A New Online Packet Scheduling Algorithm *

Hsiang-Chen Chan & Jen-Chun Chang & Hsin-Lung Wu & Tsung-Chien Wu
Department of Computer Science and Information Engineering
National Taipei University, New Taipei City, Taiwan
rasl20264@gmail.com, {jcchang,hsinlung}@mail.ntpu.edu.tw, pchome119@gmail.com

Abstract

In this paper, we study online unit-job scheduling problem. We propose a new online algorithm called VR_α *which is modified from the greedy online algorithm. We give a proof that* VR_α *obtains 2-competitiveness for any* $1 \leq \alpha \leq 2$.

1 Introduction

1.1 Online Unit-Job Scheduling Problem

The offline unit-job scheduling problem is as follows. Given a set of unit-length jobs, each job j in this set is specified by a triple (r_j, d_j, v_j) where r_j and d_j are integral release times and deadlines, and v_j is a non-negative real value. Only one job can be processed at each integer time. We define the *gain* of the set of unit-length jobs by the total values of the jobs completed before their deadline. The goal of the off-line algorithm is to compute a schedule which maximizes the gain.

In the online unit-job scheduling problem, each job arrives at their release time. At each time step, the online algorithm has to choose and schedule one of the pending jobs in the buffer without knowing those jobs which will be released later in the future. We say an online algorithm $\mathcal{A}$ is *c-competitive* if its gain on any instance is at least $1/c$ times the gain of the optimal offline algorithm. The competitive ratio of $\mathcal{A}$ is the smallest c such that $\mathcal{A}$ is c-competitive online algorithm. One can view the online problem as a game between an online algorithm $\mathcal{A}$ and an *adversary*, who issues the jobs and schedules them in order to maximize the ratio between the adversary's gain and the gain of $\mathcal{A}$. The upper bound of the competitive ratio is achieved by some online algorithms. In addition, a competitive ratio strictly less than the lower bound cannot be obtained by any online algorithm. An online algorithm is optimal if it has its competitive ratio the same as the lower bound.

Finally, we call an online algorithm *memoryless* if this online algorithm's decision of sending packets in only based on the contents its current buffer and independent of the packets that have already been released and processed.

As mentioned in [1, 5, 7], the offline unit-job scheduling problem can be solved efficiently in time $O(n^3)$ where n is the number of packets released in the input instance.

For online unit-job scheduling problem, the currently best known upper bound is $2\sqrt{2}-1 \approx 1.828$ [3]. The best known lower bound is $\frac{1+\sqrt{5}}{2} \approx 1.618$ [2, 4].

2 Modified Online Algorithms from the Greedy Online Algorithm

In this paper, we consider several online algorithms which improve the simple greedy online algorithm. Here, the greedy online algorithm always sends the pending job which has the highest weight in the buffer. It can be proved that such a greedy algorithm obtains 2-competitiveness [4, 5].

In [1], Chin et al. proposed an algorithm called EDF_α which sends the earliest-deadline packet whose weight is at least $1/\alpha$ times of the highest value of a pending packet where $\alpha \geq 1$. As mentioned in [8], they show that the competitiveness of EDF_α is at least 2 by the following example.

Example 1. *Let* (v_p, d_p) *denote a packet* p *with weight* v_p *and deadline* d_p. *At each time* $t = 0, 1, \ldots, n-1$, ϵ *is defined any constant numbers, two packets* $p_t = (1-\epsilon, t+1)$ *and* $q_t = (1, n+t+1)$ *are released. Moreover, at time* 0, *a packet* $P = (\alpha, 2n+1)$ *is released where* $\alpha > 1$. *At time* t *with* $t \leq n-1$, *the algorithm* EDF_α *sends the packet* q_t

*This work was supported in part by the National Science Council of Taiwan under contract NSC-102-2221-E-305-008-MY3.

and the packet p_t expires, and EDF_α *retains P in the buffer. At time n, the buffer only contains the packet P and* EDF_α *sends P out. Thus, the gain of* EDF_α *is $n+\alpha$. In addition, the optimal offline algorithm can send all packets out. Thus its gain is $(2-\epsilon)n+\alpha$. The competitive ratio of* EDF_α *approaches 2 as n increases.*

In order to improve EDF_α, Li [6] proposes the Modified Greedy ($\mathsf{MG}_{\alpha,\beta}$) algorithm. Before describing the MG algorithm, we need some definitions.

Definition 1. *A provisional schedule S for a set of pending packets P specifies which packet should be sent in which time step under the assumption that new packets do not arrive in the future.*

Definition 2. *An optimal provisional schedule is a provisional schedule which achieves the maximum total value among all the provisional schedules.*

Definition 3. *For two packets p and q, we define $p \prec q$ if either $d_p < d_q$, or $d_p = d_q$ and $v_p > v_q$, or $d_p = d_q$, $v_p = v_q$, and $r_p < r_q$. The relation $\prec$ is called the canonical order.*

Now we describe the algorithm MG.

Algorithm $\mathsf{MG}_{\alpha,\beta}(t)$:

1. Calculate an optimal provisional schedule S_t for the set of pending packets at time t.
2. All packets is S_t are sorted in a canonical order.
3. Let e denote the first packet and h denote the first highest-value packet.
4. If $\alpha v_e \geq v_h$, then send e else send the first packet f satisfying $v_f \geq \max\{\frac{v_h}{\alpha}, \beta v_e\}$.

The competitiveness of MG is guaranteed by the following theorem.

Theorem 1 ([6]). $\mathsf{MG}_{\alpha,\beta}$ *is 2-competitive for any $1 \leq \beta \leq \alpha \leq 2$.*

Remark 1. *At each time step t,* MG *has to calculate the optimal provisional schedule S_t for the set P_t of pending packets at time t. Note that this requires $O(|P_t|^3)$ time.*

3 Our Proposed Strategy

Let us begin with a definition.

Definition 4. *We define the remaining time $R_p(t)$ of the packet p at the time t by $R_p(t) = d_p - t$.*

The main idea of our strategy is based on the ratio between the value and the remaining time of each pending packet.

Definition 5. *For any packet $p = (d_p, v_p)$ and for any time t, we define* $\mathrm{RVR}(p,t) = \frac{v_p}{R_p(t)}$.

At each time step $t+1$, the basic idea of our strategy is to deliver the packet p which obtains the highest $\mathrm{RVR}(p,t)$ value among pending packets at time t. We modify our basic strategy to obtain the following algorithm VR.

Algorithm $\mathsf{VR}_\alpha(t)$:

1. Sort all pending packets at time $t-1$ in a decreasing order according to the value $\mathrm{RVR}(p,t-1)$. Call this sorted list S_{t-1}.
2. Let e denote the first packet in S_{t-1} and h denote the first highest-value packet in S_{t-1}.
3. If $\alpha v_e \geq v_h$, then send e else send the first packet f in S_{t-1} satisfying $v_f \geq \frac{v_h}{\alpha}$.

If the number of packets with the highest $\mathrm{RVR}(p,t)$ is more than one, we will randomly select one. The competitiveness of VR_α is guaranteed by the following theorem.

Theorem 2. VR_α *is 2-competitive for any $1 \leq \alpha \leq 2$.*

Proof. Let ADV be an adversary. At the beginning of each time step, we will modify ADV such that ADV and VR have the same buffer. For convenience, we use the following definition. Given an ordered set S and two elements $x, y \in S$, define $x <_S y$ if x appears in front of y in the list S. Assume that, at time t, VR delivers a packet f. The proof can be divided into three cases.

1. Assume that ADV sends the same packet f. Then their gains are the same. At the end, they have the same buffer.
2. Assume that ADV sends a packet $a \neq f$ with $f <_{S_{t-1}} a$ and $d_a \leq d_f$. We allow ADV sends f in this step and then we insert the packet a into ADV's buffer. At the end, VR and ADV

have the same buffer. Next, we compute their gains. Since $f <_{S_{t-1}} a$, we have $\frac{v_f}{d_f - t} \geq \frac{v_a}{d_a - t}$. Then, by the assumption that $d_a \leq d_f$, we conclude that $v_f \geq v_a$. Thus, in this case, the gain of ADV is at most $v_f + v_a \leq 2v_f$. So ADV's gain is less than or equal to 2 times of VR's gain v_f.

3. Assume that ADV sends a packet $a \neq f$ with $a <_{S_{t-1}} f$ or $d_a > d_f$. In the end of the step, we replace the packet f in the ADV's buffer by the packet a. After doing that, ADV's buffer is the same as VR's buffer. Next, we compute the gains of ADV and VR. We always have $v_f \geq \frac{v_h}{\alpha} \geq \frac{v_a}{\alpha} \geq \frac{v_a}{2}$. Thus, ADV's gain is at most 2 times of VR's gain.

By analyzing the three cases, we conclude that the gain of ADV is at most 2 times of the gain of VR. □

3.1 Performance Analysis

It is easy to see that the algorithm VR_α is better than EDF_α since, in Example 1, the gain of VR_α is close to the optimal offline gain while the gain of EDF_α is close to 2 times of the optimal offline gain.

In addition, at each time t, the time complexity of VR is $O(|P_t| \log |P_t|)$ which is much smaller than the time complexity $O(|P_t|^3)$ of MG where P_t is the set of pending packets at time t.

4 Conclusion

In this paper, we study online unit-job scheduling problem. We propose a new online algorithm called VR_α. We prove that VR_α is a 2-competitive online algorithm for any $1 \leq \alpha \leq 2$.

References

[1] F.Y.L. Chin, M. Chrobak, S.P.Y. Fung, W. Jawor, J. Sgall, and T. Tichý, " Online competitive algorithms for maximizing weighted throughput of unit jobs," *Journal of Discrete Algorithms*, 4(2), pp. 255–276, 2006.

[2] F.Y.L. Chin and S.P.Y. Fung, "Online scheduling with partial job values: Does timesharing or randomization help?," *Algorithmica*, 37(3), pp. 149–164, 2003.

[3] M. Englert and M. Westermann, "Considering suppressed packets imporveds buffer management in Qos switches," in *Proceedings of the 18th Annumal ACM SIAM Symposium on Discrete Algorithms (SODA)*, pp. 209–218, 2007.

[4] B. Hajek, "On the competitiveness of online scheduliing of unit-length packets with hard deadlines in slotted time," in *Proceedings of the 35th Annumal Conference on Information Science and Systems (CISS)*, pp. 434–438, 2001.

[5] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, M. Sviridenko, "Buffer overflow management in Qos switches," *SIAM Journal on Computing*, 33(3), pp. 434–438, 2004.

[6] F. Li, "A comprehensive study of an online packet scheduling algorithm," *Theoretical Computer Science*, Vol. 497, pp. 31–38, 2013.

[7] F. Li, J. Sethuraman, and C. Stein, " An optimal online algorithm for packet scheduling with agreeable deadlines," in *Proceedings of the 16th Annumal ACM SIAM Symposium on Discrete Algorithms (SODA)*, pp. 801–802, 2005.

[8] F. Li, J. Sethuraman, and C. Stein, "Better online buffer management," in *Proceedings of the 18th Annumal ACM SIAM Symposium on Discrete Algorithms (SODA)*, pp. 199–208, 2007.