Electronic Notes in Theoretical Computer Science 66 No. 6 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume66.html 10 pages

Online train disposition: to wait or not to wait?*

Luzi Anderegg, Paolo Penna and Peter Widmayer

Institute for Theoretical Computer Science, ETH Zentrum, CH-8092 Zürich, Switzerland, E-mail: {lastname}@inf.ethz.ch.

Abstract

We deal with an online problem arising from bus/tram/train disposition problems. In particular, we look at the case in which the delay is *unknown* and the vehicle can only wait in a station so as to minimize the passengers waiting time.

1 Problem definition

While many of the optimization problems encountered in transportation have already been studied in the early days of operations research ([5,2,4]) and have even stimulated the development of the field, this is not true for *disposition* problems. Disposition (also known as operations control) deals with the *real time reaction* against the negative effects of *unexpected events*. For railways, the goal is to maintain high service quality in spite of events such as delays due to disturbances. Problems of this sort have been attacked mostly by computer simulations [8,13] (see also [9] for a survey), and in some cases by modelling them as complex dynamical systems [6] that require heavy computations even for fairly simple problem instances. In this paper, we pursue a different approach: we aim at an understanding of the fundamental concepts, and we investigate the situation from an algorithmic perspective. In particular, we will look at *worst case* analysis of algorithms that must work with *partial information* (e.g., we know that a vehicle has been delayed, but we do not exactly know by how many time units). To this aim, we will show how these questions can be treated as an *online problem* [3]. Then, we will characterize the performance of algorithms depending on several factors like (i) number of vehicles, (ii) whether the algorithm has some "approximate" estimation of the delays,(iii) whether it can use randomization, etc. This problem is closely

* Work partially supported by the Swiss Federal Office for Education and Science under the Human Potential Programme of the European Union under contract no. HPRN-CT-1999-00104 (AMORE).

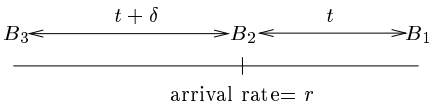

Fig. 1. The case of three buses.

related to the *delay management problem*: a schedule and a delay for one or more vehicles is given and good wait-depart decisions for the consecutive trains must be made. In contrast to our problem the exact delay is known to the algorithm. Suhl and Biederbeck [12] and Adenso-Díaz et al. [1] use simulation systems for analyzing the delay management problem. Other authors, such as Schöbel [11] (see also [7] for a survey), formulate an optimization model. They minimize the total waiting time subject to different constraints (slack times available at stations and tracks, connections can be dropped, etc.).

Consider the following scenario: we are given a station with $r > 0$ passengers arriving at each time unit on average (i.e., r is the *arrival rate of the station*). Buses reach the station regularly every t time units if no delay occurs. Whenever a bus reaches the station, it picks up all waiting passengers (i.e. the seating capacities of the bus is not our concern). This implies that the overall passenger *waiting time at the station* is $r \cdot t^2/2$ per t time units, i.e. between two buses. Now consider the case in which one bus is currently at some station and the next bus is delayed by some amount of $\delta > 0$ time units. Assume the only action we can take is to make a bus wait in a station (this is sometimes referred to as *holding* [10]). A convenient way of looking at this problem is to consider three buses B_3, B_2 and B_1 as in Fig. 1 that are travelling from left to right. Then, from the point of view of the passengers in the station, making B_2 wait for w is equivalent to "shift" B_2 leftwards by w. Indeed, the overall waiting time can be computed according to which bus passengers get into:

$$\underbrace{r(t+w)^2/2}_{B_3} + \underbrace{r(t+\delta-w)^2/2}_{B_2}, \tag{1}$$

which corresponds to the "distance" between consecutive buses (Fig. 2 shows the case $w = 0$). Clearly, *knowing* δ, the best choice (i.e., the choice that minimizes the value in Eq. 1) is $w = \delta/2$. However, in some cases we only know that B_3 is delayed because of a traffic jam and we do not exactly know δ. In this case, should B_2 leave immediately or wait for a while? In the latter case, how much should it wait for?

This kind of questions have a natural formulation as an *online* problem in which we have to choose a good w *without knowing* δ (in other words, w should be good for *all* possible delays δ). In the sequel we assume $r = 2$ and $t = 1$ (this is not too restrictive since we just rescale all the numbers). Then, the waiting time, denoted as cost, is given by $\mathsf{cost}(w, \delta) = (1+w)^2 + (1+\delta-w)^2$,

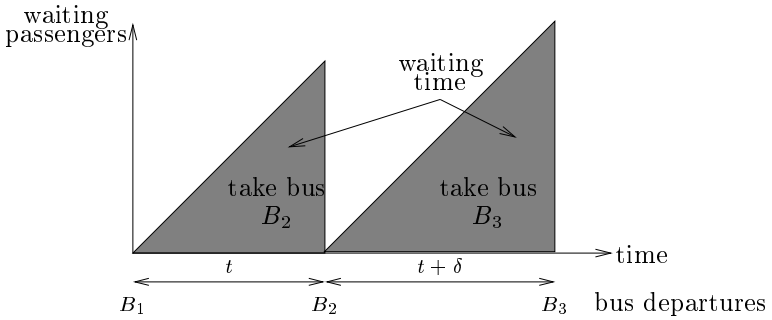

Fig. 2. Passengers waiting time when bus B_3 has a delay δ.

and the competitive ratio is

$$\rho(w,\delta) = \frac{\mathsf{cost}(w,\delta)}{\mathsf{opt}(\delta)} = \frac{(1+w)^2 + (1+\delta-w)^2}{2(1+\delta/2)^2}.$$

We are interested in online algorithms that minimize the above ratio without knowing δ, that is, algorithms that decide w in such a way $\max_{\delta\geq 0}\rho(w,\delta)$ is as small as possible. This is clearly equivalent to find $\min_{w\geq 0}\max_{\delta\geq 0}\rho(w,\delta)$.

We consider two versions of this problem: (a) the *unbounded* case in which δ can be any positive integer; (b) the *bounded* case in which $\delta \leq \Delta$, where Δ is a positive integer *known to the algorithm*, that is, an upper bound on the maximum delay that can occur.

1.1 Our contribution

We consider the above mentioned online problem and its natural extension in which a set of $n+2$ buses (instead of three) is given: bus B_1 already left the station, bus B_{n+2} has a delay δ and we have to decide the waiting time w_i for each B_i, for $i = 2,\ldots,n+1$. We prove the following tight bounds on the competitive ratio of online algorithms: $\Theta(n+1)$ in the unbounded case and $\Theta\left(1+n\left(\frac{\Delta}{2+2n+\Delta}\right)^2\right)$ in the bounded case ($\delta\leq\Delta$).

In particular, all the upper bounds are given via *deterministic algorithms*, while the lower bounds also apply to randomized ones. Indeed, we show that the deterministic algorithms we derive are tight even for the case of randomized algorithms against an oblivious adversary [3].

Paper organization.

For the sake of clarity we first consider the case $n = 1$ in Sect.s 2 and 3, for the unbounded and bounded case, respectively. We then extend the results to the case $n > 1$ in Sect. 4. Finally, in Sect. 5 we discuss further extensions and open questions.

2 Unbounded delays

We first observe that two strategies are always possible:

No wait. In this case $w = 0$ and $\rho(w, \delta) = \frac{1+(1+\delta)^2}{2(1+\delta/2)^2}$. For $\delta \to \infty$, this ratio tends to 2.

Wait "forever". This means that B_2 waits until B_3 arrives in the station, that is, $w = 1 + \delta$. Then, $\rho(w, \delta) = 2$.

The above two strategies seem quite inefficient. Indeed, a better choice might be a compromise of them (i.e. wait, but not too much). Unfortunately, the following result shows that finding such a compromise is impossible:

Theorem 2.1 *No (randomized) algorithm can be less than* 2*-competitive in the case of unbounded delays.*

Proof. Consider a randomized algorithm Alg and a positive p. Let $\overline{w}$ be the minimum $w \geq 0$ such that the probability that Alg decides to wait $w > \overline{w}$ is at most p. For every $\delta \geq 0$, the expected competitive ratio is at least $(1-p) \cdot \rho(\overline{w}, \delta)$. In particular, the adversary can choose a p sufficiently small [1] and a δ sufficiently large to make $(1-p) \cdot \rho(\overline{w}, \delta)$ close to 2 (recall that, for any fixed $\overline{w}$, $\rho(\overline{w}, \cdot)$ tends to 2), the theorem follows. □

Although the above theorem implies that both strategies above are optimal for large delays, it is clear that "no wait" is always better than "wait forever". Moreover, the former performs quite well whenever δ is small. In the subsequent section we investigate this version of the problem.

3 Bounded delays

In this section we consider the version of the problem in which $\delta \leq \Delta$, where Δ is a positive constant *known to the algorithm*. The purpose of this is twofold: by one hand we want to study whether this additional information allows for improved competitive ratios; by the other hand, we are interested in finding tight bounds that show how fast the competitive ratio tends to 2 as Δ increases. The "no wait" strategy provides a first upper bound. However, the reader can easily check that choosing $w = \Delta/2$ gives already an improvement. In the next section we give a tight bound for deterministic algorithms.

3.1 Deterministic algorithms

Our algorithm Det should choose a good value of w based solely on the information that $\delta \leq \Delta$. To this aim, we first restrict ourselves to a *weaker adversary* that chooses only $\delta = 0$ or $\delta = \Delta$. Therefore, our goal will become

$$\min_w \max\{\rho(w, 0), \rho(w, \Delta)\}. \tag{2}$$

[1] As usual, we assume the adversary to know the probability distribution of the algorithm.

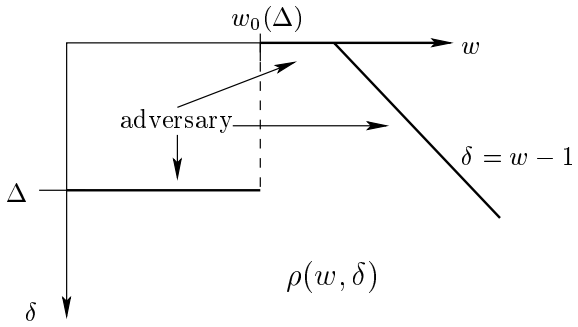

Fig. 3. The worst cases for the deterministic algorithm.

In order to determine the best value for w according to Eq. 2, we look for which values of w the adversary would give us $\delta = 0$, that is $\rho(w, 0) \geq \rho(w, \Delta)$. The latter condition is equivalent to

$$\frac{(1+w)^2 + (1-w)^2}{2} \geq \frac{(1+w)^2 + (1+\Delta-w)^2}{2(1+\Delta/2)^2},$$

which corresponds to $w \geq \frac{\Delta}{4+\Delta} = w_0(\Delta)$. Since, $\rho(w, 0)$ (respectively, $\rho(w, \Delta)$) is monotone increasing (respectively, decreasing), we have (see also Fig. 3)

$$\min_w \max\{\rho(w, 0), \rho(w, \Delta)\} = \rho(w_0(\Delta), 0) = 1 + \left(\frac{\Delta}{4+\Delta}\right)^2.$$

The following lemma is used to show that Det performs well also against an adversary choosing *any* $\delta \in [0, \Delta]$:

Lemma 3.1 *For any* $w \geq 0$, $\max_{0 \leq \delta \leq \Delta} \rho(w, \delta) \leq \max_{\delta \in \{0, \Delta\}} \rho(w, \delta)$.

Because of the above lemma and from the definition of $w_0(\Delta)$, we easily obtain the following:

Theorem 3.2 *No deterministic algorithm can be strictly better than* $1+w_0(\Delta)^2$ *competitive. Therefore,* Det *is optimal for any* $\Delta \geq 0$.

As expected, this bound tends to 2 when Δ goes to infinity (which corresponds to the case of unbounded delays).

3.2 Lower bound for randomized algorithms

In this section we show that a randomized algorithm Rand that chooses with some probability distribution between n values cannot be better than the deterministic algorithm described above. For convenience we assume that the w_i's, $i = 1, \ldots, k$ are sorted in increasing order. It is easy to see that if $w_i \leq w_0(\Delta)$, $i = 1, \ldots, k$, then the algorithm that chooses with probability 1 the value w_n has a better expected competitive ratio: the adversary can maximize *all* $\rho(w_i, \cdot)$ by choosing $\delta = \Delta$ (see Fig. 3). A similar argument also applies to the case $w_i \geq w_0(\Delta)$, $i = 1, \ldots, k$. We can thus assume $w_1 < w_0(\Delta) < w_n$.

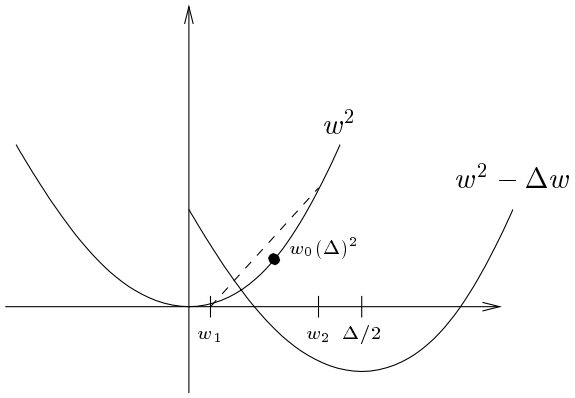

Fig. 4. The idea of the proof of Theorem 3.5.

In proving that no randomized algorithm Rand can be better than Det we make use of two properties of Det: (i) the competitive ratio of Det is maximized for $\delta \in \{0, \Delta\}$ (see Lemma 3.1), and (ii) the competitive ratio for $\delta = 0$ equals the competitive ratio for $\delta = \Delta$ (see the definition of $w_0(\Delta)$). Informally speaking, the first property allows us to restrict to a *weaker adversary* that chooses either $\delta = 0$ or $\delta = \Delta$, while the second one makes possible to compare the competitive ratios by simply comparing the cost function of the respective solutions. The following lemma, stated in a slightly more general form, formalizes this idea:

Lemma 3.3 *Let* Alg *be any algorithm such that*

(i) $\max_{\delta \in \mathcal{S}} \rho(\text{Alg}, \delta) = \max_{\delta \in \{\Delta_1, \Delta_2\}} \rho(\text{Alg}, \delta)$, *where* $\Delta_1, \Delta_2 \in \mathcal{S}$ *and*

(ii) $\rho(\text{Alg}, \Delta_1) = \rho(\text{Alg}, \Delta_2)$.

Then, for any algorithm Alg′ *such that* $\max_{\delta \in \mathcal{S}} \rho(\text{Alg}', \delta) < \max_{\delta \in \mathcal{S}} \rho(\text{Alg}, \delta)$, *the following two conditions must hold:*

$\mathsf{cost}(\text{Alg}', \Delta_1) < \mathsf{cost}(\text{Alg}, \Delta_1)$ *and* $\mathsf{cost}(\text{Alg}', \Delta_2) < \mathsf{cost}(\text{Alg}, \Delta_2)$.

The above lemma, together with some convexity property of the $\mathsf{cost}(\cdot, \cdot)$ function, will be used to prove that no randomized algorithm Rand can have a competitive ratio better than Det. Before giving the proof of the general case, we first present its ideas on a simpler case:

Example 3.4 [Two values with uniform distribution] Let Rand denote an algorithm that chooses *two values* w_1 and w_2 with *uniform distribution*. Assume Rand has a competitive ratio better than Det. By applying Lemma 3.3 with Alg = Det, Alg′ = Rand, $\Delta_1 = 0$ and $\Delta_2 = \Delta$, we get the following two conditions:

$$\mathsf{cost}(w_1, 0) + \mathsf{cost}(w_2, 0) < 2\mathsf{cost}(w_0(\Delta), 0), \tag{3}$$

$$\mathsf{cost}(w_1, \Delta) + \mathsf{cost}(w_2, \Delta) < 2\mathsf{cost}(w_0(\Delta), \Delta). \tag{4}$$

Since $\mathsf{cost}(w, 0) = 2(1 + w^2)$ and $\mathsf{cost}(w, \Delta) = \mathsf{cost}(w, 0) + 2\Delta(1 - w) + \Delta^2$, the two conditions above are equivalent to

$$w_1^2 + w_2^2 < 2w_0(\Delta)^2, \tag{5}$$

(6) $w_1^2 - \Delta w_1 + w_2^2 - \Delta w_2 < 2(w_0(\Delta)^2 - \Delta w_0(\Delta))$.

As mentioned above it must be $w_1 \leq w_0 \leq w_2$. If w_1 and w_2 have the same distance from w_0 (or w_1 is closer to w_0 than w_2) then, because of convexity (see Fig. 4), condition (5) is violated. On the other hand, if w_2 is closer than w_1 then (6) cannot be satisfied.

The following theorem generalizes the ideas of Example 3.4 to the case of any randomized algorithms.

Theorem 3.5 *For any $\Delta > 0$, no* Rand *choosing between k values w_i with uniform probability distribution can be better than* Det.

Proof. By contradiction, assume Rand has a competitive ratio better than Det. To make notation simpler let us denote $w_0(\Delta)$ as w_0. By applying Lemma 3.3 with Alg = Det, Alg$'$ = Rand, $\Delta_1 = 0$ and $\Delta_2 = \Delta$, we get the following two conditions:

$$\sum_{i=1}^{k} w_i^2 < k w_0^2, \tag{7}$$

$$\sum_{i=1}^{k} (w_i^2 - \Delta w_i) < k(w_0^2 - \Delta w_0). \tag{8}$$

We will show that these two conditions cannot hold simultaneously. The proof is based on the idea of looking at the average value of the w_i's, as we did for the Example 3.4. In particular, we consider two cases:

(i) $\sum_{i=1}^{k} w_i \geq k w_0$, i.e. the average is greater or equal w_0.

(ii) $\sum_{i=1}^{k} w_i < k w_0$, i.e. the average is less than w_0.

In both cases we will make use of the well-known Hölder inequality. We thus present it in the following convenient form. Let $x_i, i = 1, \ldots, k$ be a set of k nonnegative reals and $p \geq 1$. Then it holds that

$$\sum_{i=1}^{k} x_i^p \geq k \left(\frac{\sum_{i=1}^{k} x_i}{k} \right)^p. \tag{9}$$

- *Case 1*: Using Eq. 9 we get $\sum_{i=1}^{k} w_i^2 \geq \frac{(\sum_i w_i)^2}{k} \geq k w_0^2$, violating condition (7).
- *Case 2*: First we restate the fact $\sum_{i=1}^{k} w_i < k w_0$ in the following way: $\exists \alpha : 0 < \alpha \leq w_0 : \sum_{i=1}^{k} w_i = k(w_0 - \alpha)$.
  Using Eq. 9 we get

$$\begin{aligned} \sum_{i=1}^{k} (w_i^2 - \Delta w_i) &\geq \frac{k^2 (w_0 - \alpha)^2}{k} - \Delta k (w_0 - \alpha) \\ &= k(w_0^2 - \Delta w_0) + \alpha k (\alpha + \Delta - 2w_0). \end{aligned}$$

  Since $\Delta \geq 2w_0$, for every $\Delta \geq 0$, the second term is always positive and condition (8) does not hold.

This completes the proof. □

Notice that the above proof also works if some of the w_i are not distinct. This, together with the fact that $\mathsf{cost}(\textsc{Rand}, \delta)$ is a continuous and differentiable function, allows us to extend the result to any probability distribution:

Corollary 3.6 *For any $\Delta > 0$, no randomized algorithm* Rand *can be better than* Det.

Proof. Let p_i be the probability that Rand chooses w_i, for $i = 1, \ldots, k$ and for a given set $\{w_i\}$. We generalize the proof of Theorem 3.5 by first considering the case that all p_i's are rationale, and subsequently that in which they take real values.

rationales. Let m be the LCM of the p_i's, and let a_i such that $a_i/m = p_i$. We consider each w_i with multiplicity a_i and probability $1/m$. Notice that $\sum_i a_i = m$. Therefore, the same proof of Theorem 3.5 applies: indeed, there we did not assume that the w_i were distinct.

reals. For every p_i, $i = 1, \ldots, k-1$ define a sequence of rationales $p_i(n)$ such that $\lim_{n\to\infty} p_i(n) = p_i$, and let $p_k(n) = 1 - \sum_{i=1}^{k-1} p_i(n)$. Let $\textsc{Rand}(n)$ be the randomized algorithm choosing w_i with probability $p_i(n)$. Since $\mathsf{cost}(\textsc{Rand}, \delta)$ is a continuous and differentiable function in the p_i's, we have that $\lim_{n\to\infty} \mathsf{cost}(\textsc{Rand}(n), \delta) = \mathsf{cost}(\textsc{Rand}, \delta)$. This, together with the fact that $\textsc{Rand}(n)$ is not better than Det, implies that neither is Rand.

Finally, we can extend all the proofs to the case in which Rand chooses among infinitely many values w_i's (this is actually what we already do in the case of reals when we consider the limit for $n \to \infty$). In this case, we basically exploit the continuous version of Eq. 9. □

4 Many buses

Consider a set of $n+2$ buses $\{B_1, \ldots, B_{n+2}\}$ such that: (i) B_1 already left the station, (ii) B_{n+2} has been delayed by δ and (iii) the set $\{B_2, \ldots, B_{n+1}\}$ corresponds to the *control set* of n buses that we can delay in order to minimize the overall waiting time. Let $\vec{w} = (w^2, w^3, \ldots, w^{n+1})$ represent such waitings, i.e. bus B_i is delayed by w^i, $i = 2, \ldots, n+1$. The waiting time is clearly

$$\mathsf{cost}(\vec{w}, \delta) = \underbrace{(1+w^2)^2}_{B_2} + \sum_{i=2}^{n} \underbrace{(1 + w^{i+1} - w^i)^2}_{B_{i+1}} + \underbrace{(1 + \delta - w^{n+1})^2}_{B_{n+2}}.$$

As we do assume that B_i always proceeds B_{i+1}, for $i = 1, \ldots, n+2$ (it makes no sense to wait longer than the time B_{i+1} reaches B_i), we restrict to those $\vec{w}$ such that $w^i \leq 1 + w^{i+1}$, $i = 2, \ldots, n$. For the same reason, the adversary can only choose δ such that $w^{n+1} \leq 1 + \delta$ (see also the case $n = 1$, Fig 3).

Definition 4.1 [Balanced vector] Let the *balanced vector* $\vec{u}(w)$ be the vector

$\vec{w}$ such that $1 + w^2 = 1 + w^{i+1} - w^i, i = 1, \ldots, n$, i.e. the distance between two consecutive buses is equal, $w^i = \frac{i-1}{n} w, i = 2, \ldots, n+1$.

Fact 4.2 *For every $\vec{w}$ not balanced it holds that*

$$\forall \delta : cost(\vec{w}, \delta) > cost(\vec{u}(w^{n+1}), \delta).$$

Because of the above fact, our goal is still to choose a good value w which minimizes $\max_\delta \rho(w, \delta)$, where $\rho(w, \delta) = \mathsf{cost}(\vec{u}(w), \delta)/\mathsf{opt}(\delta)$. First of all, we can rewrite the function $\rho(\cdot, \cdot)$ as follows:

$$\rho(w, \delta) = \frac{n(1 + w/n)^2 + (1 + \delta - w)^2}{(n+1)(1 + \delta/(n+1))^2}. \tag{10}$$

The following result is a simple generalization of Theorem 2.1:

Theorem 4.3 *For the case of $n + 2$ buses and unbounded delays, no (randomized) algorithm can be less than $n + 1$-competitive.*

4.1 Bounded delays

We first consider an adversary that always picks $\delta \in \{0, \min\{\Delta, w - 1\}\}$, as in the case $n = 1$. Then, we observe that $\rho(\cdot, 0)$ (resp., $\rho(\cdot, \Delta)$) is monotone decreasing in $[0, n\Delta/(n+1)]$ (resp., monotone increasing) and monotone increasing in $(n\Delta/(n+1), \Delta+1]$. Therefore, the best deterministic algorithm is given by the value $w_0(\Delta)$ for which $\rho(w_0(\Delta), 0) = \rho(w_0(\Delta), \Delta)$. This implies that the corresponding deterministic algorithm Det has competitive ratio equal to $\rho(w_0(\Delta), 0)$. From Eq. 10, it is easy to see that $\forall w, \ \ \rho(w, 0) = 1 + w^2/n$, thus implying that the competitive ratio of Det is $1 + n\left(\frac{\Delta}{2+2n+\Delta}\right)^2$. For the more general case $\delta \in [0, \Delta]$, we simply observe that the function $\mathsf{cost}(\vec{u}(w), \cdot)$ is convex in $[0, \Delta]$. Therefore, its maximum is reached when $\delta = 0$ or $\delta = \Delta$, thus implying the following:

Theorem 4.4 *For any $n \geq 1$ and for any $\Delta \geq 0$, no deterministic algorithm can be better than $1 + w_0(\Delta)^2$. Therefore,* Det *is optimal.*

We then consider randomized algorithms and extend the result of Corollary 3.6 to the case of $n + 2$ buses, for any $n > 1$.

5 Extensions and open problems

Our model(s) can be extended in several ways. For instance, one could consider the case of more than one station. Each of the B_i, $i = 2, \ldots, n+1$, is waiting in a station, i.e., we have n equally spaced stations along a line and each station has arrival rate r. Although we might be able to suitably extend the analysis in Sect. 4, this would apply only to "non adaptive" algorithms. On the other hand, we could envision a model in which by the time B_{n+2} enters the first (leftmost) station, this information is known to all buses. In this case, an algorithm might exploit this information through a more adaptive strategy.

More in general, there are several other extensions of our model, e.g. every station S_i may have its own arrival rate r_i, the arrival rate may not be constant over time and/or be unknown to the algorithm or stations and/or buses may not be equally spaced. Finally, one could consider more bus lines sharing some stations and other more complex models.

References

[1] Adenso-Díaz, B., Gonzáles, M.O., Gonzáles-Torre, P., *On-line timetable re-scheduling in regional train services*, Transportation Research **33B** (1999), 387-398.

[2] Bertossi, A., Carraresi, P., and Gallo, G., *On some matching problems arising in vehicle scheduling models*, Networks **17** (1987), 271–281.

[3] Borodin, A., and El-Yaniv, R., "Online Computation and Competitive Analysis", Cambridge University Press, 1998.

[4] Brucker, P., Hurink, J.L., and Rolfes, T., "Routing of railway carriages: A case study", Memorandum No. 1498, Univ. of Twente, Fac. of Math. Sciences, 1999.

[5] Dantzig, G., and Fulkerson, D., *Minimizing the number of tankers to meet a fixed schedule*, Nav. Res. Logistics Q. **1** (1954), 217–222.

[6] Garcia, C., Prettand, D., and Morari, M., *Model predictive control: Theory and practice - a survey*, Automatica **25(3)** (1989), 335–348.

[7] Ginkel, A., "Event-activity networks in delay management", Master's thesis, University of Kaiserslautern, 2001.

[8] Heimburger, D.E., Herzenberg, A.J., and Wilson, N.H.M., *Using simple simulation models in operational analysis of rail transit lines: Case of study of Boston's red line*, Transportation Research Record **1677** (1999), 21–30.

[9] Mansilla, S., "Report on disposition of trains", Tech. report, ETH Zürich, 2001.

[10] O'Dell, S.W., and Wilson, N.H.M., "Optimal real-time control strategies for rail transit operations during disruptions", in: Lecture Notes in Economics and Math. Sys., Computer-Aided Transit Scheduling, 299–323. Springer-Verlag Berlin, Heildeberg, 1999.

[11] Schöbel, A., *A model for the delay management problem based on mixed-integer programming*, Electronic Notes in Theoretical Computer Science **50(1)** (2001).

[12] Suhl, L., and Biederbick, C., "Customer-oriented dispatching for railways", CASPT, 2000.

[13] Zhu, P., and Schnieder, E., "Determining traffic delays through simulation", in: Lecture Notes in Economics and Math. Sys., Computer-Aided Scheduling of Public Transport, 387–398. Springer-Verlag Berlin, Heidelberg, 2001.