EasyChair Preprint

№ 14910

New Competitive Analysis Results of Online List Scheduling Algorithm

Rakesh Mohanty, Debasis Dwibedy and Shreeya Swagatika Sahoo

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

September 16, 2024

New Competitive Analysis Results of Online List Scheduling Algorithm[⋆]

Rakesh Mohanty[1], Debasis Dwibedy[1], and Shreyaa Swagatika Sahoo[1]

Veer Surendra Sai University of Technology, Burla, Odisha 768018, India
{rakesh.iitmphd, debasis.dwibedy, shreeya.swagatika}@gmail.com

Abstract. The online algorithm has been an emerging area of interest for researchers in various domains of Computing. The online m-machine list scheduling problem introduced by Graham has gained theoretical as well as practical significance in the development of competitive analysis as a performance measure for online algorithms. In this paper, we study and explore the performance of Graham's online *list scheduling algorithm(LSA)* for independent jobs. In the literature, algorithm *LSA* has been shown $(2 - \frac{1}{m})$-competitive, where m is the number of machines. We present two new upper bound results on competitive analysis of *LSA*. We obtain upper bounds on the competitive ratio of $2 - \frac{2}{m}$ and $2 - \frac{m^2-m+1}{m^2}$ respectively for practically significant two special classes of input job sequences. Our analytical results can motivate the practitioners to design improved competitive online algorithms for the m-machine list scheduling problem by characterizing the real-life input sequences.

Keywords: Competitive Analysis · Identical Machines · Non-preemptive · Makespan · Online Scheduling.

1 Introduction

1.1 Online Algorithm

An *online algorithm* receives and processes inputs one by one in order [1, 2]. Each input is processed immediately upon its availability with no knowledge of the successive inputs. Since the algorithm has no prior idea about the entire sequence of inputs, it is constrained to make irrevocable decisions on the fly. Here, a sequence of outputs is produced by considering the past outputs and the current input. Suppose, we have a sequence of inputs $I = \langle i_1, i_2, \ldots, i_n \rangle$ of finite size n. The inputs are available to the online algorithm one at a time so that at any given time t, an input instance i_t is processed with no clue on the future inputs $i_{t'}$, where $t' > t$.
Interactive computing is indispensable in various domains such as computer science, networks, transport, medicine, agriculture, production, and industrial management [2]. Online algorithms are very much in use as a standard algorithmic framework in interactive computing. The requests arrive one by one to the

⋆ Supported by Veer Surendra Sai University of Technology

interactive system, and each request demands an immediate response. Here, the system runs an online algorithm that reacts to the current request according to the desired objectives and with no clue on the entire request sequence. Therefore, the design and analysis of online algorithms have gained a significant research interest in practice.

1.2 Competitive Analysis

Competitive analysis [4] provides a theoretical framework to measure the performance of an online algorithm. Here, the performance of an online algorithm is compared with its corresponding optimum offline algorithm, which knows all information about the inputs a priori and processes them efficiently by incurring the smallest cost. Let us consider $ALG(I)$ be the cost incurred by an online algorithm ALG for any input sequence I and $OPT(I)$ be the optimum cost obtained by the optimum offline algorithm OPT for I. We now define ALG to be k-competitive for a smallest $k \geq 1$, if $ALG(I) \leq k \cdot OPT(I)$ for all input sequences I. Here, k is referred to as the competitive ratio. For a cost minimization problem, it is always desirable to obtain a competitive ratio that is closer to 1.

1.3 Online List Scheduling

Online List scheduling(LS) [1] has been a well-studied problem in theoretical computer science. Here we are given a finite number of jobs in a list and m-machines($m \geq 2$). The output is the generation of a schedule that represents the assignments of all jobs over m machines, where the completion time of the job schedule, i.e., *makespan* is the output parameter. The objective is to attain a minimum makespan subject to some non-trivial constraints. The constraints are-input jobs are given one by one. Each received one must be scheduled irrevocably as soon as it arrives with no information about the future jobs. The assumptions are that jobs are non-preemptive and independent.

1.4 Practical and Research Motivation

Online list scheduling finds applications in areas such as multiprocessor scheduling in the interactive time-shared operating systems [2], routing of data packets on different links with balancing loads of each link in the computer networks [6], data and information processing in the distributed computing systems [7], robot navigation and exploration [8].
Online m-machine list scheduling for $m \geq 2$ has been proved to be NP-Complete by a polynomial-time reduction from the classical Partition problem [9]. The real challenge for designing of near-optimal online scheduling algorithm arises due to the unavailability of the required information on the entire job sequence before their processing. An online list scheduling algorithm is influenced by the sequence of arrival of the input jobs and their processing times. According to our knowledge, there is no attempt in the literature to classify and characterize the

input job sequences for online list scheduling based on real-world inputs. This motivates us to study and analyze the widely accepted and practically implemented online list scheduling algorithm LSA by exploring and characterizing special classes of inputs.

1.5 Our Contribution

We characterize the performance of algorithm LSA for online scheduling of independent jobs on m identical parallel machines and present a simple proof for $2 - \frac{1}{m}$ competitiveness. We analyze algorithm LSA on special classes of job sequences and obtain two new upper bounds on the competitive ratio as $2 - \frac{2}{m}$ and $2 - \frac{m^2-m+1}{m^2}$ respectively.

2 Preliminaries and Related Work

Here, we present some basic terminologies and notations, which we will use throughout the paper. We then highlight scholarly contributions related to the online list scheduling setting.

2.1 Basic Teminologies and Notations

- We specify each independent job and identical machine as J_i and M_j respectively, where m machines are represented as $M_j(j = 1, \dots, m)$ and n jobs are represented as $J_i(i = 1, \dots, n)$.
- Jobs are *independent* in the sense that jobs can execute in overlapping time slots on different machines.
- Machines are *identical* in the sense that the processing time (p_i) of any J_i is equal for all machines.
- Sometime we refer processing time (p_i) of J_i as *size* of J_i.
- We represent c_i as the completion time of any job J_i.
- We denote *makespan* obtain by any online algorithm A for input sequence I as $C^*_A(I)$. We have $C^*_A(I) = max\{c_i | 1 \leq i \leq n\}$.
- A machine is in idle state when it is not executing any job and we represent idle time of the machine as φ in the timing diagram.
- *Load*(l_j) of any machine M_j is the sum of processing time of the jobs scheduled on M_j. Suppose, n jobs are assigned to M_j, then $l_j = \sum_{i=1}^{n} p_i$. We may further define makespan as $C^*_A(I) = max\{l_i | 1 \leq j \leq m\}$.
- *Non-preemptive* scheduling of the jobs means once a job J_i with p_i starts its processing on any M_j at time t then it continues with no interruption by taking all together $t + p_i$ time prior to its completion.

2.2 Related Work

The m-machine LS problem has been studied for various setups over the years, see surveys [10-13]. According to our knowledge, the first online scheduling algorithm for multiprocessor systems was proposed by Graham in 1966 popularly known as the list scheduling algorithm(LSA) [1]. He considered the non-preemptive scheduling of a list of jobs on identical parallel machines. The goal was to obtain minimum makespan. Algorithm LSA schedules a newly available job to the most lightly loaded machine. The performance of LSA was proved to be at most $2-\frac{1}{m}$ time worse than the optimum makespan for all job sequences. Faigle et. al. [5] analyzed the performance of LSA by considering a list of 3 jobs with sizes $(1, 1, 2)$ respectively and proved that LSA is optimal for $m = 2$. Similarly, for $m = 3$, they considered 7 jobs with sizes $(1, 1, 1, 3, 3, 3, 6)$ respectively to represent the optimum competitiveness of 1.66. They obtained *lower bound(LB)* on the competitive ratio of 1.707 for $m \geq 4$ by considering a list of $2m+1$ jobs, where m jobs are of size 1 unit each, m jobs with size $1+\sqrt{2}$ unit each and a single job is of size $2(1+\sqrt{2})$ unit.
The first improvement over LSA was provided by Galambos and Woeginger [14] and achieved competitiveness of $(2-\frac{1}{m}-\epsilon_m)$, where $\epsilon_m > 0$. Bartal et. al. [15] obtained the upper bound(UB) on the competitive ratio of 1.986 for a general case of m. For $m = 3$, they proved LB of 1.4 by considering 7 jobs each with size $(1, 1, 1, 2, 1, 3, 5)$ unit respectively. Bartal et. al. [17] obtained a better LB of 1.837 for $m \geq 3454$ by examining the job sequence consisting of $4m+1$ jobs, where m jobs each with processing time of $\frac{1}{x+1}$ unit, m jobs with processing time of $\frac{x}{x+1}$ unit each, m jobs are of size x unit each, $\lfloor \frac{m}{2} \rfloor$ jobs are of size y unit each, $\lfloor \frac{m}{3} \rfloor - 2$ jobs each with size z unit, $(m+3-\lfloor \frac{m}{2} \rfloor - \lfloor \frac{m}{3} \rfloor)$ jobs are of size $2y$ unit each, where x, y, z are positive real values. We now present the summary of all important results for deterministic online scheduling algorithms for identical parallel machines in table 1.

Table 1. Summary of Important Results

Year and Author(s)	**Competitive Ratio(s)**
1966, Graham [1]	$2-(\frac{1}{m})$ for all m.
1991, Galambos and Woeginger [14]	$2-(\frac{1}{m}-\epsilon_m)$ for $m \geq 4$.
1992, Bartal et al. [15]	1.986(UB) for all m, 1.4(LB) for $m = 3$.
1996, Karger et al. [16]	1.945, for $m \geq 8$.
1994, Bartal et. al. [17]	1.837(LB) for $m \geq 3454$
1994, Chen et.al. [18]	1.7310(LB) for $m = 4$ and 1.8319(LB) for $m > 4$.
1999, Albers [19]	1.923, for $m \geq 2$.
2000, Fleicher and Wahl [20]	1.9201(UB)
2001, Rudin III [21]	1.88(LB) for all m.
2003, Rudin III and Chnadrasekharan [22]	1.732(LB) for $m = 4$
2008, Englert et.al. [23]	1.4659 for $2 \leq m \leq 30$

2.3 Graham's Online List Scheduling Algorithm

Here, we present the descriptions of algorithm *LSA* [1] for independent jobs and provide proof sketch to show its competitiveness results as follows.

Algorithm 1 LSA

```
Initially, i=1, $l_1 = l_2 = ............l_m = 0$
WHILE a new job $J_i$ arrives DO
  BEGIN
    Calculate current load for each machine $M_j$.
    Number the machines in non-decreasing order of their loads Such that $l_1 \leq l_2 \leq ...... \leq l_m$.
    Assign $J_i$ to $M_1$.
    $l_1 = l_1 + p_i$
    $i = i + 1$.
  END
Return $l_j = max\{l_j | j = 1, 2, ......m\}$
```

Theorem 1. ***Algorithm LSA is** $(2 - \frac{1}{m})$**-competitive for** $m \geq 2$.*

Proof Let us consider a list of n jobs($J_1........J_n$). Each job is available to *LSA* one by one. The processing time $p_i > 0$ for $1 \leq i \leq n$. Initially, m machines($M_1, \ldots, M_m$) are available with loads $l_1 = l_2 = \ldots = l_m = 0$. Let the size of the largest job J_k is p_k, where p_k =max$\{p_i | 1 \leq i \leq n\}$. We denote the optimal makespan as $C^*_{OPT}(I)$ and makespan obtained by algorithm *LSA* as $C^*_{LSA}(I)$ for all input sequences *I*. As per the description of *LSA*, the scheduling decision time(T) is constant for each input. Therefore, each time we ignore T, while calculating makespan.

Computation of OPT: Optimum offline strategy equally distributes the total load among all m-machines. So, the completion time of the job schedule is at least the average of total load incurred on m-machines. Therefore, we have

$$C^*_{OPT}(I) \geq \frac{1}{m}(\textstyle\sum_{i=1}^{n} p_i). \tag{1}$$

Suppose *OPT* schedules only J_k on M_1 and assigns rest $n-1$ jobs on $m-1$ machines with equal load sharing among $m-1$ machines and $\frac{1}{m-1}(\sum_{i=2}^{m} l_i) \leq l_1$, then we have

$$C^*_{OPT}(I) \geq p_k. \tag{2}$$

We now provide the computation for algorithm LSA : Algorithm *LSA* assigns a new job to the machine with least load to keep a balance in the load incurred on each machine. The worst scenario appears in this case when J_k arrives as the n^{th} job and prior to that the total load incurred by $(n-1)$ jobs are equally shared among m-machines. So, we have $l_1 \leq \frac{1}{m}(\sum_{i=1}^{n-1} p_i)$, this compels *LSA* to schedule the n^{th} job on M_1 i.e the least loaded machine. Therefore, we have

$$C^*_{LSA}(I) \leq \frac{1}{m}(\textstyle\sum_{i=1}^{n-1} p_i) + p_k, \quad \text{implies,}$$

$m.C^*_{LSA}(I) \leq \sum_{i=1}^{n-1} p_i + m.(p_k) \leq \sum_{i=1}^{n-1} p_i + p_k + (m-1).p_k \leq \sum_{i=1}^{n} p_i + (m-1).C^*_{OPT}(I)$

$C^*_{LSA}(I) \leq \frac{1}{m}(\sum_{i=1}^{n} p_i) + (\frac{m-1}{m}).C^*_{OPT}(I) \leq C^*_{OPT}(I) + (\frac{m-1}{m}).C^*_{OPT}(I) \leq C^*_{OPT}(I)(1+\frac{m-1}{m})$

$\frac{C^*_{LSA}(I)}{C^*_{OPT}(I)} \leq \frac{m+m-1}{m} \leq \frac{2m-1}{m}$

$C^*_{LSA}(I) \leq (2-\frac{1}{m})C^*_{OPT}(I).$ □

3 New Upper Bound Results on Competitiveness of Algorithm LSA

We obtain improved competitive ratios of the online deterministic *LSA* by considering two special classes of inputs. In this setting, the performance of *LSA* is evaluated through the ratio between the makespan obtained by *LSA* for the worst sequence of input jobs arrival to the makespan obtained by *OPT*. The special classes of input sequences are described as follows.

3.1 Special Classes of Input Job Sequences

Class-1(S_1): Here, we consider a list of $(m-1)^2+1$ jobs, where $(m-1)^2$ number of jobs are of size 1 unit each and a single job is of size m unit.

Class-2(S_2): Here, we consider a list of $m(m-1)+1$ jobs, where $m(m-1)$ number of jobs are of size 1 unit each and a single job is of size m^2 unit.

Theorem 2. ***LSA is ($2-\frac{2}{m}$)-competitive for S_1, where $m \geq 3$.***

Proof Let, $C^*_{OPT}(S_1)$ and $C^*_{LSA}(S_1)$ be the makespan obtained by *OPT* and *LSA* respectively for S_1. We ignore T, while scheduling each incoming job.

Computation of LSA: The worst sequence for S_1 appears when the input jobs arrive in the non-decreasing order of their processing time. So, in the worst case, jobs arrive one by one starting at time $t = 0$ in the following order $\sigma_1 = \langle J_1, J_2, \ldots, J_{(m-1)^2}, J_{(m-1)^2+1} \rangle$, where the jobs from J_1 to $J_{(m-1)^2}$ are of size 1 unit each and the $(J_{(m-1)^2+1})^{th}$ job is of size m unit. *LSA* schedules each job upon its availability and before the arrival of the next job. As we are ignoring T, so at time $t = 0$, m jobs are scheduled on m machines in one slot to complete their processing at $t = 1$. Therefore, the first m^2-2m jobs finish at $t = m-2$. Now, at $t = (m-2)$, we are left with final two jobs of sizes 1 and m respectively and are allocated to machines M_1 and M_2. So, the last job finishes at $t = 2m-2$. Therefore, we have

$$C^*_{LSA}(S_1) \leq 2m-2 \quad (3)$$

We present the Gantt chart of $C^*_{LSA}(S_1)$ in Figure 1.

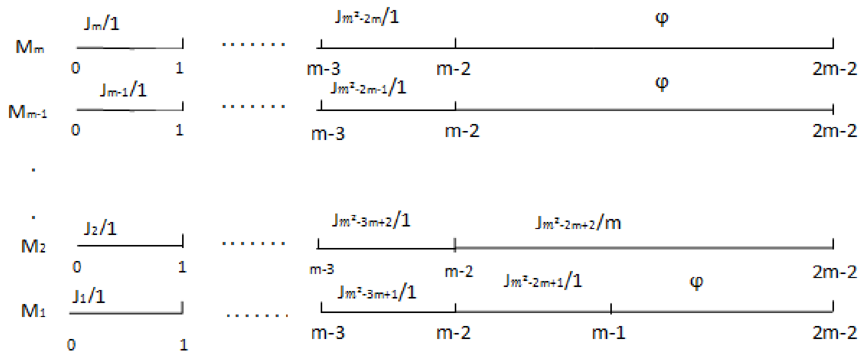

Fig. 1. Gantt chart of LSA for class 1

Computation of OPT: Here, the optimum strategy schedules the jobs according to the non-increasing order of job's size. So, at time $t = 0$, OPT assigns the largest job with size m unit to a machine along with $m - 1$ jobs of size 1 unit each to rest $m - 1$ machines. Subsequently, $(m - 1)^2$ jobs are assigned and completed at $t = m-1$ and the last job finishes at $t = m$. Therefore, we have

$$C^*_{OPT}(S_1) \geq m \tag{4}$$

The Gantt chart of $C^*_{OPT}(S_1)$ is presented in Figure 2.

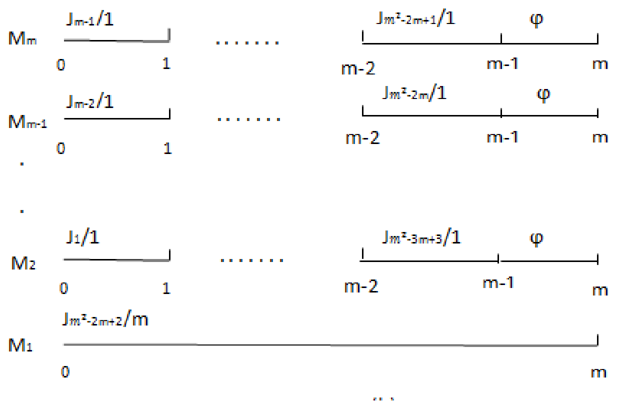

Fig. 2. Gantt chart of Optimal Scheduling for class 1

From equations (3) and (4) we have
$\frac{C^*_{LSA}(S_1)}{C^*_{OPT}(S_1)} \leq \frac{2m-2}{m} \leq (2 - \frac{2}{m})$. OPT and LSA perform equivalently for S_1 with $m = 2$ as it is required to schedule only 2 jobs. Therefore, it is proved that LSA is $(2 - \frac{2}{m})$-competitive for S_1, where $m \geq 3$. □

Theorem 3. ***LSA is*** $(2 - \frac{m^2-m+1}{m^2})$***-competitive for*** S_2***, where*** $m \geq 2$***.***

Proof Let, $C^*_{OPT}(S_2)$ and $C^*_{LSA}(S_2)$ denote the makespan of OPT and LSA respectively for S_2. We ignore T, while scheduling each incoming job.

Computation of LSA: The worst input job sequence for S_2 appears when the largest job available at the end of the input job sequence. Therefore, the sequence $\sigma_2 = \langle J_1, J_2, \ldots, J_{m^2-m}, J_{m^2-m+1}\rangle$ holds the worst sequence for S_2 where the jobs from J_1 to J_{m^2-m} are of size 1 unit each and the ${J_{m^2-m+1}}^{th}$ job is the largest job with size m^2 unit. Initially at time $t = 0$, LSA assigns m jobs on m machines in one slot and finish them at $t = 1$. Subsequently, $m(m-1)$ jobs are scheduled in $m-1$ slots and are completed at $t = m-1$. Now, at $t = m-1$, we are left with last two jobs of size 1 unit and m^2 unit respectively and the load of each machine is $m-1$. So, the last job finishes at $t = m-1+m^2$. Therefore, we have

$$C^*_{LSA}(S_2) \leq m-1+m^2 \tag{5}$$

We present the Gantt chart of LSA for S_2 in Figure 3.

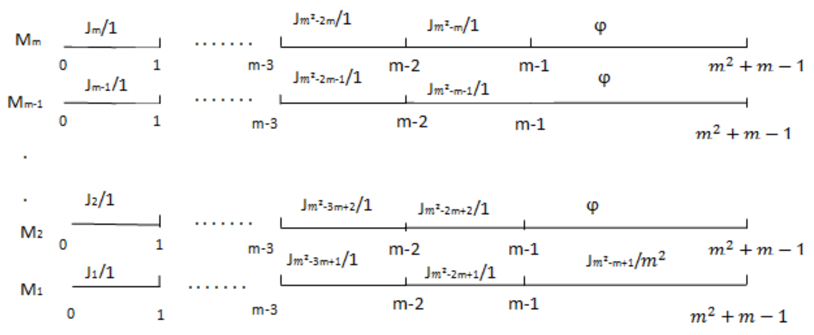

Fig. 3. Gantt chart of LSA for class 2

Computation of OPT: OPT schedules the largest job first. So, at time $t = 0$, the largest job J_{m^2-m+1} is assigned to M_1 along with $m-1$ jobs to remaining $m-1$ machines. In the same fashion, $m(m-1)$ jobs are completed at $t = m$ and the last job finishes at $t = m^2$. Therefore, we have

$$C^*_{OPT}(S_2) \geq m^2 \tag{6}$$

From equations (5) and (6) we have

$$\frac{C^*_{LSA}(S_2)}{C^*_{OPT}(S_2)} \leq 2 - (\frac{m^2-m+1}{m^2}).$$

As we are not considering the single machine case, so we have LSA is $(2 - \frac{m^2-m+1}{m^2})$- competitive for S_2, where $m \geq 2$. □

4 Conclusion and Future Scope

In this paper, we presented an alternate proof for $(2 - \frac{1}{m})$-competitiveness for algorithm LSA for independent jobs. We studied and analyzed the performance of LSA by characterizing the input sequences into two special classes. We showed that LSA is $(2 - \frac{2}{m})$-competitive for special class(S_1) of input sequence, where we considered $(m-1)^2 + 1$ jobs with processing times such as 1 unit and m unit respectively. We also proved that LSA is $(2 - \frac{m^2-m+1}{m^2})$-competitive by considering another class(S_2) of input sequence with $m(m-1)+1$ jobs of with sizes such as 1 unit and m^2 unit respectively. The competitive ratios achieved by LSA for S_1 and S_2 input sequence with different number machines are shown in Table 2. It can be observed from our analytical results that an increase in the number of machines does not help algorithm LSA to minimize the makespan for S_1. However, the performance of LSA can be improved substantially with the increase in the number of machines for S_2.

Future Scope. It can be realized that the order of availability of the jobs has a strong influence on the performance of LSA. However, the characterization of the input sequence with the known total number of jobs and their processing time can help to improve the competitive ratio of LSA. Through input characterization, theoretical input sequences can be mapped to the real-world input sequences. It will be interesting to evaluate the performance of well-known online scheduling algorithms for practical input sequences. The performance of well-known online scheduling algorithms can be improved with better competitive results based on the practical input sequences.

Table 2. Competitive Ratio of LSA for Different Number of Machines

Number of Machines	**CR for Class-1**	**CR for Class-2**
2	1.0000	1.2500
3	1.3333	1.2222
4	1.5000	1.1875
5	1.6000	1.1600
10	1.8000	1.0900
50	1.9600	1.0196
100	1.9800	1.0099

References

1. Graham R.L.(1966) "Bounds for certain multiprocessor anomalies". *Bell System Technical Journal*, **45**:1563-1581.
2. Borodin A., El-Yaniv R.(1998) "Online computation and competitive analysis". *Cambridge University Press*, Cambridge.
3. Pruhs K., Sgall J. and Torng E.(2004) "Online scheduling". *Handbook on scheduling: Algorithms, models and performance analysis, CRC Press.*
4. Robert E. Tarjan and Sleator.(1985) "Amortized computational complexity". *SIAM Journal on Algebric and Discrete Methods*, **6**(2):306-318.
5. Faigle U., Kern W. and Turan G.(1989) "On the performance of online algorithms for partition problems". *Acta Cybernetica*, **9**:107-119.
6. Aspnes J., Azar Y., Fiat A., Plotkin S. and Waarts O.(1993) "Online load balancing with applications to machine scheduling and virtual circuit routing". 25th *ACM STOC*: 623-631.
7. Bartal Y., Fiat A. and Rabani Y.(1992) "Competitive algorithms for distributed data management". *In Proceedings of* 24th *Annual ACM symposium on Theory of Computing*:39-50.
8. Baeza-Yates R.A., Culberson J.C. and Rawlins G.J.E.(1993) "Searching in the plane". *Information and Computation*, **106**: 234-252, 1993.
9. Garey M.R. and Jhonson D.S.(1979) "Computers and Intractability: A Guide to the Theory of NP-Completeness". *Freeman.*
10. Graham R.L., Lawer E.L., Lenstra J.K. and Rinnooy kan A.H.(1979) "Optimization and ap-proximation in deterministic sequencing and scheduling : A Survey". *Annals of Discrete Mathematics, Elsevier*, **5**: 287-326.
11. Chen B., Potts C.N. and Woeginger G.J.(1998) "A review of Machine scheduling: Complexity, Algorithms and Approximability". *Handbook of Combinatorial Optimization, Kluwer Academic Publishers*, **3**:21-169.
12. Sgall J.(1998) "Online scheduling: A survey". *Lecture Notes in Computer science, Springer*, **1442**:196-231.
13. Albers S.(2009) "Online scheduling: Introduction to Scheduling", edited by Y. Robert and F. Vivien. *CRC Press*: 57-84.
14. Galambos G. and Woeginger G.J.(1993) "An online scheduling heuristic with better worst case ratio than Graham's list scheduling". *SIAM Journal of Computing*, **22**(2):349-355.
15. Bartal Y., Fiat A., Karloff H. and Vohra R.(1992) "New algorithms for an ancient scheduling problem". *In Proceedings of the* 24th *ACM Symposium on the Theory of Computing*, Victoria, Canada:51-58.
16. Karger D.R., Phillips S.J. and Torng E.(1996) "A better algorithm for an ancient scheduling problem". *Journal of Algorithms*, **20**(19):400-430.
17. Y. Bartal Y., Karloff H. and Rabani Y.(1994) "A better lower bound for online scheduling". *Information Processing Letters*, **50**:113-116.
18. Chen B., Vliet A.V. and Woeginger G.J. "New lower and upper bound for online scheduling". *Operation Research Letters*, **16**:221-230.
19. Albers S.(1999) "Better bounds for Online scheduling". *SIAM Journal on Computing*, **29**:459-473.
20. Fleischer R. and Wahl M.(2000) "Online scheduling revisited". *Journal of Scheduling*, **3**:343-353.
21. Rudin III J.F.(2001) "Improved bounds for the online scheduling problem", *Ph.D Thesis. The University of Texas at Dellas.*

22. Rudin III J.F. and Chandrasekaran R.(2003) "Improved bounds for the online scheduling problem". *SIAM Journal of Computing*, **32**(3): 717-735.
23. Englert M., Ozmen D. and Westermann M.(2008) "The power of reordering for online minimum makespan scheduling". *In Proceedings* 49th *Annual IEEE Symposium on Foundations of Computer Science.*