Scheduling Trains with Small Stretch on a Unidirectional Line

Apoorv Garg[1(✉)] and Abhiram Ranade[2(✉)]

[1] Coupa Software India Pvt. Ltd., Pune 411016, Maharashtra, India
apoorv.garg@gmail.com
[2] Indian Institute of Technology Bombay, Mumbai 400076, Maharashtra, India
ranade@cse.iitb.ac.in

Abstract. We investigate the problem of scheduling trains to minimize *max-stretch*, where the stretch suffered by a train is the ratio of its actual finishing time to its minimum possible finishing time. This metric is presumably more appropriate for train scheduling because it is fairer. Our target network, introduced in [11], is an *in-comb*: a unidirectional railway line with equidistant stations, each initially having at most one train; in addition, there can be at most one train poised to enter each station. A train takes unit time to enter a station or to move from one to the next. Trains must move to their destinations such that at any time there can be at most one train at any station and on the track connecting it to the next. We prove that minimizing max-stretch is NP-hard even on this simple network. We also give an O(1)-approximation algorithm. Our problem can also be interpreted as packet scheduling on in-comb, a special case of in-trees. Packet scheduling on general graphs and some special topologies has been studied earlier with different objective functions, e.g., makespan, flowtime, and max-delay, but there has been little work on max-stretch.

Keywords: Approximation algorithms · Combinatorial optimization · In-comb network · Max-stretch minimization · NP-hard · Packet scheduling · Train scheduling · Unidirectional line

1 Introduction

In the train scheduling problem [2,8,19,21], we are interested in moving a set of trains to their destinations, respecting track capacity and minimizing an appropriate cost metric. A natural expectation is: if trains are to be late, they should be late in proportion to their planned travel times. This can be achieved by minimizing *max-stretch*, where the stretch of a train is the ratio of its actual finishing time to its minimum possible finishing time. Such fairness is not guaranteed by minimizing other metrics such as *max-delay* or *makespan*.

The problem is modeled using a graph: nodes represent stations, links represent tracks. Initially, each station may hold one or more trains—represented as point objects—to be moved to specified stations using specified paths. On each

A. Mudgal and C. R. Subramanian (Eds.): CALDAM 2021, LNCS 12601, pp. 16–31, 2021.
https://doi.org/10.1007/978-3-030-67899-9_2

link there can be at most one train at a time and it takes a specified time for a train to traverse a link. There is also a buffering constraint: each station can hold at most a specified number of trains. Our goal is to schedule the trains such as to minimize their max-stretch, where in any schedule the stretch of a train is defined as the ratio (f/f_m) between its actual finishing time f in that schedule and its minimum finishing time f_m across all possible schedules which, in our simple problem, equals the train's path-length.

This problem is also studied in packet scheduling literature, using the terms *routers*, *channels*, and *packets* in place of stations, tracks, and trains. We do not know of any work on max-stretch; for other metrics, the problem is known to be NP-hard in various versions. With bounded buffering at nodes, minimizing makespan or max-delay—even to a constant factor—is NP-hard even on *leveled directed graphs* in which packets move from the lowest numbered level to the highest numbered level [6]. Since path-lengths of all packets are the same, the hardness result also applies to max-stretch but this is rather degenerate. Assuming unbounded buffering at nodes, minimizing makespan is NP-hard on trees as well as on general graphs [20]. On trees, a 2-approximation can be obtained [20]. However, on unidirectional rings, in-trees, and out-trees, the optimal makespan can be achieved [16]. For a variant of the problem where buffers are available in links rather than at nodes, $O(1)$-approximations, using only $O(1)$ buffers in each link are known on general graphs [14,15,23,26].

There is also a large body of experimental work on train scheduling using various approaches, such as simulation, heuristics, mixed integer linear programming, multi-agent systems, genetic algorithms, reinforcement-learning, etc. [3–5,7,9,12,13,17,18,24,25,27,29], but we do not know of papers which consider stretch. In any case our goal in this paper is to establish provable bounds.

Given the practical importance of the problem, it is worth asking whether good scheduling is possible for simpler networks. We have not found any such results for max-stretch. In this paper we begin such a study by considering an *in-comb* network, a special case of in-trees. The in-comb, defined formally later, is a directed path with an extra, *branch* edge entering every node on the path. There can be a train in each node and one poised to enter it from the branch. Trains may exit the network in any node.

Our motivation for studying this network is two fold. First, for ease of management, a large railway network is often broken into sub-networks, each consisting of a *trunk route* with trains entering and exiting from and to branch lines. Each of the two directions of the route is like an in-comb. Second, the in-comb is perhaps the simplest interesting network, and it would be good to understand the computational complexity of scheduling on it with minimum max-stretch. We further simplify it, assuming identical traversal times for all trains on all links. Minimizing max-delay is known to be NP-hard on this simple network [11].

Main Results

1. Minimizing max-stretch is NP-hard for train scheduling on in-comb (Sect. 4).
2. A polytime algorithm to schedule trains on the in-comb with a max-stretch $O(1)$ times the optimal (Sect. 3).

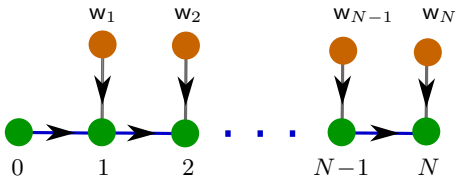

Fig. 1. The in-comb network

2 Preliminaries

2.1 Network Definition and Problem Statement

Our target network is an in-comb (Fig. 1). It consists of:

1. *Line*: sequence of stations, labeled $0, 1, \ldots, N$, and links $(s-1, s)$ $\forall s \in [N]$
2. *Branches*: for each station $s > 0$, an *outer* w_s and a link (w_s, s)

For every node (station or outer), at most one train is given to be there at time 0, along with its destination. No more trains are introduced into the network later. Trains starting at stations are called *internal trains*; those starting at outers *external trains*. Each node can hold at most one train at a time. Any train takes one *step*[1] to enter a station from its outer or to move from a station to the next, and vanishes (exits the network) on reaching its destination. An external train entering the line is called an *entry*, to distinguish it from a *movement* which means a train moving from one station to the next. The required output is a schedule for the trains, minimizing the max-stretch. (Note: A train of path-length ℓ finishing its journey at time f is said to have suffered a stretch f/ℓ).

2.2 The Chain-Hole View

Our arguments to prove the claimed results are based on a *chain-hole view* of schedules [11], summarized next.

Hole refers to a vacancy at a station s. The hole might have been at s since the very beginning (time 0), or *created* later by the exit of a train at s, or it might have come to s from upstream. Some clarifications are needed regarding holes and their progress on the line. First, for convenience we assume an infinite number of suitably numbered artificial stations[2] to the left and right of the line, with no external or internal trains, i.e., each artificial station having a hole. Second, suppose a station u has a hole h at time t. For any $v > u$, if trains at stations $u+1, \ldots, v-1$ remain stationary but the trains at stations $u-1$ and v move, the hole in station u will vanish and a hole will appear in station v.

[1] $\forall t \in \mathbb{Z}^+$, *step* t is the unit time duration $(t-1, t]$ that ends at time t.
[2] Stations $-1, ..., -\infty$ upstream of station 0, and $N+1, ..., \infty$ downstream of N.

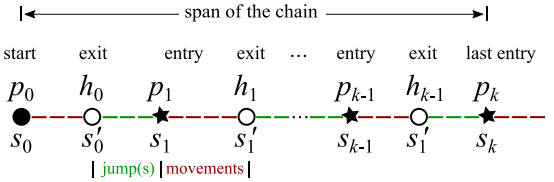

Fig. 2. Spatial view of a chain $\langle p_0, h_0, p_1, h_1, \ldots, p_{k-1}, h_{k-1}, p_k \rangle$. Station s_0 is the origin ($\bullet$) of the internal train p_0 with which the chain begins. Stations $s_1, \ldots, s_k$ are the entry points ($\star$) of external trains $p_1, \ldots, p_k$. $s'_0, s'_1, \ldots, s'_{k-1}$ are the destinations ($\circ$) of non-terminal trains $p_0, p_1, \ldots, p_{k-1}$, where the holes $h_0, h_1, \ldots, h_{k-1}$ get created when those trains exit. Links ($-$) crossed by train-movements of the chain are shown in red while those crossed by hole-jumps are in green. (Color figure online)

We define this as the hole h *jumping* from station u to station v. In this view both holes and trains can move forward, but only a hole can jump across several stations in a step. It is also useful to consider that holes and trains contend for links in order to progress down the line: across any link in any step, either a train can move or a hole can jump but not both.

An external train can enter a station s only by *filling* a hole that might have already been at s, or might jump to s from upstream. When a train p_1 exits the line at a station s'_1, it leaves behind a hole which can be used for the entry of another train p_2 at a station $s_2 > s'_1$; p_2 would exit at some station $s'_2 \geq s_2$, re-creating the hole; and so on. Such a sequence of trains is called a *chain*. Thus, a chain consists of a preexisting hole or an internal train followed by a sequence of some k external trains $p_1 \ldots p_k$; p_1 must fill a hole h_0 which is a preexisting hole or a hole created by the exit of an internal train p_0, and for $j > 1$, p_j must fill the hole h_{j-1} created by the exit of p_{j-1} (See Fig. 2). The chain is said to *begin* with the preexisting hole h_0 (or the train p_0). Clearly, in any schedule, every external train will be placed in some chain.

In order to build a schedule, we must somehow form such chains of trains. After we form chains, we can worry about how to move the trains so that the entries happen as per the chains. This is the structure of our algorithms.

3 An O(1)-Approximation of the Optimal Max-Stretch

The first ingredient of our algorithm is a strategy for scheduling any single class of external trains in an optimal manner, where trains with path-lengths between 2^{i-1} and $2^i - 1$ constitute class i. This is discussed in Sect. 3.1.

The second ingredient is: schedule classes in increasing class order. Since the path-lengths of trains in classes 1 through $i-1$ roughly add up to the path-length of a class i train, the delay caused to the class i trains by lower class trains itself does not substantially affect the stretch of class i trains. However, the movement of lower class trains causes the holes to move. So this makes it harder to apply

the lower bound on delivery time of class i trains derived in Sect. 3.1. We show in Sect. 3.2 that the movement of holes from their initial positions only causes a constant factor increase in the stretch. A second problem is that in the optimal schedule, class i trains may need to use holes created by departure of internal trains of classes up to some j. We show that we can estimate j through a pre-computation. Thus, before scheduling the movement of class i external trains, we deliver all class j internal trains. Again, this may not create the holes we need in the same positions as in the optimal. In Sect. 3.2, we also show that the drift of these holes also does not matter too much.

3.1 Schedule for a Single Class Using only Preexisting Holes

We consider how to minimize the makespan of a class i using only preexisting holes and holes created by the exits of class i external trains. For ease of exposition, rather than say "we do not use the holes created by the exit of any internal trains", we modify all internal trains to have the last station N as their destination. Positions of the preexisting holes remain unchanged. Note that since the path-lengths in a class differ at most by a factor of 2, minimizing makespan instead of max-stretch may worsen the latter at most by a factor of 2.

We show below that good schedules are possible if and only if the initial holes are well distributed among the external trains, and to the extent they are well distributed. We begin with the lower bound: good schedules are not possible if some region with many external trains has very few holes.

Lemma 1. *Suppose all internal trains go to the last station N. Suppose a contiguous sequence S of stations has w external trains of class i and h holes. Then, class i trains have a makespan at least* $\max\left\{2^{i-1}, \min\left\{\frac{w\cdot 2^{i-1}}{2h}, \sqrt{w\cdot 2^{i-1}}\right\}\right\}$.

Proof. Suppose all w external trains of class i enter by time T. Their makespan F is at least $T+d-1$, where $d=2^{i-1}$, and one of the following must be true:

1. At least $\frac{w}{2}$ trains enter in chains beginning with the holes within S: at most h can enter in step 1 (and exit in step d, re-creating those h holes), h more in step $d+1$, and so on. By time T, at most $\left(1+\frac{T-1}{d}\right)h$ trains can enter. But all do enter by time T, i.e., $\left(1+\frac{T-1}{d}\right)h \geq \frac{w}{2} \implies T+d-1 \geq \frac{wd}{2h} \implies F \geq \frac{wd}{2h}$.
2. At least $\frac{w}{2}$ trains enter in chains beginning with the holes upstream of S. Since in each step no more than one hole may jump into S from upstream, in each of the first d steps at most one train can enter. Each can exit d steps later to re-create a hole, so in the 2^{nd} set of d steps, trains can fill these holes and another d holes from upstream, i.e., two entries per step. In n^{th} set, at most n entries per step. Thus, at most $q(q+1)\frac{d}{2}+(q+1)r$ entries by time T, where $q=\lfloor\frac{T}{d}\rfloor$ and $r=T \bmod d$, i.e., $q(q+1)\frac{d}{2}+(q+1)r \geq \frac{w}{2}$ and:
   (a) If $r=0$ then $T=qd$ and $(q+\frac{1}{2})^2d^2 > wd \Rightarrow T+\frac{d}{2} > \sqrt{wd} \Rightarrow F \geq \sqrt{wd}$
   (b) If $r \geq 1$ then $(qd+d)(qd+2r) \geq wd \Rightarrow (qd+r+d-1)^2 \geq wd \Rightarrow F \geq \sqrt{wd}$

Thus, $F \geq \min\left\{\frac{wd}{2h}, \sqrt{wd}\right\}$. But $F \geq d$, the minimum class i path-length.

We next prove that if every region, or *segment* as defined below, has a large number of holes as compared to the number of external trains within it then all trains can be scheduled with small makespan.

Definition 1 (Class i segment of size w). *Any contiguous sequence of stations initially having w external trains of class i (at their outers), such that:*

1. *The first station of the sequence has a class i external train.*
2. *Either the downstream neighbor of its last station has a class i external train, or the sequence includes the downstream artificial stations $N+1, ...\infty$.*

Lemma 2. *Suppose all internal trains go to the last station N, and i, w, and h are given such that $h \geq \min\{w, 2^i\}$ and every class i segment of size w has at least h holes. Then in polytime we can schedule class i trains to finish by time $2h + \frac{w \cdot 2^i}{h} + 2^i$.*

Proof Sketch. Here we only give the main idea (the details are in Appendix A): We partition the network into a sequence of class i segments of size w and in each we form h chains, every chain having at most $w/h + 1$ trains. In the first h steps, the chain-heads enter in parallel in all segments—each filling one of the h or more initial holes of the w-sized segment upstream of it. Afterwards, the chains progress in parallel; conflicts are resolved by prioritizing external trains over internal, and downstream chains over upstream. We can show that all chains finish before time $2h + w \cdot 2^i/h + 2^i$.

Next we define *grain-size* of an instance, which tells how to apply the lemmas.

Definition 2 (Grain-size for class i). *The smallest $w \in \{1, ..., W_i\}$ for which every class i segment of size w initially has at least $\sqrt{w \cdot 2^i}$ holes, where W_i is the number of class i external trains.*

Note: Since the segment of size W_i has infinite holes, grain-size is well defined.

Theorem 1. *Suppose all internal trains go to the last station N. In polytime, we can schedule class i trains to finish by time $O(F^*)$, where F^* is their optimal makespan. Moreover, $F^* \geq \widetilde{F} = \max\left\{2^{i-1},\ \frac{1}{4}\sqrt{w \cdot 2^{i-1}}\right\}$ where w is the grain-size for class i.*

Proof. Let $d = 2^{i-1}$, the minimum class i path-length, and w be the grain-size for class i, i.e., each class i segment of size w has at least $h = \lceil\sqrt{2wd}\rceil$ holes. Then $\sqrt{2wd} \leq h < \sqrt{w \cdot 2d} + 1$ and Lemma 2 gives a schedule where class i trains finish by time $F < 3\sqrt{w \cdot 2d} + 2d + 2$. For $w \geq 2$, we know that some segment of size $\frac{w}{2}$ must have less than $\sqrt{wd}$ holes, so applying Lemma 1 we have: $F^* \geq \max\left\{d, \min\left\{\frac{1}{4}\sqrt{wd}, \frac{1}{\sqrt{2}}\sqrt{wd}\right\}\right\} = \max\left\{d, \frac{\sqrt{wd}}{4}\right\}$. For $w = 1$, $F^* \geq d \geq \max\left\{d, \frac{\sqrt{wd}}{4}\right\}$. Thus, $F^* \geq \widetilde{F} = \max\left\{d, \frac{\sqrt{wd}}{4}\right\} \geq \frac{1}{8}\sqrt{wd} + \frac{1}{2}d$.

Algorithm 1: Preprocessing

Input: Π

for $i = 1, ..., \lfloor \log N \rfloor + 1$ **do**

  $\Pi_0 = \Pi$;

  For $j > 0$: $\Pi_j = \Pi$ with all internal trains of classes $0, ..., j$ replaced by holes, and destinations of all internal trains of classes $> j$ set to N;

  for $j = 0, ..., \log N$ **do**

    $w(j) \longleftarrow$ the grain-size for class i in Π_j as per Definition 2;

    $M(j) \longleftarrow \max\left\{2^{j-1},\ 2^{i-1},\ \frac{1}{4}\sqrt{w(j) \cdot 2^{i-1}}\right\}$;

  end

  $J_i \longleftarrow \text{argmin}_j\, M(j)$; $w_i \longleftarrow w(J_i)$; $h_i \longleftarrow \lceil \sqrt{w_i \cdot 2^i} \rceil$;

end

Output: (J_i, w_i, h_i) for each class $i \in \{1, ..., \lfloor \log N \rfloor + 1\}$

3.2 The Overall Scheduling Algorithm

We now consider the scheduling of all trains, using holes created by the exits of internal trains as well as the preexisting holes. As mentioned earlier, the classes are scheduled one after another in ascending order. Scheduling of each class i is as in the previous section but with the following two crucial differences.

First, for the entry of external trains, now we can also use holes created by the exits of internal trains (in addition to the preexisting holes). Which of them to use for class i has to be carefully decided, and the makespan lower-bound accordingly adjusted. We do that in a preprocessing module.

Second, delivering the previous classes $1, ..., i-1$ delays class i and also alters the distribution of holes (preexisting as well as created) relative to its external trains. However, all those holes do become available for class i as they are re-created at the exits of the trains of previous classes, although they appear shifted somewhat downstream of their initial positions. We show in a scheduling module that the delay and the shifts are small enough—relative to the class i path-lengths—for us to still schedule class i with a max-stretch which is a weighted sum of the max-stretch lower-bounds of classes $i, i-1, ..., 1$ with the corresponding weights in a decreasing geometric progression. Since the optimal max-stretch for all trains can be no smaller than the maximum of the class-wise lower-bounds, the overall max-stretch we achieve is only a constant times the optimal (Theorem 2).

Preprocessing. This module (Algorithm 1) answers the following question: for entering class i external trains, which holes should we use? In principle, we could use the holes left behind by internal trains of any class j as well as the preexisting holes. So we create an instance Π_j by removing internal trains of classes $1, \ldots, j$ and find the grain-size $w(j)$ for class i trains in Π_j, and then use Theorem 1 to determine a lower bound $M(j)$ on the class i makespan in Π_j. Clearly, $\min_j M(j)$ is a lower bound on the class i makespan in Π. The

Algorithm 2: Scheduling

Input: $\Pi, (J_i, w_i, h_i)$ for each class $i \in \{1, \ldots, \lfloor \log N \rfloor + 1\}$

for $i = 1, ..., \lfloor \log N \rfloor + 1$ **do**

  $F_{i-1} \longleftarrow$ the number of steps already executed for classes $1, ..., i-1$;

  Execute 2^{J_i} movement steps, with no entries;

  Schedule class i using Lemma 2 with $w = \widehat{w} = r_i w_i$ and $h = \hat{h} = r_i h_i - (F_{i-1} + 2^{J_i})$, where $r_i = \lceil \frac{F_{i-1}+2^{J_i}}{h_i} + \max\{\frac{F_{i-1}+2^{J_i}}{4h_i}, \frac{2^i}{h_i}\} \rceil$;

end

corresponding j is returned as J_i. The corresponding grain-size $w(j)$ is returned as w_i, and the promised number of holes per grain as h_i. We summarize this as follows.

Lemma 3. *Let F_i^* denote the optimal makespan if only class i trains are to be delivered. Then $F_i^* \geq \max\{2^{J_i-1}, 2^{i-1}, \frac{1}{4}\sqrt{w_i \cdot 2^{i-1}}\}$, where J_i, w_i are as per Algorithm 1.*

Proof. The last two terms are as per Theorem 1. The first term arises as 2^{J_i-1} steps have to elapse in order to use the holes created by exit of class J_i trains.

Scheduling. We schedule the classes one after another in ascending order. Class i trains use the holes left behind by internal trains of classes $1, \ldots, J_i$ where J_i is as determined during preprocessing. To ensure that these trains have exited, we run 2^{J_i} movement steps. Note that these holes will not be present at the same positions as in Π_j. To account for this and also to account for all the movements that occurred while delivering class $1, \ldots, i-1$ trains, we use a somewhat larger grain-size than w_i. (See Algorithm 2.)

Lemma 4. *Let F_i denote the makespan for class i trains as per our algorithm, F_i^* the optimal class i makespan, and $F_0 = 0$. Then $F_i = \frac{3}{2}F_{i-1} + O(F_i^*)$.*

Proof. In Π_{J_i}, every class i segment of size w_i has at least $h_i \geq \sqrt{w_i \cdot 2^i}$ holes. Thus, in Π, every class i segment of size $\widehat{w} = r_i w_i$ has $r_i h_i$ or more *potential holes*, i.e., actual holes and internal trains of classes $1, ..., J_i$. During the first F_{i-1} steps, some of them turn into holes and get used for entries of previous i–1 classes but then also get re-created. By the end of the following 2^{J_i} movement steps, all of them would be available as holes but possibly downstream from their initial positions. At most one train or hole may move out of any segment in a step, hence at time $F_{i-1} + 2^{J_i}$, every segment of size $\widehat{w}$ must have at least $\hat{h} = r_i h_i - (F_{i-1} + 2^{J_i})$ holes. Simplifying, we get: $\max\left\{\frac{1}{4}(F_{i-1} + 2^{J_i}), 2^i\right\} \leq \hat{h} < \frac{1}{4}(F_{i-1} + 2^{J_i}) + 2^i + h_i$, which implies: $\hat{h} \geq 2^i \geq \min\{\widehat{w}, 2^i\}$. Thus, Lemma 2 can indeed

be used with segment-size $\widehat{w}$, and $\hat{h}$ holes per segment, to let class i trains finish by time:

$$
\begin{aligned}
F_i &= (F_{i-1} + 2^{J_i}) + 2\hat{h} + \widehat{w} \cdot 2^i/\hat{h} + 2^i \\
&< (F_{i-1} + 2^{J_i}) + \frac{1}{2}(F_{i-1} + 2^{J_i}) + 2^{i+1} + 2h_i + \frac{(\hat{h} + F_{i-1} + 2^{J_i}) \cdot w_i \cdot 2^i}{h_i \cdot \hat{h}} + 2^i \\
&\qquad\qquad \because \hat{h} < \tfrac{1}{4}(F_{i-1} + 2^{J_i}) + 2^i + h_j \text{ , } \widehat{w} = rw_i \text{ , } r = \tfrac{\hat{h}+F_{i-1}+2^{J_i}}{h_i} \\
&< \frac{3}{2}F_{i-1} + \frac{3}{2}2^{J_i} + 2h_i + 5w_i \cdot 2^i/h_i + 3 \cdot 2^i \qquad\qquad \because \tfrac{1}{4}(F_{i-1} + 2^{J_i}) < \hat{h} \\
&< \frac{3}{2}F_{i-1} + \frac{3}{2}2^{J_i} + 2\sqrt{w_i \cdot 2^i} + 2 + 5\sqrt{w_i \cdot 2^i} + 3 \cdot 2^i \\
&\qquad\qquad \because \sqrt{w_i \cdot 2^i} \le h_i < \sqrt{w_i \cdot 2^i} + 1 \\
&= \frac{3}{2}F_{i-1} + O(F_i^*)
\end{aligned}
$$

The last line follows from that $F_i^* \ge \max\{2^{J_i-1}, 2^{i-1}, \frac{1}{4}\sqrt{w_i \cdot 2^{i-1}}\}$ by Lemma 3.

Theorem 2. *Our schedule has a max-stretch $O(1)$ times the optimal.*

Proof. Let X_i^* be the optimal class i max-stretch. Clearly, the overall max-stretch $X^* \ge X_i^*$ for all i. Let X_i be the class i max-stretch in our schedule.

We know that $F_i = \frac{3}{2}F_{i-1} + O(F_i^*)$. Thus $F_i = O(1)\sum_{k=1}^{i}\left(\frac{3}{2}\right)^{i-k} F_k^*$. Since trains of class k have path-lengths $< 2^k$, we have $X_k^* > \frac{F_k^*}{2^k} \Rightarrow F_k^* < 2^k X_k^*$. Since trains of class i have path-lengths at least 2^{i-1}, we get $X_i \le \frac{F_i}{2^{i-1}} = 2^{-i+1}F_i$. Substituting we have: $X_i = 2^{-i+1} \cdot O(1)\sum_{k=1}^{i}\left(\frac{3}{2}\right)^{i-k} 2^k X_k^* = O(1)\sum_{k=1}^{i}\left(\frac{3}{4}\right)^{i-k} X_k^* = O(1)X^*\sum_{k=1}^{i}\left(\frac{3}{4}\right)^{i-k} = O(X^*)$, because $X_k^* \le X^*$.

4 NP-Hardness

We reduce from the strongly NP-hard 3-Partition problem [10], defined below.

Definition 3 (Problem 3P). *Let U be a set of positive integers, and $S(U) = \sum_{u\in U} u$. Let $B = \frac{|U|}{3}$ and $C = \frac{S(U)}{B}$ be integers and $\frac{C}{4} < u < \frac{C}{2}$ $\forall u \in U$. Can U be partitioned into B triples such that each triple adds up to the same value C?*

The core of the reduction is a *solver* widget. This contains a train for each integer in the 3P instance. If and only if the 3P instance has a solution, the three trains corresponding to each triple in the partition get linked into a single chain. For making sure that only B chains get formed, we use a *hole-blocker* widget to prevent too many holes reaching the solver. The widgets, defined next, have size polynomial in $S(U)$, the size of the 3P instance in unary.

Lemma 5. *For any 3P instance U, there exists a widget* Solver(U) *with N_U stations and integers T and L such that N_U, T, L are polynomial in $S(U)$, $N_U > 4L$, and:*

1. *If U has a solution then the trains of the widget can be scheduled with a max-stretch at most $X_U = 1 + \frac{T}{L}$, and in each step, a train can enter the solver from upstream and subsequently move forward non-stop.*
2. *If the trains can be scheduled with max-stretch $\leq X_U$ and no hole enters the widget from upstream during the first $2T$ steps, then U must have a solution.*

Proof. Let $U = \{u_1, \ldots, u_n\}$ be the 3P instance, where $n = |U|$. Suppose $\alpha = 4B$ and $N_U = 2B + \alpha BC + \alpha\frac{C}{4}$. Define *Solver(U)* as stations $s_1, ..., s_{N_U}$ such that:

1. The first B stations $s_1, ..., s_B$ have holes, labelled respectively as $h_1, ..., h_B$.
2. For each u_i, we have an external train Q_i with path-length αu_i. These trains wait at outers downstream of s_B such that paths of all Q_is are node-disjoint. Let s_D be the destination of the most downstream of Q_is, i.e., $D = B + \alpha BC$.
3. Outers of $s_{D+1}...s_{D+B}$ have trains $R_1...R_B$, each with path-length $L = \alpha\frac{C}{4}$.
4. $s_{B+1}...s_{D+B}$ have trains going to the last station N, while $s_{D+B+1}...s_{N_U}$ have holes.

Clearly, $N_U > 4L$ and L, N_U are polynomial in $S(U)$.

Now suppose U has a solution $\{U_1, \ldots, U_B\}$. Then schedule the trains as follows. For each $k \in \{1, \ldots, B\}$, construct a chain c'_k consisting of:

1. the hole h_k at station s_k,
2. the three external trains (Q_is) for the three integers in U_k, and
3. the external train R_k.

In each step $k \in [B]$, let the first train of c'_k enter using h_k. Then let all chains progress in parallel—prioritize entries over movements and arbitrarily resolve the conflicts among entries. Entries occur in at most $n + B$ steps. In other steps, for each c'_k, a non-terminal train moves on the line unless R_k has already entered; there can be at most $\sum_{u \in U_k}(\alpha u - 1) = \alpha C - 3$ such steps.[3] Then at most $T = (\alpha C - 3) + (n + B) = \alpha(C + 1) - 3$ steps occur by the time R_k has entered, i.e., all Q_is and R_ks have entered by time T. (Clearly, T is polynomial in $S(U)$.) So no train needs to halt after time T, i.e., max-delay $\leq T$. Since every train has a path-length $\geq L$, max-stretch $\leq 1 + \frac{T}{L} = X_U$. Moreover, since the entries do not use any holes from upstream of the widget, in each step a train from upstream can enter the widget and then also move ahead non-stop.

Finally, suppose the trains can be scheduled with max-stretch $\leq X_U$ such that no upstream hole enters the widget in the first $2T$ steps. Consider the set of chains induced by the schedule. Every Q_i has a path-length $\alpha u_i < \alpha\frac{C}{2} = 2L$, and R_k has path-length L. So each suffers a delay $< 2L(X_U - 1) = 2T$, i.e., enters by time $2T$, hence it can not fill a hole from upstream of the widget. All internal trains go to the last station N. Therefore, entries can use the B holes $h_1, ..., h_B$ or exit holes of other external trains, i.e., the chain-set has at most B chains, say $c'_1, ..., c'_B$. Clearly, no two R_ks can be in same chain, so each must be the terminal train of a chain. Then Q_is must be the non-terminal trains. $\forall k \in [B]$, let U_k be the set of integers corresponding to the non-terminal trains of c'_k. Then

[3] An external train with path-length l moves only $l - 1$ steps on the line.

$\{U_1, ..., U_B\}$ is a partition of U. Since R_k has a path-length L and a stretch at most $1+\frac{T}{L}$, it must enter by time $T+1$, i.e., the path-lengths of the non-terminal trains of c'_k add up to at most $T = \alpha(C+1) - 3$, i.e., U_k adds up to at most $\frac{T}{\alpha} = (C+1) - \frac{3}{\alpha} < C+1$, i.e., at most C. Then, since $\frac{C}{4} < u < \frac{C}{2}\ \forall u \in U$, the partition $\{U_1, ..., U_B\}$ must be a valid solution to the 3P instance U.

Lemma 6. *Given integers $\ell \geq 1$ and $\tau \geq 2$, $\exists$ a widget* HoleBlocker(ℓ, τ) *of size polynomial in ℓ and τ such that:*

1. *Suppose there are at least 2ℓ stations downstream of the widget, and the widget's trains can move downstream from the widget and then continue non-stop. Then they can be scheduled with a max-stretch at most $X_{HB} = 1 + \frac{\tau-1}{\ell}$.*
2. *Suppose the widget's trains can be scheduled with max-stretch at most X_{HB}. Then no holes go downstream from the widget during the first τ steps.*

Proof. Let *HoleBlocker*(ℓ, τ) consist of blocks $F_0, \ldots, F_{q-1}, E, F_q$ from upstream to downstream, where $q = \lfloor \tau/\ell \rfloor$. For each $b \in \{0, ..., q\}$, F_b has $\tau - b\ell$ stations, each with an external train going a distance ℓ and an internal train going to the last station N. The external trains are labeled $P_{b,\tau-bl}$ to $P_{b,1}$ from upstream to downstream. E has ℓ stations with only internal trains, each going to the last station N. Overall, the widget has $w = (q+1)(q\ell/2 + r)$ external trains (where $r = \tau \bmod \ell$) and $w + \ell$ internal trains. Clearly, its size is polynomial in ℓ and τ.

To prove part 1, we make τ chains $c_1, ..., c_\tau$. Each c_j consists of the external trains $P_{b,j}\ \forall b \in \{0, ..., q\}$. In each step $j \in \{1...\tau\}$, we let the first train $P_{0,j}$ of c_j enter using a hole from the artificial stations $-1, -2, ...$ After the entry of its first train, each chain *progresses non-stop*. It is easy to see that all chains can do so in parallel, and that the number of external trains that will enter by time $\tau = q\ell + r$ is $\ell q(q+1)/2 + r(q+1) = (q+1)(q\ell/2 + r) = w$, i.e., all of the $P_{b,j}$s will enter by time τ. Then all trains move non-stop to their destinations. So any external train suffers a delay at most $\tau - 1$ and hence a stretch at most $1 + \frac{\tau-1}{\ell} = X_{HB}$; any internal train suffers a delay at most τ and hence a stretch at most $1 + \tau/2\ell \leq X_{HB}$.

To prove part 2, we note that since there were no holes in the widget and all internal trains go to the last station N, entries may fill the *external* holes (from artificial stations $-1, -2, ...$) or the exit holes of other external trains. Moreover, in any step at most one external hole can enter the widget. Then, it is easy to see (similar to part 2 of the proof of Lemma 1) that by time $\tau = q\ell + r$, at most $(q\ell/2 + r)(q+1)$ trains can enter, but that is exactly the count w of the external trains. That means if in any of the first τ steps a hole goes downstream from the widget, i.e., we miss using that hole to enter one of the widget's external trains, then not all of those trains can enter by time τ, which implies at least one of them will suffer a delay of τ or more and hence a stretch at least $1 + \frac{\tau}{\ell} > X_{HB}$.

With the two widgets defined above, we can now prove the hardness result.

Theorem 3. *Minimizing max-stretch on in-comb is NP-hard.*

Proof. The proof is by reduction from the 3-Partition problem. For an instance U of the 3-Partition problem, our Train Scheduling instance is as follows.

The in-comb network consists of a HoleBlocker(ℓ, τ) followed by a Solver(U). We choose $\ell = 2L$ and $\tau = 2T + 1$, where T, L are as promised by Lemma 5. Note that the size of our solver widget is $N_U > 4L$, so the path-length of every internal train of the hole-blocker is at least $4L = 2\ell$, as required by Lemma 6. Then it can be seen from Lemmas 5 and 6 that the size of the train scheduling instance is polynomial in $S(U)$. We fix the target max-stretch X to $1 + \frac{T}{L}$.

Now, suppose the 3-Partition instance U has a solution. From Lemma 5 part 1, we can schedule trains of the solver widget with max-stretch at most X such that any trains coming into the solver from the hole-blocker can move ahead non-stop. Then, from Lemma 6 part 1, trains of the hole-blocker can also be scheduled with max-stretch at most $1 + \frac{\tau - 1}{\ell} = X$.

Conversely, suppose all trains can be scheduled with stretch at most X. From Lemma 6 part 2, we know that no hole goes downstream from the hole-blocker in the first $2T$ steps. Then, by Lemma 5 part 2, the 3-partition instance U must have a solution.

Finally, note that the above construction is broadly similar to the one used in [11] for the max-delay minimization problem, but now the crucial hole-blocker widget has to be designed more carefully in order to prove the hardness of minimizing max-stretch.

5 Conclusion

The train scheduling problem on in-comb was introduced in [11]. That work also gave the chain-hole view as an important insight into the problem and used it fruitfully to prove the hardness of max-delay minimization as well as design an $O(\log N)$ approximation algorithm for it. We have used the same chain-hole view but with entirely new ideas for lower and upper bounds to give an $O(1)$ approximation algorithm and the hardness proof for a lesser explored but presumably more relevant metric—the max-stretch.

Further work can focus on tighter bounds as well as on possible generalizations of the problem, e.g., multiple tracks between stations, multiple trains at every station, variable train speeds, etc. Parametric formulations of the problem—e.g., with a given maximum number of external trains—should also be studied.

A Proof of Lemma 2

Without loss of generality, extend the path-length of every external train (of class i) to $\ell = 2d - 1$, where $d = 2^{i-1}$. Starting from upstream, partition the network into groups of stations, each group being a class i segment of size w, i.e., containing w external trains.

The schedule is trivial for the case when $h \geq w$: external trains of one set of alternate groups can enter in the first $w \leq h$ steps and those of the other

set in the next w steps, trains of each group filling the holes of its upstream neighbour; afterwards all trains can move to their destinations during the next $\ell - 1$ steps, thus giving a makespan of $2w + \ell - 1 < 2h + 2^i$. Next, let us consider the non-trivial case: $w > h > 2d - 1$.

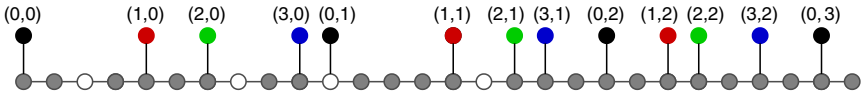

Fig. 3. Chains in a group ($i = 2, w = 13, h = 4$). Circles denote holes and grey disks denote internal trains. The other, labeled disks denote external trains, disks of a color representing trains of a chain.

Within each group, number the external trains from 0 to $w - 1$ starting upstream, and then form h chains, allocating the trains to the chains in a round-robin fashion: train p becomes the j^{th} train of the k^{th} chain, where $j = \lfloor p/h \rfloor$ and $k = p \bmod h$. We may refer to train p also as train (k, j). Thus, each chain will have some J trains where $J \leq \lceil w/h \rceil$. (See Fig. 3 for an example of chain formation in a group.)

For $k \in \{0, ..., h - 1\}$, consider train $(k, 0)$ in any group G. Let q denote the first unassigned hole downstream of train $(k, 0)$ in the previous group $G - 1$. Assign this is as the initial hole of chain k in group G. Because each segment of size w starting at any external train is guaranteed to have h holes, it can be easily seen that this assignment succeeds in finding a distinct initial hole for every chain.

Each external train has to perform one entry and $2d - 2$ movements. Let us also consider the jump of the hole that it fills as a movement of the entering train, i.e., the train has $2d-1$ movements. In each chain, number the movements from 1 to $(2d - 1) \times J$, starting upstream and numbering consecutively for all trains of the chain. We will use $\langle k, m \rangle$ to denote m^{th} movement in chain k. Note that among the movements of different chains, we have an *overlap condition*: if $k' > k$ then the movement $\langle k, m \rangle$ may overlap with $\langle k', m' \rangle$ only for $m' \leq m$.

The train movements are scheduled in three phases, as follows:

1. This has h steps: $1, ..., h$. In step i, the initial hole of chain $h - i$ in every group jumps to the first train $(h - i, 0)$ of the chain and gets filled by the train. It is easily seen that paths of the jumps in each step are disjoint.
2. In this phase, every train—except the last—in a chain completes its journey and exits the network; the re-created hole then jumps to the next train of the chain. We discuss this in more detail below.
3. In this last phase, the last trains of all chains move to their destinations. It is easily seen that there are no conflicts and this phase takes $2d-1$ steps.

In Phase 2, the paths of trains in one group do not overlap with those in other groups. So, we can consider each group separately. To resolve the conflict between overlapping movements due to take place in the same step, we use a

very simple *scheduling rule*: higher numbered chains have higher priority. Now we can use a simple *delay sequence argument*—a proof technique used earlier in [1,22,28]—to prove the time bound as follows.

Suppose $\langle k, m\rangle$ occurs in step t. Then, for $t-1$, one of the following is true:

1. Movement $\langle k, m-1\rangle$ occurred in step $t-1$.
2. Movement $\langle k', m'\rangle$ occurred in step $t-1$, delaying the (lower priority) movement $\langle k, m\rangle$. Here $k' > k$ because of the scheduling rule and $m' \leq m$ because of the overlap condition.

Thus, if the last movement of Phase 2 occurs in step $h+T$ then we can find a sequence of movements $\langle k, m\rangle$, one for each of the steps $h+T$, $h+T-1$, ..., $h+1$, such that:

1. k never decreases,
2. m never increases, and
3. at least one of the two does change.

But this can happen only $h+(2d-1)(J-1) < h+2d\cdot w/h$ times. So, the overall time for all the three phases is only less than $2h + 2d \cdot w/h + 2d$.

Figure 4, on page 16, illustrates the schedule for a small example as a space-time diagram. Note that while the space-time trajectories of trains can not cross one-another, trajectories of holes may cross those of trains because a hole can jump over stationary trains.

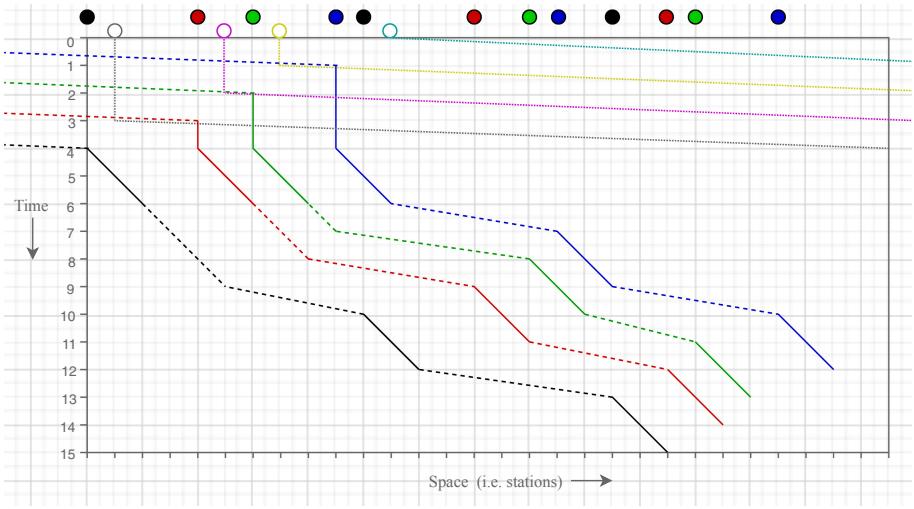

Fig. 4. Space-time diagram of a group, say G, for $i = 2, w = 12, h = 4$. Colored disks and circles at the top respectively represent external trains (of G) and holes (matched to the entries of the next group $G+1$) at their initial positions. External trains of a color belong to same chain. Thus, black lines mark the space-time trajectory of the 0^{th} chain, red lines the trajectory of the 1^{st} chain, and so on. Solid lines represent train-movements. Dashed lines represent hole-jumps for the entries of G, while dotted lines represent the hole-jumps for the entries of $G+1$. (Color figure online)

References

1. Aleliunas, R.: Randomized parallel communication (preliminary version). In: Proceedings of the First ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 1982, pp. 60–72. ACM, New York (1982)
2. Cacchiani, V., et al.: An overview of recovery models and algorithms for real-time railway rescheduling. Transp. Res. Part B: Methodol. **63**, 15–37 (2014)
3. Cai, X., Goh, C., Mees, A.I.: Greedy heuristics for rapid scheduling of trains on a single track. IIE Trans. **30**(5), 481–493 (1998)
4. Caimi, G., Chudak, F., Fuchsberger, M., Laumanns, M., Zenklusen, R.: A new resource-constrained multicommodity flow model for conflict-free train routing and scheduling. Transp. Sci. **45**(2), 212–227 (2011)
5. Chiang, T., Hau, H., Chiang, H.M., Kob, S.Y., Hsieh, C.H.: Knowledge-based system for railway scheduling. Data Knowl. Eng. **27**(3), 289–312 (1998)
6. Clementi, A., Ianni, M.D.: Optimum schedule problems in store and forward networks. In: 13th Proceedings of IEEE Networking for Global Communications, INFOCOM 1994, vol. 3, pp. 1336–1343, June 1994
7. D'Ariano, A.: Improving real-time train dispatching: models, algorithms and applications. Doctoral thesis, TRAIL Research School, Deft, The Netherlands (2008)
8. Fang, W., Yang, S., Yao, X.: A survey on problem models and solution approaches to rescheduling in railway networks. IEEE Trans. Intell. Transp. Syst. **16**(6), 2997–3016 (2015)
9. Flier, H., Mihalák, M., Schöbel, A., Widmayer, P., Zych, A.: Vertex disjoint paths for dispatching in railways. In: Erlebach, T., Lübbecke, M. (eds.) 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'10). OpenAccess Series in Informatics (OASIcs), vol. 14, pp. 61–73. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2010)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
11. Garg, A., Ranade, A.G.: Train scheduling on a unidirectional path. In: Lokam, S., Ramanujam, R. (eds.) 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017). Leibniz International Proceedings in Informatics (LIPIcs), vol. 93, pp. 29:1–29:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl (2018)
12. Iyer, R.V., Ghosh, S.: Daryn-a distributed decision-making algorithm for railway networks: modeling and simulation. IEEE Trans. Veh. Technol. **44**(1), 180–191 (1995)
13. Krasemann, J.T.: Design of an effective algorithm for fast response to the rescheduling of railway traffic during disturbances. Transp. Res. Part C: Emerg. Technol. **20**(1), 62–78 (2012). Special issue on Optimization in Public Transport+ISTT2011
14. Leighton, F.T., Maggs, B.M., Rao, S.B.: Packet routing and job-shop scheduling in o(congestion+dilation) steps. Combinatorica **14**(2), 167–186 (1994)
15. Leighton, T., Maggs, B., Richa, A.W.: Fast algorithms for finding o(congestion + dilation) packet routing schedules. Combinatorica **19**(3), 375–401 (1999)
16. Leung, J.Y.T., Tam, T.W., Young, G.H.: On-line routing of real-time messages. J. Parallel Distrib. Comput. **34**(2), 211–217 (1996)
17. Mannino, C., Mascis, A.: Optimal real-time traffic control in metro stations. Oper. Res. **57**(4), 1026–1039 (2009)

18. Mascis, A., Pacciarelli, D.: Job-shop scheduling with blocking and no-wait constraints. Eur. J. Oper. Res. **143**(3), 498–517 (2002)
19. Narayanaswami, S., Rangaraj, N.: Scheduling and rescheduling of railway operations: a review and expository analysis. Technol. Oper. Manage. **2**(2), 102–122 (2011)
20. Peis, B., Skutella, M., Wiese, A.: Packet routing: complexity and algorithms. In: Bampis, E., Jansen, K. (eds.) WAOA 2009. LNCS, vol. 5893, pp. 217–228. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12450-1_20
21. Pellegrini, P., Rodriguez, J.: Single European sky and single european railway area: a system level analysis of air and rail transportation. Transp. Res. Part A: Policy Pract. **57**, 64–86 (2013)
22. Ranade, A.G.: Fluent parallel computation. Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT, USA (1989). aAI9010675
23. Rothvoß, T.: A simpler proof for o(congestion + dilation) packet routing. CoRR abs/1206.3718 (2012)
24. Sahin, I.: Railway traffic control and train scheduling based oninter-train conflict management. Transp. Res. Part B: Methodol. **33**(7), 511–534 (1999)
25. Salim, V., Cai, X.: A genetic algorithm for railway scheduling with environmental considerations. Environ. Model Softw. **12**(4), 301–309 (1997)
26. Scheideler, C.: Universal routing strategies for interconnection networks. In: Goos, G., Hartmanis, J., van Leeuwen, J. (eds.) Lecture Notes in Computer Science, vol. 1390, pp. 57–67. W. H. Freeman & Co., New York (1998)
27. Tormos, P., Lova, A., Barber, F., Ingolotti, L., Abril, M., Salido, M.A.: A genetic algorithm for railway scheduling problems. In: Xhafa, F., Abraham, A. (eds.) Metaheuristics for Scheduling in Industrial and Manufacturing Applications. Studies in Computational Intelligence, vol. 128, pp. 255–276. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78985-7_10
28. Upfal, E.: Efficient schemes for parallel communication. J. ACM **31**(3), 507–517 (1984)
29. Šemrov, D., Marsetič, R., Žura, M., Todorovski, L., Srdic, A.: Reinforcement learning approach for train rescheduling on a single-track railway. Transp. Res. Part B: Methodol. **86**, 250–267 (2016)