Recent Results for Online Makespan Minimization  
(Extended Abstract) 

Susanne Albers  
Department of Computer Science  
Humboldt-Universität zu Berlin  
albers@informatik.hu-berlin.de

Overview: We study a classical scheduling problem that has been investigated for more than forty years. Consider a sequence of jobs $\sigma = J_1, \ldots, J_n$ that has to be scheduled on $m$ identical parallel machines. Each job $J_t$ has an individual processing time $p_t$, $1 \leq t \leq n$. Preemption of jobs is not allowed. The goal is to minimize the makespan, i.e. the maximum completion time of any job in the constructed schedule. In the offline variant of the problem all jobs of $\sigma$ are known in advance. In the online variant the jobs arrive one by one. Each incoming job $J_t$ has to be assigned immediately to one of the machines without knowledge of any future jobs $J_{t'}, t' > t$.

Already in the 1960s Graham [7] presented the famous List scheduling algorithm, which schedules each job of $\sigma$ on a machine that currently has the smallest load. List can be used in the offline as well as the online setting. Graham proved that the strategy is $(2 - 1/m)$-competitive, i.e. for any $\sigma$ the makespan of List’s schedule is at most $2 - 1/m$ times the makespan of an optimal schedule. In the 1980s Hochbaum and Shmoys [8] developed a famous polynomial time approximation scheme for the offline problem. Research over the past two decades has focused on the online problem. The best deterministic online strategy currently known achieves a competitive ratio of about 1.92, see [6]. Moreover, no deterministic online algorithm can attain a competitiveness smaller than 1.88, cf. [10].

During the last years online makespan minimization has been explored assuming that an online algorithm is given additional power or extra information while processing a job sequence $\sigma$. It turns out that usually significantly improved competitive ratios can be achieved. We survey these recent advances. The considered forms of resource augmentation are generally well motivated from a practical point of view.

Job migration: Assume that at any time an online algorithm may perform reassignments, i.e. jobs already assigned to machines may be removed and transferred to other machines. Job migration is a well-known and widely used technique to balance load in parallel and distributed systems. Sanders et al. [11] study the setting that after the arrival of each job $J_t$, jobs up to a processing volume of $\beta p_t$ may be migrated, where $\beta$ is a constant. For $\beta = 4/3$, they present a 1.5-competitive algorithm. They also devise a $(1 + \epsilon)$-competitive...
algorithm, for any $\epsilon > 0$, where $\beta$ depends exponentially on $1/\epsilon$. Albers and Hellwig [1] investigate the scenario where an online algorithm may migrate a limited number of jobs; this number does not depend on the job sequence. They give an algorithm that, for any $m \geq 2$, is $\alpha_m$-competitive, where $\lim_{m \to \infty} \alpha_m = W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2)) \approx 1.4659$. Here $W_{-1}$ is the lower branch of the Lambert $W$ function. The total number of migrations is at most $7m$, for $m \geq 11$. The competitiveness of $\alpha_m$ is best possible: No deterministic online algorithm that uses $o(n)$ migrations can achieve a competitive ratio smaller than $\alpha_m$.

**Availability of a reordering buffer:** In this setting an online algorithm has a buffer of limited size that may be used to partially reorder the job sequence. Whenever a job arrives, it is inserted into the buffer; then one job of the buffer is removed and assigned in the current schedule. For $m = 2$ machines, Zhang [12] and Kellerer et al. [9] give 4/3-competitive algorithms. Englert et al. [5] explore the setting for general $m$. They develop an algorithm that, using a buffer of size $O(m)$, is $\alpha_m$-competitive, where again $\lim_{m \to \infty} \alpha_m = W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2)) \approx 1.4659$. Furthermore they prove that if an online algorithm achieves a competitiveness smaller than $\alpha_m$, then the buffer size must depend on $\sigma$. The paper by Englert et al. [8] also considers makespan minimization with a reordering buffer on uniformly related machines.

**Information on total processing time or optimum makespan:** First assume that an online algorithm knows the sum $\sum_{t=1}^{n} p_t$ of the jobs’ processing times. The access to such a piece of information can be justified as follows. In a parallel server system there usually exist fairly accurate estimates on the workload that arrives over a given time horizon. Furthermore, in a shop floor a scheduler typically accepts orders (tasks) of a targeted volume for a given time period, say a day or a week. For $m = 2$ machines, Kellerer et al. [9] present a 4/3-competitive algorithm. For a general number $m$ of machines, Cheng et al. [4] propose a 1.6-competitive algorithm. Albers and Hellwig [2] prove that no deterministic online algorithm can attain a competitiveness smaller than 1.585. In a stronger scenario an online algorithm even knows the value of the optimum makespan, for the given $\sigma$. This framework can also be viewed as a bin stretching problem, see [3]. Obviously, the algorithm by Cheng et al. [4] is also 1.6-competitive in this setting. Azar and Regev [3] show that no online algorithm can be better than 4/3-competitive.

**References**