

PROPOSING A NEW ALGORITHM FOR JOB SCHEDULING PROBLEM WITH-DEADLINE.

by Kaustubh Deokar

FILE	16BCE0353_16BIS0175_16BCE0457.PDF (31.82M)		
TIME SUBMITTED	20-APR-2017 04:26PM	WORD COUNT	2729
SUBMISSION ID	802013909	CHARACTER COUNT	14004



DATA STRUCTURES AND ALGORITHMS

PROJECT REVIEW 3

Topic:-

PROPOSING A NEW ALGORITHM FOR JOB
SCHEDULING PROBLEM WITH-DEADLINE.

Team members:-

16BCE0353-Kaustubh Deokar (kaustubh.santosh2016@vitstudent.ac.in)

16BIS0175- Rohan Singh (rohan.singh2016@vitstudent.ac.in)

16BCE0457-Jeevanjyothi Pradhan (jeevanjyoti.pradhan2016@vitstudent.ac.in)

ABSTRACT :-

Job scheduling is a problem which completes the allotment of the jobs such that it maximises the profit ,from a set of job with each having a specific deadline.The profit is only obtained if the job is completed before or on the deadline.

Our objective is to propose a new algorithm for job scheduling . This problem is solved recursive branch and bound . This problem is useful in various dimensions of human society and applications of these problems have become an important in the world of computer science.

The algorithm which is based on recursive branch and bound algorithm has a higher percentage of accuracy and a better computational time than the rest of the algorithms in the job sequencing field for eg.greedy and dynamic programming which are shown in the results and compilation time of all the above.

KEYWORD:-

Job scheduling, deadline,branch and bound,recursive,greedy and dynamic.

I. INTRODUCTION:-

The job scheduling originally has a long history ,which has been a major components in machine industry since it's revolution. As per our background, we can distinguish two main zones about Job Scheduling task:

We mainly can recognize two basic principle times about Job Schedulers:

The centralized computer period .Work Control Language (JCL) on IBM centralized computers. At first in view of JCL usefulness to deal with conditions, this time is embodied by the improvement of refined booking arrangements, (for example, Job_Entry_Subsystem_2/3) framing some portion of the frameworks administration and mechanization toolset on the centralized computer. The open frameworks time Current schedulers on an assortment of models and working frameworks. With standard booking devices restricted to orders, for example, at and bunch, the requirement for centralized computer standard employment schedulers has developed with the expanded selection of circulated figuring conditions.

The job scheduling problems has various approaches such as the greedy approach in which the best possible work is done first which was done to the large part before better approaches such as the dynamic approach came which provided better outputs or profits over a longer periods of time.The dynamic approach related to the job sequencing approach is then beaten by the algorithm given here as the recursive branch and bound algorithm.

The following deals with the job scheduling problem and present new resolution for it.

In the next section we review the literature that has been studied earlier, and section *iii*. new resolution is proposed for solving the algorithm. Section *iv* describes the results and comparison and the evaluation. In the section *v*. we conclude the results of computing these paper.

2. LITERATURE REVIEW:-

[1] provides us the best solution of similar problem so far. It applies the tabu-search using iterative combinatorial optimization in a different and notoriously difficult problem. It dominates all other approaches. In [2] they address problems related to only one machine and also study about the different parameters on their complexity and finally [8] give the polynomial-bounded algorithm for the required problem. Now in [3] it schedules jobs on a single machine to minimize the total weighted tardiness. It uses the 2 dynamic programming and 4 branch and bound algorithms so as to generate the cardinality order respectively. In this [4] reference This is the most important citation for our document. This part talks about various sorts of sequencing issues, and depicts a class of deterministic machine planning issues has been presented in this part. This section likewise manages and talks about the single machine, parallel machine and multi-operation issues in this class, separately. The two speculations of the deterministic machine-planning model have been exhibited in the part. In the deterministic model, one has consummate data, and exploiting it in limiting the acknowledgment of an execution measure may require exponential time. In global scheduling, all eligible tasks are stored in a single priority-ordered queue; the global scheduler selects for execution the highest priority tasks from this queue. Unfortunately, he uses approach with the optimal uniprocessor scheduling algorithms, such as those with rate monotonic (RM) and the earliest-deadline-first or the (EDF) algorithms, may soon result in arbitrarily very low processor utilization in various multiprocessor systems. All the theory part and the way Reference [5] Coffman and Graham also gave an algorithm which schedules the tasks preemptively. They use the ratio of the lengths and find the optimal solution and also check it by finding the ratio of two processors and equate it to 1. In [6] that is from a book. Problems of scheduling with deadlines are done with loss functions on k parallel machines whereas our problem stands for 1 machine without loss function. Also [7] contains some several straightforward heuristic algorithms for scheduling " n " independent tasks on " m " nonidentical processors is done.

In [8] they solved the job scheduling problem with preceding constraints within polynomial time into the NP-complete one, however a good algorithm for this is highly unlikely to exist. They also illustrated with correctness proofs also. [10] The scheduling in the cloud is responsible for selection of the best suitable resources for task performance, by considering some of the various factors like the parameter and restrictions of tasks into consideration. The user's view or perspective of smooth or efficient scheduling may be based on things like task completion time total task execution cost etc. Both of those service providers and also service consumers liked to ensure that the resources are utilized efficiently to give out their best capacity.

3.PROPOSED OBJECTIVE:-

The implementation of this paper is to trace a new method which is implemented by the recursive branch and bound solution for the job scheduling with the deadline problem. Then we would compare the algorithms (new and greedy, dynamic [old]) with each other and find out the space and time complexity and also based on the used memory. First we give information about the new algorithm.

3.1 RECURSIVE BRANCH AND BOUND (new algorithm)

Suppose there are n number of days and on each day there are m number of tasks which has a specified deadline. The way to approach this job sequencing problem is by branch and bound recursive algorithm as proposed. In these algorithm we arrange the m number of tasks which are to be done on each day on n days. Then the task which has the maximum output on the n th day is assigned on that day. After which all the tasks which had to be done on the n th day are shifted in the array on $(n-1)^{\text{th}}$ day's task list. Then out of all the tasks on $n-1$ th day the tasks which are added are also checked for the maximum output and then the task corresponding to maximum output will be chosen from these all tasks, similar thing will be followed for all the remaining jobs till the 1^{st} day. Hence we will get the maximum output.

That the maximum profit should be attained which is done assigning the tasks by the reverse order. First task of n th day is decided first by considering the maximum of that day and then rest of the tasks which can also be done on the $n-1$ th day are considered with tasks to be done on $n-1$ th day tasks, hence similarly all the remaining tasks except which are assigned will be considered in all the previous days so that the maximum of them are taken and eventually we get our maximum output. This is shown by the following algorithm in the table.

ALGORITHM:-

Table 1: recursive branch and bound algorithm

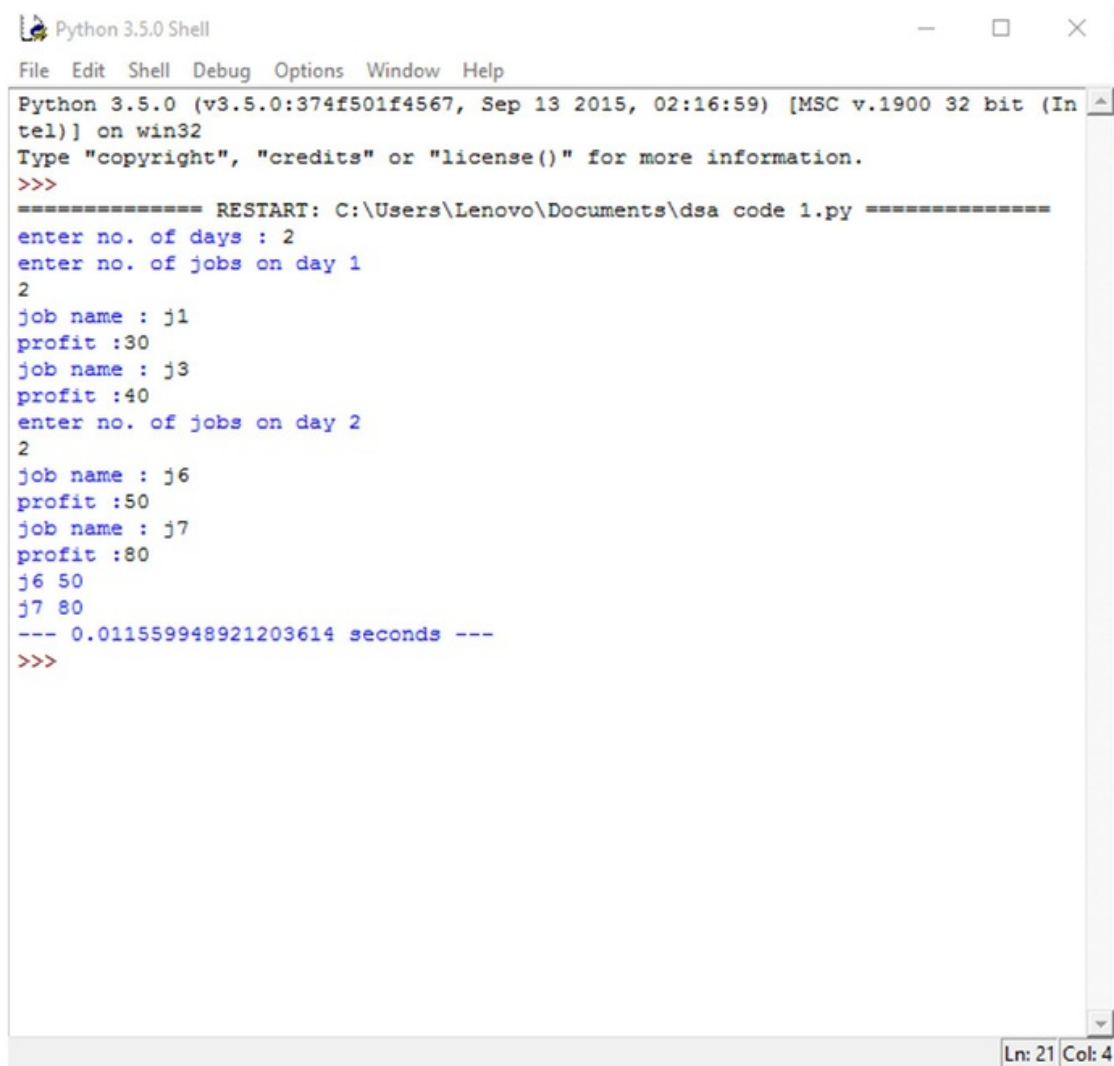
```
def job_optimise(jobs_on_day):  
    c = 0  
    t = 0  
    top_jobs = []  
    for i in jobs_on_day[::-1]:  
        t += i  
        k = max(jobs[len(jobs) - t + c:], key=lambda p: p.profit)  
        c += 1  
        jobs.remove(k)  
        top_jobs = [k] + top_jobs  
    for i in top_jobs:  
        print (i.job_name, i.profit)
```

Table 2:- quasi code for recursive branch and bound algorithm

1. Start
2. Choose the job with the maximum profit on an nth day
3. Repeat the steps 4 to 6 till the 1st day
4. Except for the job that is chosen transfer all the jobs on an nth day to the (n-1)th day task
5. Choose the maximum element of this array.
6. Store the job resulting in maximum profit in another array
7. Display the job with the profit in the ascending order of the day.

The similar technique was adopted but the only difference was the consideration of the initial point was from the day 1 as opposite to the nth day here hence the test case was ignored where the two of the jobs having the maximum profit were included in the last day. Hence this and other test cases led us to the idea about the above-proposed algorithm.

Output :-



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Lenovo\Documents\dsa code 1.py =====
enter no. of days : 2
enter no. of jobs on day 1
2
job name : j1
profit :30
job name : j3
profit :40
enter no. of jobs on day 2
2
job name : j6
profit :50
job name : j7
profit :80
j6 50
j7 80
--- 0.011559948921203614 seconds ---
>>>
```

Ln: 21 Col: 4

4. RESULTS AND DISCUSSIONS:-

Now to compare we have given two more algorithms for the same problem.

4.1 GREEDY PROGRAMMING:-

It is not a global optimum solution to any problem.

It does not look at question globally but it gives the best immediate solution to any problem useful in many cases where objects or constraints are uncertain. Generally $O(n)$ complexity, easy to implement and interpret results. It often requires sorting the data first, which is $O(n \lg n)$. In some cases (not all), greedy algorithms provide global optimum solutions (shortest paths, some job scheduling problems). In most cases they are approximate algorithms. Sometimes used as a subset of an exact algorithm (e.g., as a relaxation in an integer programming algorithm). Each job takes 1 unit of time. Here the profit will only be earned if the job is completed on or before the deadline, here there is no fraction issue here as 0/1 knapsack problem. "Profit" can be the priority of the task in a real time system that discards tasks that cannot be completed by their deadline. We want to find the list of jobs that maximizes our profit. Example use in telecom engineering and construction scheduling – Many small jobs, "profit" proportional to customers served – This is then combined with non-integer programming solution for big jobs. Greedy also used in how many machines/people problems.

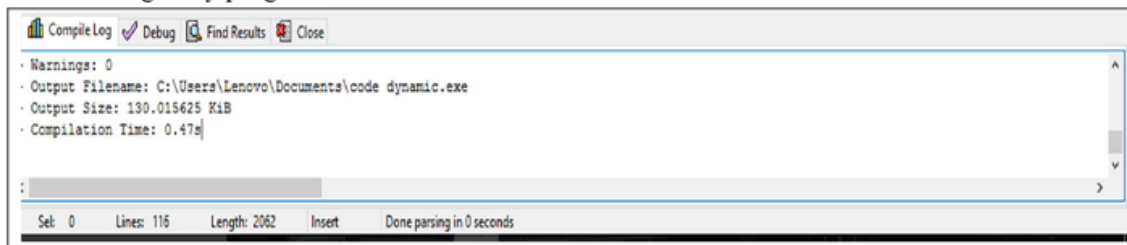
Table no 3:-quasi code for Greedy Algorithm.

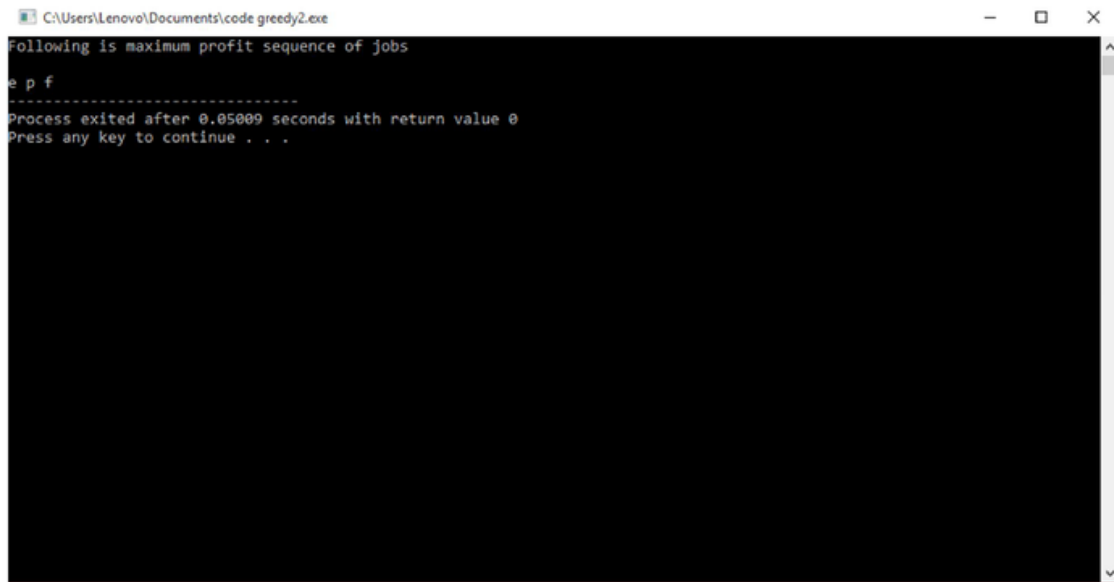
```
Start
Struct job
{
int profit;
char job_no[3];
};
enter structure job no and profit for n numbes;
for(i=0;i<n;i++)
sort(job_no[profit]);
```



```
Set a counter dmax=maximum deadline .
Set counter i=1, counter=0, k=0.
while(time_slot(dmax)!=NULL)
{
    i=1;
    k=min(dmax,deadline(i));
    while(k>=1)
    {
        if(time_slot(k)!=NULL)
            status[counter++]=job1[i].job_no(i);
        else
            k--;
    }
    i++;
}
```

Result for greedy program:-





4.2 DYNAMIC PROGRAMMING:-

Dynamic programming also known as the careful brute force is one of the basic algorithm to solve the complex problems related to data structures and other real life problems. It solves the problems by breaking them down into smaller problems and solving each subproblem just once and storing their output in such a way that if the next time the same subproblem occurs, instead of computing that subproblem once again we look up for the previously computed subproblem and save our time. Each time the sub problem's solution is indexed in such a way that we can easily call it with the values this process is called "memoization". This differs brute force from dynamic programming.

Table no 4:-code dynamic programming:-

```
//free time slots initially set to -1 [-1 denotes EMPTY]
for(i = 1; i <= dmax; i++) {timeslot[i] = -1; }
printf("dmax: %d\n", dmax);
for(i = 1; i <= n; i++)
    {k = minValue(dmax, jobs[i - 1].deadline);
    while(k >= 1) {
        if(timeslot[k] == -1) {
```

```

        timeslot[k] = i-1;
        filledTimeSlot++;
        break;} k--;}
//if all time slots are filled then stop
        if(filledTimeSlot == dmax) {break; }}
//required jobs
        printf("\nRequired Jobs: ");
        for(i = 1; i <= dmax; i++) {
            printf("%s", jobs[timeslot[i]].id);
            if(i < dmax) {printf(" --> ");}}
//required profit
        maxprofit = 0;
        for(i = 1; i <= dmax; i++)
            {maxprofit += jobs[timeslot[i]].profit;}
        printf("\nMax Profit: %d\n", maxprofit);}

```

Table 4 -quasi code for dynamic programming:-

```

1. Initially sort all the jobs according to their finish time.
2. Now apply the following recursive processes.
   //here the jobs[] is the array of all n jobs with profit and deadline.
Find maximum profit(jobs[],n)
{
    If (n=4) return jobs[0] //maximum profit only one element entered
    else return the maximum of the following 2 profits.
    1.maximum profit by excluding the traversing current job
    so, find maximum profit(jobs, n-1)
    or by
    2.maximum profit including the current jobs
}
3.) Now give the total output as maximum profit.
4.) stop

```

Compile Log
 Debug
 Find Results
 Close

- Warnings: 0
- Output Filename: C:\Users\Lenovo\Documents\code greesy.exe
- Output Size: 130.21875 KiB
- Compilation Time: 0.25s

Sel: 0 Lines: 56 Length: 1515 Insert Done parsing in 0.078 seconds

C:\Users\Lenovo\Documents\code dynamic.exe

Job	Deadline	Profit
j2	1	100
j1	2	60
j4	2	40
j3	3	20
j5	1	20

dmax: 3

Required Jobs: j2 --> j1 --> j3

Max Profit: 180

.....

Process exited after 0.01605 seconds with return value 0

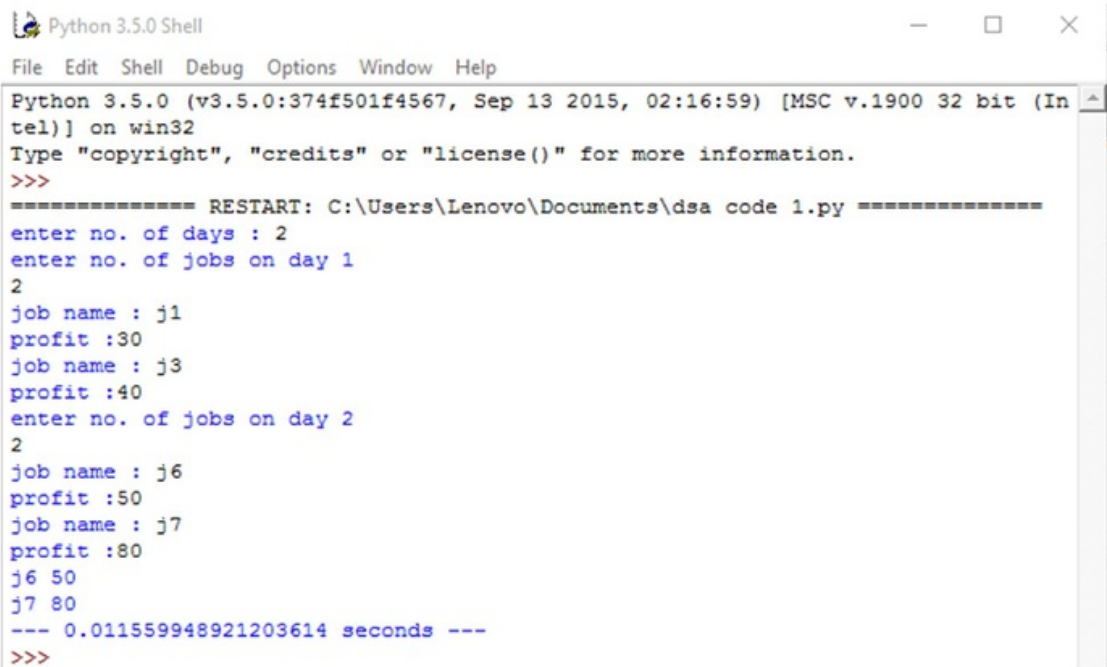
Press any key to continue . . .

Hence we see that the code for the optimal programming takes the minimum running time.

Hence we can conclude the following conclusion.

4.3 RUNNING TIME ANALYSIS:-

DYNAMIC	GREEDY	OPTIMAL
$O(\sum_{i=2}^i n^i t) * n^2)$	$O(n \log n)$	$O(n^2)$



```
Python 3.5.0 Shell
File Edit Shell Debug Options Window Help
Python 3.5.0 (v3.5.0:374f501f4567, Sep 13 2015, 02:16:59) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Lenovo\Documents\dsa code 1.py =====
enter no. of days : 2
enter no. of jobs on day 1
2
job name : j1
profit :30
job name : j3
profit :40
enter no. of jobs on day 2
2
job name : j6
profit :50
job name : j7
profit :80
j6 50
j7 80
--- 0.011559948921203614 seconds ---
>>>
```

7

5.CONCLUSION AND FUTURE WORK:-

In this paper we proposed a new algorithm on the base of branch and bound method for job sequencing with deadline which we initialized from the end and assigned the job one by one. As we see in the results and discussion we have presented both the ways other than new programming (greedy and dynamic programming) and we have shown that it is not an Optimal solution to any problem and other than that it is not always as accurate and reliable, Faster as the branch and bound algorithm that is proved by the results obtained by the compilation time or the running time of the code in languages of c language, c++ language and python language used in all three respectively. As a result of this the profit is maximized in a universal way i.e. with any conditions .In the future we can find such methods for solving this that will result in more speed and accuracy.

6.REFERENCES:-

1. Dell'Amico, Mauro, and Marco Trubian. "Applying tabu search to the job-shop scheduling problem." *Annals of Operations research* 41.3 (1993): 231-252.
2. Lenstra, Jan Karel, AHG Rinnooy Kan, and Peter Brucker. "Complexity of machine scheduling problems." *Annals of discrete mathematics* 1 (1977): 343-362.
3. Abdul-Razaq, T. S., Chris N. Potts, and Luk N. Van Wassenhove. "A survey of algorithms for the single machine total weighted tardiness scheduling problem." *Discrete Applied Mathematics* 26.2-3 (1990): 235-253.
4. Lawler, Eugene L., et al. "Sequencing and scheduling: Algorithms and complexity." *Handbooks in operations research and management science* 4 (1993): 445-522.
5. <https://pdfs.semanticscholar.org/0ff3/4d58bfdba172ed569f41e2eb7390789bbd30.pdf>
Traditionally, there have been two approaches for scheduling periodic task systems on multiprocessors: partitioning and global scheduling.
6. Lam, Shui, and Ravi Sethi. "Worst case analysis of two scheduling algorithms." *SIAM Journal on Computing* 6.3 (1977): 518-536.
7. Englewood Cliffs, N.J. : Prentice-Hall, 1963, Chapter 5 170. J. RIORDAN, Stochastic Service Systems. New York: John Wiley, 1962 171. J.G. ROOT,
8. Ibarra, Oscar H., and Chul E. Kim. "Heuristic algorithms for scheduling independent tasks on nonidentical processors." *Journal of the ACM (JACM)* 24.2 (1977): 280-289.
9. Hwang, Jing-Jang, et al. "Scheduling precedence graphs in systems with interprocessor communication times." *SIAM Journal on Computing* 18.2 (1989): 244-257.
10. S. Selvi, R. Maheswari, Dr. B. Kalaavathi. "Deadline – Cost Based Job Scheduling Using Greedy Approach in a Multi-Layer Environment". *International Journal of Computer Trends and Technology (IJCTT)* V11(2):74-79, May 2014. ISSN:2231-2803. www.ijcttjournal.org.
Published by Seventh Sense Research Group
11. (2017) Adaptive Scheduling of Task Graphs with Dynamic Resilience. *IEEE Transactions on Computers* 66:1, 17-23. CrossRef
12. S. C. Graves, A. H. G. Rinnooy Kan, P. H. Zipkin, "Mathematical programming models and methods for production planning and scheduling" in *Handbooks in Operations Research and Management Science Volume 4 Logistics of Production and Inventory*, North-Holland, 1993.

13.J. J. McCall, "Maintenance policies for stochastically failing equipment: a survey", *Manage. Sci.*, vol. 11, no. 5, pp. 493-524, 1965.

14. R. L. Daniels, P. Kouvelis, "Robust scheduling to hedge against processing time uncertainty in single-stage production", *Manage. Sci.*, vol. 41, no. 2, pp. 363-376, 1995.

15.J. S. Burton, A. Banerjee, C. Sylla, "A simulation study of sequencing and maintenance decisions in a dynamic job shop", *Comput. Industrial Eng.*, vol. 17, no.

PROPOSING A NEW ALGORITHM FOR JOB SCHEDULING PROBLEM WITH-DEADLINE.

ORIGINALITY REPORT

15%

SIMILARITY INDEX

14%

INTERNET SOURCES

5%

PUBLICATIONS

5%

STUDENT PAPERS

PRIMARY SOURCES

1

ocw.unisbank.ac.id

Internet Source

7%

2

www.scribd.com

Internet Source

3%

3

Submitted to Indian Institute of Technology
Roorkee

Student Paper

2%

4

www.coursehero.com

Internet Source

1%

5

www.citeulike.org

Internet Source

1%

6

doaj.org

Internet Source

1%

7

ce-publications.et.tudelft.nl

Internet Source

1%

8

Congram, Richard K. Potts, Chris N. van . "An
iterated dynasearch algorithm for the single-
machine total weighted tardiness scheduling

1%

problem.", INFORMS Journal on Computing, Wntr 2002 Issue

Publication

EXCLUDE QUOTES ON

EXCLUDE
BIBLIOGRAPHY ON

EXCLUDE MATCHES < 10 WORDS