Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.elsevier.com/locate/dcan

Dynamic scheduling and analysis of real time systems with multiprocessors

M.D. Nashid Anjum, Honggang Wang*

University of Massachusetts Dartmouth, 285 Old Westport Rd, Dartmouth, MA 02747, USA

ARTICLE INFO

Article history:
Received 15 June 2016
Accepted 27 June 2016
Available online 18 July 2016

Keywords:
Job-shop scheduling problems
JSP
LPT
SPT
LS
EDD
Tube-tap
MINLP

ABSTRACT

This research work considers a scenario of cloud computing job-shop scheduling problems. We consider *m* realtime jobs with various lengths and *n* machines with different computational speeds and costs. Each job has a deadline to be met, and the profit of processing a packet of a job differs from other jobs. Moreover, considered deadlines are either hard or soft and a penalty is applied if a deadline is missed where the penalty is considered as an exponential function of time. The scheduling problem has been formulated as a mixed integer non-linear programming problem whose objective is to maximize net-profit. The formulated problem is computationally hard and not solvable in deterministic polynomial time. This research work proposes an algorithm named the Tube-tap algorithm as a solution to this scheduling optimization problem. Extensive simulation shows that the proposed algorithm outperforms existing solutions in terms of maximizing net-profit and preserving deadlines.

1. Introduction

The rapid growth of realtime services and complex commercial strategies of cloud computing makes the scheduling problem a crucial challenge. In the literature of computer science the problem of scheduling multiple jobs (or tasks) on multiple machines (or processors) has been found very crucial and challenging. In terms of computer science jargon this type of optimization problems is known as job-shop scheduling problems (JSP) [1,2]. A number of variants of JSP are available in the literature focusing on different objectives and constraints. This research work considers a cloud computing scenario of real-time dynamic job-shop scheduling where multiple jobs need to be scheduled on multiple processors (i.e., machines) to maximize the net profit. The problem scenario assumes that each job has a deadline to be met, each job may have different job lengths in terms of bits, and the profit of processing a packet of one job differs from the other jobs. It is also considered that each machine may have a different processing rate (bit/s) and processing cost. The cost of processing a job on a machine per time unit may differ from one machine to another machine. The goal is to distribute the loads of the jobs to multiple machines in such a way that meets all the deadlines and maximizes the net profit, i.e., minimizing the overall processing cost.

The problem formulation considers both the hard-realtime deadlines [3] and the soft-realtime deadlines [4]. In the case of the hard-realtime deadline the execution time of a job must not exceed the given deadline. Hence, the scheduling makespan needs to meet the deadlines of each job. If the execution time of a job fails to meet the deadline, no reward or profit is gained for processing this job. On the other hand, a soft-realtime deadline allows the execution time to exceed the deadline. Hence, the scheduling makespan does not necessarily meet all the deadlines. However, the soft-deadline concept introduces a penalty function. If the makespan fails to meet the deadline of a job, a penalty is applied. The penalty is non-negative and a function of execution time. If the execution time is less than the deadline the value of the penalty is zero. But the value of the penalty keeps increasing if the execution time exceeds the deadline.

This paper formulates the scheduling optimization problem as a mixed integer programming (MIP) problem [5]. At first, the problem formulation considers only hard-deadlines. Later, the problem is extended considering soft-deadlines. The formulated optimization problem for hard-deadlines is basically a mixed integer linear programming (MILP) problem [6]. This paper considers an exponential function for the penalty and thus the formulated problem that considers the soft-deadlines is basically a mixed integer non-linear programming (MINLP) optimization problem. The MINLP problem is practically very difficult to solve, because it combines the combinatorial nature of mixed integer programming (MIP) and the difficulty in solving nonconvex (and even convex) nonlinear programming (NLP) [6].

* Corresponding author.
E-mail addresses: manjum@umassd.edu (M.D. Nashid Anjum), hwang1@umassd.edu (H. Wang).

http://dx.doi.org/10.1016/j.dcan.2016.06.004

A number of solutions are available in the literature to solve the multiprocessor job-shop scheduling problems. The solutions are precisely heuristic because of the hardness of the optimization problem. A major number of the existing job-shop scheduling algorithms deal with distributing multiple jobs on multiple machines, but in general they consider identical machines disregarding different processing rates and different processing costs of the machines. Few of them consider heterogeneous multiprocessors and real-time jobs. Primarily the existing solutions focus on minimizing the makespan and computational complexity. The existing literature has yet to find a solution which addresses the problem of maximizing the net profit of a dynamic real-time multiprocessor job-shop scheduling optimization problem considering machines with different processing rates and costs and on top of hard and soft-deadlines with penalty function.

This paper proposes an algorithm named the Tube-tap algorithm as a solution to the formulated optimization problem and an extensive simulation is carried out to compare the performance of the proposed algorithm with basic existing solutions. Simulation results show that the proposed algorithm outperforms the existing solutions.

The rest of the paper is organized as follows – Section 2 deals with related works, Section 3 explains the system model, the mathematical representation of the scheduling optimization problem is formulated in Section 4, Section 5 discusses the proposed solution, simulation results are explained in Section 6, finally the concluding remarks are drawn in Section 7.

2. Related work

A number of research works regarding the job-shop scheduling problem have been discussed in computer science and computational literature. Different versions of JSPs are introduced by different researchers focusing on distinct objective functions and constraints. The vast majority of them deal with identical multiprocessors. The most common algorithms for identical multiprocessor JSP are List-scheduling (LS) [7], Longest processing time (LPT) [8,9], Shortest processing time (SPT) [10], Weighted Shortest Processing Time (WSPT) [11], Earliest Deadline First (EDF) or Earliest Due Date (EDD) [12,13], Minimum Slack Time (MST) [14], etc. List-scheduling (LS) considers n jobs in some fixed orders and assigns the job j to the machine whose load is smallest so far. Longest processing time (LPT) sorts n jobs in descending order of processing time, and then runs the list scheduling algorithm. Shortest processing time (SPT) sorts n jobs in ascending order of processing time, and then runs the list scheduling algorithm. Earliest Due Date (EDD) sorts n jobs to be done from the job with the earliest due date to the job with the latest due date. Variants of the EDF algorithm and their applications are widely discussed in [15]. Weighted Shortest Processing Time (WSPT) sorts the jobs in non-decreasing ratio of processing time to importance weight [11]. Minimum Slack Time (MST) scheduling assigns priority based on the slack time of a process. Slack time is defined as the amount of time left after a job to meet the deadline [14].

Hodgson's Algorithm minimizes the number of tardy jobs on multiple parallel machines [16]. First it computes the tardiness of all jobs. If a tardy job is found at the kth position then the algorithm finds the LPT (longest processing time) in between position 1 and k. Johnson's Algorithm provides less computation but it works optimally only for two machines [17,18]. A genetic algorithm is proposed to solve the cloud computing scheduling in [19], where a cost function is introduced for late execution. A hybrid genetic algorithm is proposed in [20], where the job scheduling is executed using a priority rule and the priorities are defined by the genetic algorithm. In [21], the authors introduce a multiprocessor scheduling framework which integrates hard and soft real-time jobs and best-effort jobs. The focus of this work is to reduce tardiness and to improve response time of best-effort jobs by utilizing dynamic slack reclamation.

However the above-mentioned research works do not precisely address the cloud computing job-shop scheduling problem considering multiple machines with different processing speeds and processing costs, different rates of profits for different jobs with various lengths, and the penalty function for soft deadlines. This motivates us to precisely address the cloud-computing job-shop scheduling problem using a proper mathematical representation and find a realistic solution which offers less computation complexity and better net-profit.

3. System model

The problem scenario is depicted in Fig. 1. It shows there are n machines on the server-side with different processing rates (r) and processing costs (c). On the client side, there are m jobs where each one of them has different delay constraints (T), data sizes (L) and profits (p) per packet. The packet size (s) of a job may differ from others.

The notations of the presented system model are described as follows:

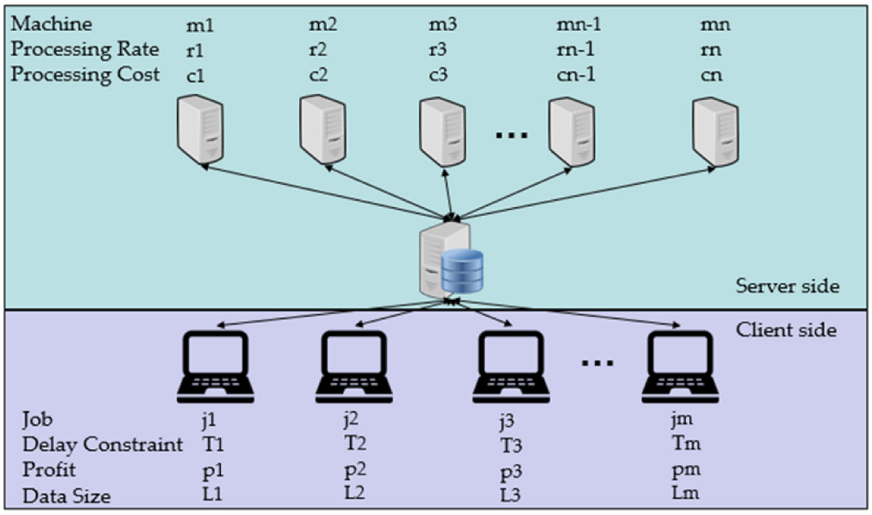

Fig. 1. System model of multiprocessor job distribution scenario.

r_i=data processing rate by the ith machine (bits/time)
$r_1 < r_2 < r_3 < \cdots < r_n$
c_i=Cost of the ith machine per unit time
$c_1 < c_2 < c_3 < \cdots < c_n$
l_{ij}=the portion (bits) of the jth job done by the ith machine
L_j=Total length of the jth job (packets)
p_j=profit of processing a packet of the jth job per packet
s_j=packet size of the jth job
$\frac{c_i}{r_i} < \frac{p_j}{s_j}$, $\forall i \, \forall j$
$p_{ij} = \frac{c_i}{r_i}$=profit of processing a bit of the jth job on the ith machine
T_j=deadline of processing the jth job
$i \in \{1, 2, 3, \ldots, n\}$; $j \in \{1, 2, 3, \ldots, m\}$

The deadlines can be either hard or soft. It is assumed that a job is schedulable within the corresponding given deadline.

4. Problem formulation

In this section we formulate the job-shop scheduling problem as a mixed integer programming problem, where the objective function is to maximize the net profit. Initially the problem considers only hard-deadlines. This implies that if the deadline of a job is not met, the profit for executing that job will be zero. Later, the problem will be extended for soft-deadlines too. In the case of soft-deadlines, if the deadline of a job is missed, a penalty will be applied for late execution. The penalty is a function of time. The amount of penalty increases as the lateness (i.e., delay) of the execution increases.

4.1. Problem formulation considering the hard-deadline

In this section we formulate the job scheduling problem as a mixed integer programming problem, where the objective function is to maximize the net profit. As described in Eq. (8), the constraints are as follows: (i) all deadlines are required to be met and (ii) all jobs should be done completely, not partially.

4.1.1. Assumptions

It is assumed that

- All deadlines are hard. Mathematically, if $\sum_{i=1}^{n} \frac{l_{ij}}{r_j} > T_j$ then $\sum_{i=1}^{n} l_{ij}(p_{ij}) = 0;\ \forall j$
- All jobs are schedulable. It implies $\frac{L_j}{\sum_{i=1}^{n} r_i} \le T_j;\ \forall j$
- $\frac{c_i}{r_i} < \frac{p_j}{s_j};\ \forall i, \forall j$
- $p_{ij} > 0;\ \forall i, \forall j$

4.1.2. Formulation

Objective function:

$$\text{Maximize} \quad z = \sum_{j=1}^{m} \sum_{i=1}^{n} l_{ij}(p_{ij}) \tag{1}$$

Subject to:

$$\sum_{j=1}^{m} \sum_{i=1}^{n} \frac{l_{ij}}{r_j} \le T_j; \quad \forall j \tag{2}$$

$$\sum_{i=1}^{n} \frac{l_{ij}}{s_j} = L_j; \quad \forall j \tag{3}$$

Eq. (2) represents the first constraint of the optimization problem that indicates the total processing time of a jth job on different machines cannot be greater than its deadline, T_j. The second constraint given in Eq. (9) means portions of the jth job on different machines should be equal to its job length L_j.

4.2. Problem extension considering soft-deadlines

4.2.1. Assumptions

- All deadlines are soft.
- All jobs are not necessarily schedulable.
- $\frac{c_i}{r_i} < \frac{p_j}{s_j};\ \forall i, \forall j$
- $p_{ij} > 0;\ \forall i, \forall j$

4.2.2. Penalty function

Rate of penalty increases as the delay increases. The function of penalty rate is chosen as follows:

$$\phi_{rate}^{j}(\tau_j) = \begin{cases} \alpha_j p_j e^{\beta_j \tau_j}, & \text{if } \tau_j > 0 \\ 0, & \text{if } \tau_j \le 0 \end{cases} \tag{4}$$

$$\tau_j = t_j - T_j \tag{5}$$

where t_j represents the execution time of jth job.

The total penalty over τ_j second delay for the jth job is calculated by

$$\phi_{total}^{j} = \sum_{k=1}^{\tau_j} \phi_{rate}^{j}(\tau_j) \tag{6}$$

Hence, the net penalty for m jobs after execution of the entire process is

$$\sum_{j=1}^{m} \phi_{total}^{j} = \sum_{j=1}^{m} \sum_{k=1}^{\tau_j} \phi_{rate}^{j}(\tau_j) \tag{7}$$

4.2.3. Formulation

Objective function:

$$\text{Maximize} \quad z = \sum_{j=1}^{m} \sum_{i=1}^{n} l_{ij}(p_{ij}) - \sum_{j=1}^{m} \phi_{total}^{j} \tag{8}$$

Subject to:

$$\sum_{i=1}^{n} \frac{l_{ij}}{s_j} = L_j; \quad \forall j \tag{9}$$

5. Proposed solution

A novel algorithm is proposed to solve the formulated scheduling optimization problem. Pseudo-code of the proposed algorithm is given in Algorithm 1. The basic idea of the proposed solution is simplified in Figs. 2 and 3. Fig. 2 shows there are three jobs $j1$, $j2$, and $j3$ sorted according to their deadlines $T1$, $T2$, and $T3$. The jobs have different sizes in terms of bits presented by different colors. This example considers that all jobs have hard-deadlines and all jobs are schedulable. The machines $M1$, $M2$, $M3$, and $M4$ are presented by tubes and sorted according to their cost per bit. The tubes are connected through taps at

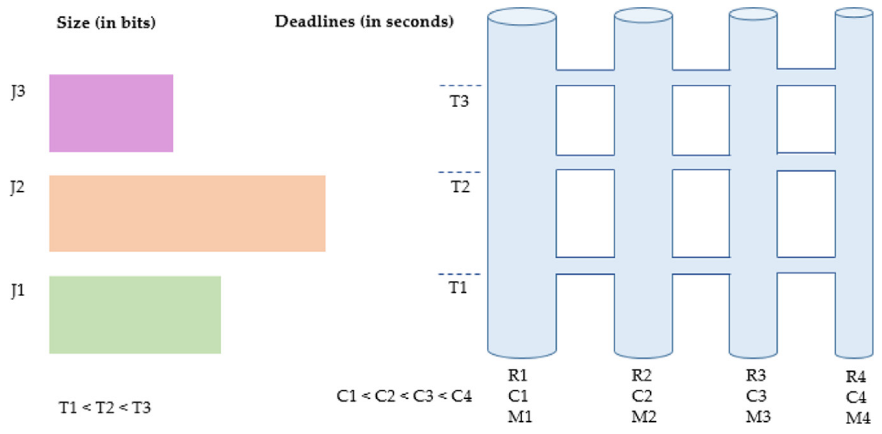

Fig. 2. Initializing the Tube-tap algorithm: (1) Jobs are sorted in ascending order with respect to deadlines. (2) Machines are sorted in ascending order with respect to cost/bit.

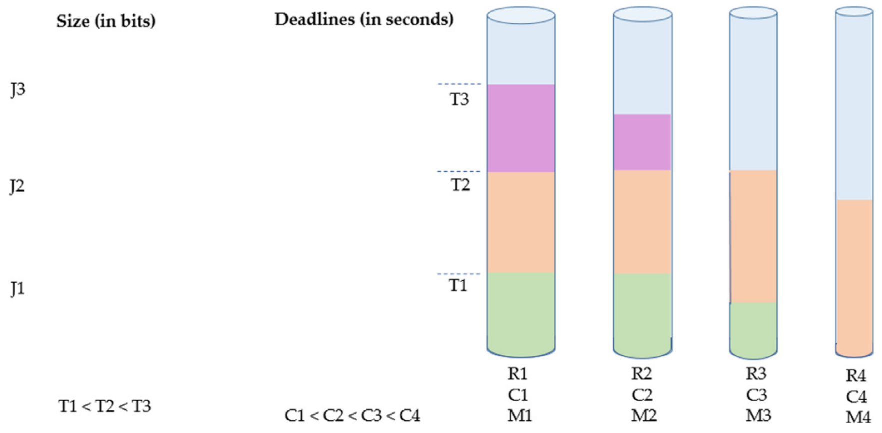

Fig. 3. Portions of different jobs distributed among different machines.

each deadline level. The width of the tube represents the processing rate of the corresponding machine and height represents the execution time. The taps are initially kept open. At first, the job *J*1 with the lowest deadline *T*1 is poured through tube *M*1. It fills the tube up to the level *T*1, then it overflows through the connected tap and fills the *M*2, and *M*3 tubes up to the levels marked by the green color. The summation of the portion of bits of Job *J*1 distributed among the machines represented by green color is equal to the job size of the *J*1. After the distribution of job *J*1 the taps at the level *T*1 are closed and *J*2 is then poured through the tube *M*1. The portion of *J*2 distributed among the tubes is presented by orange color. All the taps at level *T*2 are closed as soon as the distribution of *J*2 is done. Next, the same process is done for *J*3 which is marked by the light-blue color. Fig. 3 depicts which machine processes what portion of a job. It also indicates when a portion of a job starts and ends on different machines. Though the example shown in Figs. 2 and 3 considers the hard deadlines, the proposed algorithm is capable of dealing both hard-deadlines and soft-deadlines. To do so the proposed algorithm coins a variable named 'Threshold'. The value of the threshold is equal to the given deadline for hard-deadline jobs but the value of 'Threshold' is updated for each job in the case of soft-deadlines. The pseudo-code snippet mentioned in line-17 to line-20 in Algorithm 1 deals with only soft-deadline jobs. The rest of this section deals with some characteristics of the proposed Tube-tap algorithm.

Algorithm 1. Proposed Tube-tap algorithm.

```
1:  $sort\_c$;   {machines sorted with respect to cost/bit in ascending order}
2:  $sort\_T$;   {jobs sorted with respect to deadline in ascending order}
3:  $loads = zeros(m,n)$;   {portion of $m$th job done on $n$th tube}
4:  $job\_machine = zeros(m,n)$   {portion of $j$th job done on $i$th machine}
5:  $penalty\_TT = zeros(1,m)$;
6:  $time\_count = zeros(1,n)$;
7:  $time\_table = zeros(m,n)$;
8:  Initialize $[1 \times m]$ matrix: $\alpha$;   {penalty scaling factor}
9:  Initialize $[1 \times m]$ matrix: $\beta$;   {penalty exponential factor}
10: $L\_bit = L. * packet\_size$;   {job length in bits}
11: $L\_temp = 0$
12: for $j = 1{:}1{:}m$ do
13:     $L\_temp = L\_bit(sort\_T(j))$
14:     $i = 1$
15:     $flag = 0$
16:     $Threshold = sort\_T(j)$;   {updates the $j$th threshold of the tube}
17:     $y = \sum_{j=1}^{m} (Threshold - time\_count(i).*r(sort\_c(i))$;
18:     if $L\_temp > y$ then
19:         $Threshold = Threshold + \frac{(L\_temp - y)}{\sum_{i=1}^{n}(r)}$;
20:     end if
21:     while $flag < 1$ do
22:         $x = (Threshold - time\_count(sort\_c(i)))_{*}r(sort\_c(i))$
23:         if $L\_temp <= x$ then
24:             $loads(j, i) = L\_temp$
```

```
25:       else
26:          $loads(j, i) = x$
27:       end if
28:       $L\_temp = L\_temp - loads(j, i)$
29:       $job\_machine(sort\_T(j), sort\_c(i)) = loads(j, i)$
30:       $time\_count(sort\_c(i)) = time\_count(sort\_c(i)) + \frac{loads(j)}{r(sort\_c(i))}$;
31:       $time\_table(j, i) = \frac{loads\_TT(j,i)}{r(sort\_c(i))}$
32:       $time\_cum(i) = \sum_{j=1}^{m} time\_table(j, i)$
33:       $time(sort\_T(j)) = \max(time\_cum)$;      {Execution time of jth job}
34:       if $L\_temp == 0$ then
35:          $flag = 1$
36:       end if
37:       $i = i+1$
38:    end while
39: end for
```

5.1. Properties of Tube-tap algorithm

5.1.1. Penalty calculation

The pseudo-code for calculating the penalty for missing the soft-deadline is given as follows:

```
1.  $\tau = time - T$
2.  FOR $k=1:1:m$
3.     IF $\tau(k) > 0$
4.        FOR $l=1:1:tau(k)$
5.           STATE $penalty(k) = penalty(k) + \alpha(k)_* p(k)_* e^{\beta(k)_* l}$;
6.        ENDFOR
7.     ENDIF
8.  ENDFOR
```

5.1.2. Profit calculation for soft-deadlines

The net-profit is calculated using following pseudo-code:

```
1.  FOR $j=1:1:m$
2.     $profit(j) = \sum_{i=1}^{n}\left[\frac{job\_machine(j,i)}{packet\_size(j)} * p(j)\right] - \sum_{i=1}^{n}\left[\frac{job\_machine(j,i)}{r(i)} * c(i)\right]$;
3.  ENDFOR
4.  $net\_profit = \sum_{j=1}^{m}[profit(j) - penalty(j)]$
```

5.1.3. Profit calculation for hard-deadlines

Pseudo-code for calculating net-profit for jobs with hard-deadlines:

```
1.  FOR $x=1:1:m$
2.     IF $time(x) > T(x)$
3.        $loss(x) = p(x) * L(x)$;
4.     ENDIF
5.  ENDFOR
6.  $net\_profit = \sum_{j=1}^{m}[profit(j) - loss(j)]$
```

5.1.4. Computational complexity

Theorem 5.1. *The time complexity of the proposed Tube-tap algorithm is order of $O(n^2)$.*

Proof. Time complexity of the Tube-tap algorithm is derived from analyzing the pseudo-code given in Algorithm 1.

The algorithm contains two sorted arrays and two nested loops. The nested loop is composed of a FOR LOOP and an embedded WHILE LOOP. The complexity of a single FOR LOOP is order of $O(n)$. As like FOR LOOP the complexity of a single WHILE LOOP is also an order of $O(n)$. Since the block of the nested loop is composed of a FOR LOOP and a WHILE LOOP hence the time complexity of the nested loop block is order of $O(n^2)$. The average time complexity of a Quicksort algorithm is $O(n\log n)$ [22,23]. So, the complexity of the proposed Tube-tap algorithm is $O(n^2) + O(n\log n) + O(n\log n) \equiv O(n^2)$. □

5.1.5. Schedulability

Theorem 5.2. *Proposed Tube-tap algorithm ensures the execution of all jobs before the corresponding deadlines iff all jobs are schedulable. Mathematically,* $\sum_{j=1}^{m}\sum_{i=1}^{n}\frac{l_{ij}}{r_j} \le T_j,\ \forall j$ *iff* $\frac{L_j}{\sum_{i=1}^{n} r_i} \le T_j,\ \forall j$.

This claim is derived from analyzing the algorithm's logic and pseudo-code. The validity of this claim will be verified using an extensive simulation in Section 6.

6. Simulation result

The following section contains the MATLAB simulation results. In this section, an extensive simulation is carried out to compare the performance of the proposed Tube-tap algorithm and the other four existing algorithms called List-scheduling (LS), Longest Processing Time (LPT), Shortest Processing Time (SPT), and Earliest Due Date (EDD). MATLAB is used to conduct the simulation. The simulation is done in six parts. The first four parts deal with hard-deadlines and the latter two parts deal with soft-deadlines. The corresponding simulation parameters and corresponding results are discussed as follows.

6.1. Part-1

In the first simulation we consider 5 jobs and all the jobs are considered with hard deadlines. Each job has a different job length. The profit gained from processing a packet of a job is different from other jobs. The packet sizes of the jobs may differ too. These jobs are required to be scheduled on 4 machines with different processing speeds and processing costs. The parameter values for simulation-1 are listed in Table 1.

Fig. 4 represents the elapsed time to execute the assigned jobs on different machines. The red line represents deadlines of the corresponding jobs as mentioned in Table 1. The figure shows that the List algorithm fails to meet the deadline of job-3 and 5 and LPT algorithm fails to meet the deadlines of job 1, 3, and 5. Hence, list algorithm gains no profit for processing job-3, and 5. Similarly LPT gains no profit for processing job 1, 3, and 5. On top of that a loss is received due to processing costs on the machines. On the other hand, the SPT, EDD, and proposed Tube-tap algorithm meet all the required deadlines and thus no loss is applied for processing any job.

Table 1
Simulation parameters for simulation-1.

Parameter	Value
Number of machines, n	4
Processing rate, r	1000 ∗ 8 ∗ [1 3 2 4] bits/s
Cost of machine, c	[2 3 4 5]/10 units/s
Number of jobs, m	5
Job length, L	[100 200 150 300 100] packets
Packet size, s	[1000 1500 1200 1000 800] bytes
Profit from job, p	[6 7 8 9 8]/10 units/packet
Deadlines, T	[70 80 50 120 60] s

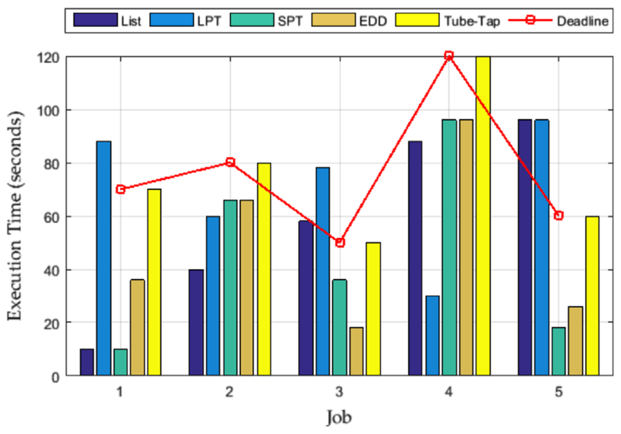

Fig. 4. Simulation result compares the net profits of applying different algorithms for the parameter set-1 as given in Table 1.

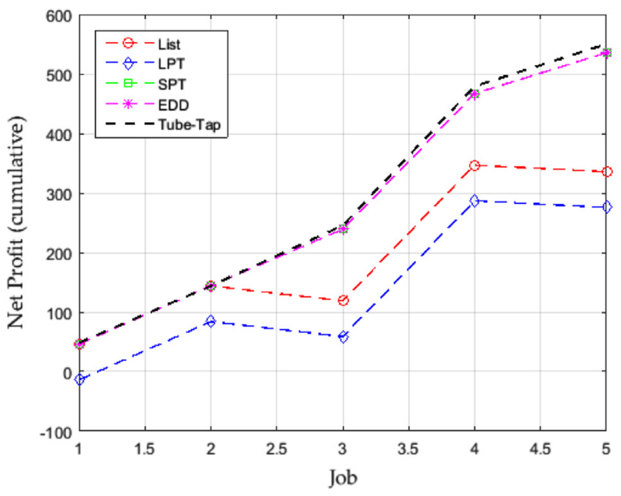

Fig. 5. The simulation results compare the net profits of applying different algorithms for the parameter set-1 as given in Table 1.

Fig. 5 represents the comparison of net profits earned by applying different algorithms for the simulation parameters described in Table 1. The red dotted line represents the List algorithm and it shows a decreasing trend in between job-2, and 3 and job-4, and 5. This decrement in profits occurs due to the failure of meeting the corresponding job deadlines. The LPT is represented by the blue dotted line and it results in a negative net profit (i.e., loss) after processing the first job. It happens because of LPT fails to meet the deadline of job-1 as found in Fig. 4 and thus it gains no profit for processing that job but it bears a cost for processing the job on the machines. On the other hand, EDD, SPT, and the proposed Tube-tap show continuous increasing trends in terms of net-profit. According to Fig. 5 the proposed Tube-tap algorithm earns the maximum net profit as indicated by the dotted black line in the figure.

Therefore, analyzing the simulation results of the first simulation we conclude that the proposed algorithm outperforms other existing algorithm in terms of profit while fulfilling all the delay constraints.

6.2. Part-2

Table 2 shows the simulation parameters for the second simulation. Unlike simulation-1 all the job lengths for simulation-2 are kept equal to 2000 packets, the packet sizes of all jobs are also kept the same (1000 bytes). The rest of the parameter values are kept equal to the simulation-1 parameter values.

Table 2
Simulation parameters for simulation-2.

Parameter	Value
Job length, *L*	[200 200 200 200 200] packets
Packet size, *s*	[1000 1000 1000 1000 1000] bytes
Profit from job, *p*	[6 7 8 9 8]/10 units/packet
Deadlines, *T*	[40 50 60 70 30] s

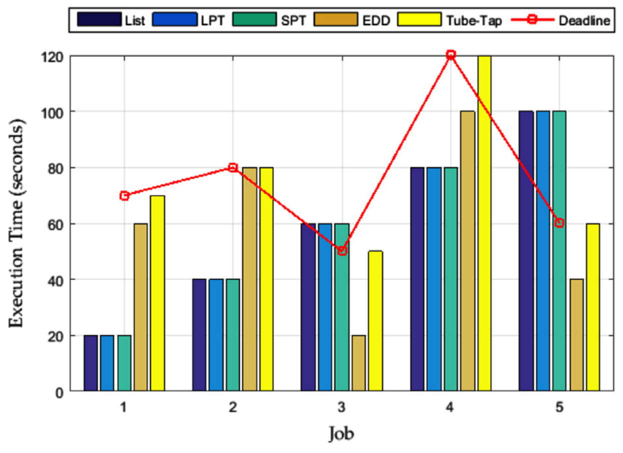

Fig. 6. The simulation results compare the elapsed time of jobs for different algorithms for the parameter set-2 as given in Table 2.

Fig. 6 compares the elapsed time to accomplish the job scheduled by different algorithms. Like simulation-1, List and LPT algorithms fail to meet the deadlines for job-3 and 5. Like these two algorithms SPT also fails to meet the deadlines for job-3 and 5. Also like simulation-1 the EDD and Tube-tap algorithms meet all the required deadlines.

Fig. 7 shows how the net-profit after accomplishing job-3 and 5 decreases for List, LPT, and SPT algorithms. Since these three algorithms missed the deadlines for job-3 and 5 as is shown in Fig. 6. From Table 2 we find that all jobs have the same length and from Fig. 6 we find that List, LPT, and SPT provide the same execution times for all five jobs. Hence, from Fig. 7 we see that all the of these three algorithms show the same net-profit trend. The

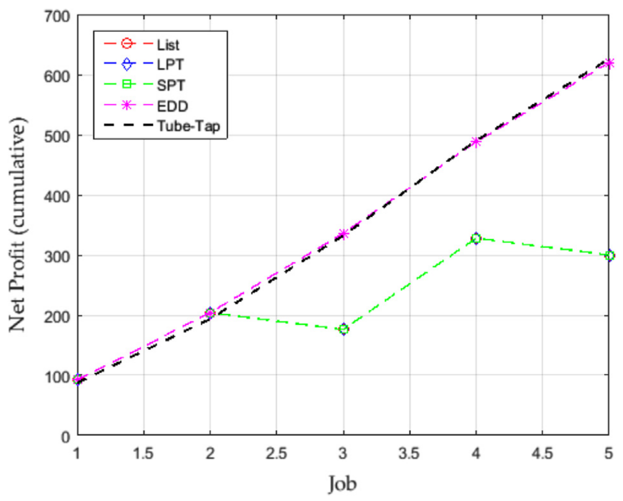

Fig. 7. The simulation results compare the net profit for different algorithms for the parameter set-2 as given in Table 2.

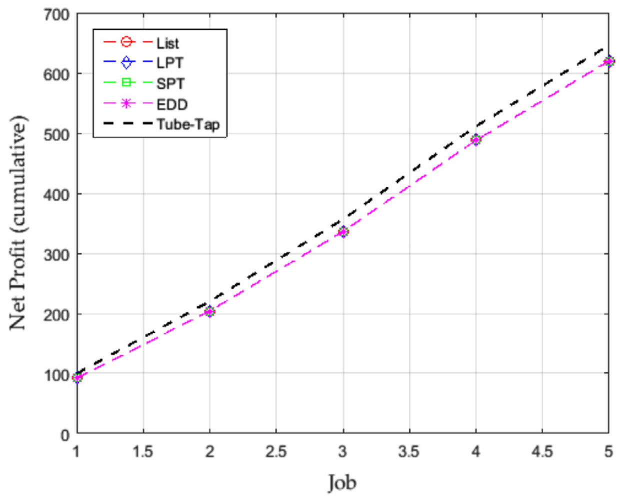

Fig. 8. The simulation results compare the net profit of jobs for different algorithms for the parameter set-3.

figure displays that the other two algorithms (i.e., EDD and Tube-tap) show greater net-profit. Since these two algorithms did not miss any deadlines, the net-profit curves show a continuous increasing trend throughout the process. Among all of the five algorithms the proposed Tube-tap algorithm shows the best net-profit.

6.3. Part-3

In the case of part-3 of the simulation, all the simulation parameter values are kept the same as simulation-2 except for the deadlines. In this simulation, it is assumed that all the five jobs have the same deadlines of 150 s.

Fig. 9 represents the elapsed time to execute the assigned jobs on different machines. This figure shows that all the algorithms meet the required deadlines (150 s). All the algorithms provide the makespan less than the deadline (150 s) except the Tube-tap algorithm. It is noteworthy that in the jargon of JSP the makespan is defined as the total processing time for executing all jobs. As we mentioned before, traditional JSP algorithms focus on minimizing the makespan and computational complexity rather than focusing on maximizing net-profit. Recall that the objective of the problem scenario of this paper is to maximize net-profit. Hence the proposed Tube-tap algorithm tries to extend the makespan focusing on processing all the jobs on the least costly machine as long as it is capable of meeting the deadlines. Moreover, from Fig. 9 all the

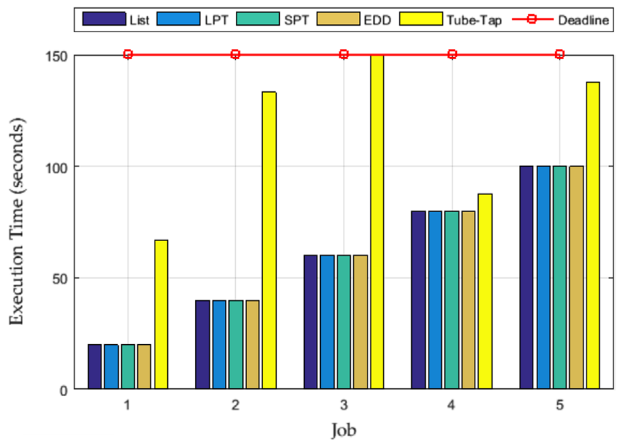

Fig. 9. The simulation results compare the elapsed time for different algorithms for the parameter set-3.

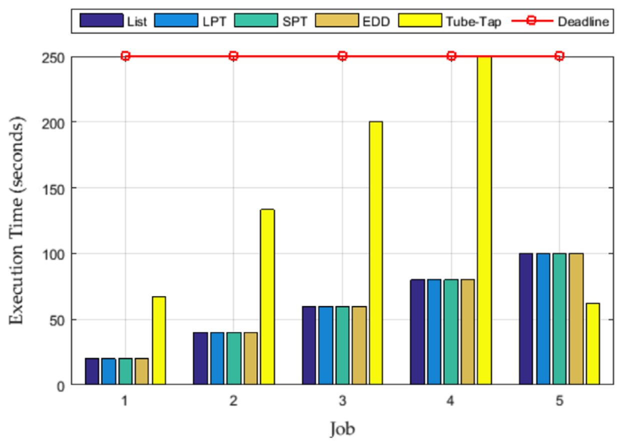

Fig. 10. The simulation results compare the elapsed time for the different algorithms for the parameter set-4.

algorithms show the same execution time for all five jobs except the Tube-tap algorithm. For this reason all the algorithms except Tube-tap show the same trend of net-profit as is shown in Fig. 8. The figure makes it clear that the proposed Tube-tap algorithm provides a far better net-profit than that of the existing algorithms.

Therefore, analyzing the simulation results of the third simulation, we conclude that the proposed algorithm still maintains best performance in terms of profit and delay constraint.

6.4. Part-4

The simulation parameters for the fourth simulation are almost the same as simulation-3 except that the deadline is extended to 250 s instead of 150 s.

The makespan of the Tube-tap algorithm is also increased to 250 s as shown in Fig. 10 and the other four algorithms show the same makespan like in the previous simulation that is equal to 100 s. Fig. 11 shows that the net-profit of the Tube-tap algorithm has also increased as the deadline is extended. It happened because Tube-tap prolongs the makespan as the deadline is extended and thus it allocates a larger portion of the jobs on the least costly machines.

From the analysis of the above-mentioned four simulations we conclude that the proposed Tube-tap algorithm is capable of meeting all the hard-deadlines if the given jobs are schedulable. Moreover, the proposed Tube-tap algorithm allocates the jobs on

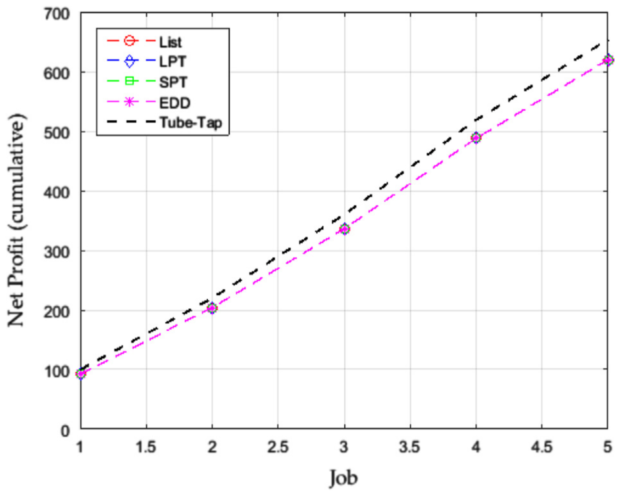

Fig. 11. The simulation results compare the net profit of the jobs for different algorithms for the parameter set-4.

Table 3
The simulation parameters for simulation-5.

Soft-deadlines, T	[70 80 50 120 60] s
α	[0.5 0.5 0.5 0.5 0.5]
β	[0.025 0.025 0.025 0.025 0.025]

the least costly machines by reducing slack time and extending the makespan. The slack time is defined as the amount of time that a task is delayed without delaying another task or impacting the deadline. This property of the proposed Tube-tap algorithm allows it to maximize the net-profit as well as meet the deadlines.

6.5. *Part-5*

This part of the simulation deals with the jobs with soft-deadlines. The simulation parameters are kept the same as simulation-1 except the deadlines are soft instead of hard. The penalty scaling factor α is *considered* 0.5 and the penalty exponent β is considered 0.025 for all five jobs as mentioned in Table 3. Practically, not necessarily all the jobs need to have the same scaling factor and exponent.

Fig. 12 shows that the execution times for the jobs with soft-deadlines using List, LPT, SPT, EDD, and Tube-tap are the same as their execution time for the hard-deadlines in simulation-1 as presented in Fig. 4. Analyzing Fig. 13 we see that the SPT, EDD, and Tube-tap do not exceed any deadlines hence the trends of the net-profit of these three algorithms are the same as simulation-1 as presented in Fig. 5. In the case of the List and LPT algorithms the net-profit curves do not show decreasing trends as in Fig. 5 though these two algorithms miss the deadlines of job-3 and 5. This is because exceeding the soft-deadlines does not necessarily result in zero profits but it applies some penalties. However, if the delay exceeds the delay for too long a time, the net profits can be zero or negative.

6.6. *Part-6*

In simulation-6 the much tighter soft-deadlines are considered to be given as follows T=[20 30 120 150 50] s. All the 5 algorithms fail to meet the deadline of job-2 as shown in Fig. 14. Yet the proposed Tube-tap algorithm misses the deadline in the smallest margin compared with the other algorithms. Moreover, the List algorithm misses the job-5 deadline and LPT misses both the job-1 and 5 deadlines. LPT misses the 1st deadline by a large margin and that results in a larger penalty and thus the net-profit after accomplishing 1st job is less than zero as shown in Fig. 15.

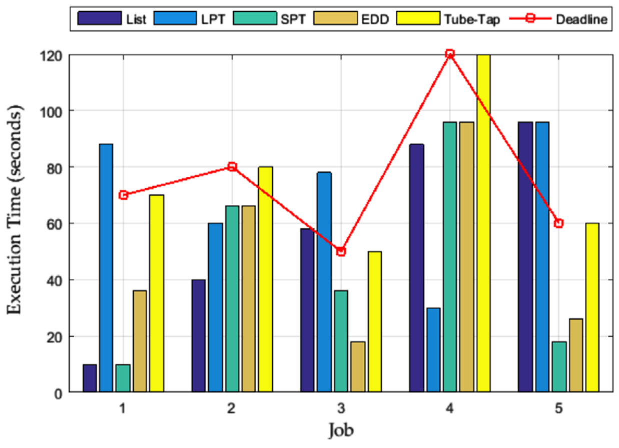

Fig. 12. The simulation results compare the net profits for different algorithms for the parameter set-5.

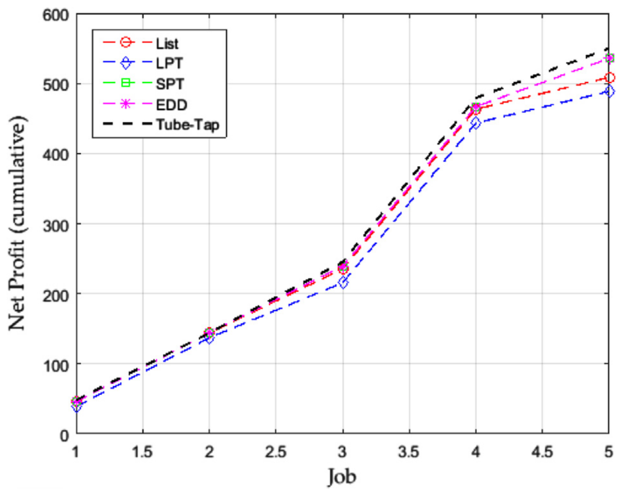

Fig. 13. The simulation results compare the net-profit of jobs for different algorithms for the parameter set-6.

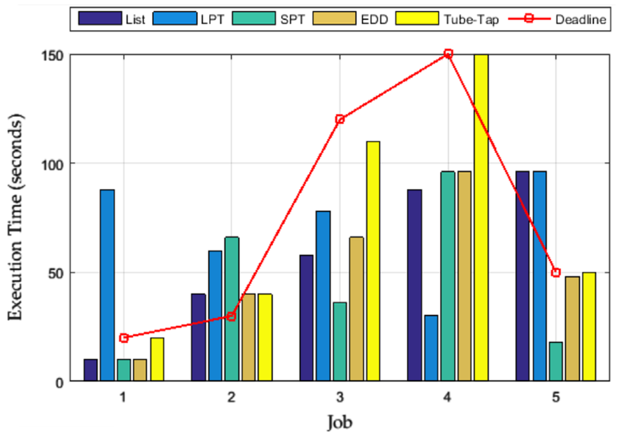

Fig. 14. The simulation results compare the elapsed time of jobs for different algorithms for parameter set-6.

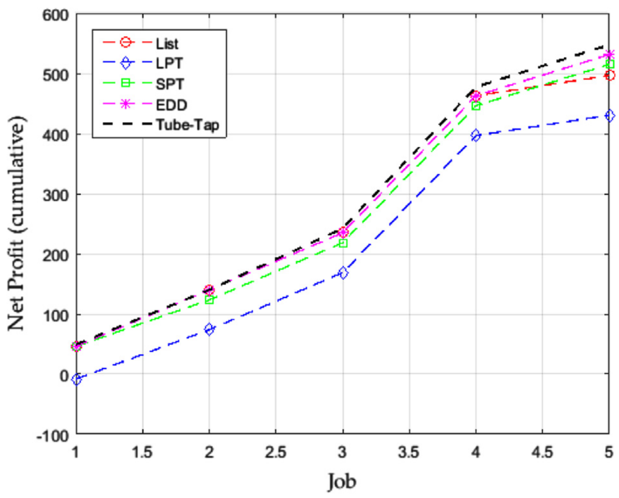

Fig. 15. The simulation results compare the net profit for different algorithms for parameter set-6.

Since the LPT and List algorithms fail to meet the deadline of job-5 hence the profit margin between job-4 and 5 is less for these two algorithms. Among all the 5 algorithms proposed Tube-tap provides the best results in terms of net-profit and execution time.

Summarizing all five simulation results mentioned above, we conclude that the proposed Tube-tap algorithm outperforms other existing solutions by preserving all the deadlines and maximizing net profit. The net profit keeps increasing as the value of the deadline increases.

7. Conclusion

This project work considers a scenario of cloud computing job-shop scheduling where multiple jobs are assigned to a server that possesses multiple processors (i.e., machines). It is considered that each job has a deadline to be met, each job may have a different job length, and the profit of processing a packet of a job can differ from other jobs. It is also considered that each machine may have different processing rates and processing costs. It is also assumed that the deadlines are either hard or soft. A penalty is applied if a job fails to meet deadline. The problem has been formulated as a mixed integer non-linear programming problem. This paper proposes a realistic solution to solve the formulated problem called the Tube-tap algorithm which offers less computational complexity. Extensive simulations are carried out to compare the performance of the proposed algorithm with existing solutions. The simulation results show that the proposed algorithm outperforms the existing solutions in terms of maximizing net profit and preserving deadlines.

Acknowledgment

This research work would not be easy to accomplish without the kind co-operation of Anusha Krishna Murthy, Sneha kolapalli and A.M. Esfar-E-Alam Turzo.

References

[1] P. Larranaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, vol. 2, Springer Science & Business Media, 2002.
[2] U.K. Chakraborty, Computational Intelligence in Flow Shop and Job Shop Scheduling, Studies in Computational Intelligence, vol. 230, Springer-Verlag, Berlin, Heidelberg, 2009.
[3] S.A. Brandt, S. Banachowski, C. Lin, T. Bisson, Dynamic integrated scheduling of hard real-time, soft real-time, and non-real-time processes, in: 24th IEEE Real-Time Systems Symposium, 2003. RTSS 2003, IEEE, Cancun, Mexico, 2003, pp. 396–407.
[4] J.A. Stankovic, K. Ramamritham, What is predictability for real-time systems? Real-Time Syst. 2 (4) (1990) 247–254.
[5] L.A. Wolsey, Mixed integer programming, in: Wiley Encyclopedia of Computer Science and Engineering, 2008.
[6] M.R. Bussieck, A. Pruessner, Mixed-integer nonlinear programming, SIAG/OPT Newsl.: Views News 14 (1) (2003) 19–22.
[7] J. Schutten, List scheduling revisited, Oper. Res. Lett. 18 (4) (1996) 167–170.
[8] L. Lu, L. Zhang, J. Yuan, The unbounded parallel batch machine scheduling with release dates and rejection to minimize makespan, Theor. Comput. Sci. 396 (1) (2008) 283–289.
[9] A.H. Kashan, B. Karimi, M. Jenabi, A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes, Comput. Oper. Res. 35 (4) (2008) 1084–1098.
[10] E. Kondili, C. Pantelides, R. Sargent, A general algorithm for short-term scheduling of batch operations I. MILP formulation, Comput. Chem. Eng. 17 (2) (1993) 211–227.
[11] A.P. Vepsalainen, T.E. Morton, Priority rules for job shops with weighted tardiness costs, Manag. Sci. 33 (8) (1987) 1035–1047.
[12] H.M. Goldberg, Analysis of the earliest due date scheduling rule in queueing systems, Math. Oper. Res. 2 (2) (1977) 145–154.
[13] T. Cheng, M. Gupta, Survey of scheduling research involving due date determination decisions, Eur. J. Oper. Res. 38 (2) (1989) 156–166.
[14] K.R. Baker, J.J. Kanet, Job shop scheduling with modified due dates, J. Oper. Manag. 4 (1) (1983) 11–22.
[15] J.A. Stankovic, M. Spuri, K. Ramamritham, G.C. Buttazzo, Deadline Scheduling for Real-Time Systems: EDF and Related Algorithms, vol. 460, Springer Science & Business Media, 2012.
[16] J.C. Ho, Y.-L. Chang, Minimizing the number of tardy jobs for m parallel machines, Eur. J. Oper. Res. 84 (2) (1995) 343–355.
[17] J.Y. Leung, Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, 2004.
[18] L. Engebretsen, Simplified tight analysis of Johnson's algorithm, Inf. Process. Lett. 92 (4) (2004) 207–210.
[19] D. Dutta, R. Joshi, A genetic: algorithm approach to cost-based multi-qos job scheduling in cloud computing environment, in: Proceedings of the International Conference & Workshop on Emerging Trends in Technology, ACM, New York, NY, USA, 2011, pp. 422–427.
[20] J.F. Goncalves, J.J. de Magalhães Mendes, M.G. Resende, A hybrid genetic algorithm for the job shop scheduling problem, Eur. J. Oper. Res. 167 (1) (2005) 77–95.
[21] B.B. Brandenburg, J.H. Anderson, Integrating hard/soft real-time tasks and best-effort jobs on multiprocessors, in: 19th Euromicro Conference on Real-Time Systems, 2007. ECRTS'07, IEEE, Pisa, Italy, 2007, pp. 61–70.
[22] J. Nievergelt, K.H. Hinrichs, Algorithms & Data Structures, vdf Hochschulverlag an der ETH, 1999.
[23] N. Wirth, Algorithms & Data Structures, Prentice-Hall, 1986.