Public Transp (2013) 5:243–266
DOI 10.1007/s12469-013-0070-z

ORIGINAL PAPER

# Online delay management on a single train line: beyond competitive analysis

**Marco Bender · Sabine Büttner · Sven O. Krumke**

Published online: 10 September 2013


**Abstract** We consider the Online Delay Management Problem (ODMP) on a network with a path topology that is served by one train. In this problem the number of delayed passengers who want to board the train is not known beforehand but revealed in an online fashion once the train arrives at the corresponding station. The goal is to decide at which station a train should wait in order to minimize the total delay of all passengers.

Competitive analysis has become one of the standard tools to evaluate the performance of algorithms in the presence of incomplete information from a theoretical point of view. The ODMP has been analyzed by means of classical competitive analysis, where one compares the output of an online algorithm with that of an optimal offline algorithm which has complete knowledge about the input data. In this paper we use different approaches to overcome the often criticized pessimism of standard competitive analysis: lookahead, comparative analysis and average-case analysis. Each of these approaches extends the classical worst-case approach of competitive analysis in different aspects. We complement these extensions by addressing the problem from the viewpoint of stochastic optimization. We discuss the theoretical benefits of the concepts and provide a case-study on real world data.

M. Bender
Institute for Numerical and Applied Mathematics, University of Goettingen, Lotzestraße 16-18, 37083 Göttingen, Germany
e-mail: m.bender@math.uni-goettingen.de

S. Büttner · S.O. Krumke (✉)
Department of Mathematics, University of Kaiserslautern, Paul-Ehrlich-Str. 14, 67663 Kaiserslautern, Germany
e-mail: krumke@mathematik.uni-kl.de

S. Büttner
e-mail: buettner@mathematik.uni-kl.de

## 1 Introduction

A central issue of managing a public transport system is how to handle unpredicted delays. In this paper we consider the question whether (and when) a train should wait for delayed passengers where we take the position of the train operator. The basic setting is that the number of passengers wishing to enter and leave the train at the stations where the train stops is known in advance. However, how many of the passengers are delayed at a station becomes known only when that station is reached. We focus on the case of a single train line. This case occurs, for instance, in a subway train line in Athens connecting the central station Syntagma to the Airport (http://www.ametro.gr/). Passengers arrive at the various stations by other means of public transportation (e.g. busses) and there is no obvious way how the train operator can take influence on the feeder lines.

The objective which we consider is to minimize the total delay of the passengers caused by (a) delayed passengers missing the train which has not waited for them and (b) on-time passengers being delayed by reaching their destinations late due to the train waiting for delayed passengers. We stress that we assume that once the train has waited, it can not catch up. This assumption is realistic if there is only a comparatively small time between stops, for instance, in urban subways.

An instance of the *Online Delay Management Problem* (ODMP) on a single train line introduced by Gatto et al. (2007a) is given by a number of stations $1, \ldots, n$ that have to be served in ascending order by a single train. Furthermore, passenger paths $P_{ij}$ carrying a fixed number $p_{ij} \geq 0$ passengers from station $i$ to $j$ are given for $1 \leq i < j \leq n$, where passengers can be either on time or have a (source) delay $\delta > 0$. If passengers are delayed, the question is whether the train should wait for them or not. We assume that the train cannot speed up in order to gain time. Hence, waiting once yields a delay of $\delta$ at all upcoming stations. If a train does not wait for delayed passengers, these passengers are dropped and they have to wait for the next train scheduled in $T \gg \delta$ time units.

Gatto et al. (2007a) used the concept of online optimization and competitive analysis to compare strategies for the ODMP. An online algorithm knows already at the first station for every station $i$ the numbers $p_{ij}$ $(j > i)$ of passengers starting their trip at station $i$. However, the numbers of delayed and on-time passengers is not known in advance. When reaching station $i$ the number $o_{ij}$ of on-time passengers starting at station $i$ is revealed, and thereby also the number of delayed passengers $d_{ij}$, since $d_{ij} = p_{ij} - o_{ij}$. If the train waits at station $k$ the total delay incurred is given by

$$D(k) = (T - \delta) \sum_{\substack{i,j:1 \leq i < k \\ i < j \leq n}} d_{ij} + \delta \sum_{\substack{i,j:1 \leq i \leq n \\ i < j \leq n}} d_{ij} + \delta \sum_{\substack{i,j:1 \leq i < j \\ k < j \leq n}} o_{ij}.$$

Here, the first term accounts for the additional delay caused by dropping delayed passengers before waiting in $k$, the second term represents the (fixed) $\delta$-delay of all previously delayed passengers and the last term is the delay caused for all initially on-time passengers who leave the train after $k$. If $k = n$, this corresponds to the train not waiting at all since no passenger path starts in station $n$. The aim of the ODMP is to find a station $k$ to wait at that minimizes $D(k)$.

A deterministic online algorithm ALG is called *c-competitive*, if for every instance $\sigma$, i.e. every setting of delayed passengers on the given passenger paths $P_{ij}$, its cost $\mathrm{ALG}(\sigma)$ satisfies:

$$\mathrm{ALG}(\sigma) \leq c\,\mathrm{OPT}(\sigma),$$

where $\mathrm{OPT}(\sigma)$ denotes the optimum offline cost for instance $\sigma$. A detailed introduction to online optimization and competitive analysis can be found in the book by Borodin and El-Yaniv (2005). In this paper we consider only deterministic online algorithms.

Competitive analysis has often been criticized for being overly pessimistic. In our situation it seems unrealistic that an algorithm (a train line operator) knows *absolutely nothing* about the number of delayed passengers at a station before it is reached. Additional information such as lookahead (i.e. information about the next $\ell \geq 2$ stations) or knowledge about the distribution of the delays should help.

The main contribution of the paper is to explore various ways to make this intuition quantifiable. Although we use different approaches which may seem only weakly connected at first glance, all of these approaches attempt to answer the same question: How much can "additional information" (such as lookahead, information about the distribution of delays etc.) help and which kind of information is most helpful? Part of our paper focusses on theoretical analysis of the problem. This is complemented by a small illustrative case-study for a subway in Athens, Greece. While some of the theoretical results are obtained only for special "delay patterns", they suggest decision strategies which we use for the case-study.

The paper is organized as follows: Our first step to go beyond standard competitive analysis is to introduce a lookahead concept for the ODMP in Sect. 2. While lookahead turns out to be essentially useless from the competitive analysis point of view, we use the comparative ratio introduced by Koutsoupias and Papadimitriou (2000) and theoretically validate the intuition that lookahead is helpful for the online player. The case-study corroborates this theoretical finding experimentally.

If one has knowledge about the delay distribution, one may ask how this can be incorporated into online algorithms. In Sect. 3 we analyze online algorithms from an average case point of view. We show how the decision made by an online algorithm changes if it does not hedge against the worst-case but against an "average worst-case" according to some distribution.

Stochastic optimization is a classical approach to deal with uncertain information (Kall and Wallace 1994; Prékopa 1995). In Sect. 4 we present a stochastic programming framework which allows to incorporate knowledge about delay distributions as well as the online feature of delay management. We show how to obtain an optimal policy.

In Sect. 5 we evaluate various strategies experimentally. Both, on random data and on the data given in the real-world case-study in Athens, the algorithms using additional information outperform the known competitive online algorithm.

### 1.1 Previous work

There exist various models and solution approaches for offline delay management, see e.g. Schöbel (2001, 2006), Heilporn et al. (2008), where mixed-integer program-

ming formulations for different objective functions are presented and also combined in a bi-criteria programming framework. An investigation of the computational complexity of delay management problems can be found in the papers by Gatto et al. (2004, 2005).

Gatto et al. (2007a) consider the problem of managing delays for the first time as an online optimization problem. They introduce the problem stated above, present a family of 2-competitive online algorithms, and prove a lower bound on the competitive ratio. Krumke et al. (2011) consider extensions to the problem. They define a 3-competitive algorithm for the case of two different source delays, and deal with a different objective function. They show that when maximizing the profit no deterministic algorithm can have bounded competitiveness, and present a randomized 2-competitive algorithm. A restricted version of the ODMP where all passenger paths end in the last station was investigated by Gatto et al. (2007b). In this setting, the online player knows that passengers on at most $k$ stations which are not known in advance are delayed. The authors connect this variant to a more general version of the Ski Rental Problem, the so-called Generalized $k$-Day Ski Rental Problem, and derive lower bounds on the competitive ratio.

Kliewer and Suhl (2011) and Bauer (2010) present various online strategies (among others priority based and ILP-based strategies) and provide simulation frameworks. The complexity of finding an optimal online policy was addressed by Berger et al. (2007) who proved this task to be PSPACE-hard. More aspects of delay management can be found in Dollevoet et al. (2011, 2012), Schachtebeck and Schöbel (2010), Suhl et al. (2001).

Standard competitive analysis has often been criticized for being overly pessimistic since it assumes a fairly weak online player. Several concepts have been introduced in the literature as a remedy. Koutsoupias and Papadimitriou suggested to use *comparative analysis* where basically two classes of online algorithms are compared relatively to each other (Koutsoupias and Papadimitriou 2000). We use this approach in Sect. 2. Another alternative is to use average-case analysis coupled with competitive analysis which was done by Fujiwara and Iwama (2002) for the Ski-rental problem. More approaches can be found in Borodin et al. (1991, 1995, 1996), Irani et al. (1992). We refer to Fiat and Woeginger (1998, Chap. 17) for a comprehensive survey.

## 2 Lookahead and comparative analysis

### 2.1 Lookahead for delay management

In the ODMP the number of delayed passengers at a station is not known in advance. We consider a *lookahead* where the information about delays at the next station(s) is known. We first show that from a competitive analysis point of view, lookahead is essentially useless for the online player. This is similar to the classical paging problem (Fiat and Woeginger 1998; Borodin and El-Yaniv 2005).

**Definition 2.1** An algorithm for the ODMP has a weak lookahead of $k$ stations if it knows at each station the number of delayed and on-time passengers for the following $k$ stations. We denote this class of algorithms by $\mathcal{L}_w(k)$.

Intuitively, lookahead gives more information to an online algorithm. However, this does not pay off in competitive analysis as in a worst case there can be sufficiently many *dummy nodes* having no outgoing passenger paths.

**Observation 2.2** *Let* $\text{ALG}' \in \mathcal{L}_w(k)$ *be a c-competitive algorithm. Then there exists an algorithm* ALG *without lookahead that is also c-competitive.*

The existence of dummy nodes is an obvious reason for the bad performance of weak lookahead in competitive analysis. Hence, we consider a modified version of the lookahead that ensures that we always obtain additional information.

**Definition 2.3** An algorithm for the ODMP has a strong lookahead of $k$ stations if it knows the number of delayed and on-time passengers at each station for the following $k$ stations that have passengers departing. We denote this class of algorithms by $\mathcal{L}_s(k)$.

**Observation 2.4** *The classes of online algorithms with different lookaheads are related to each other. By definition it holds for all* $k \geq 1$ *that*

$$\mathcal{L}(0) \subset \mathcal{L}_w(1) \subset \mathcal{L}_w(k) \subset \mathcal{L}_s(k) \subset \mathcal{L}_s(k+1) \subset \mathcal{L}(\infty),$$

*where* $\mathcal{L}(0)$ *and* $\mathcal{L}(\infty)$ *denote the class of online algorithms without lookahead, and the class of offline algorithms, respectively.*

The question arising is whether it is possible to obtain better competitiveness results for algorithms with strong lookahead. This is not the case as the following result shows.

**Theorem 2.5** *Let* $\text{ALG}' \in \mathcal{L}_s(k)$ *be a c-competitive algorithm. Then there exists for every* $\epsilon > 0$ *an algorithm* ALG *without lookahead that is* $(c+\epsilon)$*-competitive.*

*Proof* Let $I$ be an instance of the ODMP with $n$ stations. We assume that $I$ has at least one delayed passenger. Otherwise, the problem would be trivial as the online algorithm that never waits is 1-competitive.

We give a construction of an instance $I'$ by modifying $I$ as shown in Fig. 1: Multiply the number of passengers on all paths by some $\alpha \in \mathbb{N}$. Furthermore, add $k$ nodes $v_1^l, \ldots, v_k^l$ in between stations $l$ and $l+1$ for all $l \in \{1, \ldots, n-1\}$. Finally, add passenger paths $P_{l,v_1^l}$ for all $l \in \{1, 2, \ldots, n-1\}$, $P_{v_i^l, v_{i+1}^l}$ for all $i \in \{1, \ldots, k-1\}$ and $l \in \{1, \ldots, n-1\}$, and $P_{v_k^l, l+1}$ for all $l \in \{1, \ldots, n-1\}$. All of them carry one passenger and are declared to be on time.

We define ALG to work on instance $I$ as follows:

- For $\epsilon > 0$ construct $I'$ as explained above for some $\alpha$ that satisfies

$$\epsilon\alpha\delta - (\epsilon + c)(n-1)(k+1)\delta \geq 0. \tag{1}$$

- Apply $\text{ALG}'$ to instance $I'$. If $\text{ALG}'$ waits at station 1, wait at station 1. If $\text{ALG}'$ waits at any station in $\{v_1^l, \ldots, v_k^l, l\}$, wait at station $l$.

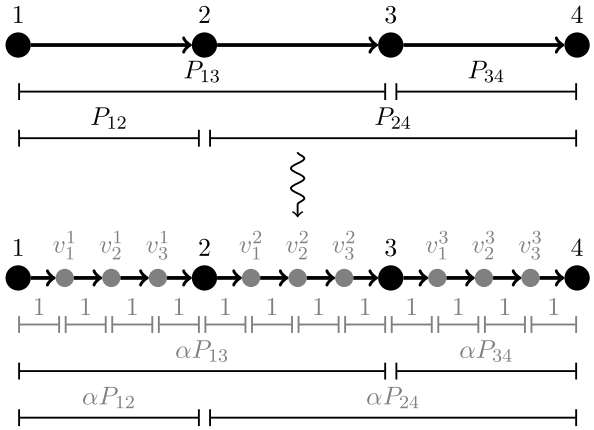



**Fig. 1** Illustration of the construction of instance $I'$ for $n = 4$ and $k = 3$ in Theorem 2.5

Observe that ALG is defined such that when it has to decide whether to wait or not, it does not use knowledge about delayed passengers at the succeeding stations in the original network $I$. Hence, we ensure that ALG does not possess a lookahead.

By construction we have

$$\text{ALG}(I) \leq \frac{1}{\alpha}\text{ALG}'(I') \tag{2}$$

and

$$\text{OPT}(I') \leq \alpha\text{OPT}(I) + (n-1)(k+1)\delta. \tag{3}$$

By the assumption that $I$ has at least one delayed passenger, there are at least $\alpha$ source delayed passengers when serving $I'$, i.e.,

$$\text{OPT}(I') \geq \alpha\delta. \tag{4}$$

Since ALG$'$ is $c$-competitive, we have

$$\begin{aligned}\text{ALG}'(I') &\leq c\ \text{OPT}(I') \\ &\overset{(1)}{\leq} c\ \text{OPT}(I') + \epsilon\alpha\delta - (\epsilon + c)(n-1)(k+1)\delta \\ &\overset{(4)}{\leq} (c+\epsilon)\big(\text{OPT}(I') - (n-1)(k+1)\delta\big).\end{aligned} \tag{5}$$

All in all we then get

$$\begin{aligned}\text{ALG}(I) &\overset{(2)}{\leq} \frac{1}{\alpha}\text{ALG}'(I') \\ &\overset{(5)}{\leq} \frac{1}{\alpha}(c+\epsilon)\big(\text{OPT}(I') - (n-1)(k+1)\delta\big) \\ &\overset{(3)}{\leq} (c+\epsilon)\text{OPT}(I).\end{aligned}$$

□

**Table 1** Lower bounds on the comparative ratio

| | |
|---|---|
| $\mathcal{R}(\mathcal{L}(0), \mathcal{L}_w(1)) \geq \frac{1+\sqrt{5}}{2} \approx 1.618$ | Gatto et al. (2007a) |
| $\mathcal{R}(\mathcal{L}(0), \mathcal{L}_w(2)) \geq 1.837$ | Krumke et al. (2011) |
| $\mathcal{R}(\mathcal{L}(0), \mathcal{L}_w(l)) \geq 2$ (for $l$ arbitrarily large) | Zeck (2011), Gatto (2007) |

## 2.2 Comparative analysis for lookahead

As we have seen, competitive analysis is too restrictive to account for the additional feature of lookahead. For a more subtle analysis of algorithms with different kinds of lookahead we make use of the concept of *comparative analysis* introduced by Koutsoupias and Papadimitriou (2000).

**Definition 2.6** For a given problem with set of instances $\Sigma$ consider two classes of algorithms $\mathcal{A}$ and $\mathcal{B}$. We call

$$\mathcal{R}(\mathcal{A}, \mathcal{B}) = \max_{B \in \mathcal{B}} \min_{A \in \mathcal{A}} \max_{\sigma \in \Sigma} \frac{A(\sigma)}{B(\sigma)} \tag{6}$$

the *comparative ratio*[1] of $\mathcal{A}$ with respect to $\mathcal{B}$. Here, $A(\sigma)$ and $B(\sigma)$ denote the costs of algorithm $A$ and $B$ on instance $\sigma$.

If we consider the case that $\mathcal{A}$ is the set of all online algorithms (without lookahead) and $\mathcal{B}$ is the set of all offline algorithms, (6) turns out to be the competitive ratio. Similar to the competitive ratio, the comparative ratio can be interpreted in a game-theoretic way: $\mathcal{B}$ tries to demonstrate its strength by choosing an algorithm $B$. $\mathcal{A}$ has to respond to this by choosing $A$. Finally, $\mathcal{B}$ selects an instance $\sigma$ such that the ratio is maximized.

When comparing the class of algorithms without and with lookahead, we have the following result.

**Theorem 2.7** *Let $L$ be a lower bound on the competitive ratio for the ODMP with $n$ stations. Then the comparative ratio of $\mathcal{L}(0)$ with respect to $\mathcal{L}_w(n-2)$ is at least $L$.*

*Proof* If we consider an instance of the ODMP with $n$ stations and an algorithm possesses a lookahead of $k = n - 2$, it can choose the same sequence as the optimal offline algorithm and serve it optimally as it possesses the same amount of information. □

Theorem 2.7 shows that lower bounds on the competitive ratio that have been derived for three, four, and arbitrary many stations, transfer to the comparative ratio in this case. Table 1 states these lower bounds using previous results cited in the right column. The $l$ in the last row is independent of the size of the instance. It simply refers

[1] In the original paper Koutsoupias and Papadimitriou (2000), the authors speak of the "comparative ratio of $\mathcal{A}$ *and* $\mathcal{B}$". We use "*with respect to*" instead to emphasize the asymmetry in the definition.

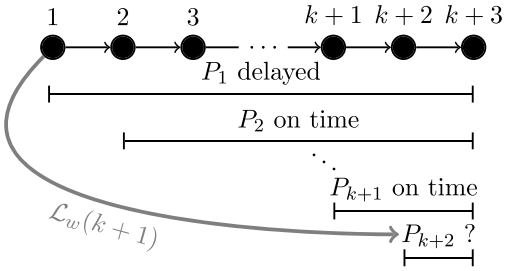



**Fig. 2** Illustration of the network used in the proof of Theorem 2.8

to the lookahead *advantage* that increases, i.e., $\mathcal{L}_w(l)$ can construct an instance with $l+1$ stations such that the comparative ratio on this instance goes to 2 as $l \to \infty$.

Note that by Observation 2.4, Theorem 2.7 also holds for the strong lookahead.

For the comparative ratio of lookahead with respect to the lookahead whose range covers one more station we have the following result.

**Theorem 2.8** *For all $k \geq 1$ the comparative ratio of $\mathcal{L}_w(k)$ with respect to $\mathcal{L}_w(k+1)$ is at least $\Phi = \frac{1+\sqrt{5}}{2}$.*

*Proof* Consider the network as shown in Fig. 2 consisting of stations $1, 2, \ldots, k+3$ and passenger paths $P_i$ connecting $i$ and $k+3$ for all $i \in \{1, \ldots, k+2\}$.

Initially, the train is at the first station. Then $P_1$ is declared to be delayed and $P_2, \ldots, P_{k+1}$ to be on time. Besides $T$ and $\delta$ (and the fixed numbers of passengers) this is the common information to both classes of algorithms. The stronger class of algorithms $\mathcal{L}_w(k+1)$ also possesses the information about the delay of the passengers on $P_{k+2}$ and, hence, can optimally choose where to wait.

In order to obtain a lower bound on the ratio between the best algorithm from each class, we do not need to consider all situations but it suffices to construct a setup along the following lines: If an algorithm from $\mathcal{L}_w(k)$ waits at station 1, $P_{k+2}$ is declared to be on time and an algorithm from $\mathcal{L}_w(k+1)$ would have chosen not to wait at all. Else, $P_{k+2}$ is declared to be delayed and the optimal strategy would have been to wait at station 1.

This situation can be summarized in a nonlinear program:

$$
\begin{aligned}
\max \quad & c \\
\text{s.t.} \quad & \delta\left(p_1 + \sum_{i=2}^{k+1} p_i + p_{k+2}\right) && \geq cTp_1 \\
& Tp_1 + \delta\left(\sum_{i=2}^{k+1} p_i + p_{k+2}\right) && \geq c\delta\left(p_1 + \sum_{i=2}^{k+1} p_i + p_{k+2}\right) \\
& T(p_1 + p_{k+2}) && \geq c\delta\left(p_1 + \sum_{i=2}^{k+1} p_i + p_{k+2}\right) \\
& c && \geq 1 \\
& T, \delta, p_1, \ldots, p_{k+2} && \geq 0.
\end{aligned}
$$

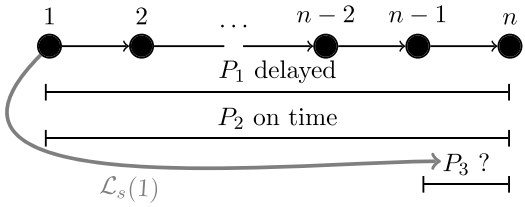



**Fig. 3** Illustration of the network used in the proof of Theorem 2.9

The aim is to find an instance (that is specified by the variables $T$, $\delta$, $p_1, \ldots, p_{k+2}$ such that the minimum ratio $c$ of any algorithm from $\mathcal{L}_w(k)$ and an optimum solution which is provided by the best algorithm in $\mathcal{L}_w(k+1)$ is as large as possible.

The first constraint in the above program is for the case that the algorithm from $\mathcal{L}_w(k)$ waits at station 1. The l.h.s. states the algorithm's delay in this case, whereas the r.h.s. corresponds to the optimal delay for this case. The second constraint corresponds similarly to the delay in the case of waiting at stations $2, 3, , \ldots, k+1$, or $k+2$. The last constraint is for not waiting at all.

Analogously to the analysis of Gatto et al. (2007a), one can show that the program is feasible for $c$ arbitrarily close to $\Phi$. Hence, we have shown that there exists an instance such that for any possible $\mathcal{L}_w(k)$-algorithm the ratio is at least $\Phi$. □

Observe that the above result even holds when comparing arbitrary lookahead variants since the considered instance does not have any dummy nodes and therefore strong and weak lookahead do not differ.

Last, we compare the weak lookahead with arbitrary range and the strong lookahead having a range of only one station. The following result shows that the type of lookahead is more important than the range.

**Theorem 2.9** *For all $k \geq 1$ the comparative ratio of $\mathcal{L}_w(k)$ with respect to $\mathcal{L}_s(1)$ is at least $\Phi = \frac{1+\sqrt{5}}{2}$.*

*Proof* Consider the following network consisting of $n = k+3$ stations and passenger paths $P_1$ and $P_2$ connecting 1 and $n$, and $P_3$ connecting $n-1$ and $n$ as shown in Fig. 3. Observe that stations $2, \ldots, n-2$ are dummy nodes.

$\mathcal{L}_s(1)$ possesses all information at the first station and can hence choose optimally where to wait.

$P_1$ is declared to be delayed and $P_2$ to be on time. The algorithm with weak lookahead has to decide whether to wait or not. If it decides to wait at station 1, $P_3$ will be declared on time and the optimal strategy would have been not to wait at all. Else, $P_3$ will be declared delayed and waiting at 1 would have been optimal.

This situation can once again be written as a nonlinear program:

$$\begin{array}{lll} \max & c & \\ \text{s.t.} & \delta(p_1+p_2+p_3) & \geq cTp_1 \\ & Tp_1+\delta(p_2+p_3) & \geq c\delta(p_1+p_2+p_3) \\ & T(p_1+p_3) & \geq c\delta(p_1+p_2+p_3) \\ & c & \geq 1 \\ & T, \delta, p_1, p_2, p_3 & \geq 0. \end{array}$$

Again, by a similar argument as above and the previous work by Gatto et al. (2007a) this program yields the desired bound. □

Gatto et al. (2007a) present a family of 2-competitive algorithms for the ODMP. Observation 2.4 and Theorem 2.5 therefore immediately yield an upper bound for the comparative ratios.

**Corollary 2.10** *The comparative ratios considered in Theorems* 2.7, 2.8, *and* 2.9 *are at most* 2.

## 3 Average-case (competitive) analysis

If there is some knowledge about the input distribution, a classical approach is to use *average-case analysis* and estimate the expected cost $\mathbb{E}_\xi[\text{ALG}(\xi)]$ of an algorithm. A related measure to overcome the pessimism of competitive analysis was introduced by Fujiwara and Iwama (2002) connecting the concepts of average-case and competitive analysis, the *average-case competitive analysis*. The average-competitive ratio of an online algorithm ALG is defined as

$$c(\text{ALG}) := \mathbb{E}_\xi\left[\frac{\text{ALG}(\xi)}{\text{OPT}(\xi)}\right].$$

From a game-theoretic perspective the adversary can still make use of the optimal offline algorithm but has to follow a distribution that is known to the online player when generating requests.

### 3.1 Analysis for a special case of the ODMP

We analyze a special case of the ODMP where we assume that there exists a *breakpoint t* (the exact position of the breakpoint should not be known to the online algorithm in advance) indicating that all passengers are *delayed* up to, and *on time* after this point. The motivation for studying this setting is derived from competitive analysis, where this transition from delaying passengers to declaring them to be on time is exactly what the adversary does as soon as an online algorithm waits.

We consider the case that the position of the breakpoint is drawn uniformly from the interval $[0, n]$, i.e., its p.d.f. is $f(t) = \frac{1}{n}1_{[0,n]}(t)$. Furthermore, we assume that

at each station one passenger embarks the train and travels to the terminal station, i.e., all paths are of the form $P_{in}$ with $p_{in} = 1$. Note that we assume in this section for the sake of better readability that the first station is 0.

These assumptions help to keep the analysis comparatively clean, and to be able to draw qualitative conclusions.

*3.1.1 Average-case analysis*

The delay for the algorithm waiting at station $k$ if the breakpoint is at $t$ is given by

$$\mathrm{ALG}_k(t) = \begin{cases} Tk + \delta(n-k) & \text{if } k < t \\ T\lceil t \rceil & \text{if } k \geq t. \end{cases}$$

The average costs of waiting at station $k$ is then

$$\begin{aligned} m(k) := \int_0^n \mathrm{ALG}_k(t) f(t) dt &= \int_0^k T\lceil t \rceil f(t) dt + \int_k^n \big(Tk + \delta(n-k)\big) f(t) dt \\ &= \frac{T}{n} \sum_{j=1}^{k} j + \frac{(n-k)(Tk + \delta(n-k))}{n} \\ &= \frac{Tk(k+1)}{2n} + \frac{(n-k)(Tk + \delta(n-k))}{n}. \end{aligned}$$

Note that $m$ is a quadratic function in $k$, and we can compute the first and second order derivatives:

$$\begin{aligned} \frac{d}{dk}\big(m(k)\big) &= (T - 2\delta)\left(1 - \frac{k}{n}\right) + \frac{T}{2n}, \\ \frac{d^2}{(dk)^2}\big(m(k)\big) &= -\frac{T - 2\delta}{n}. \end{aligned}$$

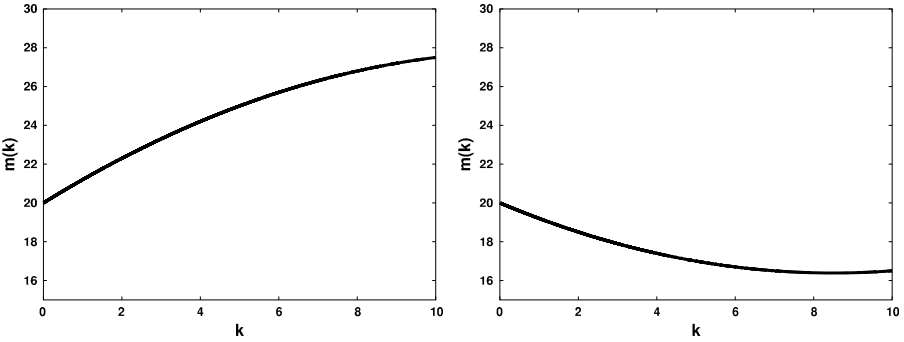
If $T \geq 2\delta$, it holds that $\frac{d}{dk}(m(k)) > 0$ for all $k \in [0, n]$, and therefore $m$ is strictly increasing on $[0, n]$. Hence, $m$ attains the minimum on $[0, n]$ for $k = 0$.

If $T < 2\delta$, we have that $\frac{d^2}{(dk)^2}(m(k)) > 0$ for all $k \in [0, n]$. In particular, the stationary point $k^* = \frac{T}{2(T-2\delta)} + n$ is the global minimum of $m$ on the real line. Furthermore, an easy computation shows that $k^* \in [0, n]$, and hence $k^*$ is the minimum on the interval $[0, n]$. Figure 4 shows an illustration of the behavior.

The results can be interpreted in the following way:

If $T \geq 2\delta$, the analysis shows that waiting at the first station leads to the minimum expected delay. This seems reasonable as dropping delayed passengers incurs a delay of $T$ per delayed passenger. Having a comparatively large $T$ therefore indicates that one should avoid this.

If $T < 2\delta$, the analysis suggests to wait at station $k^*$ (which we possibly need to round up or down if $k^*$ is not an integer). For $\delta$ being close to $T$, $k^*$ is close to $n$.




(a) $T = 5, \delta = 2$ (b) $T = 3, \delta = 2$

**Fig. 4** The average costs $m$ for $n = 10$ stations for different choices of $T$ and $\delta$

Overall, the analysis suggests to rather wait at one of the later stations. Similarly to the case before, this makes sense since a comparatively large $\delta$ corresponds to a large delay if one delays on-time passengers.

### *3.1.2 Average-case competitive analysis*

We continue to analyze the special case by means of average-case competitive analysis. To this end, we need to consider the optimal offline algorithm that knows the breakpoint $t$ in advance and can decide whether to wait at the first station or not to wait at all, yielding a total delay of

$$\text{OPT}(t) = \min\{T\lceil t\rceil, \delta n\} = \begin{cases} T\lceil t\rceil & \text{if } \lceil t\rceil < \frac{\delta n}{T} \\ \delta n & \text{if } \lceil t\rceil \geq \frac{\delta n}{T}. \end{cases}$$

The average-competitive ratio of $\text{ALG}_k$ is therefore given by

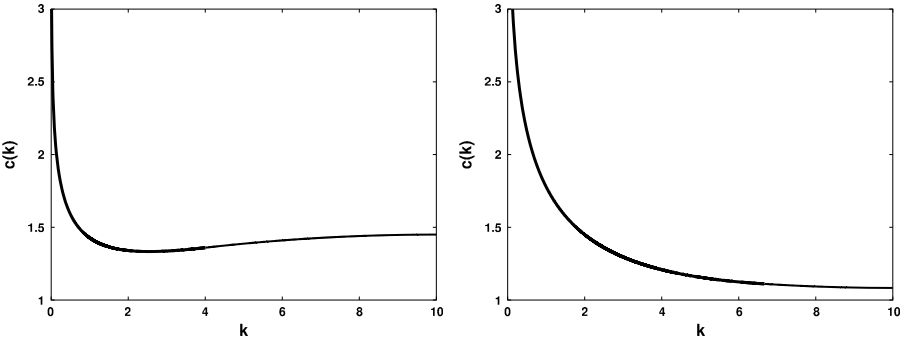
$$c(k) = \int_0^n \frac{\text{ALG}_k(t)}{\text{OPT}(t)} f(t)dt = \int_0^k \frac{T\lceil t\rceil}{\text{OPT}(t)} f(t)dt + \int_k^n \frac{Tk + \delta(n-k)}{\text{OPT}(t)} f(t)dt.$$

To be able to actually state the integrals properly, we have to distinguish two cases, and consider the following piecewise definition of $c(k)$:

$$c(k) = \begin{cases} c_1(k) & \text{if } k < \lceil t\rceil \\ c_2(k) & \text{if } k \geq \lceil t\rceil, \end{cases}$$

where

$$c_1(k) = \int_0^k \frac{T\lceil t\rceil}{T\lceil t\rceil} f(t)dt + \int_k^{\lfloor \frac{\delta n}{T} \rfloor} \frac{Tk + \delta(n-k)}{T\lceil t\rceil} f(t)dt + \int_{\lfloor \frac{\delta n}{T} \rfloor}^n \frac{Tk + \delta(n-k)}{\delta n} f(t)dt,$$




(a) $T = 5, \delta = 2$ (b) $T = 3, \delta = 2$

**Fig. 5** The average-competitive ratio $c$ for $n = 10$ stations and different choices of $T$ and $\delta$

$$c_2(k) = \int_0^{\lfloor \frac{\delta n}{T} \rfloor} \frac{T\lceil t \rceil}{T\lceil t \rceil} f(t)dt + \int_{\lfloor \frac{\delta n}{T} \rfloor}^{k} \frac{T\lceil t \rceil}{\delta n} f(t)dt + \int_k^n \frac{Tk + \delta(n-k)}{\delta n} f(t)dt.$$

Our aim is to find $k$ such that $c(k)$ is minimized. However, the analysis of $c$ is not possible in such a straightforward way as it was possible for $m$ before. The complicated structure of the functions, due to e.g., the second integral of $c_1$, forbids solving the minimization problem analytically or obtaining general statements on the monotonic behaviour. We resort to a numerical analysis of $c$ to locate minima by means of constrained nonlinear programming. Thereby, we can solve the minimization problem numerically for any given set of input parameters $T$, $\delta$, and $n$, and obtain qualitative results. Nevertheless, we observe that the location of the minimum of $c$ depends as expected on the relation between $T$ and $\delta$:

If $T < 2\delta$, average-case competitive analysis suggests to wait at the last station $n$. This is similar to the average-case analysis before that waited at $k^*$ (which was close to $n$). Again, we can argue that waiting at one of the earlier stations would incur a comparatively large delay of $\delta$ to each of the on-time passengers.

If $T \geq 2\delta$, the station chosen by the average-case competitive analysis is one of the earlier stations, which is similar to the average-case analysis. The intuition behind this is that dropping delayed passengers incurs for each of them a delay of $T$, which is large compared to $\delta$. However, the minimum is not attained for 0. This can be seen as a tendency of the measure to incorporate worst case behaviour. If it waits at the first station, the offline algorithm will have significantly less delay when the breakpoint is located slightly after this station. In this case the ratio of algorithm and optimum would therefore be large. Figure 5 illustrates the situation.

This shows how average-case competitive analysis tries to balance between these two opposing strategies.

This two-face nature of average-case competitive analysis that uses knowledge about distributions as well as hedging against worst cases can also be seen in Fig. 6.

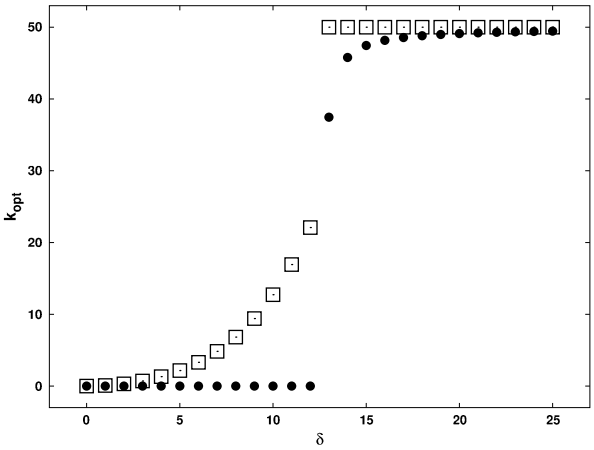



**Fig. 6** The optimal waiting strategy $k_{opt}$ for average-case analysis (*dots*) and average-case competitive analysis (*squares*) as a function of $\delta$ for $n = 50$ stations and $T = 25$

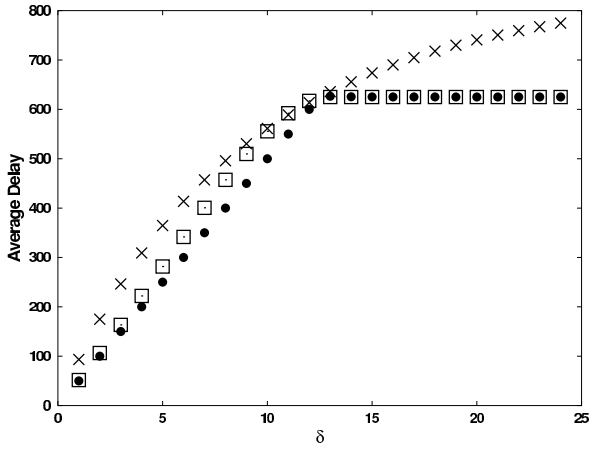



**Fig. 7** The average delay for waiting strategies suggested by average-case analysis (*dots*), average-case competitive analysis (*squares*), and the balancing algorithm (*crosses*) as a function of $\delta$ for $n = 50$ stations and $T = 25$

### *3.1.3 Comparison*

We now present a comparison of the three strategies for our special passenger path and delay structure: the strategy from Sect. 3.1.1 suggested by average-case analysis, the one from Sect. 3.1.2 obtained from average-case competitive analysis, and the 2-competitive balancing strategy of Gatto et al. (2007a).

Figure 7 shows the analytical average delay for each of the strategies assuming that the breakpoint is uniformly distributed. Here, only the optimal average-case competitive strategy had to be computed approximately in each step by means of a numerical analysis. By definition, average-case analysis yields the best results and it is not surprising that the 2-competitive balancing strategy is the worst. The values of the average-case competitive strategy lie in between the two others.

To analyze how the strategies perform in a worst case, we compared for different choices of $T$, $\delta$, and $n$ the maximum ratio between each of the strategies and the optimal offline algorithm. The results are shown in Fig. 8, and represent the theoretically derived worst-cases. By construction, the 2-competitive balancing strategy is

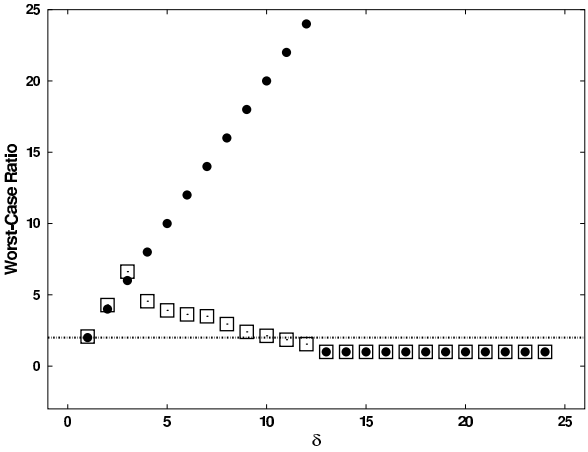



**Fig. 8** The maximum ratio of strategies suggested by average-case analysis (*dots*) and average-case competitive analysis (*squares*) to the optimal offline algorithm as a function of $\delta$ for $n = 50$ stations and $T = 25$ for waiting strategies. The *dashed line* indicates the competitive ratio of 2 that is obtained by the balancing algorithm (Gatto et al. 2007a)

never worse than twice the optimum, but often even better. The performance of the average-case strategy is poor in many cases. The worst-case ratios of the average-case competitive strategy lie again in between.

## 4 A stochastic programming framework

Stochastic Programming is a widely used framework when analyzing optimization problems that contain uncertainties. Its aim is to find a feasible policy that is optimal w.r.t. the current stage costs plus the (discounted) expected future costs. Although there are in general a lot of possible scenarios, in our problem we do not have to solve mathematical programs for each of the subproblems here, as one has to do in most cases in literature as well as applications. We make use of the nice structure of our problem and present a tailor-made solution for obtaining an optimal policy.

### 4.1 Setting of the stochastic program

A stage in our stochastic program corresponds to a station in the train network. The random variable $\xi_k$ in stage $k$ corresponds to the $(n-k+1)$-dimensional vector of delayed passengers departing from station $k$, i.e., $\xi_k = (\xi_{k,k+1}, \ldots, \xi_{kn})$. Here, $\xi_{kj}$ is the random variable of delayed passengers that want to board the train at station $k$ and travel to station $j$. We define the number of delayed passengers $d_{kl}(\xi_{kl})$ travelling from $k$ to $l$ to be a discrete random variable with values in $\{0, 1, \ldots, p_{kl}\}$. As stated before, $p_{kl}$ is assumed to be known. Hence, the number of on-time passengers is given by $o_{kl}(\xi_{kl}) = p_{kl} - d_{kl}(\xi_{kl})$.

We introduce for each station $k \in \{1, \ldots, n-1\}$ a decision variable $x_k \in \mathcal{X}_k$ which represents the waiting decision. The values allowed to be taken only depend on the previous stage decision and we define $\mathcal{X}_1 = \{\lambda, 0, 1\}$ and for $k \geq 2$

$$\mathcal{X}_k(x_{k-1}, \xi_k) = \begin{cases} \{0, 1\} & \text{if } x_{k-1} = 0 \\ \{\lambda\} & \text{if } x_{k-1} \in \{1, \lambda\}. \end{cases}$$

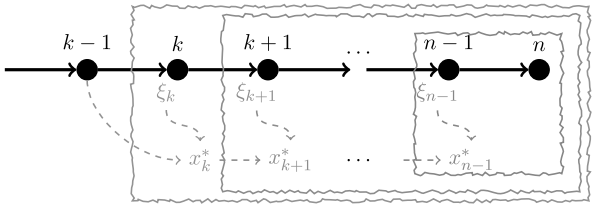



**Fig. 9** Illustration of iterative solution process for the stochastic program

If the decision variable takes the value 0, this means that we do not wait at the corresponding station, 1 represents waiting, and $\lambda \notin \{0, 1\}$ is for the case that we already waited at some station before and therefore no decision has to be taken. Depending on the waiting decision and the outcome of the random variable, costs in stage $k$ occur:

$$f_k(x_k, \xi_k) = \begin{cases} T \sum_{j:k<j\leq n} d_{kj}(\xi_{kj}) & \text{if } x_k = 0 \\ \delta(\sum_{i,j:k\leq i<j\leq n} p_{ij} + \sum_{i,j:1\leq i<k<j\leq n} o_{ij}(\xi_{ij})) & \text{if } x_k = 1 \\ 0 & \text{if } x_k = \lambda. \end{cases}$$

If we decide not to wait, all passengers at the current station are dropped and they have to wait for the next train scheduled in $T$ time units. If we wait, all on-time passengers currently on the train and all future on-time passengers are delayed by $\delta$. In case that we waited at some station before, no costs occur.

## 4.2 Obtaining an optimal policy

Our aim is then to find an optimal waiting policy, i.e., solve the $(n-1)$-stage stochastic program

$$\min_{x_1 \in \mathcal{X}_1(\xi_1)} f_1(x_1) + \mathbb{E}_{\xi_2}\Big[\min_{x_2 \in \mathcal{X}_2(x_1,\xi_2)} f_2(x_2, \xi_2) + \mathbb{E}_{\xi_3}\Big[\cdots + \mathbb{E}_{\xi_{n-1}} \times \Big[\min_{x_{n-1} \in \mathcal{X}_{n-1}(x_{n-2},\xi_{n-1})} f_{n-1}(x_{n-1}, \xi_{n-1})\Big]\cdots\Big]\Big].$$

First, we consider the last-stage problem

$$Q_{n-1}(x_{n-2}, \xi_{n-1}) = \min_{x_{n-1} \in \mathcal{X}_{n-1}(x_{n-2},\xi_{n-1})} f_{n-1}(x_{n-1}, \xi_{n-1})$$

which has optimal solution

$$x_{n-1}^*(x_{n-2}, \xi_{n-1}) = \begin{cases} 0 & \text{if } x_{n-2} = 0 \text{ and } f_{n-1}(0, \xi_{n-1}) < f_{n-1}(1, \xi_{n-1}) \\ 1 & \text{if } x_{n-2} = 0 \text{ and } f_{n-1}(0, \xi_{n-1}) > f_{n-1}(1, \xi_{n-1}) \\ \lambda & \text{if } x_{n-2} \in \{1, \lambda\}. \end{cases}$$

This solution can be interpreted as the optimal strategy when reaching station $n-1$, and is then passed on to the second-to-last stage problem to obtain an optimal waiting policy for the last three stations. This process can then be iterated as shown in Fig. 9.

In stage $k$ we already have computed $x^*_{k+1}, \ldots, x^*_{n-1}$ and need to solve the problem

$$Q_k(x_{k-1}, \xi_k) = \min_{x_k \in \mathcal{X}_k(x_{k-1}, \xi_k)} f_k(x_k, \xi_k) + \mathbb{E}_{\xi_{k+1}}\big[Q_{k+1}(x_k, \xi_{k+1})\big]$$

of minimizing the costs in stage $k$ plus the expected costs to go in the following states.

If $x_k \in \{1, \lambda\}$, no costs occur at the following stations and the expectation in the minimization problem above is zero. The more interesting case is $x_k = 0$, where we have

$$\mathbb{E}_{\xi_{k+1}}\big[Q_{k+1}(0, \xi_{k+1})\big] = \mathbb{E}_{\xi_{k+1}}\big[f_{k+1}\big(x^*_{k+1}, \xi_{k+1}\big) + \mathbb{E}_{\xi_{k+2}}\big[Q_{k+2}\big(x^*_{k+1}, \xi_{k+2}\big)\big]\big]. \quad (7)$$

If after the realization of $\xi_{k+1}$, $x^*_{k+1}(\xi_{k+1}) = 1$ is chosen, we have expected costs of

$$\underbrace{\delta\Bigg(\sum_{i,j:k+1\le i<j\le n} p_{ij} + \sum_{i,j:1\le i<k+1<j\le n} o_{ij}(\xi_{ij})\Bigg)}_{=f_{k+1}(1,\xi_{k+1})} + \underbrace{\mathbb{E}_{\xi_{k+2}}\big[Q_{k+2}(1, \xi_{k+2})\big]}_{=0}. \quad (8)$$

If $x^*_{k+1}(\xi_{k+1}) = 0$ is chosen, the expected costs are

$$\underbrace{T \sum_{j:k+1<j\le n} d_{k+1,j}(\xi_{k+1,j})}_{=f_{k+1}(0,\xi_{k+1})} + \underbrace{\mathbb{E}_{\xi_{k+2}}\big[Q_{k+2}(0, \xi_{k+2})\big]}_{=:\mathcal{A}_{k+2}}, \quad (9)$$

where $\mathcal{A}_{k+2}$ is known from the iteration before. Hence, the expression in (7) becomes:

$$\pi_1 \delta\Bigg(\sum_{i,j:k+1\le i<j\le n} p_{ij} + \sum_{i,j:1\le i<k+1<j\le n} o_{ij}(\xi_{ij})\Bigg) + \pi_2\Bigg(T\mathbb{E}_{\xi_{k+1}}\Bigg[\sum_{j:k+1<j\le n} d_{k+1,j}(\xi_{k+1,j}) \,|\, (8) > (9)\Bigg] + \mathcal{A}_{k+2}\Bigg), \quad (10)$$

where $\pi_1 = \Pr((8) \le (9))$ and $\pi_2 = \Pr((8) > (9))$.

Observe that we do not need a conditional expectation for the first summand in (10) as the numbers of the corresponding on-time passengers are already known at this stage. Combining what we just obtained, the strategy

$$x^*_k(x_{k-1}, \xi_k) = \begin{cases} 0 & \text{if } x_{k-1} = 0 \text{ and } f_k(0, \xi_k) + (10) < f_k(1, \xi_k) \\ 1 & \text{if } x_{k-1} = 0 \text{ and } f_k(0, \xi_k) + (10) > f_k(1, \xi_k) \\ \lambda & \text{if } x_{k-1} \in \{1, \lambda\} \end{cases}$$

is optimal for the problem in stage $k$. This yields, together with the previously obtained strategies, a policy $(x^*_k, x^*_{k+1}, \ldots, x^*_{n-1})$ that minimizes the expected costs after reaching station $k$.

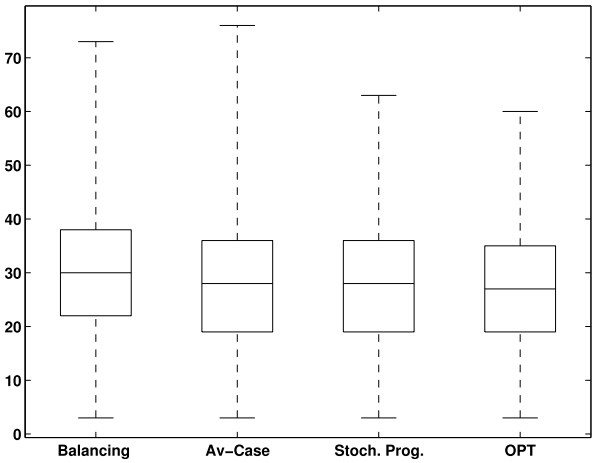



**Fig. 10** Boxplots for the average delay of the different strategies

This procedure is continued until we reach the first station, i.e., $k = 1$, and we obtain an optimal waiting policy $(x_1^*, \ldots, x_{n-1}^*)$, saying at each station whether it is better to wait or to continue on time.

For the computation of the conditional expectation in (10) we can use the convolution to obtain the distribution of a sum of random variables for the case of independent random variables. Otherwise, if no closed form is at hand, one has to perform sampling and use the empirical distribution function.

### 4.3 Numerical evaluation

In the following we show how the stochastic programming strategy performs on randomly generated data. For small networks of three stations, we compare stochastic programming, average-case analysis, and the balancing algorithm (Gatto et al. 2007a). We generated 10,000 instances where the parameters for each instance were chosen with a uniform distribution: $p_{ij}$ from $\{0, \ldots, 20\}$, $d_{ij}$ from $\{0, \ldots, p_{ij}\}$, and $T$ from $\{1, \ldots, 5\}$.

We obtained the boxplots shown in Fig. 10 for the average delays. Although this yields a first insight into the performance of the strategies, we cannot draw any statistical conclusions from it. Therefore, we analyzed the number of cases where the strategies differed. In fact, the stochastic programming strategy and the balancing strategy suggested the same strategy for 69.21 % of the cases. Average-case analysis and stochastic programming coincided even on 93.36 % of the instances.

Hence, we focused only on those instances where the delays of the strategies differ. We computed the difference of stochastic programming and each of the respective strategies. Negative values therefore correspond to instances where stochastic programming is superior.

Figure 11 shows that stochastic programming outperforms the 2-competitive balancing strategy by far. Figure 12 shows the difference between stochastic programming and average-case analysis. Although the difference is not as significant as in the previous case, the observations indicate that stochastic programming is slightly superior.

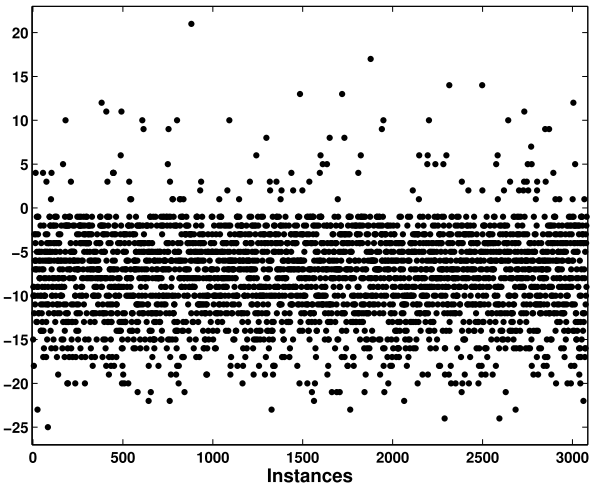



**Fig. 11** Difference of the delays of stochastic programming and balancing algorithm (the instances where the delays coincide have been left out)

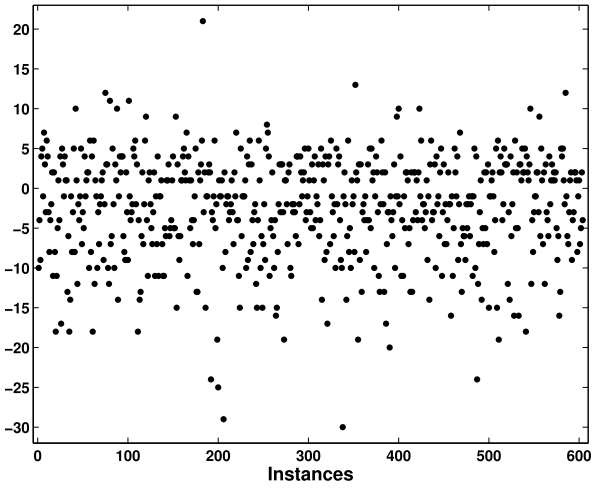



**Fig. 12** Difference of the delays of stochastic programming and average-case analysis (the instances where the delays coincide have been left out)

## 5 Experimental case study

In this section, we compare the concepts introduced before on a real world data set taken from the Athens Metro as shown in Fig. 13. We consider the single train line connecting the station Syntagma in the center of Athens with the airport and a total of 13 stations.

The number of passengers for all origin-destination pairs was taken from the LinTim software package http://lintim.math.uni-goettingen.de/. In the following, we apply and compare different concepts for the ODMP introduced before:

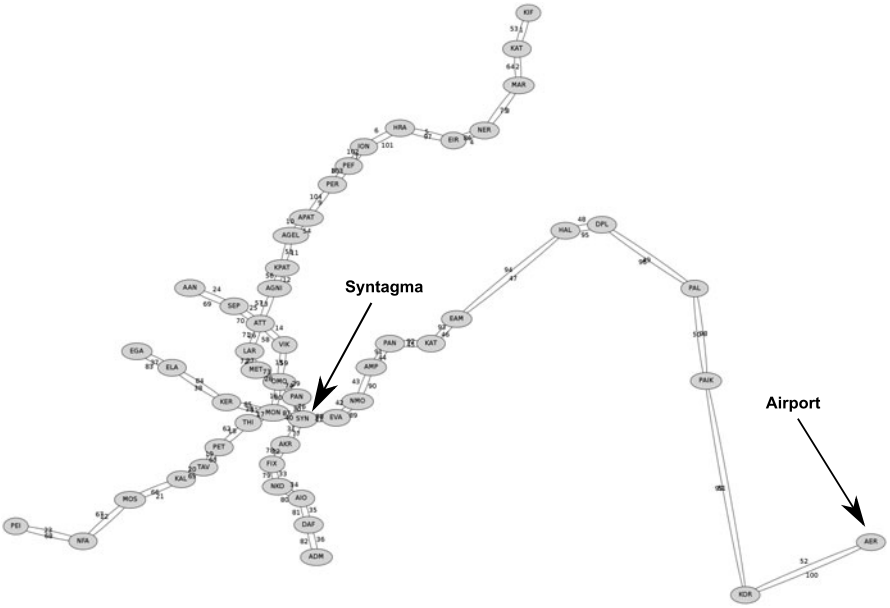



**Fig. 13** Illustration of the Athens Metro http://lintim.math.uni-goettingen.de/. The station Syntagma (SYN) is located in the center of Athens, the airport (AER) in the south-east

- the 2-competitive balancing algorithm from Gatto et al. (2007a): This algorithm fixes a parameter $t \in [T - \delta, T]$ and decides to wait at the first station $k$, where

$$t \sum_{\substack{i,j:1 \le i < k \\ i < j \le n}} d_{ij} \ge \delta \left( \sum_{i,j:1 \le i < k < j \le n} o_{ij} + \sum_{i,j:k \le i < j \le n} p_{ij} \right). \tag{11}$$

  (If there is no such $k$, then the algorithm does not wait at all.) Equation (11) states that the total delay caused to delayed passengers missing the train before $k$ is balanced with the potential delay for all future and on-board passengers which are or might be on time.
- two variants of algorithms using a lookahead of $l$ stations as introduced in Sect. 2:

  The first variant, the *dynamic lookahead* version, uses its knowledge about the delays at the next $l$ stations and takes these actual numbers into consideration in a balancing argument. It waits at station $k \le n - l - 1$, if

$$t \sum_{\substack{i,j:1 \le i < k \\ i < j \le n}} d_{ij} \ge \delta \left( \sum_{i,j:1 \le i < k+l < j \le n} o_{ij} + \sum_{i,j:k+l \le i < j \le n} p_{ij} \right). \tag{12}$$

  Once it reaches station $n - l - 1$ and has not waited so far, it possesses all information about future delays and chooses optimally for the remaining stations. Hence, it can be seen as a generalization of the balancing algorithm.

  The *static lookahead* algorithm is a naïve version of the dynamic algorithm. It actually ignores its additional information until it has reached station $n - l - 1$.

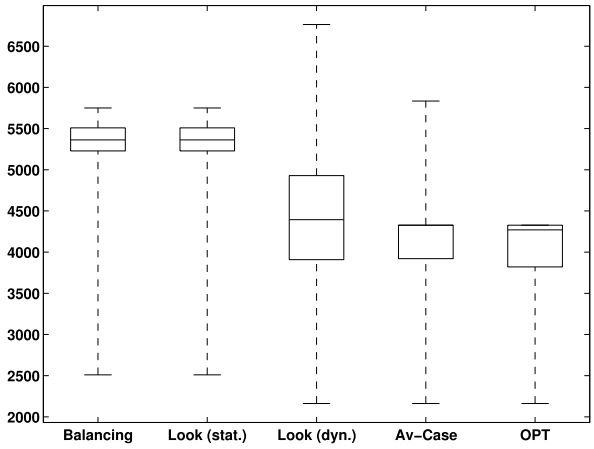



**Fig. 14** Boxplots for the average delays for $\delta = 1$, $T = 2$, and 100,000 runs

Until this station, it uses the algorithm from Gatto et al. (2007a). At station $n - l - 1$, it behaves like the dynamic lookahead algorithm. Note that the naïve static lookahead algorithm is the natural generalization to a lookahead algorithm as it is by construction never worse than the balancing algorithm, and optimal on networks with up to $l + 2$ stations.

The train line in our case study contained only a total of 13 stations, so we limited the lookahead in both the dynamic and the static version to $l = 1$.

- the algorithm from Sect. 3.1.1 which chooses a station $k^*$ according to the results from the average-case analysis. Note that the analysis in Sect. 3.1.1 was carried out only for a special pattern of the delays and from a theoretical point of view. However, from a practical point of view, $k^*$ can still be used on arbitrary data.
- the optimum offline algorithm OPT which, given all input data, chooses a station $k$ to wait at such that $D(k)$ is minimized.

We did not evaluate the stochastic programming approach of Sect. 4 for the case study. The computation of the conditional expectations is complex and too time consuming for a larger number of stations. The average-case competitive algorithm from Sect. 3.1.2 behaved slightly worse than the algorithm using the average-case solution (see also Sect. 3.1.3) and thus is also omitted in the following comparison.

Due to lack of the actual delay distributions, in our case study we assume that the number of delayed passengers $d_{ij}$ travelling from station $i$ to $j$ is a uniformly distributed random variable with values in $\{0, 1, \ldots, p_{ij}\}$ and independent of the other variables. For this setting we generated 100,000 instances for every different choices of the parameters $\delta$ and $T$. Figure 14 shows boxplots for the average delays of each of the strategies for parameters $\delta = 1$ and $T = 2$. Figure 15 shows the associated worst-case ratios, i.e., the maximum that was observed over all instances for the ratio of each of the strategies and the optimal offline algorithm.

We observe that the *static lookahead* algorithm that uses its additional information only when reaching station $n - 2$ does not help to improve over the balancing algorithm from Gatto et al. (2007a). This can be explained by the special structure of the passenger paths. There are many passengers boarding the train at one of the first stations, and therefore the main delay is caused by passengers at these stations.

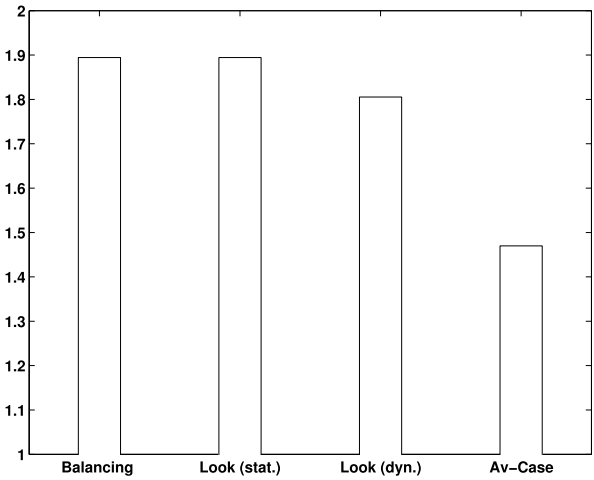



**Fig. 15** Maximum ratio that was observed for each of the strategies compared to the optimal solution

The algorithm waits in most cases "too early", since the *break-even point* of balancing is reached before it can make use of the lookahead. In contrast, *dynamic lookahead* algorithm takes its additional lookahead knowledge into consideration at every station. Although this does not always guarantee to outperform the balancing strategy (as can be seen in Fig. 14, there are actually instances where the dynamic lookahead performed worse than all other strategies), it is more suitable for taking the structure of the passenger paths into account and obtains better results overall.

The best strategy (except for the optimum offline strategy, of course) in the real world example is to follow the decisions motivated by the *average-case analysis*. In our setting, it chooses to wait at the first station which coincides with the optimal offline strategy in almost all cases.

The worst-case ratios of the different strategies have the same order of magnitude. The average-case analysis algorithm provides a decent worst-case behaviour on the real data. Recall that we saw in Sect. 3 that it might be significantly worse than the other strategies. This behavior cannot be observed in the case study. The ratio of the balancing algorithm is fairly close to its theoretical competitive ratio of 2.

We note that we could observe qualitatively similar results for other choices of the parameters.

## 6 Conclusion

We have studied extensions and alternatives to competitive analysis for the online delay management problem on a single train line (ODMP). The results indicate that competitive analysis is, in fact, overly pessimistic. The balancing algorithm from Gatto et al. (2007a) is tailored for the worst-case and exhibits the same performance on real data. Additional information such as lookahead or knowledge about the distribution of the input can help an algorithm both theoretically and in practice. In order to obtain a theoretical quantification of the improvement, one has to deviate from standard competitive analysis: we used comparative analysis and two versions of

average-case analysis in order to prove the actual benefit. This improvement also shows in experimental results both on artificially generated instances and on those on the basis of real-world data.

Our work raises a couple of challenging questions: Can analogous results be derived for more complex delay management problems? Can the theoretical results (in particular average-case analysis) be extended to more general distributions of the input? Finally, although we presented a stochastic programming framework for computing an optimum policy, at the current stage, the effort to implement this policy for a larger number of stations is excessive. Thus, it would be interesting to see if one could develop efficient means to compute those policies, maybe using the special structure of a given real-world application.

**Acknowledgement** We thank the anonymous referees for making suggestions to make the paper more consistent and easier to read. In particular, the comments about the greater outline were very helpful. We also thank the LinTim team for providing the Athens data.

## References

Bauer R (2010) Theory and engineering for shortest paths and delay management. PhD thesis, Fakultät für Informatik, Karlsruher Institut für Technologie (KIT)

Berger A, Hoffmann R, Lorenz U, Stiller S (2007) Online delay management: PSPACE hardness and simulation. Tech Report 0097, ARRIVAL Project

Borodin A, El-Yaniv R (2005) Online computation and competitive analysis. Cambridge University Press, Cambridge

Borodin A, Irani S, Raghavan P, Schieber B (1991) Competitive paging with locality of reference. In: Proceedings of the 23th annual ACM symposium on the theory of computing, pp 249–259

Borodin A, Irani S, Raghavan P, Schieber B (1995) Competitive paging with locality of reference. J Comput Syst Sci 50:244–258

Borodin A, Kleinberg J, Raghavan P, Sudan M, Williamson DP (1996) Adversarial queueing theory. In: Proceedings of the 23rd annual ACM symposium on the theory of computing, pp 376–385

Dollevoet T, Huisman D, Schmidt M, Schöbel A (2012) Delay management with rerouting of passengers. Transp Sci 46(1):74–89

Dollevoet T, Schmidt M, Schöbel A (2011) Delay management including capacities of stations. In: Proceedings of ATMOS 2011—11th workshop on algorithmic approaches for transportation modeling, optimization, and systems, pp 88–99

Fiat A, Woeginger GJ (eds) (1998) Online algorithms: the state of the art. Lecture notes in computer science, vol 1442. Springer, Berlin

Fujiwara H, Iwama K (2002) Average-case competitive analyses for ski-rental problems. In: Algorithms and computation, pp 157–189

Gatto M (2007) On the impact of uncertainty on some optimization problems: combinatorial aspects of delay management and robust online scheduling. PhD thesis, ETH, Zürich

Gatto M, Glaus B, Jacob R, Peeters L, Widmayer P (2004) Railway delay management: exploring its algorithmic complexity. In: Lecture notes in computer science, vol 3111. Springer, Berlin, pp 199–211

Gatto M, Jacob R, Peeters L, Schöbel A (2005) The computational complexity of delay management. In: Lecture notes in computer science, vol 3787. Springer, Berlin, pp 227–238

Gatto M, Jacob R, Peeters L, Widmayer P (2007a) Online delay management on a single train line. In: Lecture notes in computer science, vol 4359. Springer, Berlin, pp 306–320

Gatto M, Nunkesser M, Schöbel A, Widmayer P (2007b) Delay management with restricted adversaries and ski rental. Tech Report 0118, ARRIVAL Project

Heilporn G, de Giovanni L, Labbé M (2008) Optimization models for the single delay management problem in public transportation. Eur J Oper Res 189:762–774

Irani S, Karlin A, Phillips S (1992) Strongly competitive algorithms for paging with locality of reference. In: Proceedings of the 3rd annual ACM-SIAM symposium on discrete algorithms, pp 228–236

Kall P, Wallace SW (1994) Stochastic programming. Wiley, New York
Kliewer N, Suhl L (2011) A note on the online nature of the railway delay management problem. Networks 57(1):28–37
Koutsoupias E, Papadimitriou CH (2000) Beyond competitive analysis. SIAM J Comput 30:394–400
Krumke SO, Thielen C, Zeck C (2011) Extensions to online delay management on a single train line: new bounds for delay minimization and profit maximization. Math Methods Oper Res 74:1–23
Prékopa A (1995) Stochastic programming. Kluwer Academic, Dordrecht
Schachtebeck M, Schöbel A (2010) To wait or not to wait-and who goes first? Delay management with priority decisions. Transp Sci 44(3):307–321
Schöbel A (2001) A model for the delay management problem based on mixed-integer programming. Electron Notes Theor Comput Sci 50(1):1–10
Schöbel A (2006) Optimization in public transportation: stop location, delay management and tariff planning from a customer-oriented point of view. Springer, Berlin
Suhl L, Mellouli T, Biederbick C, Goecke J (2001) Managing and preventing delays in public transportation by simulation and optimization. In: Pursula M, Niittymäki J (eds) Mathematical methods on optimization in transportation systems. Kluwer Academic, Dordrecht
Zeck C (2011) Efficient algorithms for online delay management and railway optimization. PhD thesis, University of Kaiserslautern, Germany