MULTIPROCESSOR SCHEDULING USING ANT COLONY OPTIMIZATION WHERE JOB VALUES ARE THE POWER FUNCTION OF COMPLETION TIME

Sachin Shrivastava[1], Deepak Chaudhary[2]

[1]*M.Tech,CSE, IET, Alwar, Rajasthan,(India)*

[2]*Assistant Professor, IET, Alwar, Rajasthan,(India)*

ABSTRACT

This paper deals with an anology of scheduling jobs on prarallel processors using Ant Colony Optimization where job values are given as a power function of the job execution times. We consider a scheduling problem in a machine of parallel processors with the objective of minimization of the total loss of job values is considered as criterion which reflects the minimization of waiting time of the jobs. This paper establishes the computationl complexity of the problem by the application of Ant Colony Optimization on the problem. An augmented experimentation is conducted to evaluate the performance of the parallel processor scheduling using ACO which is strong NP hardness of its general versions and NP hardness of its single machine case. Moreover, some special cases of the problem in the polynomial time.

Keywords*: *Ant Colony, ACO, job values, loss function, Parallel Processor, Scheduling, Scheduling Algorithm

I. INTRODUCTION

The aim of this Multi-processor scheduling using ant colony optimization is to describe and solve a scheduling problem where execution time of jobs are described as a power function of the job completion or execution times. In this paper we consider the scheduling problem over parallel processors with the objective of minimizing the total loss value (average waiting time) of the jobs by application of Ant Colony Optimization on parallel processor scheduling. [1]

Our aim in the present work is to develop an efficient algorithm for solving the parallel processor scheduling problem over the heuristics available in the literature and we propose the ant colony optimization technique over multi-processor scheduling to formulate the results.

Another new kind of the scheduling problems that we had studied in this paper .Namely the in which processing times of all the jobs are some values which are fixed in advance and constant during optimization process but their values deteriorate over time. The process description of the problem can be illustrated by an application example, which characterizes the utilization process of the components from some used up computers. Therefore

there is given a set of some used up computer which cannot be used anymore because their further utilization is connected with a high risk of a breakdown.

There are some applications which require faster processors or simply of their components are already broken. However some of their components (e.g monitors, floppy disks drives, network drives ,) can be utilized as a spare parts in some other computers. Thus the problem of disassembling computers into their components appears nearest can be shown that the values of the computer component decreases over time as a power function of the speed of this different for the particular component .In disassembling process we are interested in the component values which are determined at the moments at they are available for utilization . The component is ready to be used after it is completely removed from one computer and its proper functionality is confirmed. Thus the order in which the computer will be disassembled has a significant influenced on the total profit, i.e. the sum of the component values. Therefore, maximization of the total component value is considered as an optimization criterion [2]

II LITRATURE SURVEY

The single machine scheduling problem with constant setup times has been well covered in the literature. Koulamas [1994] presents a review of the total tardiness minimization problem and also points out that various approaches are available for the single machine problem. He proposes the classification of the proposed methods as *optimal* or *heuristic*. The first class includes dynamic programming, branch and bound and hybrids including both dynamic as well as branch & bound. The author points out that dynamic programming algorithm are superior and most efficient method was proposed by Potts & Van Wassenhove [1982]. The class of heuristic methods can be further broken down to sub-classes including constructive algorithms, local search methods and decomposition methods. Results indicate that local search and decomposition approaches are generally more effective than construction heuristics.[3]

Two main theoretical models must be pointed out that concerned to the single machine scheduling tardiness-minimization problem. Emmons [1969] developed the dominance condition and several authors [Lawler, 1977] [Potts & Van Wassenhove, 1982] wrote on the subject of the decomposition principles. These contributions bring the development of optimal solution procedures but they also inspired the construction of various heuristics [Della Croce et al., 1998].

In addition to the characteristics of sequence dependent algorithms, heuristics increases the complexity of the problem when provides the solution to minimize the total tardiness on a single machine. This characteristic invalidates the supremacy principle as well as the decomposition principle [Rubin & Ragatz, 1995]. Du & Leung [1990] seem to have been the first to show that this problem is NP-hard.[4]

From a practical point of view, many industrial situations require the explicit consideration of setups and the development of appropriate scheduling tools. Previous authors have described cases highlighting this situation. Pinedo [1995] describes the situation of a manufacturing plant making paper bags where setups are required when the type of bag changes. The duration of a setup depends on the similarity of the bags made in the preceding lot. A similar situation was observed in the plastics industry by Das et al. [1995] and Franca et al. [1996]. The printing industry also has setups that are sequence dependent because various cleaning operations were required when the print colors are changed [Conway et al., 1967]. The aluminum industry has casting operations where setups, mainly affecting the holding furnaces, are required between the castings of different

alloys [Gravel et al, 2000]. The textile, pharmaceutical, chemical and metallurgical industries present other practical examples where sequence dependent setups are frequently observed.

Despite the copious literature in scheduling, it was only in 1999 that two reviews of problems with sequence dependent setups were published [Allahverdi et al., 1999] [Yang & Liao, 1999]. These authors have proposed different classifications of the field but arrive at similar conclusions. Allahverdi et al. [1999] point out gaps in existing research, including in the underlying theoretical underpinnings and in the treatment of multiple objectives. The general conclusion of this review is that scheduling where sequence dependent setups are required is a fertile area for further research. Yang & Liao [1999] observe that there are few comparisons of the solution methods developed for this problem. They make the same observation concerning the applications of the various methods available in practical situations.[2]

The literature shows [Allahverdi et al., 1999] that while many industrial applications have sequence dependent setups, few papers have treated this characteristic in combination with the objective of meeting delivery dates. Few authors have treated the problem described in the previous section. Among these Ragatz [1993] proposed a branch and bound algorithm for the exact solution of smaller problems. A genetic algorithm and a local improvement method were proposed by Rubin & Ragatz [1995] while Tan & Narasimhan [1997] tackle this same problem through simulated annealing. Finally Tan et al. [2000] present a comparison of these four approaches and conclude, following a statistical analysis, that the local improvement method offers better performance than simulated annealing, which is turn better than branch and bound.
In this comparison, the genetic algorithm had the worst performance. The authors propose an ant colony optimization (ACO) algorithm for the solution of the single machine scheduling problem with sequence dependent setups. This industrial scheduling problem from the aluminum industry consists in the scheduling of a set of released jobs on a casting rig and has been formulated as a single machine with sequence dependent setups and multiple objectives. We found the basic ACO to be effective in terms of solution quality and that it had relatively low computation times [Gravel et al., 2001]. In this paper, we report on the addition of extensions to the basic algorithm and on numerical experiments that compare our results to those found using other metaheuristics.

III SCHEDULING

A schedule is a tangible plan or document, such as a bus or a class schedule. A schedule usually tells us when things are supposed to happen; it shows us a plan for the timing of certain activities and answers the question, “If all goes well, when will a particular event take place?”

The aim of this scheduling is to describe and solve a scheduling problem where jobs values (or losses of jobs values) changes during their execution and are described as a power function of the job completion times. [3]
In this paper the multiprocessor scheduling with job values depending on completion time using ant colony optimization is studied. Furthermore, in order to improve the performance of the system, increase the revenue and reduce processing times of deteriorating jobs, we have used job selection and pheromone update formula.

3.1 Notations

Regardless of its nature, every scheduling problem S can be formulated as a quadruple, S = (J, M, P, L), where J is a set of non preemptive jobs immediately available for processing time 0. M is a set of identical parallel machines entities that will perform the available jobs. P is the processing time of the job, where pj >0 and L is the loss function as lj(t) describing the loss of its value at time t. [4]

Processing time (pij) - The pij represents the processing time of job j on machine i. The subscript i is omitted if the processing time of job j does not depend on the machine or if job j is only to be processed on one given machine.

Exponential loss rate (aj)-This is the loss rate denoted by the aj, aj>0 and it is associated with the job ji performing on machine M.

Proportional loss rate (wj)-This is another loss rate denoted by the wj, wj>0 and associated with the job ji performing on machine M.

Loss function (lj(t))-This is loss rate function characterizing the loss of its value at time t. The loss rate function is a non decreasing power function of time. Morever the loss rate is calculated only at the completion time Cj

Three basic pieces of information that help to describe jobs in the single-machine case are:

- Processing time (pj) the amount of processing required by job j
- Exponential loss rate (aj) the exponential loss rate of job j
- Proportional loss rate (wj)The proportional loss rate of job j
- Completion time (Cj) The time at which the processing of job j is finished.

3.2 Scheduling Algorithm and its Complexity Theory

A useful perspective on the relation of scheduling problems and their solution techniques comes from developments in a branch of computer science known as complexity theory. The notion of complexity refers to the computing effort required by a solution algorithm. Computing effort is described by order-of-magnitude notation. For example, suppose we use a particular algorithm to solve a problem of size n. (Technically, n denotes the amount of information needed to specify the problem.) The number of computations required by the algorithm is typically bounded from above by a function of n. If the order of magnitude of this function is polynomial, as n gets large, then we say the algorithm is polynomial. For instance, if the function has order of magnitude n2, denoted O(n2), then the algorithm is polynomial. On the other hand, if the function is O(2n), then the algorithm is non polynomial (in this case, exponential). Other things being equal, we prefer to use a polynomial algorithm because as n grows large, polynomial algorithms are ultimately faster. [5]

IV RELATED WORK

The paper deals with the problem of scheduling jobs on multi-processors of similar type, where job values are taken as power function of the job execution times. Minimization of the total loss of job values is considered as criterion. This paper establishes the computational complexity of the problem of strong NP hardness of its general versions and NP hardness of the single machine case. Moreover, some special cases of the problem in the polynomial time. Finally the paper constructs and experimentally test ant colony optimization algorithm along with some elimination properties improving its efficiency. In order to solve the significant real life problems like single machine scheduling tardiness-minimization problem, sequence independent scheduling using ant colony optimization is proposed. Besides ants' activity, ant colony optimization algorithm includes two more procedures: pheromone trail evaporation and daemon

actions. In order to solve the general version of the problem, we also construct and experimentally test a number of heuristic algorithms and mathematically define the comparison between various heuristics & ant colony optimization algorithm.

4.1 Mathematical Formulation

There are given a set of m identical parallel machines M = {M1,…., Mm} and a set of n independent and non preemptive jobs J = {J1,…..Jn} that are immediately available for processing at time t = 0. Each job Ji ϵ J is characterized by its processing time pi>0 and a loss function lj(t) which characterize the loss of its value at time t. The loss function is given by non decreasing power function of time.

$L_j(t) = w_j t^{a_j}$ (1)

$$TLV(II) = \sum_{i=1}^{m}\sum_{j=1}^{n_i} l_{\pi_{i(j)}}(C_{\pi_{i(j)}}) = \sum_{i=1}^{m}\sum_{j=1}^{n_i} w_{\pi_{i(j)}} C_{\pi_{i(j)}}^{ce_{\pi_{i(j)}}} \rightarrow \min$$

The decrease of the computer components values can be calculated by the following time function $v(t) = \omega t^{-\alpha}$, where the values of α and ω are experimentally calculated for various type of the processors, random access memory, hard disks ,monitors etc.Therefore the values of the computer components decrease fast at the beginning when the components are quite new, and the values decrease slowly when the component are quite old. Thus such situation can be also represented by the scheduling problem considered in the present paper as follows. The value vj(t) of job Jj decreases over time and can be decreased by the following expression.

$$V_j(t) = v_j^0 - l_j(t) = v_j^0 - w_j t^{a_i} \quad (2)$$

Where, $v_j^0 > 0$ is its initial value (since $I_j(0)$ and $a_j \in (0,1)$ (the function $v_j(t)$ is convex for the concave function $I_j(t)$ given by Expression. The specific values of v_j^0, w_j, and a_j for particular components can be obtained experimentally, taking into account the appropriate values of ω and α. In such a case, we can also assume that the job values $v_j(t)$ is positive at any moment t of the optimization process.

Thus, the following condition should be satisfied

$v_j(t) = v_j^0 - w_j t^{\alpha j} > 0$ for t belongs $[0, \sum_{j=1}^{n} p_j]$.

The problem is to find such a solution π that the sum of job values calculated at their completion times is maximal, i.e,

$$\left(\sum_{i=1}^{m}\sum_{j=1}^{n_i} v_{\pi_{i(j)}}(C_{\pi_{i(j)}}) \right. \quad (3)$$

$$\left. = \sum_{i=1}^{m}\sum_{j=1}^{n_i} v_{\pi_{i(j)}}^0 - \sum_{i=1}^{m}\sum_{j=1}^{n_i} l_{\pi_{i(j)}}(C_{\pi_{i(j)}})\right) \rightarrow \max,$$

Since, $\sum_{i=1}^{m}\sum_{j=1}^{n_i} v_{\pi_{i(j)}}^0 = \sum_{j=1}^{n} v_j^0$ is constant, thus, maximization of the sum of job values is equivalent to minimization of the total loss of jobs values present in the system.

V ALGORITHM FOR PARALLEL PROCESSOR SCHEDULING

In this problem we have given a set of m identical parallel machines and a set of n independent and non-preemptive jobs immediately available for processing at time 0. Each job J_J is belongs to J charcterised by its processing time and a loss function.

Firstly we assign the jobs to the machines according to their processing time. The job j having the minimum processing time is assigned to the first machine and as goes on. The number of jobs and number of machines are randomly generated by the user and we generate the sequence number with processing time.

Secondly we generate another sequence number by assigning the jobs to the machines on the basis of their loss function. The loss function is denoted by $l_j(t)$ characterizing the loss of its value at time 0. The loss function is denoted by non decreasing power function.[6]

For this section we describe the most efficient algorithms and the result of the performed experimentally analysis.All the algorithms A1LST-A3LST are based on the list strategy but they use three different priority dispatching rules to obtain an order of jobs in an input list. [7]

5.1 Algorithm A1LST

Step 1 Construct the list of jobs L, according to the algorithm Al, i € {1, 2, 3}.

Algorithm A1 and A2 construct solutions by sequencing all the jobs in the non increasing order of a_j and ($a_j.w_j/p_j$ respectively).

Algorithm A3LST

Step1. Set $\pi = 0$, U= {1….n} and $C=\sum_j=p_j$.

Step2. Find a job j€ U with the minimal value of w_jC^j and move it from U at the beginning of π.Set $C = C - p_j$.

Step3. If $U \neq 0$ then go to step 2 the job order is given by π.

The computational complexity of algorithm A1 and A2 is equl to O(nlogn), while the complexity of algorithm A3 is equal to $O(n^2)$. Finally in order to improve the job sequence on the machines the following procedure is launched when the jobs are scheduled. [8] [9]

5.1.1 Algorithm Doubleswap

In this paper we apply double swap to all the above mentioned algorithms, i.e. A1LST-A3LST as per following rules & notations. Preliminary tests showed that DoubleSwap procedure do not improve the schedules in all cases. Thus, in the sequel we will add a suffix ‘‘+DS’’ to the name of the algorithm which uses DoubleSwap. On the other hand, algorithm A2 without DoubleSwap is completely inefficient and we will not consider it as an efficient algorithm. Hence, we will consider five heuristic algorithms: A1LST, A1LST+DS, A2LST+DS with the computational complexity equal to O(nlogn) and A3LST, A3LST+DS with the computational complexity O(n2).[5][19]

Step1: For j=1,…..,n-1 swap jobs in positions j and j+1 if

$$l_{\pi_{1(j)}}(C_{\pi_{1(j)}}) + l_{\pi_{1(j+1)}}(C_{\pi_{1(j+1)}}) \quad (4)$$

$$> l_{\pi_{1(j+1)}}(C_{\pi_{1(j-1)}} + P_{\pi_{1(j+1)}}) + l_{\pi_{1(j)}}(C_{\pi_{1(j+1)}})$$

Step2: For j=n,…..,2 swap jobs in positions j and j-1 if

$$l_{\pi_{1(j)}}(C_{\pi_{1(j)}}) + l_{\pi_{1(j-1)}}(C_{\pi_{1(j-1)}}) \quad (5)$$

$$> l_{\pi_{1(j-1)}}(C_{\pi_{1(j)}}) + l_{\pi_{1(j)}}(C_{\pi_{1(j-2)}} + P_{\pi_{1(j)}})$$

VI ANT ALGORITHM

The ant colony optimisation meta-heuristic (ACO) was first described by Marco Dorigo in his PhD thesis, and was initially applied to the travelling salesman problem(TSP). It has since been applied to many other combinatorial optimisation problems, as well as other tough problems. It is inspired by the abilities of real ants to find the shortest path between their nest and a food source. Ant algorithm is a biological inspiration, and describes the mechanisms the approach uses to find good solutions to tough problems.[9]

6.1 Algorithm used in ACO

There is a pool of jobs having constant processing time (prt_j) and initial job value (C_j) but varying deterioration (d_j) depending on completion time of the jobs. The algorithm for parallel processor scheduling using Ant Colony Optimization may be represented as follows:

Step1. Initialize the pheromone $ph_{j,i}$ with initial pheromone value(tau0/τ_0). Pheromone $ph_{j,i}$ is accumulated between jobs and the deterioration interval. [10]

Step2. All ants start making its tour.

2a) Ants select a job first. Selection of job depends on two parameter- (1) processing time and (2) next deterioration, which is drive by a probability distribution function. Job j is selected for deterioration interval i according to formula [11]

$j = \max (phj, i * d_i^{1/2}/prt_j)$ if $q<q_0$

$j=p(j)=(ph_{j,i} * d_i^{1/2}/prt_j) / \sum(ph_{j,i} * d_i^{1/2}/prt_j)$ otherwise

Where, d_i=deterioration value for interval i (6)

prt_j = processing time for job j

2b) Ants select a processor (p) for selected job (j) according to formula

$$p= \max[(t_{pc}-t_{pi-1})/(t_{pi}-t_{pi-1}) / i_p]$$

Where, t_{pc} =completion time on processor p

t_{pi}= end point of interval i for processor p

i_p= interval for processor p

2c) Update the pheromone (local updation)

$$ph_{j,pi}=(1-\rho)* ph_{j,pi} +\rho *\tau_0$$

2d) Repeat steps '2a' thtough '2c' for all jobs.

Step3. Select the best tour among tours of all ants.

3a) Best tour with maximum job value is selected.

3b) Update the pheromone (global updation)

$$ph_{j,i}=(1-\alpha)* ph_{j,i} +\alpha *(i/D)$$

where, D= total deterioration in best tour

Repeat steps 1 to 3 for each iteration

6.2 Steps for Solve the problem by ACO

The basic steps, which are to be followed for solving the problem by ACO, are as follows [12]

- Represent the problem in the form of set of components and transitions, or by a set of weighted graphs, on which ants can build solutions.
- Define the meaning of the pheromone trail.
- Define the heuristic preference for the ant while constructing a solution.
- If possible implement a efficient local search algorithm for the problem to be solved.
- Choose a specific ACO algorithm and apply to problem being solved.
- Tune the parameter of the ACO algorithm.

6.3 Ant System

Ant System is the first ACO algorithm proposed in the literature. Its main characteristic is that at every iteration the pheromone values are updated by all the 'm' ants that built a solution in the iteration itself. The pheromone τ_{ij}, associated with the edge joining cities i and j, is updated as follows:[13]

$\tau_{ij} \leftarrow (1-\rho).\tau_{ij} + \sum_{k-1}^{m} \Delta_{ij}^{k}$

where ρ is the evaporation rate, m is the number of ants, and $\Delta\tau^{k}_{ij}$ is the quantity of pheromone laid on edge (i, j) by ant k:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if any k used edge } (i,j) \text{ in its tour,} \\ 0 \end{cases}$$

Where, Q is a constant, and L_k is the length of the tour constructed by ant k. In the construction of a solution, ants select the city to be visited through a stochastic mechanism. When ant 'k' is in city i and has so far constructed the partial solution s^p, the probability of going to city j is given by

$$p_{ij}^{k} = \begin{cases} \sum \frac{\tau_{ij}^{\alpha}.\pi_{ij}^{\beta}}{i,l \in N(iP)\tau_{ij}^{\alpha}.\pi_{ij}^{\beta}} & \text{if } c_{ij} \in N(S^p) \\ 0 & \text{otherwise,} \end{cases} \qquad (7)$$

where $N(s^p)$ is the set of feasible components i.e., edges (i,l) where l is a city not yet visited by the ant k. The parameters α and β control the relative importance of the pheromone versus the heuristic information η_{ij}, which is given by:

$\eta_{ij} = 1/d_{ij}$, where d_{ij} is the distance between cities i and j. [14]

6.3.1 MIN-MAX Ant System (MMAS)

This algorithm is an improvement over the original Ant System. Its characterizing elements are that only the best ant updates the pheromone trails and that the value of the pheromone is bound. The pheromone update is implemented as follows: [20]

$$\tau_{ij} \leftarrow \left[(i-\rho).\tau_{ij} + \Delta\tau_{ij}^{best}\right]_{\tau_{min}}^{\tau_{max}} \qquad (8)$$

where τ_{max} and τ_{min} are respectively the upper and lower bounds imposed on the pheromone; the operator $[x]_b^a$ is defined as:

$$[x]_b^a = \begin{cases} a & \text{if } x > a, \\ b & \text{if } x < b, \\ x & \text{otherwise;} \end{cases}$$

And $\Delta\,\tau_{ij}^{best}$ is:

$$\Delta\,\tau_{ij}^{best} = \begin{cases} 1/L_{best} & \text{if } (i,j) \text{ belongs to best tour} \\ 0 & \text{otherwise,} \end{cases} \quad (9)$$

where L_{best} is the length of the tour of the best ant. This may be (subject to the algorithm designer decision) either the best tour found in the current iteration—iteration-best, L_{ib} or the best solution found since the start of the algorithm i.e. best-so-far, L_{bs} or a combination of both. Concerning the lower and upper bounds on the pheromone values, τ_{min} and τ_{max}, they are typically obtained empirically and tuned on the specific problem considered. Nonetheless, some guidelines have been provided for defining τ_{min} and τ_{max} on basis of the analytical considerations defined earlier. [16]

6.4 Ant Colony System

The most interesting contribution of ACS is the introduction of a local pheromone update in addition to the pheromone update performed at the end of the construction process (called offline pheromone update). The local pheromone update is performed by all the ants after each construction step. Each ant applies it only to the last edge traversed: [17]

$$\tau_{ij} = (1 - \phi) \cdot \tau_{ij} + \phi \cdot \tau_0, \quad (10)$$

where $\phi \in (0, 1]$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone.[20]

The main goal of the local update is to diversify the search performed by subsequent ants during iteration by decreasing the pheromone concentration on the traversed edges, ants encourage subsequent ants to choose other edges and hence to produce different solutions. This makes it less likely that several ants produce identical solutions during one iteration. The offline pheromone update, similarly to MMAS, is applied at the end of each iteration by only one ant, which can be either the iteration-best or the best-so-far. However, the update formula is slightly different:

$$\tau_{ij} \leftarrow \begin{cases} (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} & \text{if } (i,j) \text{ belongs to best tour,} \\ \tau_{ij} & \text{otherwise} \end{cases} \quad (11)$$

As in MMAS, $\Delta\tau_{ij} = 1/L_{best}$, where L_{best} can be either L_{ib}or L_{bs}. Another important difference between Ant Colony System and Ant System is in the decision rule used by the ants during the construction process. [18][19]

In Ant Colony System, the so-called pseudorandom proportional rule is used, the probability for an ant to move from city i to city j depends on a random variable q uniformly distributed over [0,1], and a parameter q_0; if $q \leq q_0$, then

$$p_{ij}^k = \begin{cases} \sum \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{i,l \in N(iP) \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}} & \text{if } c_{ij} \in N(S^p) \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

VII RESULTS

This paper solves the problem of parallel processor scheduling. It is analyzed that the final results of parallel processor scheduling using ant colony optimization technique are better than the results of heuristics that have been already developed earlier. We test the system for up to 500 jobs that are executing on the parallel machines.

7.1 Results of Heuristics

	VAL 1	VAL 2	VAL 3	VAL 4	VAL 5
A1	7.897	8.9764	7.1234	9.8765	6.7453
A2	8.899	7.8976	8.9907	5.4532	6
A3	3.241	2.1342	4.3321	2.1311	4.6754
DA1	9.877	7.8564	7.3521	8.3409	6.4409
DA2	7.898	6.5643	5.3421	7.3452	8.9856
DA3	3.214	2.1045	3.1232	1.5643	4.6867
ACS1	2.132	2.1001	1.9987	1.3342	3.0934

Table 7.1 Result of Scheduling For 100 jobs

	VAL 1	VAL 2	VAL 3	VAL 4	VAL 5
A1	7.868	8.4532	7.8889	5.3421	9.5632
A2	8.896	6.7843	8.9786	6.2311	6.3444
A3	4.123	3.1232	3	1.9906	3.0568
DA1	9.453	6.5543	8.4251	9.1765	6.3245
DA2	8.923	8.1232	7.5643	4.1242	8.6421
DA3	3.123	3.1096	2.7865	1.9874	2.3421
ACS1	2.134	1.2321	2.3339	1.0078	2.2222

Table 7.2 Result of Scheduling For 200 jobs

	VAL 1	VAL 2	VAL 3	VAL 4	VAL 5
A1	4.563	6.6543	9.4532	6.6665	7.3421
A2	6.765	6.7888	7.5643	7.5643	7.3421
A3	5.675	3.2213	5.6543	4.5632	4.5621
DA1	8.895	8.6554	8.7865	9.7854	8.4654
DA2	9.112	6.3452	5.6754	7.7644	5.6754
DA3	4.445	3.2218	4.3211	2.1068	3.5632
ACS1	4.401	2.4321	3.3305	2.1005	2.5612

Table 7.3 Result of Scheduling For 300 jobs

	VAL 1	VAL 2	VAL 3	VAL 4	VAL 5
A1	7.675	8.9653	5.5643	7.5652	8.7651
A2	4.342	8.8865	7.6743	9.7633	7.7665
A3	3.342	4.3421	2.4311	6.6572	5.7622
DA1	8.46	6.6763	9.7897	7.0923	6.7687
DA2	9.856	9.7643	8.4321	8.8328	7.7832
DA3	3.332	4.4321	1.9087	5.5634	4.3421
ACS1	2.118	3.3211	1.0093	5.0012	4.3421

Table 7.4 Result of Scheduling For 400 jobs

	VAL 1	VAL 2	VAL 3	VAL 4	VAL 5
A1	9.876	6.9878	8.6782	7.8217	9.8923
A2	7.821	7.7823	6.7222	5.7323	8.8932
A3	4.819	3.0291	3.2811	2.9218	4.9021
DA1	7.219	8.8912	9.8921	8.8921	7.8921
DA2	8.821	9.9218	8.2187	7.8921	9.8922
DA3	4.123	3.0191	3.1111	2.1256	3.5421
ACS1	3.711	2.1272	1.4545	0.0919	0.8212

Table 7.5 Result of scheduling for 500 jobs

* RED COLOR: MAXIMUM LOSS VALUE SKY BLUE COLOR: MINIMUM LOSS VALUE

VIII CONCLUSION

In this paper the parallel machines scheduling problem with time dependent losses of job values has been investigated. It has been shown that the general version of the problem is strongly NP- hard and its single machine case is NP-hard .Therefore in order to solve the problem the use of branch and bound algorithm supported by some elimination properties which improve its efficiency. It delivers optimal solution of the instances of 25 jobs in a few minutes on the average, using some heurisitics algorithm with low computational complexities – O(nlogn) and $O(n^2)$.

The experimentally analysis has revealed , that their efficiencies strongly depend on the instance size and the values of the problem parameters.However ,the analysis has delivered also a manner of choosing an appropriate algorithm in order to obtain near solutions.

At last comparison is made with ant colony algorithms and we conclude that ant colony gives more optimized result as compare to heuristics.

REFERENCES

[1] Principles of sequencing and scheduling by Baker & Trietch, John Wiley & Sons 2009 ed.

[2] Scheduling Theory, Algorithms and Systems 3/e Michael L Pine do Springer Press.

[3] Bachman, A., Janiak,A.,2000. Minimizing maximum lateness under linear deterioration . European Journal of Operational Research 126,557-566.

[4] The Many Facts of Natural Computing by Lila Kari & Rozenberg, Communications of the ACM,October 2008 ,Volume 51 ,Number 10

[5] Adam Janiak ' Tomasz Krysiak ' Costas P. Pappis 'Theodore G. Voutsinas "A scheduling problem with job value given as a power function of their completion time"

[6] Ant Colony optimization Marco Dorigo, Mauro Birattari, and Thomas St¨utzle Universi´e Libre de Bruxelles

[7] Neumann, J. The Computer and the Brain. Yale University Press, 1958.

[8] Brunc, J.L., Coffman, E.G., Sethi,R.,1974. Scheduling independent tasks to reduce mean finishing time. "C.E., Kovalyov. M.Y., 1995. Single machine batch scheduling with deadlines and resources dependent processing times. Operations Research letters 17, 243-249.

[9] Koza, J. Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.

[10] Cheng T.C.E., Liu. Parallel machine scheduling to minimize the sum of quadratic completion times.

[11] Della Croce, F.,Szware. The Many Facets of Natural Computing by Lila Kari & Rozenberg, Communications of the ACM,October 2008 ,Volume 51 ,Number 10

[12] Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem by Marco Dorigo, Senior Member, IEEE, and Luca Maria Gambardella, Member, IEEE.

[13] Single processor scheduling using ant colony optimization, Reena Thakur & Patel, IEEE 2012

[14] Adam Janiak ' Tomasz Krysiak ' Costas P. Pappis 'Theodore G. Voutsinas "A scheduling problem with job value given as a power function of their completion time"

[15] Kuan Yeu Wong, Phen Chiak "A new minimum pheromone threshold strategy (MPTS) for max–min ant system" Applied Soft Computing, Elsevier June 2009

[16] J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels, "The self-organizing exploratory pattern of the Argentine ant," Journal of Insect Behavior, vol. 3, p. 159, 1990.

[17] S. Goss, S. Aron, J.-L. Deneubourg, and J.-M. Pasteels, "Self-organized shortcuts in the Argentine ant," Naturwissenschaften, vol. 76, pp. 579–581, 1989.

[18] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Dipartimento di Elettronica, Politecnico di Milano, Italy, Tech. Rep. 91-016, 1991.

[19] Adam Janiak ' Tomasz Krysiak ' Costas P. Pappis 'Theodore G. Voutsinas "A scheduling problem with job value given as a power function of their completion time" 2009 ed

[20] Sachin Shrivastava, Deepak Chaudhary, Kapil Shrivastava, "Ant Colony Optimization on scheduling problem with job values given as a power function of their completion times." International Journal of Scientific & Engineering Research, ISSN 2229-5518