Online Algorithms for Scheduling Unit Jobs

Jiří Sgall*

In this abstract we give a short survey of results from [5, 3] on the following *unit-job scheduling* problem. We are given a set of unit-length jobs, with each job j specified by a triple (r_j, d_j, w_j) where r_j and d_j are integral release times and deadlines, and w_j is a non-negative real weight. We have a single machine, i.e., one job can be processed at each integer time. We use the term *weighted throughput* or *gain* for the total weight of the jobs completed by their deadline. The goal is to compute a schedule that maximizes the *weighted throughput*. In Graham's notation, this problem is $1|p_j = 1, r_j| \sum_j w_j U_j$.

We focus on the online problem, where each job arrives at its release time. At each time step t, an online algorithm needs to schedule one of the pending jobs (i.e., j with $r_j \leq t < d_j$ and not scheduled before) without any knowledge of the jobs that will be released later in the future. An online algorithm is R-competitive, if its gain on any instance is at least $1/R$ times the optimum (offline) gain. This scheduling problem is equivalent to the online *bounded delay buffer problem* recently introduced [7, 1, 8] to model the trade-offs arising in managing buffers for storing packets in QoS networks.

In addition to the general problem, we study some restricted versions proposed in the literature [7, 1, 4, 8]. In *s-bounded instances*, the span of the jobs (defined as the difference between the deadline and the release time) is at most s, and in *s-uniform instances* the span of each job is exactly s. Finally, an instance is *similarly ordered* if the release times and deadlines are similarly ordered, that is $r_i < r_j$ implies $d_i \leq d_j$ for any two jobs i and j.

The known bounds are summarized in the table below. A blank entry indicates that the bound in this entry follows from another bound in the same column. The table proceeds from more general to more restricted classes of instances, with the exception of the last two lines. For these, note that similarly ordered instances include both s-uniform and 2-bounded instances, but not s-bounded ones for $s \geq 3$.

	deterministic		randomized	
	upper bound	lower bound	upper bound	lower bound
General	1.939... [5]		1.582... [3]	
s-bounded	$2 - 2/s + o(1/s)$ [3]			
4-bounded	1.732... [3]			
3-bounded	1.618... [3]			
2-bounded	1.618... [7]	1.618... [1, 4, 6]	1.25 [3]	1.25 [4]
2-uniform	1.377... [5]	1.377... [5]		1.172.... [3]
s-uniform	1.75 [2]			1.25 for $s \to \infty$ [3]
Similarly ordered	1.838... [5]			

Some algorithms

A simple greedy algorithm that always schedules the heaviest available job is 2-competitive [7, 6]. The example on which its ratio approaches 2 exploits the fact that this algorithm prefers the heaviest job, even in presence of more urgent jobs with almost the same weight. One natural idea for an improvement

*Mathematical Institute, AS CR, Žitná 25, CZ-11567 Praha 1, Czech Republic. Email: sgall@math.cas.cz

Dagstuhl Seminar Proceedings 05031
Algorithms for Optimization with Incomplete Information
http://drops.dagstuhl.de/opus/volltexte/2005/69

is to schedule the earliest-deadline pending job among those with weight at least some fixed fraction of the heaviest job. This yields the algorithm EDF$_\alpha$ below. It gives an improved competitive ratio for the s-bounded case, and in particular EDF$_{\phi-1}$ is $\phi \approx 1.618$-competitive for 3-bounded instances, matching the lower bound. However, for general instances, EDF$_\alpha$ is only 2-competitive.

Algorithm EDF$_\alpha$**:** At each step, let h be the heaviest pending job and let f be the earliest-deadline pending job such that $w_f \geq \alpha w_h$. Execute f.

A small technical note: whenever the algorithms need to break ties, the heaviest job is chosen among those with the same deadline.

Our randomized algorithm RMIX presented below resembles EDF$_\alpha$ with the change that it sets the threshold randomly, independently for each time. This algorithm is $\frac{e}{e-1} \approx 1.582$-competitive, giving the currently best algorithm for general instances. Note that the competitive ratio is smaller than the deterministic lower bound, thus randomization provably helps for this problem.

Algorithm RMIX**.** At each step, let h be the heaviest pending job. Select a real $x \in [-1, 0]$ uniformly at random. Let f be the earliest-deadline pending job with $w_f \geq e^x w_h$. Execute f.

Finally, we present a deterministic algorithm for general instances which is better than 2-competitive. The general idea is to alternate the greedy rule (schedule the heaviest job) and the EDF rule (schedule the earliest-deadline job among the sufficiently heavy jobs). This simple algorithm, as stated, still has ratio no better than 2. After some minor modifications, the algorithm GENFLAG presented below is $64/33 \approx 1.939$-competitive deterministic algorithm for unit-job scheduling. An algorithm based on the same ideas is 1.838-competitive for similarly ordered instances.

Algorithm GENFLAG**:** We use parameters $\alpha = 7/11$, $\beta = 8/11$, and a Boolean variable *eflag*, initially set to *false*, that stores information about the previous step. At a given time step t, let h be the heaviest pending job and e the earliest-deadline job among the pending jobs with weight at least αw_h. Schedule either e or h according to the following procedure:

`if` *eflag* = *false*
  `then` schedule e; `if` $e \neq h$ `then` set *eflag* $\leftarrow$ *true*
  `else` `if` $d_e = t + 1$ and $w_e \geq \beta w_h$ `then` schedule e `else` schedule h;
    set *eflag* $\leftarrow$ *false*

Related models and open problems

FIFO buffering. Kesselman *et al.* [7, 8] consider a different model related to the s-uniform case. Here packets do not have individual deadlines, instead they are stored in a buffer of capacity s and they are required to be served in FIFO order (i.e., if a packet is served, all packets in the buffer that arrived before the served one are dropped). Any algorithm in this FIFO model applies also to s-uniform instances in the scheduling model and has the same competitive ratio [8].

Similarly as in unit-job scheduling, a simple greedy algorithm is 2-competitive for this problem. (Here the greedy algorithm upon arrival of a new packet inserts it in the buffer if there is space or if there is a smaller packet in the buffer—then the smallest packet is dropped; the first packet in the buffer is scheduled after handling all the new packets.) The algorithm PG presented below was proposed in [8] and later in [2] it was proven to be 1.75-competitive. (In [2] a slight modification of PG is analyzed, but it is easy to see that its competitive ratio is not smaller than that of PG.) This implies a 1.75-competitive algorithm for the s-uniform case of unit-job scheduling. The best lower bound for FIFO buffering is 1.419, while it is known that PG is not better than $1 + \sqrt{2}/2 \approx 1.707$ competitive (unpublished).

Algorithm PG**:** Upon an arrival of a new packet of weight w:

(i) if the buffer contains a packet of weight at most $w/4$, remove the first such packet from the buffer;

(ii) if the buffer is full and the smallest packet has weight at most w, remove it from the buffer;

(iii) if now the buffer is not full, insert the new packet at the end of the buffer.

After all new packets are handled, schedule the first packet from the buffer.

A very interesting problem is to combine the FIFO model with the case of similarly ordered instances of unit-job scheduling. For similarly ordered instances the optimum can also be transformed so that it obeys the FIFO rule. The greedy algorithm from the FIFO model is 2-competitive in this case as well. But no better algorithm is known: the competitive ratio of PG is 2 on similarly ordered instances, on the other hand, GenFlag and its variants do not obey the FIFO order. Designing a better algorithm for similarly ordered instances that obeys the FIFO order would combine the good features of both models; it could also help to develop new algorithms for both models—note that in both cases there is a significant gap between upper and lower bounds.

Memoryless algorithms. Another interesting question is whether good algorithms can be memoryless. An algorithm is *memoryless* if (i) the decision as to which job to execute is based only on the weights of the pending jobs, and (ii) the algorithm is invariant under scaling.

The optimal deterministic algorithm $\text{Edf}_{\phi-1}$ for 3-bounded instances as well as the optimal randomized algorithm for 2-bounded instances are memoryless. On the other hand, the optimal deterministic algorithm for 2-uniform instances is not memoryless and the same competitive ratio cannot be achieved by a memoryless algorithm [3, 5]. Our algorithm GenFlag for general instances is not memoryless, as it needs the one bit of the boolean variable *eflag*. It would be surprising if this single bit of memory is needed for a better than 2-competitive algorithm. Nevertheless, whether it is possible to reduce the ratio of 2 with a memoryless algorithm for general instances remains an open problem.

Closing the gaps. Despite the progress in designing online algorithms for unit-job scheduling and FIFO buffering, the gap between the lower and upper bounds is still wide both for deterministic and randomized algorithms. Closing or substantially reducing these gaps seems to be a challenging open problem.

Acknowledgments. Without many discussions with Marek Chrobak, Wojciech Jawor, Tomáš Tichý, and other coauthors, there would be no results to cover.

References

[1] N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies in QoS switches. In *Proc. 14th SODA*, pp. 761–770. ACM/SIAM, 2003.

[2] N. Bansal, L. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko. Further improvements in competitive guarantees for QoS buffering. In *Proc. 31st ICALP*, LNCS 3142, pp. 196–207. Springer, 2004.

[3] Y. Bartal, F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, R. Lavi, J. Sgall, and T. Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. In *Proc. 21st STACS*, LNCS 2996, pp. 187–198. Springer, 2004.

[4] F. Y. L. Chin and S. P. Y. Fung. Online scheduling for partial job values: Does timesharing or randomization help? *Algorithmica*, 37:149–164, 2003.

[5] M. Chrobak, W. Jawor, J. Sgall, and T. Tichý. Improved online algorithms for buffer management in QoS switches. In *Proc. 12th ESA*, LNCS 3221, pp. 204–215. Springer, 2004.

[6] B. Hajek. On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. In *Conference in Information Sciences and Systems*, pp. 434–438, 2001.

[7] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in QoS switches. *SIAM J. Comput.*, 33:563–583, 2004.

[8] A. Kesselman, Y. Mansour, and R. van Stee. Improved competitive guarantees for QoS buffering. In *Proc. 11th ESA*, LNCS 2832, pp. 361–372. Springer, 2003.