Forwarding Packets Greedily on the Line

Joan Boyar
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

Lene M. Favrholdt
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

Kim S. Larsen
Department of Mathematics and Computer Science, University of Southern Denmark, Odense, Denmark

Kevin Schewior
Department of Computer Science, University of Cologne, Germany

Rob van Stee
Department of Mathematics, University of Siegen, Germany

Abstract

We consider the problem of forwarding packets arriving online with their destinations in a line network. In each time step, each router can forward one packet along the edge to its right, and the packet arrives at the next router one time step later. Packets are forwarded until they reach their destination. The flow time of a packet is the elapsed time between its release and its arrival at its destination. The goal is to minimize the maximum flow time.

This problem was introduced by Antoniadis et al. in 2014, with a focus on line networks. They proposed several natural algorithms. For one, they proved that it is not $O(1)$-competitive; for others, they claimed analogous lower bounds, seemingly leaving no natural candidate for an $O(1)$-competitive algorithm.

In this paper, we study a natural algorithm not considered in that work. Our algorithm, simply called Greedy, selects packets according to their projected flow time under the assumption that they are not delayed any further. We focus on the special case in which each packet needs to be forwarded by one or two routers; this case captures core difficulties. We show that Greedy achieves a competitive ratio of exactly $2 - 2^{1-k}$, where k is the number of active routers in the network.

We also give the first nontrivial general lower bound, which applies even to randomized algorithms: using the same type of instances as in our lower bound for Greedy, we show that no algorithm can be $(4/3 - \varepsilon)$-competitive for any $\varepsilon > 0$.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Online algorithms, Packet scheduling, Greedy algorithm

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.42

Funding The first four authors were supported in part by the Independent Research Fund Denmark, Natural Sciences, grants DFF-0135-00018B and DFF-4283-00079B.

1 Introduction

We consider packet scheduling in a network. Routers often have backlogs of packets awaiting transmission, so a policy is needed to decide which packet to forward at each time. Ideally, such a policy would optimize the Quality of Service (QoS) experienced by application-layer clients. This is difficult, however, because routers are generally not informed of which packets belong to which clients. As in Antoniadis et al. [3], we therefore optimize the QoS of the packets themselves by minimizing their flow time.

More specifically, we study online packet forwarding on a line network. Packets are released in an online manner, each with an origin and a destination router. Time is discrete. In each time step, any number of packets may be released at any number of routers, and each router may forward at most one packet to its neighbor on the right. The buffer at each router has unbounded capacity. Routers do not know the states of other routers. The objective is to minimize the maximum flow time, where the flow time of a packet is the time between its release and its arrival at its destination.

We measure algorithms by their competitive ratios. Let Opt be an optimal offline algorithm for packet routing on the line. For any algorithm Alg, we let $\textsc{Alg}(I)$ denote the maximum flow time achieved by Alg on the input sequence I. For any $c \geq 1$, an online algorithm Alg is said to be c-competitive if there exists a constant b such that, for any input sequence I, $\textsc{Alg}(I) \leq c \cdot \textsc{Opt}(I) + b$. The (asymptotic) competitive ratio of Alg is the infimum over all values c for which Alg is c-competitive.

This work is motivated by the results by Antoniadis et al. [3], who defined this appealingly clean but "surprisingly challenging" (a predicate we tend to agree with) problem and also focused on the line network. Antoniadis et al. showed that a natural algorithm, Earliest Arrival, is not $O(1)$-competitive, and claimed the same for three other natural *classes of* algorithms, seemingly leaving no natural algorithm as a candidate for being $O(1)$-competitive. They posed the existence of such an algorithm as an open problem and revealed that there was a modest wager on its resolution. In the first progress on this problem since their paper appeared more than a decade ago, we study the natural Greedy algorithm. Although we do not settle the wager, our results advance the understanding of this problem and identify Greedy as a natural candidate for being $O(1)$-competitive.

Specifically, we develop a good understanding of the case in which all packets have a *length* of at most two, where the length of a packet is the number of routers that need to forward it. As detailed below, core difficulties already arise for such packets. We therefore hope that our work helps pave the way towards solving the general problem.

Contributions

Minimizing the maximum flow time on the line seems to entail a fundamental trade-off between the following two proxy objectives:

- Give highest priority to packets with the earliest release times. This is the Earliest Arrival algorithm, which is not $O(1)$-competitive [3], essentially because it ignores how far the packets still have to travel.
- Prioritize packets by how far they still have to travel (their remaining *length*). This is the Furthest-To-Go algorithm, which Antoniadis et al. claim is not $O(1)$-competitive [3]. Indeed, a length-1 packet can be repeatedly delayed by a continuous stream of length-2 packets, even though forwarding it first would yield a maximum flow time of 3. The algorithm fails because it ignores when packets were released, namely the first proxy objective.

We formalize that there *is* a non-trivial trade-off by providing the first general lower bound on the achievable competitive ratio. In Section 4, we show that even randomized algorithms cannot be better than 4/3-competitive. Our construction repeatedly forces the algorithm to choose between the two proxy objectives. Remarkably, the lower bound requires only packets of length 1 or 2.

We propose an algorithm that we term Greedy, which resolves the trade-off in a simple way: it prioritizes packets according to the sum of the two proxy objectives, namely how long they have been in the system plus the distance they still have to travel. (We give a

different, equivalent formulation later.) In yet other words, this algorithm prioritizes the packets by the flow time they are going to get under the optimistic assumption that they are not delayed any further – therefore the name. So Greedy does not act by a "proxy objective" but by the *real* objective. Viewed this way, the reader may agree that this is the most natural policy to consider.

Our main result is an analysis of Greedy in the case mentioned above – packets of length 1 or 2. Although this may seem restrictive, the fundamental trade-off already arises in this setting. We prove that Greedy is $(2 - \frac{1}{2^{k-1}})$-competitive, where k is the number of active routers in the line network, and establish a matching lower bound for Greedy. These results are presented in Section 3, where we begin with the lower bound to build intuition.

We conjecture that the competitive ratio of Greedy is constant, possibly even 2. Indeed, we do not see how longer packets could be used to raise the lower bound on Greedy further. If our conjecture is true, we believe that our techniques will be useful in proving it.

Related work

Most previous work on routing algorithms under the adversarial model has focused on stability (whether the number of packets in the system remains bounded over time) and throughput maximization. We refer to [2, 7, 10, 18, 20, 21] for representative papers on stability and to [4, 6, 11, 15, 17, 19, 23] for work on throughput maximization.

In the aforementioned work [3], Antoniadis et al. show that Earliest Arrival and Furthest-To-Go are scalable for the maximum-flow-time objective, i.e., $O(1)$-competitive with arbitrarily small speed augmentation. They also show that no $O(1)$-competitive algorithm exists for average flow time. Havill [14] considered online packet routing on bidirectional paths and rings with the objective of minimizing the makespan. They showed that Longest In System and Moving Priority have competitive ratio 2 on linear arrays and also gave a 2-competitive algorithm for bidirectional ring routing. Im and Moseley [16] considered speed scaling for job scheduling on machines connected by a tree, with the objective of minimizing total flow time, and gave constant-competitive algorithms with constant speed augmentation. Disser et al. [12] give approximation algorithms for bidirectional scheduling on a path to minimize total completion time and show that this problem is NP-hard. Erlebach and Jansen [13] study the problem of establishing and completing a given set of calls as early as possible for bidirectional and directed calls in various classes of networks. Adamy et al. [1] studied call control in rings. Given a ring network with edge capacities and a set of paths, the goal is to find a maximum-cardinality subset of the paths such that no edge capacity is violated. They give a polynomial-time algorithm to solve the problem optimally.

The problem we consider can be viewed as a special case of job shop scheduling with unit processing times [5], in which each job must be processed on a contiguous set of machines in left-to-right order. Soukhal et al. [22] considered flow shop scheduling on two machines. Wei and Yuan [24] considered two-machine flow shop scheduling with equal processing times. See also Chen [9].

2 Preliminaries

We let k denote the number of active routers in the line network, i.e., routers with a neighbor to their right that they can forward packets to. Thus, the last router on the line, which can only receive packets, is ignored. Similarly, if router i is the last router that needs to forward a packet p, we call router i the *last router* of p, even though the *destination* of p is router $i + 1$. When a router forwards a packet, we also say that it *processes* the packet.

The release time of a packet p is denoted $r(p)$. The *length* $\ell(p)$ of p is the number of routers that need to process p, so $\ell(p) \in \{1, 2\}$. At any given time $t \geq r(p)$, p's remaining length (the number of routers that still need to process p) is denoted $\ell(p, t)$. The *completion time*, $C(p)$, of a packet p is the first time t such that $\ell(p, t) = 0$, and its *flow time* is $C(p) - r(p)$.

We define the *delay* of packet p at time t by

$$d(p, t) = t - r(p) - (\ell(p) - \ell(p, t))$$

and the *priority* of packet p at time t by

$$\pi(p, t) = \ell(p) + d(p, t) = t - r(p) + \ell(p, t).$$

The delay of a packet is thus the number of time steps it has spent in the system minus the number of time steps in which it has been processed so far. Consequently, once a packet's priority equals its eventual flow time, it is processed in every subsequent time step until completion.

Without loss of generality, we assume that Opt is *zealous* on each router, meaning that whenever at least one packet is waiting to be processed on a router, Opt processes a packet.

3 The Algorithm Greedy

Since the priority of a packet at any point in time is a lower bound on its flow time, a natural greedy choice is to forward packets with high priority. This is what the greedy algorithm does.

Greedy: In each time step, every router with at least one waiting packet forwards a packet of highest priority. Ties may be broken arbitrarily.

By providing matching upper and lower bounds, we show that the competitive ratio of Greedy is exactly $2 - 1/2^{k-1}$ for k routers and packets of lengths 1 and 2.

3.1 Lower bound for Greedy

We begin with the lower bound on Greedy's competitive ratio, thus building intuition from concrete examples before proving the matching upper bound. With only one router, Greedy is optimal, since both it and Opt are zealous. As a warm-up to the general result (Theorem 2), we first consider $k = 2$ routers, using the same notation as in the proof of Theorem 2.

▶ **Proposition 1.** *The competitive ratio of Greedy is at least* $3/2$*, even with only* $k = 2$ *routers.*

Proof. Let $h \geq 4$. The input sequence I contains the following packets:

- At time 0, h packets are released that need to be processed on router 1; denote this set by A_1.
- At time 2, h packets are released that need to be processed on routers 1 and 2; denote this set by B_1.
- At time $h + 3$, $2h$ packets are released that need to be processed on router 2; denote this set by B_2.

Because the packets in B_1 are released two time steps later than those in A_1, Greedy prioritizes the packets in A_1 on router 1. On router 2, Greedy prioritizes the packets in B_1 over those in B_2. Thus, Greedy processes B_1 on router 1 during time steps h to $2h - 1$ and on router 2 during time steps $h + 1$ to $2h$. Consequently, the packets in B_2 are processed during time steps $2h + 1$ to $4h$, resulting in a maximum flow time of $3h - 2$.

An optimal schedule, in contrast, achieves a maximum flow time of $2h$ by giving priority to the packets in B_1 on router 1, thus processing the packets in A_1 during time steps 0 to 1 and $h+2$ to $2h-1$, the packets in B_1 on router 1 during time steps 2 to $h+1$ and on router 2 during time steps 3 to $h+2$, and the packets in B_2 during time steps $h+3$ to $3h+2$. This way, both the last packet in A_1 and the last packet in B_2 have flow time $2h$.

Hence, $\textsc{Greedy}(I) = \frac{3}{2}\,\textsc{Opt}(I) - 2$, yielding a lower bound of $\frac{3}{2}$ on Greedy's competitive ratio. ◀

We now extend this construction to any $k \geq 2$ routers. The construction for $k = 4$ is shown in Figure 1.

▶ **Theorem 2.** *For k routers, the competitive ratio of Greedy is at least $2 - \frac{1}{2^{k-1}}$.*

Proof. We first define a family of input sequences with parameters k and $h \geq 2$. The parameter h can be arbitrarily large, which is necessary to make the result asymptotic. The sequence I_k^h consists of the blocks of packets $A_1^h, A_2^h, \ldots, A_{k-1}^h$ and $B_1^h, B_2^h, \ldots, B_k^h$ defined below. In the remainder of the proof, we omit the superscript h on the blocks A_i^h and B_i^h.

All packets have length 2, except those in A_1 and B_k, which have length 1. This choice strengthens the lower bound by improving its dependence on the number of routers k. The release times of the packets in blocks A_i and B_i are determined by the total number of packets in $B_1, B_2, \ldots, B_{i-1}$, based on:

$$r_i = \left(\sum_{j=1}^{i-1} |B_j| \right) + \max\{0, i-2\}.$$

All packets in a block are released at the same time. Let $r(X)$ denote the release time for the packets in a block X. For $1 \leq i \leq k-1$:

- Block A_i: $2^{k-1-i} \cdot h$ packets released on router $\max\{1, i-1\}$ at time $r(A_i) = r_i$.
- Block B_i: $(2^{k-1} - 2^{k-1-i})h$ packets released on router i at time $r(B_i) = r_i + 2$.

Block B_k: $2^{k-1} \cdot h$ packets released on router k at time $r(B_k) = r_k + 2$.

To compare the largest flow times achieved by Greedy and Opt, respectively, we use the following identities, valid for $1 \leq i \leq k-1$:

$$|A_i| + |B_i| = |B_k| = 2^{k-1} \cdot h \tag{1}$$

$$\sum_{j=1}^{i} |A_j| = \sum_{j=1}^{i} 2^{k-1-j} \cdot h = \left(2^{k-2} + \cdots + 2^{k-1-i}\right) h = \left(2^{k-1} - 2^{k-1-i}\right) h = |B_i| \tag{2}$$

For $k = 4$ routers, the schedules of Greedy and Opt are depicted in Figure 1. First, we verify that Greedy's schedule is as shown.

Each B-block is released two time steps after the corresponding A-block, and A_i and B_i are released at least $|B_{i-1}|$ time steps after A_{i-1} and B_{i-1}, respectively. Thus, the release times and Greedy's priorities π imply the following relations at every time t and on every router:

- if a packet a from A_i and a packet b from B_i are both waiting to be processed, then $\pi(a,t) > \pi(b,t)$;
- if a packet b from B_i and a packet a' from A_{i+1} are both waiting to be processed, then $\pi(b,t) > \pi(a',t)$.

We show below that the first B_i packet arrives on router $i+1$ before B_{i+1} is released, so that Greedy processes packets in order A_1, B_1, A_2, B_2, …, B_{k-1}, B_k, as depicted in Figure 1. The following calculations determine the completion time and flow time of the last packet in each block for Greedy under this processing order.

Before the first packet in A_i is processed, all blocks A_j and B_j with $j < i$ have already been completed. Moreover, because all packets except A_1 and B_k have length 2, they require an additional time step to move from one router to the next. GREEDY finishes the last packet of A_1 at time $e_{\text{GREEDY}}(A_1) = 2^{k-2}h$, and its flow time is

$$f_{\text{GREEDY}}(A_1) = 2^{k-2}h. \tag{3}$$

For $2 \leq i \leq k-1$, GREEDY finishes the last packet of A_i by time

$$e_{\text{GREEDY}}(A_i) = \sum_{j=1}^{i} |A_j| + \sum_{j=1}^{i-1} |B_j| + i - 1. \tag{4}$$

Subtracting its release time from Equation (4) and using Equation (2) shows that its flow time is

$$f_{\text{GREEDY}}(A_i) = \sum_{j=1}^{i} |A_j| + 1 = \left(2^{k-1} - 2^{k-1-i}\right) h + 1. \tag{5}$$

For $1 \leq i \leq k-1$, GREEDY finishes the last packet of B_i by time

$$e_{\text{GREEDY}}(B_i) = \sum_{j=1}^{i} (|A_j| + |B_j|) + i, \tag{6}$$

because B_i starts on the router where A_i ends. Subtracting its release time from Equation (6) shows that its flow time is

$$f_{\text{GREEDY}}(B_1) = 2^{k-1} \cdot h - 1 \text{ and}$$

$$f_{\text{GREEDY}}(B_i) = \left(\sum_{j=1}^{i} |A_j|\right) + |B_i| = |B_i| + |B_i| = \left(2^k - 2^{k-i}\right) h, \tag{7}$$

using Equation (2).

GREEDY finishes the last packet of B_k by time

$$e_{\text{GREEDY}}(B_k) = \sum_{j=1}^{k-1} (|A_j| + |B_j|) + |B_k| + k - 1, \tag{8}$$

because B_k is processed on only one router. Subtracting its release time from Equation (8) shows that its flow time is

$$f_{\text{GREEDY}}(B_k) = \left(\sum_{j=1}^{k-1} |A_j|\right) + |B_k| - 1 = |B_{k-1}| + |B_k| - 1 = \left(2^k - 1\right) h - 1, \tag{9}$$

by Equation (2).

We now verify that B_{i+1} is released only after GREEDY has completed A_i (and hence also B_{i-1}), ensuring that GREEDY is processing B_i on router $i+1$ when B_{i+1} is released. For $1 \leq i \leq k-1$,

$$\begin{aligned} r(B_{i+1}) &= \left(\sum_{j=1}^{i} |B_j|\right) + i - 1 + 2 = |B_i| + \left(\sum_{j=1}^{i-1} |B_j|\right) + i + 1 \\ &= \left(\sum_{j=1}^{i} |A_i| + \sum_{j=1}^{i-1} |B_j|\right) + i + 1, \text{ by Equation (2)} \\ &= e_{\text{GREEDY}}(A_i) + 2, \text{ by Equation (4).} \end{aligned}$$

Finally, since $r(A_{i+1}) = r(B_{i+1}) - 2$, the calculation above also gives $e_{\text{GREEDY}}(A_i) = r(A_{i+1})$. Thus, for every $i \geq 1$, GREEDY finishes processing A_i exactly when A_{i+1} is released.

We have now verified all assumptions used above, so Equations (4)–(9) are valid, as is the depiction of GREEDY's schedule in Figure 1, with the blocks scheduled one after another in order of their priorities. Comparing the flow times for A_i, B_i, and B_k from Equations (3), (5), (7), and (9), we obtain

$$\text{GREEDY}(I_k^h) = f_{\text{GREEDY}}(B_k) = \left(2^k - 1\right) h - 1. \tag{10}$$

We now analyze OPT's schedule; again the reader can use Figure 1 as a guide. The depicted schedule for OPT is feasible: the first packet of each B-block starts one time step after its release, and the A-blocks have much larger delays relative to their release times. The adversary releases the B-packets earlier than OPT requires in order to increase the flow time of GREEDY: GREEDY's maximum flow time occurs on B_k, whereas OPT's occurs on A_{k-1}.

For $1 \leq i \leq k-1$, OPT finishes the last packet of B_i by time

$$e_{\text{OPT}}(B_i) = \sum_{j=1}^{i} |B_j| + i + 2. \tag{11}$$

Subtracting its release time from Equation (11) gives that its flow time is

$$\begin{aligned} &f_{\text{OPT}}(B_1) = 2^{k-2}h + 1 \text{ and} \\ &f_{\text{OPT}}(B_i) = |B_i| + 2 = \left(2^{k-1} - 2^{k-1-i}\right) h + 2. \end{aligned} \tag{12}$$

Block B_k starts when B_{k-1} finishes, at time $\sum_{j=1}^{k-1} |B_j| + (k-1) + 2 = r(B_k) + 1$, by Equation (11). It finishes after $|B_k|$ further time steps, so its flow time is

$$f_{\text{OPT}}(B_k) = |B_k| + 1 = 2^{k-1}h + 1. \tag{13}$$

We next consider the blocks A_i. OPT finishes the last packet of A_1 at time $|A_1| + |B_1| = 2^{k-1}h$. Thus, since $r(A_1) = 0$, $e_{\text{OPT}}(A_1) = f_{\text{OPT}}(A_1) = 2^{k-1}h$. For $2 \leq i \leq k-1$, OPT finishes the last packet of A_i by time

$$\begin{aligned} e_{\text{OPT}}(A_i) &= |A_i| + e_{\text{OPT}}(B_i) - 1, \text{ since } A_i \text{ ends on the router where } B_i \text{ starts} \\ &= |A_i| + \left(\sum_{j=1}^{i} |B_j| + i + 2 \right) - 1, \text{ by Equation (11).} \end{aligned} \tag{14}$$

Subtracting its release time from Equation (14) gives that its flow time is

$$\begin{aligned} f_{\text{OPT}}(A_i) &= |A_i| + \sum_{j=1}^{i} |B_j| + i + 1 - r(A_i) \\ &= |A_i| + |B_i| + 3 \\ &= 2^{k-1} \cdot h + 3, \text{ by Equation (1).} \end{aligned} \tag{15}$$

This flow time is obtained for every $i \geq 2$, and in particular for A_{k-1}.

Comparing Equations (12), (13), and (15), we see that OPT's flow time for A_{k-1} is slightly larger than its flow time for B_k. Therefore,

$$\text{OPT}(I_k^h) = f_{\text{OPT}}(A_{k-1}) = 2^{k-1} \cdot h + 3. \tag{16}$$

Solving Equation (16) for h gives

$$h = \frac{\textsc{Opt}(I_k^h) - 3}{2^{k-1}}.$$

Then,

$$\begin{aligned}\textsc{Greedy}(I_k^h) &= \left(2^k - 1\right) h - 1, \text{ by Equation (10)} \\ &= \left(2^k - 1\right) \frac{\textsc{Opt}(I_k^h) - 3}{2^{k-1}} - 1 \\ &= \left(2 - \frac{1}{2^{k-1}}\right) \textsc{Opt}(I_k^h) - 3\left(2 - \frac{1}{2^{k-1}}\right) - 1.\end{aligned}$$

Since $\textsc{Opt}(I_k^h)$ grows without bound as h increases, the competitive ratio of Greedy cannot be smaller than $2 - \frac{1}{2^{k-1}}$. ◀

Note that in the proof of Theorem 2, the cost of Opt is never greater than that of Greedy on any prefix of I_k^h, so Opt's schedule is an online-bounded optimal [8] solution for Greedy. Thus, Greedy has an online-bounded ratio of at least $2 - \frac{1}{2^{k-1}}$. In online-bounded analysis, the offline optimal algorithm that an online algorithm competes against is constrained to never perform worse than the online algorithm on any prefix, and therefore cannot fully exploit its knowledge of when the input ends.

3.2 Upper bound for Greedy

We now turn to the main result of the paper, the upper bound for Greedy, beginning with some definitions. A packet p is *alive* for a given algorithm at any time t such that $t \geq r(p)$ and $\ell(p,t) > 0$.

Fix an optimal offline algorithm, Opt.

▶ **Definition 3.** *Let $a_i(t)$ be the number of alive packets in Opt's schedule that still need to be processed on router i at time t. Similarly, let $g_i(t)$ be the number of alive packets that Greedy still needs to process on router i at time t.*

The optimal flow time is bounded below by the maximum value of $a_i(t)$ over all time steps and routers.

▶ **Definition 4.** *We define $\Delta_i(t) = g_i(t) - a_i(t)$ for all routers i and times t.*

Thus, $\Delta_i(t)$ is the number of packets by which Opt is ahead on router i at time t: it is the number of packets that Opt has processed on router i before time t minus the number that Greedy has processed there before time t. The value $\Delta_i(t)$ may be positive or negative.

▶ **Lemma 5.** *For each time $t \geq 0$, we have $\Delta_1(t) = 0$.*

Proof. Both Opt and Greedy are zealous, and packets arrive at router 1 at the same times for both algorithms. Therefore, the number of alive packets on router 1 is always the same, and $\Delta_1(t) = 0$. ◀

We wish to bound $\Delta_i(t)$, the number of packets by which Opt is ahead on router i at time t. This is the purpose of Lemmas 6–7. The basic idea of the proof is the following: Each unit by which Δ_i exceeds Δ_{i-1} corresponds to a packet whose last router is $i-1$ and which Greedy has processed there but Opt has not. The reason Opt did not process it is that it was processing another packet released on router $i-1$ that contributes to its lead

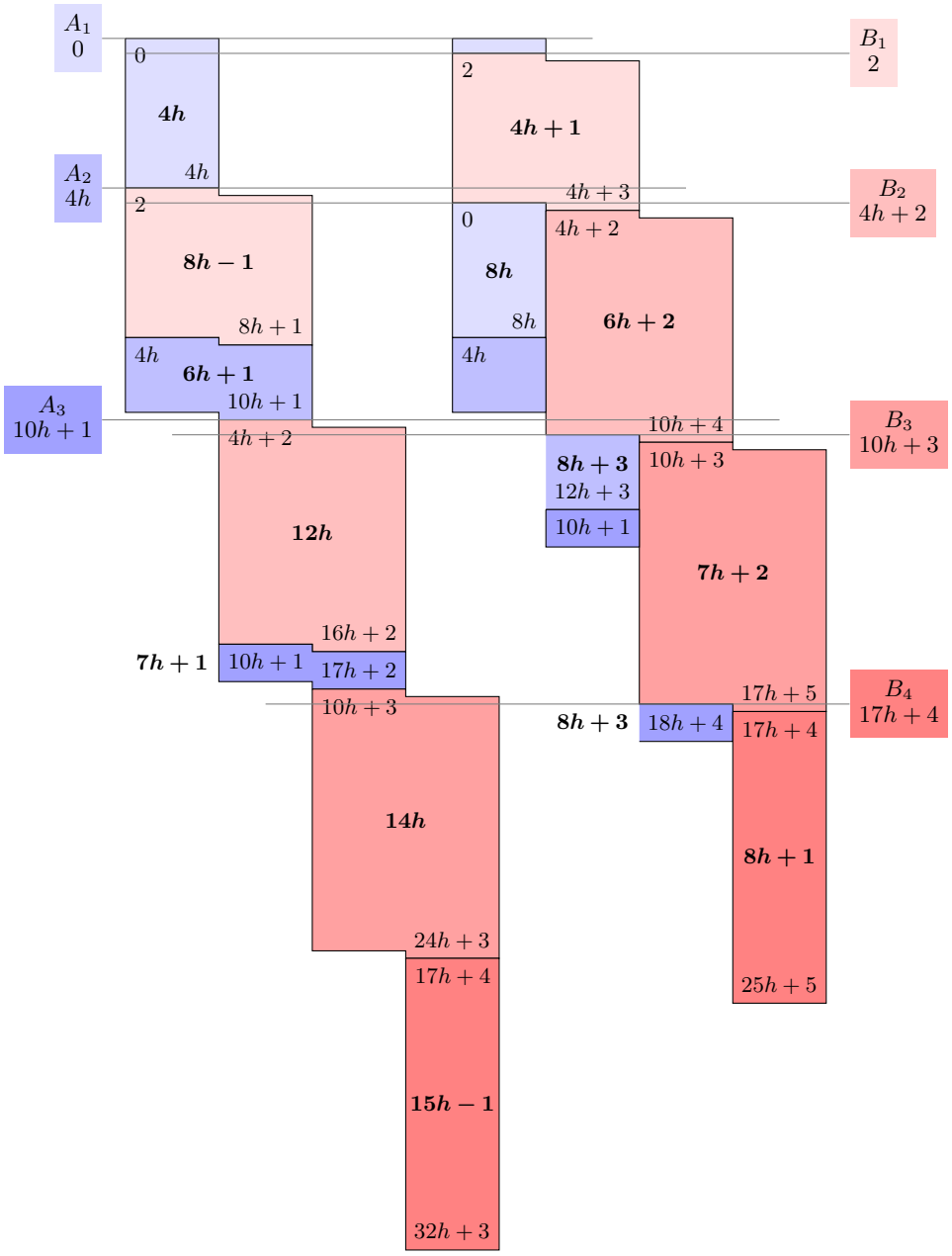

Figure 1 Example of the lower-bound construction for $k = 4$ routers. The small boxes on the left and right give the release times of the blocks; the choice of side is only for readability and has no algorithmic significance. The horizontal guide lines indicate the corresponding release times. In the center, GREEDY's schedule is shown on the left and OPT's on the right. A block that is open to the left indicates that its processing started on the previous router. Within each block, the release time appears at the upper left, the completion time at the lower right, and the flow time in the center.

on router i. If there are ℓ such packets, then OPT still has to process ℓ packets on router $i-1$. Each of these packets must have priority at least that of the packets OPT processed, because GREEDY preferred them. Processing them takes OPT ℓ additional steps, giving a lower bound on its cost.

▶ **Lemma 6.** *If $\Delta_i(t+1) = \Delta_i(t) + 1 > \Delta_{i-1}(t) \geq 0$, then, at time t, there are $\Delta_i(t+1) - \Delta_{i-1}(t)$ packets with current priority at least $\Delta_i(t+1)+1$ that OPT still needs to process on router $i-1$.*

Proof. The equality $\Delta_i(t+1) = \Delta_i(t)+1$ implies that GREEDY is idle on router i at time t, so GREEDY has processed all packets released on router i. Thus, by time t, OPT has processed at least $\Delta_i(t+1)$ length-2 packets that were released on router $i-1$ and that GREEDY has not yet processed there. Let P be this set of packets, and let p be a packet of minimum release time in P. Note that

$$t - r(p) \geq |P| \geq \Delta_i(t+1),$$

since OPT processes P on router $i-1$ between $r(p)$ and t.

Now consider the sets of packets that GREEDY and OPT process on router $i-1$ before time t. Call these sets G and O, respectively, and note that

$$|O| = |G| + \Delta_{i-1}(t). \tag{17}$$

Moreover, since $P \subseteq O \setminus G$,

$$|O \setminus G| \geq |P| \geq \Delta_i(t+1), \tag{18}$$

so

$$\begin{aligned} |G \setminus O| &= |O \setminus G| - \Delta_{i-1}(t), \text{ by (17) and removing } O \cap G \text{ from both } O \text{ and } G \\ &\geq \Delta_i(t+1) - \Delta_{i-1}(t), \text{ by (18).} \end{aligned}$$

Since all packets in P are released on router $i-1$, they become available to GREEDY there immediately upon release. Therefore, by the definition of GREEDY and the fact that p has length 2, each packet in $G \setminus O$ is at least as old as p and of length 2 or is at least one time step older than p. It follows that, at time t, OPT has at least $\Delta_i(t+1) - \Delta_{i-1}(t)$ packets, each with priority at least $\Delta_i(t+1)+1$, that still need to be processed on router $i-1$. ◀

For an illustration of Lemma 6, see the lower-bound construction depicted in Figure 1. This input sequence uses the blocks defined at the beginning of the proof of Theorem 2. Letting $i = 4$, we consider

$$t = 17h + 2,$$

since $17h+2$ is the last time before completion at which GREEDY is idle on router 4. Since OPT processes the last two packets of B_3 at times $t = 17h+2$ and $t+1 = 17h+3$ on router 3, the set P consists of the first $7h-2$ packets in B_3, and

$$\Delta_4(t+1) = \Delta_4(17h+3) = 7h-2.$$

The oldest packet p in P was released at time

$$r(p) = 10h + 3$$

(all packets in P have this release time). By this time, Opt has completed all but the last packet of B_2, while Greedy has completed only the first packet of B_2, so Opt is ahead of Greedy by

$$\Delta_3(r(p)) = 6h - 2$$

packets from B_2 on router 3.

At time $r(p)$ on router 3, Opt has not yet begun processing any of the h packets in A_3 on router 3. At time $t = 17h + 2$, each has priority

$$\pi(A_3, t) = 2 + (17h + 2) - (10h + 1) - (2 - 1) = 7h + 2.$$

These are the packets in $G \setminus O$ from Lemma 6. According to the lemma, the last packet in A_3 that Opt processes on router 3 has priority (and flow time) at least $\Delta_4(t + 1) = 7h - 2$, and there are at least

$$|A_3| = \Delta_4(t + 1) - \Delta_3(t) = (7h - 2) - (6h - 2) = h$$

packets in A_3. In this example, the number of packets remaining for Opt on router 3 matches the lower bound in the lemma, and the priority $7h + 2$ exceeds the guaranteed value $\Delta_4(t+1)+1 = 7h-1$. As noted below in the proof of Lemma 7, at least one of these packets has flow time at least

$$2\Delta_4(t + 1) - \Delta_3(t) = 2(7h - 2) - (6h - 2) = 8h - 2$$

in Opt's schedule (its actual flow time is $8h + 3$).

▶ **Lemma 7.** *For any router $i \geq 1$,*

$$\max_{t \geq 0, j \leq i} \Delta_j(t) \leq \left(1 - \frac{1}{2^{i-1}}\right) \mathrm{Opt}.$$

Proof. We prove the lemma by induction on i. For $i = 1$, the claim follows from Lemma 5, establishing the base case.

For the inductive step, fix a router $j \geq 2$ and let $t + 1$ be the first time at which the maximum value of Δ_j is attained. By Lemma 6, at time t there are $\Delta_j(t + 1) - \Delta_{j-1}(t) > 0$ packets that Opt still needs to process on router $j - 1$ and whose priority is already at least $\Delta_j(t + 1) + 1$. The first of these packets may have flow time $\Delta_j(t + 1) + 1$, but the last has flow time at least $2\Delta_j(t + 1) - \Delta_{j-1}(t)$. Thus,

$$\begin{aligned}
\mathrm{Opt} &\geq 2\Delta_j(t + 1) - \Delta_{j-1}(t) \\
&\geq 2\Delta_j(t + 1) - (1 - 2^{2-j})\,\mathrm{Opt}, \text{ by the induction hypothesis} \\
&\geq 2\Delta_j(t + 1) - (1 - 2^{2-i})\,\mathrm{Opt}, \text{ since } j \leq i.
\end{aligned}$$

Hence,

$$(1 - 2^{1-i})\,\mathrm{Opt} \geq \Delta_j(t + 1).$$

This completes the induction. ◀

▶ **Theorem 8.** *If there are k routers and all packets have length 1 or 2, Greedy is $\left(2 - \frac{1}{2^{k-1}}\right)$-competitive.*

Proof. Fix any router $i \geq 2$ and any packet p that is released at time $r(p)$ on router $i-1$ or i and has router i as its last router. The priority of every packet that remains alive in Greedy's schedule increases by 1 in each time step in which it is not processed. Therefore, no packet released at time $t' = r(p) + 2$ or later, on any router, can delay p, because p has higher priority.

At time t', there are $g_{i-1}(t')$ packets alive (on router $i-2$ or $i-1$) that will eventually need to be processed on router $i-1$, and

$$\begin{aligned} g_{i-1}(t') &= a_{i-1}(t') + \Delta_{i-1}(t') \\ &\leq \textsc{Opt} + \left(1 - \frac{1}{2^{i-2}}\right) \textsc{Opt}, \text{ by Lemma 7} \\ &= \left(2 - \frac{1}{2^{i-2}}\right) \textsc{Opt}. \end{aligned}$$

Likewise, at time t', there are

$$g_i(t') = a_i(t') + \Delta_i(t') \leq \left(2 - \frac{1}{2^{i-1}}\right) \textsc{Opt}$$

packets alive that need to be processed on router i.

Thus, if p is released at router i, its completion time is no later than $t' + g_i(t')$. If p is released at router $i-1$, it reaches router i no later than at time $t' + g_{i-1}(t')$. Because no packet released at time t' or later can delay p, it cannot be delayed on router i beyond time $t' + g_i(t')$. Therefore, the completion time of p is at most $t' + \max\{g_{i-1}(t') + 1, g_i(t')\}$.

Since $t' = r(p) + 2$, we conclude that p is completed no later than at time

$$\begin{aligned} r(p) + 2 + \max\{g_{i-1}(t') + 1, g_i(t')\} &< r(p) + 2 + \left(2 - \frac{1}{2^{i-1}}\right) \textsc{Opt} + 1 \\ &\leq r(p) + \left(2 - \frac{1}{2^{k-1}}\right) \textsc{Opt} + 3, \text{ since } i \leq k. \end{aligned}$$

Hence, the flow time of p is at most $\left(2 - \frac{1}{2^{k-1}}\right) \textsc{Opt} + 3$. ◀

Since this matches the lower bound of Theorem 2, this proves that the competitive ratio of Greedy is exactly $\left(2 - \frac{1}{2^{k-1}}\right)$ on inputs with only k active routers. Since one is dealing with a restricted Opt when using the online-bounded ratio, $\left(2 - \frac{1}{2^{k-1}}\right)$ is also the online-bounded ratio.

4 General lower bounds

We start with a warm-up construction before proving the 4/3 lower bound for randomized algorithms.

▶ **Theorem 9.** *Any randomized algorithm has a competitive ratio of at least* 6/5 *even with only* $k = 2$ *routers.*

Proof. We give a family of input sequences $\{I_h\}_{h \in \mathbb{N}}$. Each sequence I_h begins with the following packets.

- At time 0, $2h$ packets are released that need to be processed on router 1. We call these *short packets*.
- At time h, h packets are released that need to be processed on routers 1 and 2. We call these *long packets*.

Let $y \in [0, h]$ be the expected number of long packets that ALG processes on router 1 before time $2h$. The expected maximum flow time among the short packets must be at least $2h + y$. If no further packets are released, the optimal maximum flow time is $2h + 1$, achieved by giving priority to the short packets at router 1. Thus, since ALG is 6/5-competitive, there must be a constant a such that

$$2h + y \leq \frac{6}{5} \cdot (2h + 1) + a,$$

or equivalently, by solving for y,

$$y \leq \frac{2h}{5} + b, \text{ where } b = a + \frac{6}{5}.$$

Thus, the expected number of packets that remain for ALG to process on router 2 at time $2h + 1$ is at least

$$h - y \geq \frac{3h}{5} - b.$$

If now $3h$ additional *jam packets* are released at time $2h + 1$ that need to be processed by router 2, the expected number of packets that ALG still needs to process on router 2 is at least

$$\frac{3h}{5} - b + 3h = \frac{18h}{5} - b$$

so the expected maximum flow time in ALG's schedule is at least this number.

The optimal maximum flow time for the entire instance is $3h$, achieved by giving priority to the long packets at router 1 and thereby avoiding any conflict between the long packets and the jam packets. Hence, $\text{ALG}(I_h) \geq \frac{6}{5} \text{OPT}(I_h) - b$. For h large enough, this contradicts that ALG is $(6/5 - \varepsilon)$-competitive. ◀

Next, we prove a lower bound of 4/3 for randomized algorithms, using the same idea as in the proof of Theorem 9, using "short" and "long" packets iteratively to build up how much behind OPT the randomized algorithm, ALG, is on a specific router. The proof also reuses the idea of restricting ALG to maintaining the goal competitive ratio during each iteration (since it is online), but allowing OPT to perform worse than ALG until the end where there are many "jam" packets. Unfortunately, this means that the lower bound we obtain for the competitive ratio does not immediately imply the same result for the online-bounded ratio, even though the lower bound of Theorem 2 for GREEDY immediately implies a lower bound for GREEDY's online-bounded ratio.

We call an instance I (t, i, U, L)-*critical* if there exists an offline algorithm OFF satisfying the following conditions:

- **(i)** The cost of OFF on I is at most U.
- **(ii)** At time t, OFF has no packets left to process on routers $i, i + 1, \ldots, k$.
- **(iii)** At time t, for every 4/3-competitive randomized algorithm ALG, the expected number of packets that ALG has left to process on router i is at least L.

First, we show the following lemma.

▶ **Lemma 10.** *For any 4/3-competitive algorithm ALG, there exists a constant b such that, for any nonnegative L and any positive even U, the existence of a (t, i, U, L)-critical instance, I, implies the existence of a $(t', i+1, 3U/2, L+U/6-b)$-critical instance, I', for some $t' \in \mathbb{N}$.*

Proof. We extend I to I' by releasing the following additional packets:

- At time t, U *short packets* that need to be processed at router i.
- At time $t + U/2$, $U/2$ *long packets* that need to be processed at routers i and $i + 1$.

Let P consist of the L packets waiting on router i at time t and the U packets released there at time t. Let Alg be a 4/3-competitive algorithm. Let $y \in [0, U/2]$ be the expected number of long packets released at time $t + U/2$ that Alg processes on router i before time $t + U$. Then the expected time by which Alg has processed all packets in P is at least $t + L + U + y$. Hence, the expected maximum flow time among the $L + U$ packets in P is at least $L + U + y$.

Based on Off, we can define another offline algorithm Off$'$ with Off$'(I') \leq U{+}1$ by processing all short packets before the long packets. Because Alg is 4/3-competitive and Opt performs at least as well as Off$'$, there must exist a constant a, independent of U and L, such that

$$L + U + y \leq \frac{4}{3}(U + 1) + a.$$

Equivalently,

$$y \leq \frac{U}{3} - L + b, \text{ where } b = a + \frac{4}{3}.$$

Thus, the expected number of packets that Alg still has to process on router $i' = i + 1$ at time $t' = t + U{+}1$ is at least

$$\frac{U}{2} - y \geq \frac{U}{6} + L - b.$$

Finally, from Off we construct another offline algorithm Off$''$ by processing the long packets as soon as they become available and before the remaining short packets. The schedule of Off$''$ has cost at most $(3/2) \cdot U$ and has no packets left to process on routers $i', i' + 1, \ldots, k$ from time t'. ◀

We now prove the theorem by iteratively applying this lemma.

▶ **Theorem 11.** *Any randomized algorithm has a competitive ratio of at least* 4/3.

Proof. Let $\varepsilon > 0$. Suppose Alg is $(4/3 - \varepsilon)$-competitive. We start with the empty instance $I^{(0)}$, which is $(0, 0, q, 0)$-critical for any positive integer q. Let i and ℓ be positive integers, where ℓ depends on ε. We set $q = 2^i \cdot \ell$, allowing us to apply Lemma 10 iteratively i times to $I^{(0)}$ (the lemma requires the third entry of the 4-tuple to be a multiple of 2).

This yields an instance $I^{(i)}$ that is (t, i, U_i, L_i)-critical for some $t \in \mathbb{N}$, where

$$U_i = \left(\frac{3}{2}\right)^i \cdot q,$$

and, for some constant b,

$$L_i = \sum_{j=0}^{i-1} \left(\frac{1}{6} \cdot U_j - b\right) \geq \left(\sum_{j=0}^{i-1} \left(\frac{2}{3}\right)^{i-j} \cdot \frac{1}{6}\right) \cdot U_i - bi = \left(\sum_{j=1}^{i} \left(\frac{2}{3}\right)^{j} \cdot \frac{1}{6}\right) \cdot U_i - bi$$
$$= \left(\frac{1}{3} - \frac{2^i}{3^{i+1}}\right) U_i - bi.$$

By choosing i and ℓ sufficiently large, we obtain

$$L_i > \left(\frac{1}{3} - \frac{\varepsilon}{4}\right) U_i - bi \geq \left(\frac{1}{3} - \frac{\varepsilon}{2}\right) U_i.$$

Thus, by Lemma 10, ALG, being a 4/3-competitive algorithm, has in expectation more than $(1/3 - \varepsilon/2) \cdot U_i$ packets left to process on router i at time t. Releasing U_i additional packets at time t that need to be processed on router i therefore gives ALG an expected cost greater than $(4/3 - \varepsilon/2) \cdot U_i$. In contrast, the definition of critical guarantees that OFF, and hence OPT, can schedule $I^{(i)}$ together with these additional packets with cost U_i, because no packet remains to be processed on router i at time t. Since ℓ, and hence U_i, can be chosen arbitrarily large, this contradicts the assumption that ALG is $(4/3 - \varepsilon)$-competitive. ◀

5 Open Problem

We conjecture that GREEDY has a constant competitive ratio, possibly 2, for online packet routing even when packet lengths are unbounded. A difficulty in extending Lemma 6 to packets of length greater than two is that the packets delayed by OPT, namely those in $G \setminus O$, need not all be located on one router. In the worst case, they could be distributed evenly across the preceding routers, for example. The maximum packet length may therefore enter the competitive analysis, perhaps only through an additive constant.

References

1 Udo Adamy, Christoph Ambühl, R. Sai Anand, and Thomas Erlebach. Call control in rings. *Algorithmica*, 47(3):217–238, 2007. doi:10.1007/s00453-006-0187-4.

2 W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Adaptive packet routing for bursty adversarial traffic. *Journal of Computer and System Sciences*, 60(3):482–509, 2000. doi:10.1006/jcss.1999.1681.

3 Antonios Antoniadis, Neal Barcelo, Daniel Cole, Kyle Fox, Benjamin Moseley, Michael Nugent, and Kirk Pruhs. Packet forwarding algorithms in a line network. In *11th Latin American Theoretical Informatics Symposium (LATIN)*, volume 8392 of *Lecture Notes in Computer Science*, pages 610–621. Springer, 2014. doi:10.1007/978-3-642-54423-1_53.

4 Baruch Awerbuch, Yossi Azar, and Serge A. Plotkin. Throughput-competitive on-line routing. In *34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 32–40. IEEE, 1993. doi:10.1109/SFCS.1993.366884.

5 Nikhil Bansal, Tracy Kimbrel, and Maxim Sviridenko. Job shop scheduling with unit processing times. *Mathematics of Operations Research*, 31(2):381–389, 2006. doi:10.1287/moor.1060.0189.

6 Martin Böhm, Nicole Megow, and Jens Schlöter. Throughput scheduling with equal additive laxity. In *12th International Conference on Algorithms and Complexity (CIAC)*, volume 12701 of *Lecture Notes in Computer Science*, pages 130–143. Springer, 2021. doi:10.1007/978-3-030-75242-2_9.

7 Allan Borodin, Jon M. Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. Adversarial queuing theory. *Journal of the ACM*, 48(1):13–38, 2001. doi:10.1145/363647.363659.

8 Joan Boyar, Leah Epstein, Lene M. Favrholdt, Kim S. Larsen, and Asaf Levin. Online-bounded analysis. *Journal of Scheduling*, 21(4):328–441, 2018. doi:10.1007/s10951-017-0536-y.

9 Bo Chen. Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *Journal of the Operational Research Society*, 46(2):234–244, 1995. doi:10.1057/jors.1995.28.

10 Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. Adversarial queuing on the multiple access channel. *ACM Transactions on Algorithms*, 8(1):5, 2012. doi:10.1145/2071379.2071384.

11 Rathish Das and Hao Sun. Approximation hardness of resource scheduling. In *37th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 46–61. ACM, 2025. doi:10.1145/3694906.3743340.

12 Yann Disser, Max Klimm, and Elisabeth Lübbecke. Scheduling bidirectional traffic on a path. In *42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 9134 of *Lecture Notes in Computer Science*, pages 406–418. Springer, 2015. doi:10.1007/978-3-662-47672-7_33.

13 Thomas Erlebach and Klaus Jansen. Call scheduling in trees, rings and meshes. In *30th Annual Hawaii International Conference on System Sciences (HICSS)*, volume 1, page 221. IEEE, 1997. doi:10.1109/HICSS.1997.667220.

14 Jessen T. Havill. Online packet routing on linear arrays and rings. In *28th International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of *Lecture Notes in Computer Science*, pages 773–784. Springer, 2001. doi:10.1007/3-540-48224-5_63.

15 Dylan Hyatt-Denesik, Mirmahdi Rahgoshay, and Mohammad R Salavatipour. Approximations for throughput maximization. *Algorithmica*, 86(5):1545–1577, 2024. doi:10.1007/s00453-023-01201-4.

16 Sungjin Im and Benjamin Moseley. Scheduling in bandwidth constrained tree networks. In Guy E. Blelloch and Kunal Agrawal, editors, *27th ACM on Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 171–180. ACM, 2015. doi:10.1145/2755573.2755576.

17 Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004. doi:10.1137/S0097539701399666.

18 F. T. Leighton, Bruce M. Maggs, and Satish B. Rao. Packet routing and job-shop scheduling in O(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994. doi:10.1007/BF01215349.

19 Qingsong Liu and Zhixuan Fang. Online task scheduling and termination with throughput constraint. *IEEE/ACM Transactions on Networking*, 32(6):4629–4643, 2024. doi:10.1109/TNET.2024.3425617.

20 Rafail Ostrovsky and Yuval Rabani. Universal O(congestion + dilation + $\log^{1+\epsilon} N$) local control packet switching algorithms. In *29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 644–653. ACM, 1997. doi:10.1145/258533.258659.

21 Boaz Patt-Shamir and Will Rosenbaum. Space-optimal packet routing on trees. In *IEEE Conference on Computer Communications (INFOCOM)*, pages 1036–1044. IEEE, 2019. doi:10.1109/INFOCOM.2019.8737596.

22 Ameur Soukhal, Ammar Oulamara, and Patrick Martineau. Complexity of flow shop scheduling problems with transportation constraints. *European Journal of Operational Research*, 161(1):32–41, 2005. doi:10.1016/j.ejor.2003.03.002.

23 Pavel Veselý, Marek Chrobak, Łukasz Jeż, and Jiří Sgall. A ϕ-competitive algorithm for scheduling packets with deadlines. *SIAM Journal on Computing*, 51(5):1626–1691, 2022. doi:10.1137/21M1469753.

24 Hongjun Wei and Jinjiang Yuan. Two-machine flow-shop scheduling with equal processing time on the second machine for minimizing total weighted completion time. *Operations Research Letters*, 47(1):41–46, 2019. doi:10.1016/j.orl.2018.12.002.