Online Delay Management on a Single Train Line

Michael Gatto, Riko Jacob, Leon Peeters, and Peter Widmayer

Institute of Theoretical Computer Science, ETH Zurich
{gattom,rjacob,leon.peeters,widmayer}@inf.ethz.ch

Abstract. We provide competitive analyses for the online delay management problem on a single train line. The passengers that want to connect to the train line might arrive delayed at the connecting stations, and these delays happen in an online setting. Our objective is to minimize the total passenger delay on the train line.

We relate this problem to the Ski-Rental problem and present a family of 2-competitive online algorithms. Further, we show that no online algorithm for this problem can be better than Golden Ratio competitive, and that no online algorithm can be competitive if the objective accounts only for the optimizable passenger delay.

1 Introduction

In the everyday operation of a railroad, it is unfortunately not uncommon for a train to arrive at a station with a delay. In such a situation, some of the train's passengers may miss a connecting train, resulting in an even larger delay for them since they have to wait for the next train. If, on the other hand, the connecting train waits, then it is delayed itself, and so are all the passengers it is carrying. Delay management consists of deciding which connecting trains should wait for what delayed feeder trains, usually with the objective of minimizing the overall discomfort faced by the passengers. Although railway optimization and scheduling problems have been studied quite intensively the last decade, the management of delayed trains has received much less attention.

Various approaches to delay management have been considered the last years, such as simulations, Linear and Integer Programming, or complexity and algorithmic analyses (Section 1.1 provides an overview of related research). However, except for very few exceptions, these papers consider delay management as an offline problem, where the delays are known a priori, and for which a global optimal solution is sought. But delays in a railway system are by nature not known a priori (even though they may be correlated). Therefore, delay management is inherently an online problem.

This paper studies delay management as an online problem, with a focus on competitive analyses. Given the lack of research on online delay management, we consider the basic case of a single train operating on a train line consisting of several consecutive legs. Moreover, previous research on offline delay management problems indicates that railway networks with a path topology are easier to analyze. At each intermediate stop of the train line, some passengers wish to board the train. Each passenger has a destination, and possibly an initial delay, meaning that she arrives at the transfer station with a delayed feeder train. Should a passenger miss her connection at the transfer station, then she has to wait for the next train. Further, we assume a timetable without buffer times, so a train cannot catch up on any of its delay.

The problem is to decide, for each intermediate station, whether the train waits for delayed passengers or not. If it waits, then all its passengers face an arrival delay, including the ones that were so far on time. The same holds for all passengers boarding the train at subsequent stations, since the train cannot catch up on its delay. If the train departs on time, then all connecting passengers will miss their train and have to wait for the next train. A waiting policy specifies at which intermediate stations the train waits for delayed passengers. The goal of the online delay management problem is to find a waiting policy that minimizes the total passenger delay, without knowing beforehand whether the connecting passengers will arrive with a delay at the subsequent intermediate stations. Although our descriptions are in terms of railways, we remark that our model and results are also applicable to other modes of scheduled public transportation, such as bus or metro.

1.1 Related Research

To the best of our knowledge, the only other theoretical online analysis of a delay management problem is by [1]. They consider a bus station with buses arriving at regular time intervals, and passengers arriving with a fixed arrival rate. For this problem, the objective is to decide which buses should wait for how long at the bus station such that the overall passenger waiting time is minimized.

A fair amount of research has been done on offline delay management. Several network-based Mixed Integer Programming (MIP) formulations were introduced [8,9], both for single criteria and bi-criterial objective functions. In particular, some formulations allow special cases to be solved to optimality efficiently, and the model's structure can be used to derive an appropriate Branch-and-Bound algorithm for solving the delay management problem to optimality.

Recently, [3] described polynomial time algorithms for special cases of the delay management problem, such as a limited number of transfers, or a railway network with a path topology. In a follow-up paper [4], a more general variant of the delay management problem was shown to be NP-complete both with and without slack times (or buffer times) in the timetable.

Another approach, based on simulation, applies deterministic waiting rules [2]. The delays are introduced on trains randomly over time with an exponential distribution, and the quality of the waiting rule is derived with an agent-based simulation tool.

Finally, [5,7] considered simulation studies for delay management, which are less related to this paper because we are interested in strategies with a theoretically guaranteed performance.

1.2 Results

We consider the above described setting of a simple railway network with a path topology, which is a natural next step after the station analysis by [1]. We relate this setting of the online delay management problem to a variant of the well-known ski rental problem [6]. Based on this relation, we propose a family of 2-competitive online algorithms, and show that the competitive analysis for this family of algorithms is tight. Further, we prove that no online algorithm for this setting can be better than Φ-competitive, where $\Phi \approx 1.618$ is the golden ratio. Finally, we consider the slightly different objective function of minimizing only the additionally faced delays of the passenger paths. For this objective, we show that no deterministic online algorithm can have bounded competitive ratio. Remarkably, the only strategy for this case not having infinite competitive ratio is the trivial strategy of waiting for any delayed passengers, and departing on time otherwise.

1.3 Outline

The next section defines the problem statement and our assumptions on the problem. Section 3 first introduces some notation, and discusses the relation between the online delay management problem and the so-called Discounted Ski Rental problem. Next, we present the family of 2-competitive online algorithms, as well as the tightness of the competitive analysis. Section 4 contains the two competitive ratio bounds for any online algorithm for the single train line, after which Section 5 concludes the paper and suggest some topics for further research.

2 Problem Statement

We consider a directed graph $G = (V, E)$, with the vertices $V = \{v_1, \ldots, v_n\}$. Let $E = \{(v_i, v_{i+1}) | i \in \{1, \ldots, n-1\}\}$, i.e. the graph forms a simple directed path from v_1 to v_n. The vertices $v \in V$ represent the stations which are served by a single railway line. An edge $(u, v) \in E$ represents a direct link served by the train along the railway line. Hence, the train serving this line starts its journey at station v_1, and traverses the graph through the directed edges. We assume that the train stops at every intermediate station $v_i, i \in \{2, \ldots, n-1\}$ and ends its journey at station v_n. At each stop, passengers may board or alight from the train. We refer to the passenger streams as passenger paths. Each passenger path boards the train at a station v_i, and alights at station $v_j, j > i$. We say the passenger paths enter at station v_i. These passenger paths represent passengers either starting their journey at station v_i, or having arrived at station v_i with another train, a feeder train, and wishing to connect to the train line. The passenger paths may continue their journey outside the train line, but this is not part of the considered problem. The passenger paths are known a priori, as well as the number of passengers travelling along each of them.

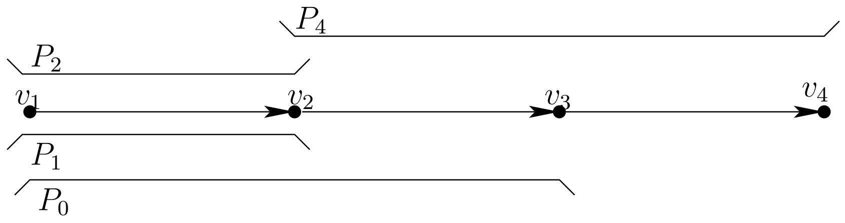

Fig. 1. An illustration of a railway line. The railway line has two intermediate stations (v_2 and v_3), and four paths: P_1, P_2, P_3 and P_4.

An illustration of a railway line and its paths is given in Figure 1. In the example, the passenger path P_1 connects to the train at station v_1 and alights from the train at station v_2. Similarly, path P_0 connects to the train at station v_1 and alights from the train at station v_3, where it continues its journey outside our track.

Some of the connecting passenger paths at the stations $v_i \in V$ might arrive there delayed. We say these paths enter the train line delayed, or that they have a source delay. This reflects the model of feeder trains arriving delayed at station v_i. For simplicity, we assume that all source delays equal δ time units. In order to allow such delayed paths to connect to the train, the train must wait for the delayed paths. Once delayed, the train is not able to catch up on any of this delay, and will hence arrive at its target station v_n with a delay of δ time units. Furthermore, the next train departs T time units later, and will run on time. Hence, should a passenger path miss its connection, it will catch the next train along the line and arrive at its destination with a total delay of T time units. We refer to passenger paths which have missed a connection as dropped paths, and to those facing an arrival delay as delayed paths.

It is not a priori known which passenger paths entering at station $v_i \in V$ start delayed. The delays occur in an online fashion. The online setting, which can be thought of as an adversary, notifies the online algorithm which of the entering paths at station $v_i \in V$ are delayed only when the train arrives at station v_i. Hence, only at that point in time can an online algorithm figure out how many passengers reach the station delayed, and how many are still on time.

When the train reaches station $v_i \in V$, the traffic control must decide whether the train departs on time, or whether it waits for the delayed passenger paths boarding the train in v_i. If the train departs on time, the delayed entering passengers miss their connection and will have to board the next train along this route. If the train waits for the delayed passengers, it will be delayed by δ time units. As the delays of the passenger paths are of the same size, a decision to wait will immediately guarantee the connections for all future delayed entering passenger paths. Hence, the decision to be taken is from which station on the train will wait, from then on allowing transfers of all future delayed entering passenger paths. Naturally, non-delayed passengers paths which are influenced by this decision thereby face a delay of δ time units.

Our objective is to minimize the total passenger delay occurring on the train line, i.e. the sum of the delays over all passengers which travel with the line. Notice that, in general, the objective includes the unavoidable delay of δ time units per passenger of the passenger paths that enter the line delayed.

As mentioned, we analyze the online setting of this problem. To solve the offline problem we merely have to decide where to start waiting. Therefore, we can efficiently enumerate the n different waiting policies. The adversary can decide which of the passenger paths to delay. When the train arrives at station v_i, the adversary must notify which passenger paths entering at that station are delayed. This implies that the online algorithm, when arriving at station v_i, knows exactly which passenger paths were delayed at stations $v_j, j \in \{1, \ldots, \text{i-1}\}$, and which passenger paths are delayed at station v_i. As giving the least information to the online algorithm is advantageous for the adversary, we assume the adversary does not reveal the delayed passenger paths connecting at stations $v_k, k > j$ until we reach these stations.

3 Competitive Online Algorithms

This section presents a family of 2-competitive online algorithms for the described delay management problem. First, we introduce some notation and some inequalities needed to prove the competitiveness of the online algorithms. Next, we point out the similarity between the presented online delay management problem and the Ski-Rental problem. Finally, we move the focus to the online algorithms and their analysis.

3.1 Notation and A-Priori Knowledge

As mentioned earlier, we assume the passengers paths on the train line to be known. Table 1 introduces a set of variables representing the number of passengers having a specified status at their connecting station (delayed, on time). These variables reflect the a-posteriori knowledge of the delays, i.e. the number of passenger in a specific state when the delay configuration is entirely known. If the train starts to wait for delayed entering passenger paths at station j, the value of our objective $\varDelta(j)$ is:

$$\varDelta(j) = (T - \delta) \sum_{i<j} D^i + \delta \sum_{i=1}^{n} D^i + \delta O^{\geq j}$$

We wish to find j such, that $\varDelta(j)$ is minimal. Notice that we include the delay $\delta \sum_{i=1}^{n} D^i$, which we cannot optimize, in the objective. In Section 4 we show that if we do not include this delay, no online algorithm can be more competitive than applying a trivial strategy having unbounded competitiveness.

We define the variables of Table 1 also for the point of view of an online algorithm, see Table 2. The variables are identified by the same letters used for the offline algorithm, but by using lowercases. The following relations hold:

Table 1. The passenger variables for the offline setting

$O^{\geq i}$	The number of passengers which stay on the train at station i plus on time connecting passengers at and after station i, i.e. passengers newly subject to a delay if the train waits at station i.
D^i	The number of connecting passengers which arrive delayed at station i, i.e. passengers subject to be dropped at station i if the train departs on time from station i.

Table 2. The passenger variables for the online setting

$o^{\geq i}$	The sum of the number of passengers which are on the train at station i, the number of on time passengers connecting at i, and the number of connecting passengers at future stations. Note that the online algorithm believes that the latter passengers will be on time. These are the passengers subject to be newly delayed if the train waits at station i.
d^i	The number of connecting passengers which arrive delayed at station i. These will be dropped if the train departs on time from station i.

$$O^{\geq i+1} \leq O^{\geq i} \tag{1}$$

$$o^{\geq i+1} \leq o^{\geq i} \tag{2}$$

$$O^{\geq i} \leq o^{\geq i} \tag{3}$$

$$o^{\geq i} = O^{\geq i} + \sum_{j>i} D^j \leq O^{\geq i} + \sum_{j} D^j \tag{4}$$

Inequalities (1) and (2) hold, because at each station passengers may alight from the train. Hence, the number of passengers influenced by a delay monotonically decreases if the train starts to wait at a later station. Inequality (3) holds, as the online algorithm does not know which passengers will arrive delayed after station i. Hence, $o^{\geq i}$ is an upper bound on the number of passengers that will be delayed by a waiting decision. Finally, this overestimate equals the actual number of passengers influenced by this decision (i.e. with the a-posteriori knowledge of the delayed passenger paths), plus the number of passengers that will be delayed after station i. This number can naturally be bounded as shown in inequality (4).

When at a station i, the online algorithm knows the correct number of delayed passengers: $d^j = D^j, \forall j \leq i$.

3.2 Relation to the Ski-rental Problem

In the Ski-Rental problem, a skier wishes to go skiing. He does not known how many times he will actually go skiing, because he will only ski if the weather is nice and there is enough snow. This reflects the online situation. Initially, the skier does not own a pair of skis. Each time he goes skiing, he can either rent the skis at a fixed price or buy the skis. Obviously, buying is more expensive than renting. With the a-posteriori knowledge on the number of times he went skiing, it is clear that it only makes sense to buy the skis if the overall renting costs exceed the price of buying the skis. Furthermore, in that case the skier should buy the skis the first time he goes skiing. Indeed, the only decision to be taken is whether to buy the skis or to rent them. A well known online strategy is to buy the skis as soon as the overall renting costs would exceed the costs of buying the skis. This online strategy is 2-competitive, as analyzed in [6].

We introduce a slight variant of the above Ski-Rental problem. As the skier rents the skis always at the same shop, the shop owner gives him a discount on buying the skis. The discount is proportional to the amount of money the skier has already paid for renting skis, and the proportionality factor is α. Next, we allow the renting price to be variable, but known each time the skier wishes to rent the skis. We call this problem the Discounted Ski-Rental problem. The usual strategy for the Ski-Rental problem can also be applied to the Discounted Ski-Rental problem. The skier buys the skis as soon as the overall renting costs would exceed the actual costs of buying the skis, i.e. the price of the skis with the discount.

In the following, we present a one to one correspondence between a restricted version of the online delay management problem and the Discounted Ski-Rental problem. We restrict the online delay management problem as follows: all passenger paths have as destination v_n. The objective is still to minimize the overall passenger delay, including the delay δ of passenger paths entering delayed at the stations along the train line. Note that the latter part of the objective cannot be optimized. In this setting, the contribution to the objective of $\delta \sum_i D^i + \delta O^{\geq j}$, i.e. the delay of the paths arriving with a delay δ at their destination, plus the δ delay of all dropped paths, is independent of the waiting decision.

We map this constant sum of delays to the original, undiscounted, price of buying the skis in the Discounted Ski-Rental problem. Moreover, the sum of the costs of the dropped paths (i.e. T time units per passenger) at station v_j corresponds to the renting price of the skis on that day. Therefore, there is a bijection between waiting at station v_j and buying the skis on day j. Hence, the cost of buying the skis on day j corresponds to the delay caused by waiting at station v_j, whereas the cost of renting the skis on day j corresponds to the cost of dropping the delayed paths at station v_j. By setting the proportional discount factor to $\alpha = \frac{\delta}{T}$ we complete the mapping. Indeed, the cost of waiting at station v_j is the same as buying the skis on day j:

$$\text{ski-cost}(j) = T \sum_{i<j} D^i + \delta \sum_i D^i + \delta O^{\geq j} - \frac{\delta}{T} T \sum_{i<j} D^i = \Delta(j)$$

Our analysis of the online delay management problem hence also provides a 2-competitive algorithm for the above Discounted Ski-Rental problem. Further, the mapping also provides some intuition for the online algorithm in Section 3.3. In the 2-competitive algorithm for the Discounted Ski-Rental problem, the skier buys the skis as soon as the overall renting costs exceed the costs of buying the skis. In the next section we show that a similar strategy is 2-competitive for the online delay management problem.

We point out that the general setting where passengers can alight from the train at any station along the path, is still related to a Ski-Rental problem. However, the mapping is much less intuitive, and therefore we omit it. Finally, differently from the Ski-Rental problem, the maximum number of times the decision is to be taken is known a priori in the online delay management problem, as the number of stations on the train line is known in each instance.

3.3 A Family of 2-Competitive Online Algorithms

In this section, we describe a family of 2-competitive online policies for the delay management problem described in Section 2. Recall that paths may end before the last station of the line. Hence, the problem is structurally different from both the Discounted Ski-Rental problem and the general Ski-Rental problem. In fact, in most Ski-Rental problems the key decision for an optimal adversary is whether or not to buy skis. Hence, the decision is boolean. On the contrary, for an optimal adversary of the online delay management problem, the decision is not only whether to wait or not, but additionally where the online algorithm should start to wait in order to achieve the optimal policy.

Nevertheless, the family of online algorithms resembles the classical online algorithm for the Ski-Rental problem. Loosely speaking, the train should wait at station v_j if the delay caused by dropping passengers up to and including station v_j exceeds the delay caused to on time passengers by waiting in v_j.

As we present a family of algorithms, we must be a little more precise. For $t \in [T - \delta, T]$, the online algorithm $\textsc{Alg}(t)$ of the family lets the train wait at station j if

$$t \sum_{i \leq j} d^i \geq \delta o^{\geq j}.$$

Note that the two extremal values of t lead to two extremal behaviors within the same policy. By setting $t = T - \delta$ we obtain the algorithm that starts to wait as late as possible. By setting $t = T$, we obtain the algorithm of the family that starts to wait as early as possible. Below, we show that both extremal algorithms are 2-competitive.

Intuitively, these algorithms achieve the competitive ratio of 2 by a similar argument as for the Ski-Rental problem: the algorithms drop all source delayed passenger paths, until the accumulated delay balances the delay which would occur if the train started to wait. At this point, an adversary could leave all other remaining passenger paths to be on time, thus causing again the same amount of delay to the online algorithm, whereas it would have been optimal not to wait at all. On the other hand, should the online algorithm have waited earlier, the analysis shows that the adversary must also have had a delay equal to the one of the online algorithm.

Theorem 1. *The family of online algorithms* $\textsc{Alg}$*(t) which start to wait at station j if $t \sum_{i \leq j} d^i \geq \delta o^{\geq j}, t \in [T - \delta, T]$, is 2-competitive on the single train line with fixed passenger paths.*

Proof. Let j^* be the station where the optimal offline algorithm waits and j be the station where the online algorithm $\textsc{Alg}(t)$ waits. The analysis is subdivided into two cases. In fact, if not optimal, the online algorithm either started waiting too early ($j < j^*$) or too late ($j^* < j$). In a worst-case scenario, for analyzing the first case, we take the algorithm of the family which waits the earliest, i.e. $\textsc{Alg}(T)$. Similarly, for analyzing the case where the algorithm waits too late, we take the algorithm of the family which waits the latest, i.e. $\textsc{Alg}(t-\delta)$.

Case $j^* < j$: we compare the objective value of the two solutions:

$$\begin{aligned}\Delta(j) =& (T-\delta)\sum_{i<j} D^i + \delta \sum_i D^i + \delta O^{\geq j}\\ =& \underbrace{(T-\delta)\sum_{i<j^*} D^i + \delta\sum_i D^i + \delta O^{\geq j*}}_{\Delta(j^*)} - \delta O^{\geq j*} + \delta O^{\geq j} + (T-\delta)\sum_{i=j^*}^{j-1} D^i\end{aligned}$$

First, we note that as $j^* < j$, inequality (1) implies the following: $-\delta O^{\geq j*} + \delta O^{\geq j} \leq 0$.

Second,

$$\sum_{i=j^*}^{j-1} D^i \leq \sum_{i\leq j-1} D^i.$$

Since the train did not wait at station $j-1$, and we use $t = T-\delta$, the following inequalities hold:

$$\begin{aligned}(T-\delta)\sum_{i\leq j-1} D^i \leq \delta o^{\geq j-1} &\overset{(4)}{\leq} \delta\left(O^{\geq j-1} + \sum_i D^i\right)\\ &\leq \delta\left(O^{\geq j^*} + \sum_i D^i\right) \leq \Delta(j^*).\end{aligned}$$

Concluding,

$$\Delta(j) \leq \Delta(j^*) - \delta O^{\geq j*} + \delta O^{\geq j} + \delta\left(O^{\geq j^*} + \sum_i D^i\right) \leq 2\Delta(j^*).$$

Case $j^* > j$: similarly to the previous case, we compare the values of the two solutions:

$$\Delta(j) = \Delta(j^*) - \delta O^{\geq j^*} + \delta O^{\geq j} - (T-\delta)\sum_{i=j}^{j^*-1} D^i.$$

Inequality (3) and waiting at j with $t = T$ imply

$$\delta O^{\geq j} \leq \delta o^{\geq j} \leq T\sum_{i\leq j} D^i \leq T\sum_{i\leq j^*} D^i.$$

Since

$$T \sum_{i \le j^*} D_i \le (T-\delta) \sum_{i \le j^*} + \delta \sum_{i} D_i \le \Delta(j^*),$$

we finally have

$$\Delta(j) \le \Delta(j^*) - \delta O^{\ge j^*} + \Delta(j^*) - (T-\delta) \sum_{i=j}^{j^*-1} D^i \le 2\Delta(j^*). \qquad \square$$

We have shown that the family of online algorithms $\text{ALG}(t)$, $t \in [T-\delta, T]$ is 2-competitive.

Corollary 1. *The analysis of the family of online algorithms* $\text{ALG}(t)$*, with* $t \in [T-\delta, T]$*, is tight.*

Proof. We show that the analysis is tight, both when the online algorithm starts waiting earlier or later than optimum. For each case, we analyze the worst-case scenario for the family $\text{ALG}(t)$: for starting to wait too late, we analyze the criterion which will wait the latest, i.e. $(T-\delta)\sum_{i \le j} d_i \ge \delta o^{\ge j}$; for starting to wait too early, we analyze the criterion of the family which will wait the earliest, i.e. $T \sum_{i \le j} d_i \ge \delta o^{\ge j}$.

We start by analyzing the case where the online algorithm starts waiting after the optimal solution. Consider the simple train line built by three stations, $V = \{v_1, v_2, v_3\}$, $E = \{(v_1, v_2), (v_2, v_3)\}$. We introduce two passenger paths $P_0 = \{v_1, v_2\}$ and $P_1 = \{v_2, v_3\}$, both connecting from other feeder trains. Potentially, these passenger paths could be delayed. Let p_0 be the number of passengers following path P_0, $p_1 = \frac{(T-\delta)}{\delta} p_0 + \epsilon$ the number of passengers following path P_1. At station v_1, the adversary declares P_0 to be delayed and P_1 to be on time. As $(T-\delta)p_0 < \delta p_1 = (T-\delta)p_0 + \delta\epsilon$, the online algorithm will not wait. Upon the arrival of the train at v_2, the adversary also delays P_1. Then, the online algorithm will certainly wait in v_2, as it will not delay any other passengers. The optimal offline solution would already have waited at v_1, hence online algorithm started to wait after the optimum. The ratio between the two solutions is:

$$\frac{\text{Online}}{\text{Opt}} = \frac{Tp_0 + \delta p_1}{\delta(p_0 + p_1)} = 2\frac{Tp_0 + \delta\epsilon}{Tp_0 + \delta\epsilon} - \frac{\delta p_0 + \delta\epsilon}{Tp_0 + \delta\epsilon} = 2 - \frac{\delta p_0 + \delta\epsilon}{Tp_0 + \delta\epsilon} \tag{5}$$

For $\epsilon \to 0$ the ratio converges to $r_1 = 2 - \frac{\delta}{T}$. For $\frac{\delta}{T} = \epsilon'$, and by letting $\epsilon' \to 0$, we get arbitrarily close to 2. For analyzing the case where the online algorithm starts waiting too early, we consider a train line similar to the above, but with different passenger paths. Let $P_0 = \{v_1, v_2, v_3\}$, $P_1 = \{v_1, v_2\}$, $P_2 = \{v_2, v_3\}$, carrying p_0, p_1 and $p_2 = \frac{T}{\delta} p_0 - \epsilon$ passengers, respectively. Initially, the adversary shows P_0 to be delayed. The online algorithm does not wait at station v_0, as the other two paths are assumed to be on time: indeed, $\delta(p_1 + p_2) > Tp_0$, for $\delta p_1 > \delta\epsilon$. When the train arrives at station v_1, the adversary leaves P_2 on time.

But then, $Tp_0 > \delta p_2 = Tp_0 - \delta\epsilon$, hence the online algorithm starts to wait at v_2, interestingly enough, for nobody. The optimal algorithm would not have waited anywhere, and would only have dropped the path P_0. This gives the competitive ratio of

$$\frac{\text{Online}}{\text{Opt}} = \frac{Tp_0 + \delta p_2}{Tp_0} = \frac{2Tp_0 - \delta\epsilon}{Tp_0} \overset{\epsilon \to 0}{=} 2. \tag{6}$$

This case is thus independent from the ratio $\frac{\delta}{T}$, and the analysis directly shows its tightness. □

4 Competitiveness of Online Algorithms

In the following, we show two bounds on the competitive ratio for all online algorithms on the train line. First, we discuss that, if the objective accounts only for the delay which can be optimized, we cannot be better than $\frac{T}{\delta}$-competitive. This actually implies that we cannot do better than applying the trivial strategy of waiting as soon as there is a delayed entering passenger path, and to stay on time otherwise. We then analyze the objective discussed in Section 3.1 and show that no online algorithm can be better than Φ-competitive, where $\Phi = \frac{\sqrt{5}+1}{2}$ is the Golden Ratio.

4.1 Competitiveness with Additional Delay Objective

In this section, we consider the so-called additional delay objective function, which accounts only for the delay which can be optimized on the network, i.e. without the unavoidable delay $\delta \sum_i D^i$. With the previously introduced notation, the objective value occurring if the train waits at station j is defined as

$$\Delta_{\text{ADD}}(j) = (T - \delta) \sum_{i<j} D^i + \delta O^{\geq j}$$

Theorem 2. *No online algorithm on a single train line with fixed passenger paths can be better than $\frac{T}{\delta}$-competitive when minimizing only the additional delay.*

Proof. We analyze the network shown in Figure 2. The line we wish to optimize travels between stations A and C, and has an intermediate stop in B. We introduce two passenger paths, P_1 connecting to the train line in A and carrying $p_1 = 1$ passengers, P_2 connecting to the train line in B and carrying $p_2 = \frac{T(T-\delta)}{\delta^2}$ passengers. When in A, the adversary announces that the passenger path P_1 is delayed by δ time units. He can still choose if or not he will delay the passenger path P_2.

If the online algorithm decides to wait in A, the adversary leaves P_2 on time. Thus, the optimal offline policy is to stay on time to the end of the trip. The delay accumulated by the online algorithm is $\frac{T(T-\delta)}{\delta^2}\delta = \frac{T(T-\delta)}{\delta}$, the optimal strategy accumulates only $(T - \delta)$ delay. Hence, the online algorithm is $\frac{T}{\delta}$-competitive.

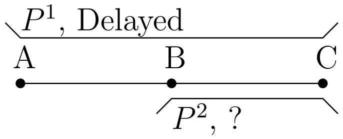

Fig. 2. The train network used for showing the non-competitiveness of the additional delay-objective

If the online algorithm decides to leave A on time, the adversary will delay P_2 as well. The optimal offline policy in this case is to wait in A, which produces a zero valued objective. The online algorithm produces an additional delay of at least T, hence the competitive ratio in this case is infinite. □

The setting for proving the $\frac{T}{\delta}$-competitiveness might seem peculiar, as in one case the train travels empty between stations A and B. This can be resolved by introducing an on time passenger path between A and C carrying just one passenger. The same setting can then be used to prove a lower bound on the competitiveness of $\frac{T}{\delta} - 1$. This is asymptotically the same as above, and in practice it does not change significantly if $\frac{T}{\delta}$ is large.

4.2 Φ-Competitiveness for Total Delay Objective

In the following section, we prove that no online algorithm can be better than Φ-competitive if we use the objective function accounting for all delays introduced in Section 3.1.

Theorem 3. *No online algorithm on a train line with fixed passenger paths can be better than Φ-competitive when minimizing the total delay, where $\Phi = \frac{\sqrt{5}+1}{2}$ is the Golden Ratio.*

Proof. We introduce a network similar to the one of the previous proof (see Figure 3). This time, we introduce three passenger paths: P_1, carrying p_1 passengers and connecting to the train line at station A; P_2, carrying p_2 passengers, starting their journey in A; P_3, carrying p_3 passengers, connecting to the train line in B. At the beginning, the adversary declares P_1 to be delayed and P_2 to be on time.

Now, an online algorithm must decide whether to wait in A or not. The situation the adversary wants to enforce is the following: whatever decision the online algorithm takes, it is c-competitive. Then, he chooses the parameters such that c is maximal. The following mathematical program describes the adversary's parameter choices:

$$\max c$$

$$\delta(p_1 + p_2 + p_3) \geq cTp_1 \tag{7}$$

$$Tp_1 + \delta(p_2 + p_3) \geq c\delta(p_1 + p_2 + p_3) \tag{8}$$

$$T(p_1 + p_3) \geq c\delta(p_1 + p_2 + p_3) \tag{9}$$

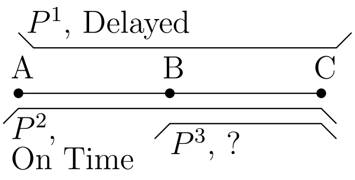

Fig. 3. The simple network used in the proof for not competitiveness of the all-delay objective

Inequality (7) reflects the situation when the online algorithm waits in A, thus delaying all three paths, but it would have been better not to wait, as P_3 was on time. The left hand side (LHS) reflects the costs of the online algorithm, the right hand side (RHS) the costs of the optimal offline solution, weighted with the competitive ratio c. Inequality (8) reflects the situation where the online algorithm does not wait in A but waits in B, as the adversary then delays P_3, and it would have been better to wait in A. Again, the LHS describes the costs of the online algorithm, the RHS the costs of the optimal delay policy weighted with the competitive ratio. Finally, inequality (9) describes the situation where the online algorithm decides not to wait at all even if P_3 delays, and it would have been better to wait in A. The LHS and RHS of the inequality describe the costs as before. For this last online policy we do not consider the case where the optimal offline strategy waits in B. Were this strategy better than waiting in A, we would only make the bound on the competitive ratio bigger than what we show here.

For simplicity, we normalize all passenger numbers with respect to p_1, and the delays with respect to δ. Hence, P_1 carries 1 passenger, and $\delta = 1$. Due to this normalization, in the following we should formally refer to the drop delay as T' and to the passenger numbers as p'_2 and p'_3. To improve readability, we omit the primes. The mathematical program becomes:

$$\max c$$

$$1 + p_2 + p_3 \geq cT \tag{10}$$

$$T + p_2 + p_3 \geq c(1 + p_2 + p_3) \tag{11}$$

$$T(1 + p_3) \geq c(1 + p_2 + p_3) \tag{12}$$

We restrict our attention to the case where $p_2 \leq (T-1)p_3$. In this case, (11) is tighter than (12), so we can omit the latter equation. As we are constructing a specific solution to the mathematical program, we let inequality (10) be tight. Note that by choosing (11) to be tight, we can construct an example with the same competitive ratio as shown below. Now we can set $p_3 = cT - 1 - p_2$. Thus, substituting into (11):

$$T + p_2 + cT - 1 - p_2 \geq c + cp_2 + c^2T - c - cp_2$$

$$T(1 + c) - 1 \geq c^2T$$

Let $c = \Phi - \epsilon$, and recall that $\Phi + 1 = \Phi^2$:

$$\begin{aligned} T(1+\Phi) - T\epsilon - 1 &\geq \Phi^2 T - 2\Phi\epsilon T + \epsilon^2 T \\ (2\Phi - 1 - \epsilon)\epsilon T &\geq 1 \end{aligned} \tag{13}$$

As long as $(2\Phi - 1 - \epsilon)\epsilon \geq 0$, we can choose T such that (13) is satisfied. The condition is satisfied for $0 < \epsilon \leq 2\Phi - 1$, and we can set $T = \frac{1}{(2\Phi-1-\epsilon)\epsilon}$. By letting $\epsilon \rightarrow 0$, we can get arbitrarily close to Φ. In all, this shows that the competitive ratio of any online algorithm cannot be better than Φ.

Notice that a closer inspection of the constructed instance shows that within the setting of this example, we cannot prove the competitive ratio to be greater than Φ, as choosing a negative ϵ leads to a contradiction. □

5 Conclusion

We considered the online delay management problem for a single train line with passengers transferring from other feeder trains. Since such a feeder train may arrive at a transfer station with an arrival delay, the connecting passengers may be delayed as well. In this online setting, the train line only knows the delays of the entering passengers at its current station, as well as at the previous stations on the line.

As such, we provided a natural next step to the research in [1], who considered the online situation of delays at a single station. We proposed a family of Ski Rental-like 2-competitive online algorithms, and presented lower bounds on the competitive ratio that hold for any online algorithm for the single train line. As we do not know of any other theoretical work on online delay management problems, our results provide a first step in the direction of online delay management for more general networks and with more realistic assumptions.

Indeed, the extension of our results to two crossing train lines, or to a railway network with a tree topology are interesting topics for further research. Other directions for future research include different arrival delays for the connecting passengers, and the inclusion of timetable buffer times.

References

1. Anderegg, L., Penna, P., Widmayer, P.: Online train disposition: to wait or not to wait? In: Wagner, D. (ed.) Electronic Notes in Theoretical Computer Science, vol. 66, Elsevier, Amsterdam (2002)
2. Biederbick, C., Suhl, L.: Improving the quality of railway dispatching tasks via agent-based simulation. In: Allan, J., Brebbia, C.A., Hill, R.J., Sciutto, G., Sone, S. (eds.) Computers in Railways IX, Proceedings of the 9th International Conference on Computer Aided Design, Manufacture and Operation in the Railway and Other Mass Transit Systems (COMPRAIL), Dresden, Germany, pp. 785–795. WIT Press, Southampton, Boston (2004)

3. Gatto, M., Glaus, B., Jacob, R., Peeters, L., Widmayer, P.: Railway delay management: Exploring its algorithmic complexity. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 199–211. Springer, Heidelberg (2004)
4. Gatto, M., Jacob, R., Peeters, L., Schöbel, A.: The computational complexity of delay management. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 227–238. Springer, Heidelberg (2005)
5. Heimburger, D., Herzenberg, A., Wilson, N.: Using simple simulation models in the operational analysis of rail transit lines: A case study of the MBTA's red line. Transportation Research Record 1677, 21–30 (1999)
6. Karlin, A., Manasse, M., Rudolph, L., Sleator, D.: Competitive snoopy caching. Algorithmica 3(1), 79–119 (1988)
7. O'Dell, S., Wilson, N.: Optimal real-time control strategies for rail transit operations during disruptions. In: Computer-Aided Transit Scheduling. Lecture Notes in Economics and Math. Sys. pp. 299–323. Springer, Heidelberg (1999)
8. Schöbel, A.: A model for the delay management problem based on mixed-integer-programming. In: Zaroliagis, C. (ed.) Electronic Notes in Theoretical Computer Science, vol. 50, Elsevier, Amsterdam (2001)
9. Schöbel, A.: Optimization in Public Transportation: Stop Location, Delay Management and Tariff Zone Design in a Public Transportation Network. In: Optimization and Its Applications, vol. 3, Springer, Heidelberg (2007)