms are well accepted algorithms, which have been the base for most of the algorithms and have good performances with comparatively short run times.

In this study, we have considered n\m\p\$C_{max}$ problem for m>2 with simple linear deterioration function $p_{ij} = \alpha_{ij} t_{ij}$, where i refers to order of sequence of the job on machine and j refers to order of machine itself. The problem under consideration has been denoted in the form of n\m\p,$p_{ij} = \alpha_{ij} t_{ij}$\$C_{max}$, where it is assumed that all jobs are available on $t_0 = 1$. As mentioned earlier this problem is NP-complete for m>2.

It is assumed that a job's initially arriving order is introduced by *i*, but jobs are allocated to all the machines with permutation order which is identified by an array like I. in the following we use of these denotes: The notations are as follows:

$OI_i = [k]$: The kth job in array I is ith initial arrived job
$S[k]_j$: The start time of operation on kth job in array I on jth machine
$P[k]_j$: The process time of kth job in array I on jth machine
$C[k]_j$: The process completion time of kth job in array I on jth machine
$C[k]$: The completion time of kth job in array I on all machines
$\alpha[k]_j$: The deterioration coefficient for kth job in array I on jth machine

Now assume that the initial order of jobs is rearranged and allocated to machines by the order of array I, so for the first job in array I on the first machine we should have:

$$S_{[1]1} = 1$$
$$P_{[1]1} = 1 \times \alpha_{[1]1} = \alpha_{[1]1}$$
$$C_{[1]1} = S_{[1]1} + P_{[1]1} = 1 + \alpha_{1[1]}$$

For the second job in array I on the first machine we should have:

$$S_{[2]1} = C_{[1]1} = 1 + \alpha_{1[1]}$$
$$P_{[2]1} = S_{[2]1} \times \alpha_{[2]1} = \left(1 + \alpha_{[1]1}\right)\alpha_{[2]1}$$
$$C_{[2]1} = S_{[2]1} + P_{[2]1} = \left(1 + \alpha_{[1]1}\right)\left(1 + \alpha_{[2]1}\right)$$

Likewise, for the kth job in array I on the first machine we should have:

$$S_{[k]1} = C_{[k-1]1} = \prod_{l=1}^{k-1} \left(1 + \alpha_{[L]1}\right)$$
$$P_{[k]1} = S_{[k]1} \times \alpha_{[k]1} = \alpha_{[k]1} \times \prod_{l=1}^{k-1} \left(1 + \alpha_{[L]1}\right)$$
$$C_{[k]1} = \prod_{L=1}^{k} \left(1 + \alpha_{[L]1}\right)$$

Similarly for the kth job in array I on the jth machine we should have:

$$S_{[k]j} = \text{Max}\left\{C_{[k-1]j}, C_{[k]j-1}\right\}$$
$$P_{[k]j} = S_{[k]j} \times \alpha_{[k]j}$$
$$C_{[k]j} = S_{[k]j}\left(1 + \alpha_{[k]j}\right) \qquad k = 1,\ldots,n \quad j = 1,\ldots,m$$

We have solved this problem with 4 different methods:

- Modifying classic CDS and Palmer
- Fortified Local search (LS) for finding lower bound
- Neighbor Search (NS)
- Using of Electro Magnetism (EM) meta-heuristic algorithm with Neighbor search (EMN)

BENCH MARKED ALGORITHMS

Modifying classic CDS and palmer: Johnson (1954) recommended that for n\2\p\C_{max} problem, in optimal array the initial ith job should be before initial jth job if:

$$\min\{p_{j,1}, p_{j,2}\} = \min\{p_{i,1}, p_{i,2}\}$$

Campbell *et al.* (1970) recommended the following procedure for solving n\m\p\C_{max} problem for m>2, in their algorithm, which is well known as CDS:

Step 1 : k=1
Step 2 : $P_{iB} = \sum_{j=m-k+1}^{m} P_{ij}$, $P_{iA} = \sum_{j=1}^{k} P_{ij}$
Step 3 : Treat A and B as two Johnson machines and find the optimal answer for it. Denote this answer by $C_m(k)$
Step 4 : If k = m-2 then k+1 ¬k and return to step 2
Step 5 : The best near optimal answer is the array with smallest $C_m(k)$ for 1 = k = m-1

We have modified this algorithm by substituting α_{ij} with p_{ij}. As mentioned earlier, in the case of m=2 the Johnson's algorithm gives optimal solution for n\2\p,$p_{ij} = \alpha_{ij}t_{ij}$\$C_{max}$ problem. Since CDS is a direct development of Johnson's algorithm and gives good near optimal result for classic n\m\p\$C_{max}$ problem, we hypothesized that modified CDS would also give good near optimal solution for n\m\p,$p_{ij} = \alpha_{ij}t_{ij}$\$C_{max}$.

Palmer (1965) recommended in that order to get a good near optimal solution for n\m\p\C_{max} problem in a very short time, the following procedure should be used:

For each initial ith job find the:

$$S_i = -\sum_{j=1}^{m} \{m - (2j-1)\}P_{ij}$$

Sorting jobs in descending order of S_i leads to a good near optimal result.

Similar to CDS we have modified classic Palmer by substituting α_{ij} with p_{ij} and logically hypothesized that this procedure will provide us with a good near optimal solution in a short time.

Fortified Local Search (LS) for finding lower bound: This algorithm requires Initial Sample Number (ISN) and Algorithm's Iterations Number (AIN) for each sample array. At first it provides random different arrangement of job order in ISN. Then for each sequential order, like array I, it follows the following procedure in the AIN:

- Start with array I
- Exchange the position of first job with the second one in array I and name this new array as I'
- If the make-span for array I' is smaller than I, substitute with I (or delete I and rename I' to I)
- Repeat steps 2 and 3 for first job in updated array I and all subsequent jobs
- Focus on the second job in the updated array and repeat steps 2, 3 and 4 for this job and all subsequent jobs
- Repeat step 5 for all jobs in updated array like second job
- Gives out array I

This algorithm searches in feasible space with a simple systematic logic. Since it checks many arrays systematically, the algorithm converges towards the optimal point. However, it takes considerable time to run this algorithm and for this reason, it is not at all feasible to solve large scale problems. We have used this algorithm here only to find a lower bound, so that we can compare the solution of other algorithm with this one.

Neighborhood Search (NS): This is very similar to LS algorithm, but we have made it simpler. It is mainly used in EM for improving the array in its neighborhood. It does not take as much time as LS. Like LS, it requires Initial Sample Number (ISN) and Number of Algorithm's Iteration (AIN). At first it provides different random arrangement of job order in ISN. Then for each sequential order like I it does the following procedure in AIN:

- Start with array I
- Put K = 1
- Exchange the position of Kth job in array I with (K+1) th job in array I and name new array I'
- If the make-span for array I' is smaller than I, substitute it with I (or delete I and rename I' to I)
- K+1→K
- If the K<n then repeat from step 3
  Give out array I.

ELECTRO MAGNETISM

Introducing base electro magnetism: Birbil and Fang (2003) introduced this meta-heuristic algorithm. The method utilizes an attraction-repulsion mechanism to move the sample points towards the optimality. They studied a special class of optimization problems with bounded variables in the form of:

$$\min f(x)$$
$$\text{s.t. } x \in [l,u]$$
$$[l,u] := \{x \in R^n \mid l_k \le x \le_k u_k, k = 1,...,n\}$$

Where:
n = Dimension of the problem
u_k = Upper bound in the kth dimension
l_k = Lower bound in the kth dimension
f(x) = Pointer to the function that is minimized

Similar to that in elementary electromagnetism, they have assumed each sample point as a particle that is released to a space. In their approach the charge of each point relates to the objective function value, which is tried to be optimized. This charge also determines the magnitude of attraction or repulsion of the point over the sample population- the better the objective function value, the higher the magnitude of attraction. After calculating these charges, they have used them to find a direction for each point to move in subsequent iterations. They have selected that direction by evaluating a combination force exerted on the point via other points. The same as electromagnetic, force is calculated by adding the forces in vector space from each of the other points which is calculated separately.

The EM consists of four main phase which is depicted in brief:

Initialization: This procedure is used to sample m points (arrays for our problem) randomly from the feasible domain, which is an n dimensional hyper-cube (n job position for our problem). After a point is sampled from the space, the objective function value for the point is calculated using the function pointer f(x) and the point that has the best function value is named x^{best}.

Local search: This procedure is used to gather the local information for a point and improve the point along its local area. They have suggested using arbitrary local search methods and in their paper have considered a local search method for example.

Calculation of total force vector: At first they have computed the charge of each point like i according to its objective function, the objective function of x^{best} and other points objective function by the following formula:

$$q^i = \exp\left(-n\frac{f(x^i)-f(x^{best})}{\sum_{k=1}^{m}\left(f(x^k)-f(x^{best})\right)}\right)$$

Note that objective function of each point changes from iteration to iteration and the sign of the charge depends on the other point which is compared with. In comparing two points, the point with higher objective function attracts other point and the point with lower objective function repulses other point. Finally, the total force F^i exerted on point i is computed by the following equation:

$$F^i = \sum_{j\neq i}^{m}\begin{Bmatrix}(x^j-x^i)\dfrac{q^iq^j}{\|x^j-x^i\|^2} & \text{if} & f(x^j)<f(x^i)\\ (x^i-x^j)\dfrac{q^iq^j}{\|x^j-x^i\|^2} & \text{if} & f(x^j)\geq f(x^i)\end{Bmatrix},\forall_i$$

Movement according to the total force: After evaluating the total force vector F^i, the point i is moved in the direction of the force by a random step length, λ. This random step length is used in order to ensure that the points have a nonzero probability to move to the unvisited regions along the movement direction. The new point i is calculated from the former point i with the following equation:

$$x^i = x^i + \lambda\frac{F^i}{\|F^i\|}(RNG)$$

In which the (RNG) is a vector whose components denote the allowed feasible movement toward the upper bound, u^k, or the lower bound, l^k, for the corresponding dimension.

There can be define various termination criteria for the EM, Birbil and Fang (2003) proposed using predefined iteration number. They have demonstrated by some example that heuristic EM performs very well in convergence towards the optimal point.

Modified EM for n\m\p,$p_{ij} = \alpha_{ij}t_{ij}$\$C_{max}$ problem: As earlier mentioned Birbil and Fang (2003) have proposed heuristic EM for special application. We assume that the position of each job in an array I is denoted by x^I_i, in other words, if the position of initial ith job in the array I is k, it is denoted by:

$$x^I_i = k$$

We have modified some aspects of base EM for using in our problem which are as follows:

Use of EM for discrete variables: The base EM was designed for continuous variables. We have used EM for discrete variables, this customization leads to some modification in the movement procedure and we have used integer function to round the produced points. If the power is positive we have rounded the points up and if the power is negative we have rounded the points down.

Adding constraints to base optimization model: In the base EM model there was just one upper and lower bound constraint. But, in this problem, we have more constraints, for example, two or more initial jobs cannot be allocated in the same place in an array I, in notation:

$$x^I_i \neq x^I_j \qquad \text{if } i\neq j$$

So, after assigning new position of each job in each updated array, the algorithm checks whether this position was formerly allocated to another job if it was allocated earlier, the process is repeated 10 times. If some jobs cannot be allocated after the process repetitions, they are stored in a set. Finally in our new array we may have some positions which are empty of job and some jobs which are never allocated, these two sets will have equal members. We sort the unallocated jobs ascending by their former position in the old array and then these jobs are allocated one by one to empty positions in new updated array.

Prioritizing jobs movement: There is no position constraint in the base EM model, so two or more variables could have the same value and is not any priority in getting value either and all variables get value independent of others. But in present model as explained earlier, the variables cannot take equal value, so we should allocate value to variables one by one and value of each variable depends on the former variables. But it is possible a value which is better for a variable allocated former. For preventing such occurrence to occur, we have prioritized value allocation to variables. We have sorted variables (or jobs for getting new position in new array) by the absolute value of their total force which was calculated previously. The bigger absolute value of total force for each job in an array means, that job is allocated in bad position and its position should be changed immediately. Therefore, at first we allocate value to the variable with strong total force.

We indeed are interested in testing some parameters of base EM and find proper value for them.

The powers parameter in the force formula: Birbil and Fang (2003) have used exactly the electromagnetism formula in their heuristic EM. In that formula the power for

charge is 1 and the power for the distance is 2. But we are interested to know if those parameters are proper for our problem. For investigation we assumed 3 levels:

- The power of charge = 0.5
  The power of distance = 2
- The power of charge = 1
  The power of distance = 1
- The power of charge = 2
  The power of distance = 1

In the first scenario, distance has more importance. In the second scenario, distance and charge have equal importance. In the third scenario, charge has more importance.

Initial Sample Number (ISN) and Iteration Number of the Algorithm (AIN): We are interested in investigating the importance of ISN via AIN. So, we have considered three different scenarios:

$$ISN = \left[\frac{n}{3}\right]$$
$$AIN = \left[\frac{2n}{3}\right]$$
$$ISN = \left[\frac{n}{2}\right]$$
$$AIN = 20$$
$$ISN = \left[\frac{2n}{3}\right]$$
$$AIN = \left[\frac{n}{3}\right]$$

Updating new sample arrays: In the base EM, after each iteration, all new arrays replace all old arrays, we name it continuous updating. We have proposed new approach in which, after producing new arrays out of old ones, we investigate all old and new arrays and accept half of total arrays with better objective function and we name this new approach discrete updating.

We have used neighborhood search which was introduced in former sections, as the local improver in the heuristic EM for solving $n\backslash m\backslash p, p_{ij} = \alpha_{ij}t_{ij}\backslash C_{max}$ problem.

COMPUTATIONAL DESIGN

For solving $n\backslash m\backslash p, p_{ij} = \alpha_{ij}t_{ij}\backslash C_{max}$ problem, we consider n in five levels (5, 15, 25, 35, 45) and *m* in three levels (3, 4, 5). For each 15 combination of levels, we generate 5 instances by randomly predefined deterioration coefficient such that:

Table 1: Factors and levels of EMN algorithms

	Levels		
Factor name	1	2	3
Force parameter	P0	P1	P1
ISN-AIN	S1	S2	S3
Updating procedure	C	D	-

Table 2: Thirity six different algorithms used for solving $n\backslash m\backslash p$, $p_{ij} = \alpha_{ij}t_{ij}\backslash C_{max}$

EM		LS	NS	Classic	Random
EMNCP0S1	EMNDP0S2	LSLS1	BNS1	CDS	RND
EMNCP1S1	EMNDP1S2	LSBS1	CNS1	PALMER	
EMNCP2S1	EMNDP2S2	LSLS2	DNS1		
EMNDP0S1	EMNCP0S3	LSBS2	BNS2		
EMNDP1S1	EMNCP1S3	LSLS3	CNS2		
EMNDP2S1	EMNCP2S3	LSBS3	DNS2		
EMNCP0S2	EMNDP0S3		BNS3		
EMNCP1S2	EMNDP1S3		CNS3		
EMNCP2S2	EMNDP2S3		DNS3		

$$\alpha_{ij} \sim uni(0,1)$$

We solve each instance twice, each time by 36 different algorithms from 5 main categories as following:

Category 1: 18 different EMN algorithms produced by combinations of three levels of force parameter, three levels of ISN-AIN and two levels of updating procedure, which is shown in Table 1.

Category 2: Nine different NS algorithms are produced. Six of them by combination of three levels of ISN_AIN and two levels of updating procedure and three of them are produced by a big initial sample number which is equal to ISN*AIN and repeats just once; the later one is called BNS.

Category 3: Six different LS algorithms are proposed. Three of them correspond to three ISN-AIN and three of them with a big sample size equal to ISN*AIN and repeats just once.

Category 4: Two modified CDS and Palmer algorithms.

Category 5: One randomly produced arrangement of jobs.

Finally, 75 different instances have been generated and in total 5400 solutions have been produced for those instances. In Table 2 different algorithms are shown.

We have used MATLAB7 for coding the algorithms and SAS9 for analysis of variance (ANOVA). All of these have been done on a notebook with CPU speed 2, cash 4, RAM 1024. The summary of our computational result is reported in Appendix 1.

COMPARISON OF ALGORITHMS AND CONCLUSION

At first, we have analyzed our data with simple mean statistic. For more accuracy we have performed nested factorial analysis with ANOVA models on present data. Present conclusion in brief is as follows:

- If we add an operation to our problem (increase m by one unit) and keep the number of job unchanged, the make-span for new problem will be about 2.23 times greater than the former problem. If we do not change operation numbers and add another job to our problem, the new make-span will be about 1.55 times greater than the former problem.
- After doing ANOVA at $\alpha = 0.05$ the performance of algorithms are grouped and sorted as following:

  LS
  EMN
  NS (LCN)
  CDS, Palmer
  NS (BN)
  NS (LDN)
  Random allocating

- Modified classic CDS and Palmer have distinguished good performance, so that the mean make-span of these algorithms is 0.21 times smaller than the mean make-span of random allocating of jobs. CDS do a little better than Palmer which is distinguished at $\alpha = 0.05$.
- The best make-span of LS algorithms is on the average 0.83 times smaller than the mean make-span of EM algorithms, but the mean run time of LS algorithms is approximately 12 times larger than mean run time of EM algorithms. So we conclude that LS algorithms improve the objective function a little only by sacrificing the run time
- All EM algorithms have better performance compared to all other algorithms (except LS) at $\alpha = 0.05$.
- Since, there is distinguished differentiation at á=0.05, the performance of EMN algorithm is not dominated by internal NS algorithm.
- After performing ANOVA with $\alpha = 0.05$, we conclude that using same power, 1, for both charge parameters and distance parameters leads to a better result.
- After performing ANOVA with $\alpha = 0.05$ and interaction tests, we conclude that in small number of jobs (5, 15 and around 25) the Number of Initial Sample (ISN) is more important than the algorithm's iteration number (AIN).
- Doing ANOVA at $\alpha = 0.05$ and interaction tests have also helped us to conclude that using discrete updating approach leads to a better result than using continuous approach, when the problem size is relatively large (number of jobs is bigger than 25 and number of machines is bigger than 3).

From these tests now we know that EMN algorithms lead to reasonable answer in a relatively less time. Further more we can adjust the parameters of EMN algorithms correspond to our problem size to better results.

The scopes for further study are as follows:

- Considering n\m\p,pij = $a_i+\alpha_{ij}t_{ij}\backslash C_{max}$ problem with different heuristic and meta heuristic algorithms
- Solving problem that has been considered here (n\m\p,pij = $\alpha_{ij}t_{ij}\backslash C_{max}$) with other algorithms and compare them
- Considering parameter λ in movement process of EM algorithm

Appendix 1: Our test problems report is presented in the following table

Jobs No.	Machines No.	CDS	Palmer	Random ARRY	EMN's mean	Best EMN	NS's mean	Best NS	LS's mean	Best LS
5	3	16	16	23	15	14	15	14	14	14
	4	33	31	49	30	28	30	28	29	27
	5	54	58	94	55	52	54	52	52	52
15	3	794	898	2,509	744	715	839	764	710	710
	4	2,626	2,457	8,530	2,035	1,809	2,413	2,203	1,703	1,672
	5	4,953	8,013	15,081	4,404	3,816	5,314	4,341	3,469	3,457
25	3	65,302	74,472	230,675	66,634	65,366	76,394	68,675	65,302	65,302
	4	117,866	103,036	582,487	84,846	77,436	121,582	98,548	68,766	67,414
	5	241,278	249,314	1,183,794	176,780	157,184	245,657	192,723	135,719	128,057
35	3	5,325,580	5,271,280	31,167,300	4,323,770	4,221,010	5,521,064	4,864,320	4,189,163	4,188,980
	4	6,483,760	6,645,520	52,847,270	4,522,562	4,331,200	5,807,006	4,839,380	4,194,543	4,171,934
	5	15,662,920	19,553,640	79,720,100	11,034,873	9,458,390	18,154,457	13,777,190	6,834,972	6,729,850
45	3	176,950,400	166,900,000	1,031,607,000	152,469,100	149,354,800	186,334,444	156,184,300	148,273,533	148,140,200
	4	222,345,400	223,166,400	1,730,438,000	136,708,744	121,227,600	232,038,411	179,617,500	93,730,800	90,838,900
	5	520,280,000	597,990,000	6,232,300,000	419,038,539	358,225,900	716,958,300	513,903,000	280,830,767	280,200,000

REFERENCES

Alidaee, N.K.W., 1999. Scheduling with time dependent processing times: Review and extensions. J. Operat. Res. Soc., 50: 711-720.

Birbil, S.I. and S.C. Fang, 2003. An Electromagnetism-like mechanism for global optimization. J. Global Optimiz., 25: 263-282.

Campbell, H.G., R.A. Dudek and M.L. Smith, 1970. A heuristic algorithm for the n-job, m-machine sequencing problem. Manage. Sci., 16: 630-637.

Cheng, T.C.E., Q. Ding and B.M.T. Lin, 2004. A concise survey of scheduling with time-dependent processing times. Eur. J. Operat. Res., 152: 1-13.

Gawiejnowicz, S., W. Kurc and L. Pankowska, 2006. Parallel machine scheduling of deteriorating jobs by modified steepest descent search. In: Parallel Processing and Applied Mathematics, 6th International Conference, September 11-14, Poznan, Poland, pp: 116-123.

Gupta, J.N.D. and S.K. Gupta, 1988. Single facility scheduling with nonlinear processing times Comput. Ind. Eng., 14: 387-393.

Hsieh, Y.C. and D.L. Bricker, 1997. Scheduling linearly deteriorating jobs on multiple machine. Comput. Ind. Eng., 32: 727-734.

Ji, M. and T.C.E. Cheng, 2008. Parallel-machine scheduling with simple linear deterioration to minimize total completion time Eur. J. Operat. Res., 188: 342-347.

Johnson, S.M., 1954. Optimal two and three stage production schedules with setup times included. Naval Res. Logist. Q., 1: 61-68.

Kang, L. and C.T. Ng, 2007. 'A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs. Int. J. Prod. Econ., 109: 180-184.

Khalil, S.H. and M. Samson, 2001. Scheduling linearly deteriorating jobs on parallel machines: A simulated annealing approach Prod. Plann. Control, 12: 76-80.

Kunnathur, A.S. and S.K. Gupta, 1990. Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problems. Eur. J. Operat. Res., 47: 56-64.

Lee, W.C., C.C. Wu, C.C. Wen and Y.H. Chung, 2007. A two-machine flowshop makespan scheduling problem with deteriorating jobs Comput. Ind. Eng., 54: 737-749.

Lee, W.C. and C.C. Wu, 2008. Multi-machine scheduling with deteriorating jobs and scheduled maintenance. Applied Math. Modell., 32: 362-373.

Mosheiov, G., 1994. Scheduling jobs under simple linear deterioration. Comput. Operat. Res., 21: 653-659.

Mosheiov, G., 1995. Scheduling jobs with step-deterioration, minimizing Makespan on a single and multi-machine Comput. Ind. Eng., 28: 869-879.

Mosheiov, G., 1998. Multi-machine scheduling with linear deterioration. INFOR., 36: 205-214.

Mosheiov, G., 2002. Complexity analysis of job-shop scheduling with deteriorating jobs. Discrete Applied Math., 117: 195-209.

Palmer, D.S., 1965. Sequencing Jobs through in a multi stage process in the minimum total time: A quick method of obtaining a near optimum. Operat. Res. Q., 16: 101-107.

Pinedo, M., 1995. Scheduling, Theory, Algorithms and Systems. 1st Edn. Prentice-hall, Englewood Cliffs, NJ.

Ruiz, R. and M. Concepcion, 2005. A comprehensive review and evaluation of permutation flowshop heuristics. Eur. J. Operat. Res., 165: 479-494.

Sundararaghavan, P.S. and A.S. Kunnathur, 1994. Single machine scheduling with start time-dependent processing times: Some solvable cases. Eur. J. Operat. Res., 78: 394-403.

Taillard, 1993. Benchmark for basic scheduling problems. Eur. J. Operat. Res., 47: 278-285.

Wang, J.B., C.T. Daniel Ng, T.C.E. Cheng and L.L. Liu, 2006. Minimizing total completion time in a two-machine flow shop with deteriorating jobs Applied Math. Computat., 180: 185-193.

Wu, C.C. and W.C. Lee, 2006. Two-machine flowshop scheduling to minimize mean flow time under linear deterioration. Int. J. Prod. Econ., 103: 572-584.