Research Article

Quanchang Zheng*, Yueyang Zhao, and Jiahe Wang

Online makespan minimization for MapReduce scheduling on multiple parallel machines

https://doi.org/10.1515/dema-2024-0040
received January 6, 2023; accepted June 7, 2024

Abstract: In this work, we investigate the online MapReduce processing problem on m uniform parallel machines, aiming at minimizing the makespan. Each job consists of two sets of tasks, namely, the map tasks and the reduce tasks. A job's map tasks can be arbitrarily split and processed on different machines simultaneously, while its reduce tasks can only be processed after all its map tasks have been completed. We assume that the reduce tasks are preemptive, but cannot be processed on different machines in parallel. We provide a new lower bound for this problem and present an online algorithm with a competitive ratio of $2 - \frac{1}{m}$ (m is the number of machines) when the speeds of the machines are 1.

Keywords: MapReduce scheduling, online algorithm, competitive ratio, parallel machines, lower bound

MSC 2020: 90B35, 68M20

1 Introduction

MapReduce [1] is a popular model in many big data-processing frameworks such as search indexing, distribution sort, log analysis, etc. In general, MapReduce processing consists of two phases: the map phase and the reduce phase. When a job is submitted, its computation always contains these two phases. In the map phase, there are many map tasks inputting raw data and outputting key-value pairs. These key-value pairs are used as inputs of the reduce phase. In the reduce phase, the machines process the pairs and output the final results.

In this article, we study the online MapReduce scheduling problem where jobs are released by over-list. Each job consists of map tasks and reduce tasks, and the processing times of both tasks become known to the decision-maker once the job is revealed. For each job, we assume: (1) the job's reduce tasks can be processed after finishing its all map tasks; (2) the map task is fractional, i.e., it can be arbitrarily split and processed between the machines simultaneously, while the reduce task is not fractional; (3) we assume preemption on reduce task is allowed, i.e., any reduce task may be interrupted and resumed at a later time during processing. The problem can be formally described as follows. A set of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ arrive one by one and must be allocated into m uniform machines $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$. Let the speed of machine σ_i be s_i. Without loss of generality, we may assume $s_1 \geq s_2 \geq \ldots \geq s_m$. Each job J_j consists of a set of map tasks $M_j = \{m_j^1, m_j^2, \ldots, m_j^{v_j}\}$ and a set of reduce tasks $R_j = \{r_j^1, r_j^2, \ldots, r_j^{t_j}\}$, namely, each job has v_j map tasks and t_j reduce tasks. Our goal is

*** Corresponding author: Quanchang Zheng,** Institute of Operational Research, Qufu Normal University, Rizhao, Shandong, 276800, P. R. China, e-mail: zhengquanchang1@163.com
Yueyang Zhao: School of Management Science, Gachon University, Gyeonggi-do, 826802, South Korea, e-mail: 772511022@qq.com
Jiahe Wang: Institute of Operational Research, Qufu Normal University, Rizhao, Shandong, 276800, P. R. China, e-mail: wangjiahe98@foxmail.com

to minimize the makespan, i.e., the completion time of the last job that finishes. Adopting the classical threefold notion, we denote the aforementioned problem as $Qm|MR(pmtn), \text{online}|C_{\max}$, and $Pm|MR(pmtn), \text{online}|C_{\max}$ when $s_i = 1$, $i = 1, 2, \ldots, m$.

The quality of an online algorithm A is normally measured by its competitive ratio. An algorithm A is called ρ-competitive if, for any instance I, $C^A(I) \le \rho \cdot C^*(I)$ holds, where $C^A(I)$ denotes the objective function value produced by A and $C^*(I)$ denotes the optimal objective value.

In recent years, the scheduling problem has been one of the active research topics in MapReduce because of the need to guarantee the performance of the system like response times, system utilization, etc. A number of empirical works [2–5] have been done to provide some new scheduling strategies or heuristics and give the experimental results for different models. Besides, several theoretical works [6–9] have also emerged. Most of these works focus on the offline scheduling problem, where job arrivals are known beforehand. However, in practice, job scheduling is often decided without all the information in advance. Thus, the theoretical research of online scheduling needs to be solved urgently. To the best of our knowledge, there are few studies that consider the online scheduling problem in the MapReduce system [6,7,9–12].

For the online MapReduce scheduling problem, Moseley et al. [6] model the MapReduce system as the two-stage classical flexible flow shop problem and the objective of the problem is to minimize the total flow-time. Then they present an online $1 + \varepsilon$-speed $O(\frac{1}{\varepsilon^2})$-competitive algorithm for this problem, where $0 < \varepsilon \le 1$. However, there is no guarantee of a competitive ratio without resource augmentation. To solve this problem, Zheng et al. [7] construct a slightly weaker criteria called efficiency ratio because no online algorithm can achieve a constant competitive ratio for nonpreemptive tasks. Then they provide an online algorithm called available shortest remaining processing time (ASRPT) with a very small (less than 2) efficiency ratio and show that it outperforms the state-of-the-art schedulers. Chang et al. [10] focus on minimizing the total completion time, and design an online algorithm that achieved 30% shorter completion time of all jobs compared to the original FIFO-based scheduling via simulation. For minimizing makespan, Jiang et al. [13] consider the problem on two uniform machines, i.e., $Q2|M(frac)R(pmtn)|C_{\max}$. They provide an optimal online algorithm with a competitive ratio of $\frac{\sqrt{s^2+2s+5}+1-s}{2}$, where $s \ge 1$ is the speed ratio of two machines. Under the assumption that a job's reduce tasks are unknown until its map tasks are finished, Luo et al. [9] present online optimal algorithms with the same competitive ratio of $2 - \frac{1}{m}$ for both the preemptive and non-preemptive reduce tasks. When jobs are released over time, Chen et al. [14] present a non-preemptive algorithm MF-LPT with a competitive ratio $2 - \frac{1}{m}$ and an optimal preemptive algorithm for two machines.

Many scholars have conducted in-depth research on the MapReduce task scheduling problem in a heterogeneous environment. Jeyaraj et al. [15] devised two methods, a roulette wheel scheme and constrained 2-dimensional bin packing, for a batch of heterogeneous MapReduce jobs on heterogeneous virtual machine capacities, to improve makespan and resource utilization. Li et al. [16] discussed the problem of scheduling MapReduce tasks to heterogeneous geo-distributed data centers, where tasks have different deadlines. The goal is to minimize the total tardiness. They provided a Task Scheduling on Heterogeneous GeoDistributed Data Centers algorithm (TSGC) and demonstrated through experiments that it is effective for the considered problem. Wang et al. [17] studied the task scheduling problem with throughput as the objective in a heterogeneous environment. They provided a heterogeneous throughput-driven task scheduling algorithm (HTD), which can quickly obtain a reasonable sequence of job execution to ensure that the job set can be completed in the shortest possible time in a heterogeneous environment.

In this article, for problem $Qm|MR(pmtn), online|C_{\max}$, we give a lower bound γ^*, where γ^* is the positive root of the following quadratic equation $\sum_{i=2}^{m} s_i\gamma^2 + (s_1 - \sum_{i=2}^{m} s_i)\gamma - \sum_{i=1}^{m} s_i = 0$. Note that $\gamma^* = \frac{\sqrt{s^2+2s+5}+1-s}{2}$ when $m = 2$, where $s \ge 1$ is the speed ratio of two machines, so it is a general result for the problem in in the study by Jiang et al. [13]. Then we devise an $2 - \frac{1}{m}$-competitive online algorithm for $Pm|MR(pmtn), online|C_{\max}$.

The rest of this article is organized as follows: In Section 2, we derive a lower bound for the problem $Qm|MR(pmtn), online|C_{\max}$. In Section 3, we present an approximate algorithm for the problem $Pm|MR(pmtn), online|C_{\max}$. Finally, we conclude the article in Section 4.

2 Notations and lower bound for $Qm|MR(pmtn), online|C_{max}$

Throughout the rest of this article, the following notations are used.

M_j: the set of the map tasks of the job J_j.

R_j: the set of the reduce tasks of the job J_j.

r_j^i: the reduce task in R_j or the length of this task.

$p(\mathcal{S})$: the total length of the tasks in the set $\mathcal{S}$.

$C^A(\mathcal{J})$: the makespan produced by algorithm A after scheduling all jobs in the sequence of jobs $\mathcal{J}$.

$C^*(\mathcal{J})$: the optimal makespan after scheduling all jobs in the sequence of jobs $\mathcal{J}$.

Throughout the rest of this article, we will not use another notation to denote the task length. For example, the r_j^i can denote a reduce task or the length of this task. We assume that $r_j^1 \geq r_j^2 \geq \ldots \geq r_j^{t_j}$ for every $1 \leq j \leq n$. Let $P_j = \sum_{i=1}^{j}(p(M_i) + p(R_i))$ be the total length of the first j jobs and $S_k = \sum_{i=1}^{k} s_i$.

We now give a lower bound of the preemptive version $Qm|MR(pmtn), online|C_{\max}$ below.

Theorem 2.1. *The competitive ratio of any online algorithm for $Qm|MR(pmtn), online|C_{\max}$ is at least γ^*, where γ^* is the positive root of the following quadratic equation*:

$$\sum_{i=2}^{m} s_i\gamma^2 + \left(s_1 - \sum_{i=2}^{m} s_i\right)\gamma - \sum_{i=1}^{m} s_i = 0. \tag{1}$$

Proof. Suppose that the competitive ratio of the algorithm A is γ. Let $\mathcal{J}_i = \{J_1, J_2, \ldots, J_i\}$, the set of the first i arrived jobs. We first consider a sequence of jobs $\mathcal{J}_{2m-2}$ as follows:

- $M_i = \{1\}$ and $R_i = \varnothing$ which means that M_i has only one task with a size of 1, $1 \leq i \leq m$;
- $M_{m+i} = \varnothing$ and $R_{m+i} = \left\{\frac{m(S_m)^{i-1}s_1}{(S_m - s_1)^i}\right\}$ which means that R_{m+i} has only one task with a size of $\frac{m(S_m)^{i-1}s_1}{(S_m - s_1)^i}$, $1 \leq i \leq m-2$.

Let x_i, $1 \leq i \leq m$, be the last time when at least i machines are always busing in the interval $[0, x_i]$, right after all jobs in $\mathcal{J}_{2m-2}$ are scheduled by algorithm A. Then we have $x_1 \geq x_2 \geq \ldots \geq x_m$ as shown in Figure 1 and clearly

$$\sum_{i=1}^{m} s_i x_i = \sum_{i=1}^{2m-2}(p(M_i) + p(R_i)) = \frac{m(S_m)^{m-1}}{(S_m - s_1)^{m-1}}. \tag{2}$$

We claim that for any $1 \leq i \leq m-1$,

$$C^A(\mathcal{J}_{2m-1-i}) \geq x_i. \tag{3}$$

Since the algorithm A is an online algorithm, the schedule for $\mathcal{J}_{2m-1-i}$ is obtained from the schedule for $\mathcal{J}_{2m-2}$ by removing the last $i-1$ jobs. At the time x_i, there are at least i jobs running according to the definition

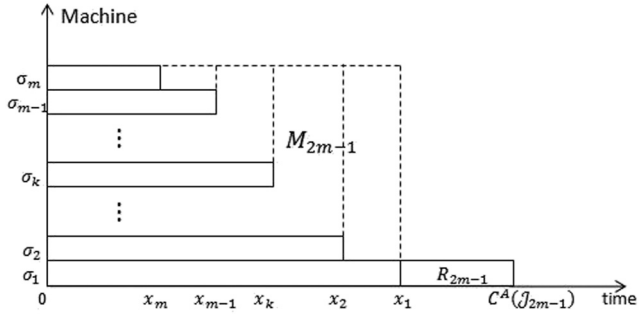

Figure 1: Definition of x_i and the schedule of J_{2m-1}.

of x_i, even after removing $i-1$ jobs from the schedule at least one job running at time x_i remains, and thus, $C^A(\mathcal{J}_{2m-1-i}) \geq x_i$.

Noting that the competitive ratio of the algorithm A is γ, we must have $C^A(\mathcal{J}_{2m-1-i}) \leq \gamma C^*(\mathcal{J}_{2m-1-i})$ for any $1 \leq i \leq m-1$. By computing the job sequence $\mathcal{J}_{2m-1-i}$, we can obtain that $C^*(\mathcal{J}_{2m-1-i}) = \frac{m(S_m)^{m-i-2}}{(S_m-s_1)^{m-i-1}}$. By combining it with (3), we obtain that for any $1 \leq i \leq m-1$,

$$x_i \leq \gamma \frac{m(S_m)^{m-i-2}}{(S_m-s_1)^{m-i-1}}. \tag{4}$$

After scheduling all the jobs in $\mathcal{J}_{2m-1-i}$, the last job $\mathcal{J}_{2m-1} = \{M_{2m-1}, R_{2m-1}\}$ arrives, where $M_{2m-1} = \{S_m x_1 - \sum_{i=1}^{m} s_i x_i\}$ and $R_{2m-1} = \left\{\frac{m(S_m)^{m-2}s_1}{(S_m-s_1)^{m-1}}\right\}$.

By the value of M_{2m-1}, we can conclude that the completion time of M_{2m-1} is at least x_1. Noting that the job J_{2m-1} has only one reduce task with size of $\frac{m(S_m)^{m-2}s_1}{(S_m-s_1)^{m-1}}$, its completion time is at least $x_1 + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}}$. Hence, we have $C^A(\mathcal{J}_{2m-1}) \geq x_1 + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}}$. It is not difficult to obtain that

$$\begin{aligned} C^*(\mathcal{J}_{2m-1}) &= \frac{p(M_{2m-1})}{S_m} + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}} \\ &= \frac{S_m x_1 - \sum_{i=1}^{m} s_i x_i}{S_m} + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}} \\ &= x_1 - \frac{m(S_m)^{m-2}}{S_m(S_m-s_1)^{m-2}} + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}}. \end{aligned}$$

By (4), we obtain

$$\begin{aligned} \gamma \geq \frac{C^A(\mathcal{J}_{2m-1})}{C^*(\mathcal{J}_{2m-1})} &\geq \frac{x_1 + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}}}{x_1 - \frac{m(S_m)^{m-2}}{S_m(S_m-s_1)^{m-2}} + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}}} \\ &\geq \frac{\gamma\frac{m(S_m)^{m-3}}{(S_m-s_1)^{m-2}} + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}}}{\gamma\frac{m(S_m)^{m-3}}{(S_m-s_1)^{m-2}} - \frac{m(S_m)^{m-2}}{S_m(S_m-s_1)^{m-2}} + \frac{m(S_m)^{m-2}}{(S_m-s_1)^{m-1}}} \\ &= \frac{\gamma + \frac{S_m}{S_m-s_1}}{\gamma + \frac{S_m}{S_m-s_1} - 1}, \end{aligned}$$

i.e.,

$$f(\gamma) \doteq \sum_{i=2}^{m} s_i \gamma^2 + \left(s_1 - \sum_{i=2}^{m} s_i\right)\gamma - \sum_{i=1}^{m} s_i \geq 0.$$

Since $f(0) = -1 < 0$, the quadratic equation $f(\gamma) = 0$ has a positive root and a negative root. Denote by γ^* the positive root and thus we have $\gamma \geq \gamma^*$.

3 An approximate algorithm for $Pm|MR(pmtn), online|C_{\max}$

In this section, we provide an approximate online algorithm A with a competitive ratio of $2 - \frac{1}{m}$ for the problem $Pm|MR(pmtn), online|C_{\max}$. Let l_j^i denote the completion time of machine σ_i at the moment right after the job J_j has been scheduled, $l_j^{i'}$ denote the completion time of machine σ_i at the moment right after the map

tasks of the job J_j has been scheduled, $i = 1, 2, \ldots, m$. Denote by C_j^A and C_j^* the makespan produced by algorithm A and the optimal makespan for the first j jobs, respectively.

Before presenting our algorithm, we introduce a procedure $P(M_j, R_j)$ to schedule the job $J_j = (M_j, R_j)$ for the case, where $l_{j-1}^1 = l_{j-1}^2 = \ldots = l_{j-1}^m$.

Procedure $P(M_j, R_j)$

(1) Assign all map tasks in M_j evenly to m machines.

(2) Use McNaughton's wrap-around rule to schedule all the reduce tasks in R_j.

(3) Reindex machines such that $l_j^1 \geq l_j^2 \geq \ldots \geq l_j^m$.

Now we present our algorithm A as follows:

Algorithm A

1. If $p(M_j) < (m-1)l_{j-1}^1 - \sum_{i=2}^m l_{j-1}^i$,

1.1 Schedule all map tasks in M_j between machines to finish them as early as possible,

1.2 Take the longest reduce task r_j^k in R_j not yet processed,

if $l_{j-1}^{i'} + r_j^k \geq l_{j-1}^{1'}$, where $l_{j-1}^{i'}$ is the least loaded processor, partition r_j^k into two parts $r_j^{k_1}$ and $r_j^{k_2}$ such that $l_{j-1}^{i'} + r_j^{k_1} = l_{j-1}^{1'}$. Then schedule $r_j^{k_1}$ on σ_i, let $l_{j-1}^{i'} = l_{j-1}^{i'} + r_j^{k_1}$;

else schedule r_j^k on σ_i, let $l_{j-1}^{i'} = l_{j-1}^{i'} + r_j^k$;

1.3 Repeat the aforementioned steps until the reduce tasks in R_j are scheduled once, or the completion time of each machine is $l_{j-1}^{1'}$ reindex machines such that $l_{j-1}^{1'} \geq l_{j-1}^{2'} \geq \ldots \geq l_{j-1}^{m'}$. Then use McNaughton's rule to schedule the leftover reduce tasks in R_j from the time $l_{j-1}^{1'}$.

1.4 Reindex machines such that $l_j^1 \geq l_j^2 \geq \ldots \geq l_j^m$.

2. If $p(M_j) > (m-1)l_{j-1}^1 - \sum_{i=2}^m l_{j-1}^i$,

2.1 schedule the portion $p(M_j^i) = l_{j-1}^1 - l_{j-1}^{i+1} (i = 1, 2, \ldots, m-1)$ of map tasks on σ_{i+1} such that $l_{j-1}^{i+1} + p(M_j^i) = l_{j-1}^1 (i = 1, 2, \ldots, m-1)$ and denote the remainder of map tasks as $M_j' = M_j \backslash \cup_{i=1}^{m-1} M_j^i$.

2.2 Run procedure $P(M_j', R_j)$ for the job $J_j' = (M_j', R_j)$.

Remark. From the aforementioned algorithm A and procedure P, we always have $l_j^1 \geq l_j^2 \geq \ldots \geq l_j^m$ after scheduling the job J_j.

Theorem 3.1. *For any* $1 \leq j \leq n$, *we have* $\frac{C_j^A}{C_j^*} \leq 2 - \frac{1}{m}$.

Proof. We show the result by induction on j.

First, we consider the case $j = 1$, i.e., the assignment of the first job J_1. Noting that $l_{j-1}^1 = l_{j-1}^2 = \ldots = l_{j-1}^m = 0$, the algorithm schedules the job J_1 by procedure P. By procedure P, we conclude that $C_1^A = C_1^*$, i.e., $\frac{C_1^A}{C_1^*} = 1 \leq 2 - \frac{1}{m}$.

Hence, the result holds at $j = 1$.

Suppose that the result holds at $j-1 (j \geq 2)$, i.e., $\frac{C_{j-1}^A}{C_{j-1}^*} \leq 2 - \frac{1}{m}$. We consider the assignment of J_j below.

(1) If $M_j < (m-1)l_{j-1}^1 - \sum_{i=2}^m l_{j-1}^i$, we discuss two cases.

Case 1: After scheduling the job J_j, we have $l_{j-1}^1 = l_j^1 \geq l_j^2 \geq \ldots \geq l_j^m$. Thus, we conclude that $C_j^A = C_{j-1}^A$, $C_j^* \geq C_{j-1}^*$. It yields that $\frac{C_j^A}{C_j^*} \leq 2 - \frac{1}{m}$.

Case 2: After scheduling the job J_j, we have $l_{j-1}^1 < l_j^1$. Suppose that r_j^k is the last finished reduce task and its start time is s, the makespan produced by Algorithm A is $s + r_j^k$. We conclude that $C_j^A = s + r_j^k \leq (\sum_{i=1}^j (p(M_i) + p(R_i)) - r_j^k)/m + r_j^k$ and $C_j^* \geq \max\left\{\frac{\sum_{i=1}^j (p(M_i) + p(R_i))}{m}, \frac{M_j}{m} + r_j^1\right\}$. Thus, $\frac{C_j^A}{C_j^*} \leq ((\sum_{i=1}^j (p(M_i) + p(R_i)) - r_j^k)/m + r_j^k)/C_j^* \leq 2 - \frac{1}{m}$.

(2) If $p(M_j) \geq (m-1)l_{j-1}^1 - \sum_{i=2}^{m} l_{j-1}^i$, we discuss two cases.

Case 1: $r_j^1 \leq \frac{p(R_j)}{m}$. In this case, we have $C_j^A = \frac{\sum_{i=1}^{j}(p(M_i)+p(R_i))}{m} = C_j^*$. Thus, the desired result holds.

Case 2: $r_j^1 > \frac{p(R_j)}{m}$. In this case, we have $C_j^A = (\sum_{i=1}^{j-1}(p(M_i)+p(R_i)) + M_j)/m + r_j^1$ and $C_j^* \geq \max\left\{\frac{\sum_{i=1}^{j}(p(M_i)+p(R_i))}{m}, \frac{p(M_j)}{m} + r_j^1\right\}$. Thus, $\frac{C_j^A}{C_j^*} \leq ((\sum_{i=1}^{j-1}(p(M_i)+p(R_i)) + p(M_j))/m + r_j^1)/C_j^* \leq ((\sum_{i=1}^{j}(p(M_i)+p(R_i)) - p(R_j))/m + r_j^1)/C_j^* \leq 2 - \frac{1}{m}$.

Hence, the result holds.

By now, the result holds for any $j \geq 2$ and the proof is complete.

4 Conclusion

We study online scheduling on m parallel machines in a MapReduce-like system where the map tasks can be arbitrarily split and processed in parallel on multiple machines.

There are some related problems that deserve further study. A natural question is to study whether the lower bound of the non-preemptive version $Qm|MR|C_{\max}$ is the same as that of $Qm||C_{\max}$. Can we apply the algorithm for $Qm||C_{\max}$ to tackle the problem $Qm|MR|C_{\max}$ and obtain an online algorithm? It is also interesting to design a preemptive online algorithm.

Acknowledgement: The authors thank the reviewers for their constructive remarks on their work.

Funding information: The work is supported by the National Science Foundation of China under Grant 12001313, the Natural Science Foundation of Shandong Province of China under Grant ZR2020QA023 and the National Natural Science Foundation of China 12271295.

Author contributions: All authors have accepted responsibility for the entire content of this manuscript and consented to its submission to the journal, reviewed all the results and approved the final version of the manuscript.

Conflict of interest: The authors declare that they have no conflicts of interest.

References

[1] J. Dean and S. Ghemawat, *MapReduce: simplified data processing on large clusters*, Commun. ACM **51** (2008), 107–113, DOI: https://doi.org/10.1145/1327452.1327492.

[2] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar and A. Goldberg, *Quincy: Fair scheduling for distributed computing clusters*, Proceedings of ACM SIGOPS 22nd Symposium on Operating Systems Principles, (Big Sky Montana USA), 209, October 11–14, pp. 261–276, DOI: https://doi.org/10.1145/1629575.1629601.

[3] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy, S. Shenker and I. Stoica, *Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling*, Proceedings of Fifth EuroSys Conference 2010, (Paris France), 2010, April 13–16, pp. 265–278, DOI: https://doi.org/10.1145/1755913.1755940.

[4] L. Kolb, A. Thor, and E. Rahm, *Load balancing for mapreduce based entity resolution*, Proceedings of 2012 IEEE 28th International Conference on Data Engineering, 2012, April 1–5, pp. 618–629, DOI: https://doi.org/10.1109/ICDE.2012.22.

[5] A. Bechini, F. Marcelloni, and A. Segatori, *A MapReduce solution for associative classification of big data*, Inf. Sci. **332** (2016), 33–55, DOI: https://doi.org/10.1016/j.ins.2015.10.041.

[6] B. Moseley, A. Dasgupta, R. Kumar, and T. Sarlós, *On scheduling in Map-Reduce and flow-shops*, Proceedings of 23rd ACM Symposium on Parallelism in Algorithms and Architectures, (San Jose California USA), 2011, June 4–6, pp. 289–298, DOI: https://doi.org/10.1145/1989493.1989540.

[7] Y. Zheng, N. Shroff, and P. Sinha, *A new analytical technique for designing provably efficient MapReduce schedulers*, Proceeding of 2013 Proceedings IEEE INFOCOM, 2013, April 14–19, pp. 1600–1608, DOI: https://doi.org/10.1109/INFCOM.2013.6566956.

[8] Y. Zhu, Y. Jiang, W. Wu, L. Ding, A. Teredesai, D. Li, et al., *Minimizing makespan and total completion time in MapReduce-like systems*, Proceeding of IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 2166–2174, DOI: https://doi.org/10.1109/INFOCOM.2014.6848159.

[9] T. Luo, Y. Zhu, W. Wu, Y. Xu, and D. Du, *Online makespan minimization in MapReduce-like systems with complex reduce tasks*, Optim. Lett. **2015** (2015), 271–277, DOI: https://doi.org/10.1007/s11590-015-0902-7.

[10] H. Chang, M. Kodialam, R. Kompella, T. Lakshman, M. Lee, and S. Mukherjee, *Scheduling in MapReduce-like systems for fast completion time*, Proceeding of IEEE INFOCOM, 2011, pp. 3074–3082, DOI: https://doi.org/10.1109/infcom.2011.5935152.

[11] Y. Jiang, W. Zhou, and P. Zhou, *An optimal preemptive algorithm for online MapReduce scheduling on two parallel machines*, Asia-Pac. J. Oper. Res. **35** (2018), 185003, DOI: https://doi.org/10.1142/S0217595918500136.

[12] J. Huang, F. Zheng, Y. Xu, and M. Liu, *Online MapReduce processing on two identical parallel machines*, J. Comb. Optim. **35** (2018), 216–223, DOI: https://doi.org/10.1007/s10878-017-0167-4.

[13] Y. Jiang, P. Zhou, T. Cheng, and M. Ji, *Optimal online algorithms for MapReduce scheduling on two uniform machines*, Optim. Lett. **13** (2019), 1663–1676, DOI: https://doi.org/10.1007/s11590-018-01384-8.

[14] C. Chen, Y. Xu, Y. Zhu, and C. Sun, *Online MapReduce scheduling problem of minimizing the makespan*, J. Comb. Optim. **33** (2017), 590–608, DOI: https://doi.org/10.1007/s10878-015-9982-7.

[15] R. Jeyaraj, V.S. Ananthanarayana, and A. Paul, *Improving MapReduce scheduler for heterogeneous workloads in a heterogeneous environment*, Concurr. Comput. Pract. Exp. **32** (2020), 1–10, DOI: https://doi.org/10.1002/cpe.5558.

[16] X. Li, F. Chen, R. Ruiz, and J. Zhu, *MapReduce task scheduling in heterogeneous geo-distributed data centers*, IEEE Trans. Serv. Comput. **15** (2022), 3317–3329, DOI: https://doi.org/10.1109/TSC.2021.3092563.

[17] X. Wang, C. Wang, M. Bai, Q. Ma, and G. Li, *HTD: heterogeneous throughput-driven task scheduling algorithm in MapReduce*, Distrib. Parallel. Dat. **40** (2022), 135–163, DOI: https://doi.org/10.1007/s10619-021-07375-6.