Fast Approaches to Improve the Robustness of a Railway Timetable

Matteo Fischetti*, Domenico Salvagnin+, and Arrigo Zanette+

(*) DEI, University of Padova
(+) DMPA, University of Padova

December 2007; Revised, October 2008

Abstract. The Train Timetabling Problem (TTP) consists in finding a train schedule on a railway network that satisfies some operational constraints and maximizes some profit function which counts for the efficiency of the infrastructure usage. In practical cases, however, the maximization of the objective function is not enough and one calls for a robust solution that is capable of absorbing as much as possible delays/disturbances on the network. In this paper we propose and analyze computationally four different methods to improve the robustness of a given TTP solution for the aperiodic (non cyclic) case. The approaches combine Linear Programming (LP) and ad-hoc Stochastic Programming/Robust Optimization techniques. We compare computationally the effectiveness and practical applicability of the four techniques under investigation on real-world test cases from the Italian railway company (Trenitalia). The outcome is that two of the proposed techniques are very fast and provide robust solutions of comparable quality with respect to the standard (but very time consuming) Stochastic Programming approach.

Keywords: Timetabling, Integer Programming, Robustness, Stochastic Programming, Robust Optimization.

1 Introduction

The Train Timetabling Problem (TTP) consists in finding an effective train schedule on a given railway network. The schedule needs to satisfy some operational constraints given by capacities of the network and security measures. Moreover, it is required to exploit efficiently the resources of the railway infrastructure.

In practice, however, the maximization of some objective function is not enough: the solution is also required to be robust against delays/disturbances along the network. Very often, the robustness of optimal solutions of the original problem turns out to be not enough for their practical applicability, whereas easy-to-compute robust solutions tend to be too conservative and thus unnecessarily inefficient. As a result, practitioners call for a fast yet accurate method to

find the most robust timetable whose efficiency is only slightly smaller than the theoretical optimal one. This is typically obtained by adding "buffer times" to the schedule according to certain simple rules (see §2.2 in [12]).

The purpose of the present paper is to propose and evaluate new methods to improve the robustness of a given TTP solution. In particular, we address the aperiodic (non cyclic) TTP version described in [3]. Our approach combines Linear Programming (LP) with Stochastic Programming (SP) and Robust Optimization techniques.

We propose the following three-stage framework as a practical tool for improving and testing robust solutions for the TTP:

stage 1) nominal problem solution: the TTP is modeled without taking into account robustness, and solved (not necessarily to optimality) by a standard MIP solver or by using ad-hoc heuristics.

stage 2) robustness training: borrowing an expression typical of the Artificial Intelligence field, starting from the previous stage solution the model is "trained to robustness", typically by exploiting a restricted set of samples (scenarios).

stage 3) robustness validation: the robustness of the final solution found in stage 2 is evaluated by using a validation tool, thus allowing for a fair comparison of different training methods.

In the present work we focus mainly on the second stage, robustness training. We assume nominal solutions are given in input while, as far as the validation stage is concerned, we use a simple LP validation model.

The paper is organized as follows. In Section 2 the TTP literature is quickly reviewed. In Section 3 we present the TTP in detail and give a natural event-based MIP formulation. Section 4 is a short introduction to the SP paradigm and to the sampling techniques that are used in practice to implement it. In Sections 5 and 6 we present the main building blocks of our solution framework. Extensive computational results are given in Section 7, showing that two of the new methods we propose are very fast and provide robust solutions of comparable quality with respect to the standard (but very time consuming) Stochastic Programming approach. Finally, some conclusions are drawn in Section 8.

An early version of the present work was presented at ATMOS 2007 [8].

2 Literature review

The TTP problem has two main variants: the *periodic* and *aperiodic* versions. The periodic TTP's goal is to design a timetable that is operated cyclically after a (small) period of time; this is a typical requirement for passenger trains in order to come up with an easy-to-remember timetable. The first authors who developed a model for generating periodic timetables were Serafini and Ukovic [23], who introduced a mathematical model called *Periodic Event Scheduling Problem* (PESP). In PESP, a set of repetitive events is scheduled under periodic time-window constraints. Consequently, the events are scheduled for one cycle

in such a way that the cycle can be repeated. Most models for periodic TTP are based on PESP. A natural LP formulation of PESP is quite weak, due to kind of big-M constraints (where M is the period). An alternative stronger formulation is treated in Nachtigall [20] and Liebchen and Peeters [14, 21] among others, and is based on cycle bases and separation of valid inequalities.

As to robustness, Kroon et al. [12] describe a stochastic optimization variant of PESP. Their model explicitly takes into account stochastic disturbances of the railway processes, distinguishing between a planned timetable and several realizations of the timetable under pre-determined stochastic disturbances. The model can be used to allocate time supplements and buffer times to the processes in the planned timetable in such a way that the average delay of the realizations of the trains is minimized. In order to keep the computation times within an acceptable bound, they start with an existing timetable and fix the precedences among trains. They show that a substantial increase in robustness can be achieved by taking into account stochastic disturbances in the design of the timetable. For the case of one trip serving 10 stations, Liebchen and Stiller [16] provide a theoretical explanation for the effects observed empirically in Kroon et al. [12]. Very recently, a new notion of robustness, called *recoverable robustness*, has been proposed in [13], which integrates the notion of robustness and recoverability into a common framework. Applications to integrated timetabling/delay management in railway systems are described and evaluated in [13, 15, 5].

The aperiodic TTP is especially relevant on heavy-traffic, long-distance corridors, where the capacity of the infrastructure is limited due to greater traffic densities, and competitive pressure among the train operators is expected to increase in the near future. In the Caprara et al. [3] setting, a train operator applies for allocating its trains on the infrastructure, and specify a profit for the "ideal timetable" they are asking for. Then the infrastructure manager collects all requests from train operators and computes a feasible timetable maximizing the overall profit, i.e., the difference between the profits of the scheduled trains and a cost-penalty function, which takes into account the deviations of the final timetables with respect to the ideal ones (possibly leaving some trains unscheduled).

Different ILP models based on a graph representation of the problem were presented in [3, 4]. In these papers the problem is modeled by means of a directed acyclic multi-graph, in which nodes correspond to arrival and departure events from the stations and arise at some discretized time instants, and arcs correspond to train stops within a station or to train trips. A Lagrangian relaxation method is used to derive bounds on the optimal solution value as well as to drive a heuristic procedure.

3 The Nominal Model

In this section we describe the specific aperiodic TTP problem we consider, and give a basic event-based formulation for the "nominal" version where robustness is not taken into account.

Following [3], the aperiodic TTP can be described as follows: given a railway network, described as a set of stations connected by tracks, and an ideal train timetable, find an actual train schedule satisfying all the operational constraints and having a minimum distance from the ideal timetable.

The entities involved in modelling the problem are the following:

railway network: a graph $N = (\mathcal{S}, \mathcal{L})$, where $\mathcal{S}$ is the set of stations and $\mathcal{L}$ is the set of tracks connecting them.

trains: a train corresponds to a simple path on the railway network N. The set of trains is denoted by T. For each train $h \in T$ we have an ideal profit π_h (the profit of the train if scheduled exactly as in the ideal timetable), a stretch penalty θ_h (the train *stretch* being defined as the difference between the running times in the actual and ideal timetables) and a shift penalty σ_h (the train *shift* being defined as the absolute difference between the departures from the first station in the actual and ideal timetables).

events: arrivals and departures of the trains at the stations. The set of all the events is denoted by E. With a small abuse of notation, we will denote by t_i^h both the i-th event of train h and its associated time. We also define

- A: set of all arrival events
- D: set of all departure events

whereas A_S, D_S and E_S denote the restriction of the above sets to a particular station S. Each train h is associated with an ordered sequence of length $len(h)$ of departure/arrival events t_i^h such that $t_{i+1}^h \geq t_i^h$, the first and last event of train h being denoted by t_1^h and $t_{len(h)}^h$, respectively. In addition, let $\overline{t_i^h}$ denote the ideal time for event t_i^h.

(partial) schedule: a time assignment to all the events associated with a subset of trains.

objective: maximize the overall profit of the scheduled trains, the profit of train h being computed as

$$\rho_h = \pi_h - \sigma_h \, shift_h - \theta_h \, stretch_h$$

where

$$shift_h = |t_1^h - \overline{t}_1^h|$$

and

$$stretch_h = (t_{len(h)}^h - t_1^h) - (\overline{t}_{len(h)}^h - \overline{t}_1^h)$$

denote the shift and stretch associated with train h, respectively. Trains with negative profit are intended to remain unscheduled and do not contribute to the overall profit.

Operational constraints include:

time windows: it is possible to shift an event from its ideal time only within a given time window;

headway times: for safety reasons, a minimum time distance between two consecutive arrival/departure events from the same station is imposed;

track capacities: overtaking between trains is allowed only within stations (assumed of infinite capacity).

As already mentioned, in the present paper we do not address the solution of the nominal TTP problem explicitly, in that a nominal solution is assumed to be provided in input. Nevertheless, we next outline the structure of a MIP formulation for the nominal TTP problem, since a relaxed version of it is at the basis of the LP models used in Sections 5 and 6.

Although in the nominal problem one is allowed to leave some trains unscheduled, to simplify our presentation we consider the situation where one is required to schedule all the trains. A natural event-based model in the spirit of the Periodic Event Scheduling Problem (PESP) formulation used in the periodic (cyclic) case [23] can be sketched as follows:

$$z^* = \max \sum_{h \in T} \rho_h$$

$$t^h_{i+1} - t^h_i \geq d^h_{i,i+1} \quad \forall h \in T, i = 1, \ldots, len(h) - 1 \tag{1}$$

$$|t^h_i - t^k_j| \geq \Delta_a \quad \forall t^h_i, t^k_j \in A_S, \forall S \in \mathcal{S} \tag{2}$$

$$|t^h_i - t^k_j| \geq \Delta_d \quad \forall t^h_i, t^k_j \in D_S, \forall S \in \mathcal{S} \tag{3}$$

$$t^h_{i+1} < t^k_{j+1} \Leftrightarrow t^h_i < t^k_j \quad \forall t^h_i, t^k_j \in D_S, \forall S \in \mathcal{S} \tag{4}$$

$$\rho_h = \pi_h - \sigma_h |t^h_1 - \bar{t}^h_1| - \theta_h((t^h_{len(h)} - t^h_1) - (\bar{t}^h_{len(h)} - \bar{t}^h_1)) \quad \forall h \in T \tag{5}$$

$$l \leq t \leq u \quad \forall t \in E \tag{6}$$

Constraints (1) impose a minimum time difference $d_{i,i+1}$ between two consecutive events of the same train, thus imposing minimum trip duration (trains are supposed to travel always at the maximum allowed speed for the track) and minimum stop time at the stations.

Constraints (2)-(3) model the headway times between two consecutive arrival or departure events in the same station (Δ_d and Δ_a being the minimum departure and arrival headway, respectively). Since these constraints are nonlinear and we do not know in advance the order in which events occur at the stations, in our MIP model we introduce a set of binary variables $x^{h,k}_{i,j}$ to be set to 1 iff $t^h_i \leq t^k_j$ along with a big-M coefficient M, so that conditions

$$|t^h_i - t^k_j| \geq \Delta$$

can be translated to

$$t^h_i - t^k_j \geq \Delta - M x^{h,k}_{i,j}$$

$$t^k_j - t^h_i \geq \Delta - M x^{k,h}_{j,i}$$

$$x_{i,j}^{h,k} + x_{j,i}^{k,h} = 1$$

Given the linearization of constraints (2)-(3), it is easy to translate the track capacity constraints (4) as

$$x_{i,j}^{h,k} = x_{i+1,j+1}^{h,k}$$

Constraints (5) define the profits of the trains, whereas constraints (6) model the user-defined time windows of each event.

It is important to notice that, although we are interested in integer values (minutes) for the events to be published in the final timetable, we do not force the integrality on variables t_j. This has the important consequence that, after fixing the event precedence variables x, the model becomes a plain linear model. On the other hand, the possible fractional value of the final time variables t need to be handled somehow in a post-processing phase to be applied before publishing the timetable. For arrival events, one can just round the corresponding fractional times to the nearest integer since there is no problem of an arrival arises a little earlier (or later) than published. An easy procedure for departure times is instead to simply round down all the t-values even if this results into a slightly infeasible published timetable, so as to guarantee that all events arise not earlier than their published time value. In a sense, this policy amounts to using an "infinite" time discretization during the optimization phase, the difference between the actual and the published event times being perceived by the travelers as a small (less than one minute) delay.

As far as the objective function is concerned, the nonlinear term

$$|t_1^h - \bar{t}_1^h|$$

giving the shift s_h of train h can be easily linearized as

$$s_h \geq t_1^h - \bar{t}_1^h$$

$$s_h \geq \bar{t}_1^h - t_1^h$$

4 The Stochastic Programming Paradigm

Having stated the problem as a MIP, a well known tool to find robust solutions is the (two-stage) *Stochastic Programming* approach; see [2],[22] for an introduction to the SP methodology. In SP, the set of constraints is decomposed in *structural* constraints, which represent the deterministic part of the model, and *control* constraints which have a stochastic nature and whose coefficients depend on the particular scenario. Roughly speaking, the approach allows one to take decisions in the first stage by ignoring the stochastic part of the model, but enforces some costly *recourse* action when indeterminacy will eventually occur. Thus a natural optimization objective for this two-stage strategy is to minimize the total expected cost:

$$\min_{x \in X} c^T x + \mathbb{E}[Q(x, \xi(\omega))]$$

where x denotes the first-stage decision whose feasibility set is X, $\omega \in \Omega$ denotes a *scenario* that is unknown when the first-stage decision x has to be made, and $Q(x, \xi(\omega))$ represents the optimal value of the second-stage recourse problem corresponding to first-stage decision x and parameters $\xi(\omega)$.

If Ω contains a finite number of scenarios $\{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ with associated probabilities p_k, $k \in 1, 2, \ldots, |\Omega|$, then the expectation can be evaluated as a finite sum, and the two-stage model (with linear recourse) becomes a standard linear model:

$$w^* = \min_{x \in X} c^T x + \sum_{k=1}^{|\Omega|} p_k q_k^T r_k, \quad r_k \in Y_k, \forall k = 1 \ldots |\Omega| \tag{7}$$

where r_k are the recourse variables with linear costs q_k, and Y_k is a polyhedron depending on the first-stage variables x.

As $|\Omega|$ is often very large, various sampling-based approaches have been proposed to estimate the second-stage objective function. *Interior sampling* methods use samples during the algorithm execution to estimate, from time to time, algorithmic parameters such as function values, gradients, optimality cuts. *Exterior sampling* methods, instead, use the *Sample Average Approximation* (SAA) algorithm to estimate the optimal objective. We have chosen exterior sampling, since it has some advantages over interior sampling [24]: ease of numerical implementation, good theoretical convergence properties [25], well developed statistical inference (validation and error analysis, stopping rules). Furthermore, it is easily amenable to variance reduction techniques, ideal for parallel computations.

4.1 The Sample Average Approximation Method

The SAA consists in approximating the mean of the random variable $Q(x, \xi(\omega))$ with the sample mean of $\{Q(x, \xi(\omega_1)), Q(x, \xi(\omega_2)), \ldots, Q(x, \xi(\omega_N))\}$ independent and identically distributed (i.i.d.) samples from the distribution of $Q(x, \xi(\omega))$. If $Q(x, \xi(\omega))$ has finite mean and variance, the sample mean $\bar{Q}_N(x, \xi(\omega_i)) = \frac{1}{N} \sum_{i=1}^{N} Q(x, \xi(\omega_i))$ is an unbiased estimator of the actual mean:

$$\mathbb{E}[\bar{Q}_N(x, \xi(\omega_i))] = \mathbb{E}[Q(x, \xi(\omega))]$$

and it satisfies the following central limit theorem:

$$\sqrt{N}[\bar{Q}_N(x, \xi(\omega_i)) - \mathbb{E}[Q(x, \xi(\omega))]] \Rightarrow \mathcal{N}(0, \sigma^2) \text{ as } N \to \infty$$

where $\Rightarrow$ denotes convergence in distribution, $\mathcal{N}(0, \sigma^2)$ is a normal random variable with zero mean and variance $\sigma^2 = \text{var } Q(x, \xi(\omega))$.

The SAA approximation of (7) reads:

$$w_N^* = \min_{x \in X} c^T x + \frac{1}{N} \sum_{k=1}^{N} q_k^T r_k, \quad r_k \in Y_k, \forall k = 1 \ldots N \tag{8}$$

Under mild assumptions it was proved that the optimal value of SAA problem (8) converges with probability 1 to w^* (the optimal value of stochastic problem) as N tends to infinity (see [25]).

Also, it is possible to use SAA to estimate the optimality gap by deriving lower and upper bounds on w^*. These bounds will be used to quantify the confidence intervals of the optimal solution of the stochastic model (see Section 7, Figure 9). Indeed, an *upper bound* on w^* is

$$c^T\hat{x} + \mathbb{E}[Q(\hat{x}, \xi(\omega))] = c^T\hat{x} + \mathbb{E}[\bar{Q}_N(\hat{x}, \xi(\omega_i))] \tag{9}$$

where $\hat{x}$ is a given feasible, yet suboptimal, first-stage decision vector. The expectation in the right hand side of (9), by its own, can be estimated as the mean of $N' \ll N$ (say) independent SSA $\bar{Q}^j_N(\hat{x}, \xi(\omega^j_i))$, obtaining:

$$\overline{UB} = \frac{1}{N'}\sum_{j=1}^{N'} \bar{Q}^j_N(\hat{x}, \xi(\omega^j_i)) \tag{10}$$

$$\sigma^2_u = \text{var } \bar{Q}_N(\hat{x}, \xi(\omega_i)) = \frac{1}{(N'-1)N'}\sum_{j=1}^{N'}(\bar{Q}^j_N(\hat{x}, \xi(\omega^j_i)) - \bar{UB}) \tag{11}$$

It is easy to show (see [17]) that a lower bound on w^* is given by $\mathbb{E}[w^*_N]$. Again, we can compute this expectation by sampling:

$$\overline{LB} = \frac{1}{N'}\sum_{j=1}^{N'} w^{*j}_N \tag{12}$$

$$\sigma^2_l = \text{var } \bar{w}^*_N = \frac{1}{(N'-1)N'}\sum_{j=1}^{N'}(w^{*j}_N - \overline{LB}) \tag{13}$$

4.2 Sampling

Sampling of delays has been carried out using the following per-line model. A *line* $\mathcal{L}$ is defined as a sequence of stations operated by trains during the 24 hours. Each line section (the path between two consecutive stations i and j) can have a certain probability $P_{(i,j)}$ to be affected by delay. Also, each time interval $[l, k]$ in the 24-hour time horizon can have a certain probability of delay, say $P_{[l,k]}$. Then each single train h arrives at the last station with a cumulative random delay δ^h. The actual delay incurred by train h operating on section (i, j) in time interval $[l, k]$ is computed using the following formula:

$$\delta^h_{(i,j)}([l, k]) = \delta^h P_{[l,k]} \frac{P_{(i,j)}}{\sum_{(i,j)\in\mathcal{L}} P_{(i,j)}} \tag{14}$$

where we normalize sections delay probabilities in order to distribute the cumulative delay $\delta^T P_{[l,k]}$ incurred by train T operating on line $\mathcal{L}$ through each line

section (i, j). Note that $P_{(i,j)}$ and $P_{[l,k]}$ could be deterministic numbers between 0 and 1, but typically they are treated as random variables.

When using random sampling, the outcome can be affected by a large variance, making it difficult to interpret. So we decided to use *Latin Hypercube* (LH) variance reduction technique when sampling from each distribution of $P_{(i,j)}$, $P_{[l,k]}$ and δ^h. Other techniques such as, e.g., Importance Sampling [6] can in principle fit our simulation setting as well, but they are quite involved. On the contrary, LH sampling is of general applicability and comes with a straightforward implementation. The standard approach to get a sample from a particular probability distribution is to apply the inverse of the desired Cumulative Distribution Function (CDF) to a sample drawn from a uniform distribution. The process is then repeated until the required number of samples N is collected. Using LH sampling, instead, we first subdivide the $[0, 1]$ interval in N equal subintervals, and from each of them we draw a sample from a uniform distribution spanning the subinterval. Then the obtained sample vector is inverted through the CDF and randomly permuted. For more theoretical insights on LH sampling, the interested reader is referred to [18].

LH sampling proved to be quite effective in our application. Figure 1 shows the reduction in variance σ when sampling from an exponential distribution with or without LH sampling. In our computational testing, we observed an even larger variance reduction (one order of magnitude or more).

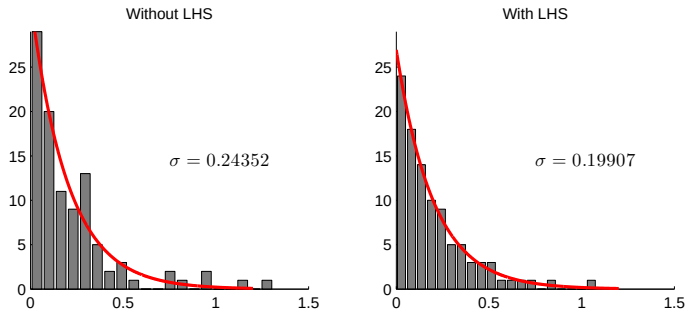

Fig. 1. Reduction of variance σ with LH when approximating, through sampling, the exponential probability distribution function (solid line).

5 Validation Model

An important component in our framework is robustness validation. Validation is often carried out inside the model itself, as is the case when a SP approach is used. However, we decided to implement an external simulation-based validation module that is independent from the optimization model itself, so that it can

be of general applicability and allows one to compare solutions coming from different methods. The module is required to simulate the reaction of the railway system to the occurrence of delays, by introducing small adjustments to the planned timetable received as an input parameter. Validation is a major topic on its own. Indeed, the set of actions the railway system can make to react to disruptions is quite large—see for example [9]—and the decision making process is often complicated by strict real-time requirements and complex business rules. Validation can be carried out by optimization methods, as proposed in [13, 15, 5]. However, the complexity of such models grows rapidly as soon as we allow complex decisions to be made. Thus, simulation methods are often used to measure empirically the robustness of a given timetable—see, for example, [1].

For the purpose of the present paper, we decided to implement a simple LP-based validation tool based on the following simplified assumptions.

- Limited adjustability in response to delays with respect to the given timetable: In this paper, timetabling robustness is *not* concerned with major disruptions (which have to be handled by the real time control system and require human intervention) but is a way to control delay propagation, i.e., a robust timetable has to favor delay compensation without heavy human action. As a consequence, at validation time no train cancellation is allowed, and event precedences are fixed with respect to the planned timetable.
- Speed of validation: The validation tool should be able to analyze quickly the behavior of the timetable under many different scenarios.

Given these guidelines, we designed a validation model which analyzes a single delay scenario ω at a time. As all precedences are fixed according to the input solution to be evaluated, constraints (1-3) all simplify to linear inequalities of the form:

$$t_i - t_j \geq d_{i,j}$$

where $d_{i,j}$ can be a minimum trip time, a minimum rest or an headway time. We will denote with $\mathcal{P}$ the set of ordered pairs (i, j) for which a constraint of type (1) can be written. The problem of adjusting the given timetable t under a certain delay scenario ω can thus be rephrased as the following simple linear programming model with decision variables t^ω:

$$\min \sum_{j \in E} w_j \left(t_j^\omega - t_j \right) \tag{15}$$

$$t_i^\omega - t_j^\omega \geq d_{i,j} + \delta_{i,j}^\omega \; \forall (i,j) \in \mathcal{P} \tag{16}$$

$$t_i^\omega \geq t_i \quad \forall i \in E \tag{17}$$

Constraints (16) correspond to the linear inequalities just explained, in which the nominal right-hand-side value $d_{i,j}$ is updated by adding the (possibly zero) extra-time $\delta_{i,j}^\omega$ from the current scenario ω. Weights w_j appearing in objective function (15) are related to the relative importance of the events, and typically depend on the number of passengers affected.

Constraints (17) are non-anticipatory constraints stating the obvious condition that one is not allowed to anticipate any event with respect to its published value in the timetable. Since these values are known, these constraints act as simple lower bounds on the decision variables. Instead, we impose no upper bounds since we allow for an unlimited stretch of the timetable to recover from delays, i.e., a feasible timetable is always achievable.

The objective function is to minimize the "cumulative delay" on the whole network.

Given a feasible solution, the validation tool keeps testing it against a large set of scenarios, one at a time, gathering statistical information on the value of the objective function and yielding a concise figure (the average cumulative delay) of the robustness of the timetable.

6 Finding Robust Solutions

In this section we present three different approaches to cope with robustness. We introduced two simplifying hypotheses: (1) all input trains have to be scheduled; (2) all event precedences are fixed according to a given input "nominal" solution. These strong assumptions were made to obtain tractable (LP) models.

6.1 A Fat Stochastic Model

Our first attempt to solve the robust version of the TTP is to use a standard scenario-based SP formulation. The model can be outlined as:

$$\min \frac{1}{|\Omega|} \sum_{j \in E, \omega \in \Omega} \left(t_j^\omega - t_j\right)$$

$$\sum_{h \in T} \rho_h \geq (1-\alpha) z^* \tag{18}$$

$$t_i^\omega - t_j^\omega \geq d_{i,j} + \delta_{i,j}^\omega \ \forall (i,j) \in \mathcal{P}, \forall \omega \in \Omega \tag{19}$$

$$t_i^\omega \geq t_i \quad \forall i \in E, \forall \omega \in \Omega \tag{20}$$

$$t_i - t_j \geq d_{i,j} \qquad \forall (i,j) \in \mathcal{P} \tag{21}$$

$$l \leq t \leq u \tag{22}$$

The structure of the model is similar to that used in the validation tool, but takes into account several scenarios at the same time. Moreover, the nominal timetable values t_j are now viewed as first-stage decision variables to be optimized–their optimal value will define the final timetable to be published. The model is composed by the original one and a copy of it with a modified right hand side for each scenario. The original variables and the correspondent second-stage copies in each scenario are linked through non-anticipatory constraints.

The objective is to minimize the cumulative delay over all events and scenarios. The original objective function $\sum \rho_h$ is taken into account through constraint (18), where $\alpha \geq 0$ is a tradeoff parameter and z^* is the objective value of the reference solution.

For realistic instances and number of scenarios this model becomes very time consuming (if not impossible) to solve–hence we called it "fat". On the other hand, also in view of its similarity with the validation model, it plays the role of a kind of "perfect model" in terms of achieved robustness, hence it has been used for benchmark purposes.

6.2 A Slim Stochastic Model

Being the computing time required by the full stochastic model quite large, we present an alternative model which is simpler yet meaningful for our problem. In particular, we propose the following recourse-based formulation:

$$\min \sum_{(i,j)\in\mathcal{P},\omega\in\Omega} w_{i,j} s^{\omega}_{i,j} \tag{23}$$

$$\sum_{h\in T} \rho_h \geq (1-\alpha) z^* \tag{24}$$

$$t_i - t_j + s^{\omega}_{i,j} \geq d_{i,j} + \delta^{\omega}_{i,j} \; \forall (i,j) \in \mathcal{P}, \forall \omega \in \Omega \tag{25}$$

$$s^{\omega}_{i,j} \geq 0 \; \forall (i,j) \in \mathcal{P}, \forall \omega \in \Omega \tag{26}$$

$$t_i - t_j \geq d_{i,j} \qquad \forall (i,j) \in \mathcal{P} \tag{27}$$

$$l \leq t \leq u \tag{28}$$

In this model we have just one copy of the original variables, plus the recourse variables $s^{\omega}_{i,j}$ which account for the unabsorbed extra times $\delta^{\omega}_{i,j}$ with respect to the minimum train trip times. It is worth noting that the above "slim" model is inherently smaller than the fat one. Moreover, one can drop all the constraints of type (25) with $\delta^{\omega}_{i,j} = 0$, a situation that occurs very frequently in practice since most extra-times in a given scenario are zero.

As to objective function, it involves a weighted sum of the recourse variables. Finding meaningful values for weights $w_{i,j}$ turns out to be very important. Indeed, we will show in Section 7 how to define these weights so as to produce solutions whose robustness is comparable with that obtainable by solving the (much more time consuming) fat model.

6.3 Light Robustness

A different way to produce robust solutions is to use the *Light Robustness* (LR) approach proposed recently by Fischetti and Monaci [7]. This method is based on the consideration that, roughly speaking, robustness is about putting enough

slack on the constraints of the problem. In its simpler version, the LR counterpart of the LP model

$$\min\{c^T x : \quad Ax \leq b, \quad x \geq 0\}$$

reads

$$\min f(\gamma) \tag{29}$$
$$Ax - \gamma \quad \leq b - \beta \tag{30}$$
$$c^T x \leq (1+\alpha) z^\star \tag{31}$$
$$x \geq 0 \tag{32}$$
$$0 \leq \gamma \leq \beta \tag{33}$$

where β_i is a parameter giving the desired *protection level* (or slack) on constraint i, and $\gamma_i \geq 0$ is a decision variable giving the corresponding *unsatisfied slack*. The objective is to minimize a given function f of the γ variables (typically, a linear or quadratic expression). Moreover, (31) gives a bound (controlled by α) on the efficiency loss due to the increased robustness of the solution, where $z^\star$ is the value of the input nominal solution.

In our TTP model, a typical constraint reads

$$t_i - t_j \geq d_{i,j}$$

and its LR counterpart is simply

$$t_i - t_j + \gamma_{i,j} \geq d_{i,j} + \Delta_{i,j} \quad \gamma_{i,j} \geq 0$$

where $\Delta_{i,j}$ is the required protection level parameter.

7 Computational Results

We carried out tests on four single-line medium-size TTP instances provided by the Italian railway company, Trenitalia. Data refers to unidirectional traffic on different corridors.

Instance	#Stations	#Sched. Trains
BZVR	27	127
BrBO	48	68
MUVR	48	48
PDBO	17	33

Table 1. Nominal solution characteristics

An almost-optimal heuristic solution for each of these instances was computed by using the algorithm described in [3]. The algorithm is a Lagrangian

heuristic based on the computation of paths on a time-expanded network, whose computing time was in the order of minutes on a Pentium IV, 2.4 GHz PC. The corresponding solutions were used as the input nominal solutions to freeze the event precedences and to select the trains to schedule. Solution characteristics are given in Table 1.

We implemented our framework in C++ and carried out our tests on a AMD Athlon64 X2 4200+ computer with 4GB of RAM. ILOG CPLEX 10.1 [10] was used as MIP solver.

According to the sampling model described in Section 4.2, we generated an extra time $\delta^h(\omega)$ corresponding to each train h and to each scenario ω, drawing them from an exponential distribution with mean $\mu = 5\%$. In lack of more detailed data from the Italian railways operator about the actual distribution of delays in line sections, we assume a proportional distribution of delays along line segments. Accordingly, probabilities $P_{(i,j)}$ in (14) are proportional to the length of train segments, barring a small additive white Gaussian noise (standard deviation $\sigma = 0.01$, i.e., a random adjustment of 1-2%), and probabilities $P_{[l,k]}$ are deterministically set to 1.

Given this setting, the first test we performed was aimed at comparing four different training methods for each reference solution, with different values of the tradeoff parameter α, namely 1%, 5%, 10% and 20%. We compared the following alternative methods:

- *fat*: fat stochastic model (50 scenarios only)
- *slim1*: slim stochastic model with uniform objective function–all weights equal (400 scenarios)
- *slim2*: slim stochastic model with enhanced objective function (400 scenarios), where events arising earlier in each train sequence receive a larger weight in the objective function. More specifically, if the i-th event of train h is followed by k events, its weight in (23) is set to $k + 1$. The idea behind this weighing policy is that unabsorbed disturbances $s^{\omega}_{i,j}$ in a train sequence are likely to propagate to the next ones, so the first ones in the line are the most important to minimize.
- *LR*: Light Robustness model with objective function as in *slim2* (using the *slim1* objective function produces significantly worse results). Protection level parameters are set to $\Delta = -\mu \ln \frac{1}{2}$, where μ is the mean of the exponential distribution. This is the protection level required to absorb a delay drawn from such a distribution with probability $\frac{1}{2}$. For example, setting a buffer of 1 minute we can absorb half of the times an exponentially distributed disturbance of mean 1.44 minutes.

As to the validation model, weights w_j appearing in objective function (15) are assumed to be equal to 1, i.e., all events are considered equally important.

The results are shown in Table 2, while graphical representations are given in Figures 2 and 3.

According to the figures, *slim2* always yields a very tight approximation of *fat*, while *slim1* is often poorer. As to *LR*, it usually produces good results that are only slightly worse than *slim2*, mainly in the most-realistic cases where the

tradeoff parameter α is small. As to computing times (reported in Table 2), the *fat* model is one order of magnitude slower than *slim1* and *slim2*, although it uses only 50 scenarios instead of 400. *LR* is much faster than any other method—more than two orders of magnitude w.r.t the fat stochastic models. Therefore, *LR* qualifies as the method of choice for addressing large-scale real cases, as it guarantees good levels of robustness and requires very short computing times.

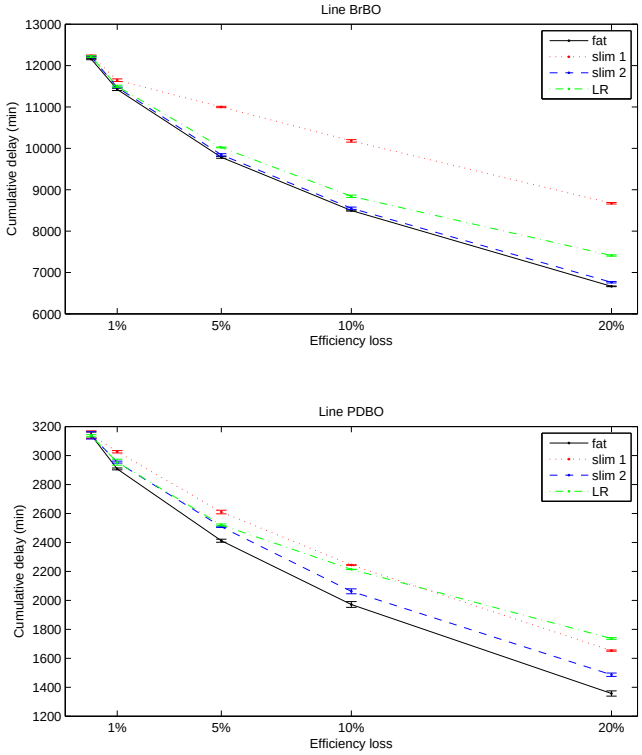

Fig. 2. Comparison of different training models applied to the best reference solution for each instance. On the x-axis there is the efficiency loss (α) while the y-axis reproduces the confidence intervals of the validation figure (run with 500 scenarios).

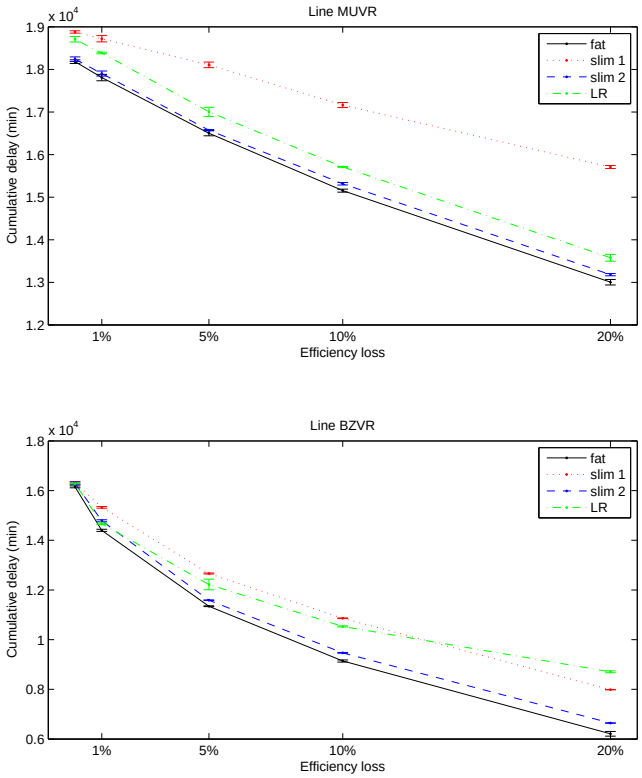

Fig. 3. Comparison of different training models applied to the best reference solution for each instance. On the x-axis there is the efficiency loss (α) while the y-axis reproduces the confidence intervals of the validation figure (run with 500 scenarios).

α	Fat			Slim1			Slim2			LR		
Line	Delay	WAD(%)	Time(s)	Delay	WAD(%)	Time(s)	Delay	WAD(%)	Time(s)	Delay	WAD(%)	Time(s)
0% BZVR	16149	–	9667	16316	–	532	16294	–	994	16286	–	2.27
0% BrBO	12156	–	384	12238	–	128	12214	–	173	12216	–	0.49
0% MUVR	18182	–	377	18879	–	88	18240	–	117	18707	–	0.43
0% PDBO	3141	–	257	3144	–	52	3139	–	63	3137	–	0.25
Tot:	49628	–	10685	50577	–	800	49887	–	1347	50346	–	3.44
1% BZVR	14399	16.4	10265	15325	45	549	14787	17	1087	14662	18	2.13
1% BrBO	11423	21.6	351	11646	42	134	11472	21	156	11499	23	0.48
1% MUVR	17808	12.9	391	18721	37	96	17903	12	120	18386	8	0.48
1% PDBO	2907	15.6	250	3026	51	57	2954	11	60	2954	13	0.27
Tot:	46537	66.5	11257	48718	175	836	47116	61	1423	47501	62	3.36
5% BZVR	11345	15.9	9003	12663	48	601	11588	19	982	12220	22	1.99
5% BrBO	9782	18.9	357	11000	50	146	9842	22	164	10021	23	0.51
5% MUVR	16502	14.5	385	18106	41	86	16574	13	107	17003	11	0.45
5% PDBO	2412	14.7	223	2610	44	49	2508	20	57	2521	19	0.28
Tot:	40041	64	9968	44379	183	882	40512	74	1310	41765	75	3.23
10% BZVR	9142	21.4	9650	10862	50	596	9469	24	979	10532	33	2.01
10% BrBO	8496	19.1	387	10179	51	132	8552	20	157	8842	23	0.51
10% MUVR	15153	14.7	343	17163	49	84	15315	15	114	15710	13	0.43
10% PDBO	1971	19.9	229	2244	49	50	2062	27	55	2314	37	0.25
Tot:	34762	75.1	10609	40448	199	862	35398	86	1305	37398	106	3.2
20% BZVR	6210	28.5	9072	7986	50	538	6643	31	1019	8707	52	2.04
20% BrBO	6664	22.1	375	8672	53	127	6763	23	153	7410	30	0.52
20% MUVR	13004	17.1	384	15708	52	91	13180	18	116	13576	19	0.42
20% PDBO	1357	28.4	230	1653	49	55	1486	34	60	1736	53	0.28
Tot:	27235	96.1	10061	34019	204	811	28072	106	1348	31429	154	3.26
40% BZVR	3389	35.4	10486	4707	50	578	3931	37	998	5241	51	2.31
40% BrBO	4491	27.7	410	6212	52	130	4544	29	166	6221	52	0.53
40% MUVR	10289	21.8	376	13613	52	95	10592	25	108	11479	34	0.45
40% PDBO	676	37.1	262	879	49	55	776	41	57	1010	52	0.28
Tot:	18845	122	11534	25411	203	858	19843	132	1329	23951	189	3.57

Table 2. Comparison of different training methods with respect to computing time, percentage WAD and validation function (cumulative delay in minutes), for different lines and tradeoff α.

We also tried a variation of the *slim2* (and *LR*) objective function. The variation is motivated by observations in [12] about the optimal distribution of buffers on a single corridor. There, it was observed that buffers that are placed too early risk to be left unused, because the probability to face any delay at this early position is too small. As a consequence, it might be worthwhile to lower the weights $w_{i,j}$ arising in the early sections of the line. Figure 4 plots different parametric variants of the *slim2* objective function. All of the them obey a common formula, namely:

$$(1 - e^{-\lambda i})(len(h) - i)$$

parametrized in λ ($\lambda = \infty$ gives the original *slim2* weighing scheme). Table 3 reports the percentage improvements with respect to case $\lambda = \infty$ for *slim2* and *LR*, respectively. It turns out that the new objective function typically produces slightly worse results for *LR*, while *slim2* takes advantage of it for large values of λ. In any case, the improvement is not substantial (up to 3-4%).

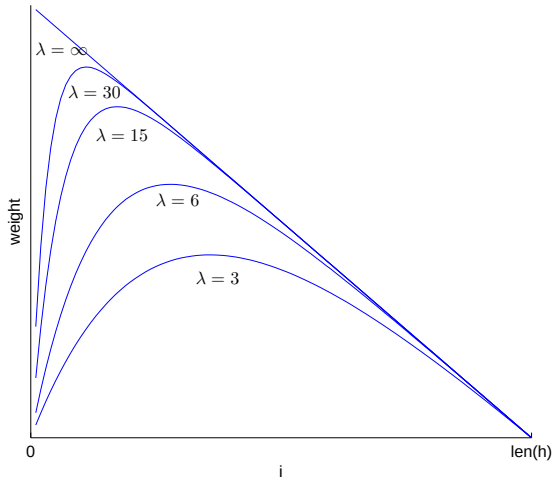

Fig. 4. Alternative weighing functions for *slim2* and *LR*, giving weight w_{ij} as a function of position i in the line.

One might also wonder what is the effect of the input nominal solution to the subsequent robustness improving phase. To answer this question, we performed the following experiment. We took the scheduled trains in the heuristic nominal solutions of [3] used in the previous experiments, and we constructed the MIP model described in Section 3, where the choice of precedences is left open. Then we collected a series of heuristic nominal solutions of increasing efficiency for that

Slim2

λ	BZVR				BrBO				MUVR				PDBO			
$\alpha =$	0.01	0.05	0.10	0.20	0.01	0.05	0.10	0.20	0.01	0.05	0.10	0.20	0.01	0.05	0.10	0.20
3	-1.9	2.6	1.7	3.5	0	-2.5	-2.3	-0.1	-1.2	-0.8	-1.2	-5.3	-2.3	0.2	-0.1	-1.7
6	-0.7	2.6	1.6	3	0.4	-1.3	0.6	2.8	-0.8	-0.9	0.2	-1.5	-1.8	0.2	1	3.6
15	0	3.7	1.7	4.9	1.1	1.2	3.4	3.8	-0.6	-0.3	0	1	-0.3	0.7	1.8	1.4
30	0.4	3.6	0.1	3.5	0.8	1.8	2.2	3.7	0	0.2	0.3	0.7	0.3	-0.2	1.1	1.8

LR

λ	BZVR				BrBO				MUVR				PDBO			
$\alpha =$	0.01	0.05	0.10	0.20	0.01	0.05	0.10	0.20	0.01	0.05	0.10	0.20	0.01	0.05	0.10	0.20
3	-0.3	-0.2	1.1	-2.2	-0.1	0.2	-0.5	-0.5	0.1	-0.4	1.2	0.2	-0.7	-0.7	-2.2	0.2
6	-0.2	-0.8	2	-2	0.1	0.6	0.5	-0.7	0.4	0.2	-0.1	1.3	-1.5	-0.3	0.2	-1.7
15	-0.3	-0.5	1.7	-1.1	0.1	0.3	0.4	-0.5	-0.2	0.7	1.3	0.6	-1.3	0	-0.7	1.3
30	0.1	-0.1	1.2	-0.7	0.3	0.2	0	-0.7	-0.4	0.4	0.9	-0.8	-0.4	-0.8	-0.4	-0.9

Table 3. Percentage robustness improvement with respect to $\lambda = \infty$ for the different weighing functions plotted in Figure 4; a negative value corresponds to a worse robustness.

model. This was obtained by running the MIP solver with a 5-minute time limit and by storing all the incumbent solutions produced during the run. Moreover, we ran the solver with a 1-hour time limit so as to produce an almost optimal solution of value, say, z_{ref}. (For all instances, the optimality gap after 1 hour was less than 4%.) Then, we compared the robustness achieved by our *fat* model when starting from these solutions, by allowing for a relative efficiency loss α with respect to z_{ref}. The left-hand side part of Table 4 gives the outcome of the experiment, for two instances (BrBO and MUVR). Columns correspond to the 10 best solutions obtained within the 5-minute time limit, sorted from left to right by increasing efficiency. E.g., for instance BrBO, the 10 solutions have a loss of efficiency ranging from 5.5% to 0.4% with respect to z_{ref}. Rows correspond to the different thresholds α used (1%, 5%, 10%, and 20%). The table entries then give the percentage increase in robustness (as measured by the validation tool) with respect to robustness measured when starting from the almost optimal solution of value z_{ref}. E.g., for BrBO, if we allow for a 10% efficiency loss with respect to z_{ref} and start from a nominal solution which is already 4.5% worse, we lose 13.9% in terms of robustness achievable through the *fat* training method. Missing entries correspond to infeasible cases.

As expected, starting from a worse nominal solution reduces the degree of freedom in the subsequent training phase, leading to a robustness loss. This negative effect could in principle be counterbalanced by the different precedence structure of the solution, in the sense that a less-efficient solution could involve precedence patterns leading to improved robustness. However, our experiments

seem to indicate that the precedence structure of the solutions plays only a secondary role. This support the viability of our approach, where only the most-efficient nominal solution available is "trained" for robustness.

To better quantify the effect of fixing all precedences when improving robustness of the nominal solution, we performed a second experiment consisting of solving the MIP version of the *LR* model where all precedences are left unfixed. Note that this is only viable for *LR*, since the other models are too large to be attacked by a general-purpose MIP solver. As in our previous experiments, we considered a loss of efficiency α ranging from 1 to 20% with respect to the almost-optimal solution value z_{ref}. The solution of value z_{ref} was also used to provide a first incumbent to the MIP solver. In these runs the MIP solver performed quite well, in that the optimality gap after 1 hour of computing time was less than 1% for all instances. (Note however that the model does not take into account the possibility of leaving some trains unscheduled.)

The results of our second experiment are reported in the right-hand side part of Table 4. For PDBO and BZVR, the MIP model did not find any better solution than the incumbent, so these cases are not reported in the table. The last column of Table 4 reports the percentage robustness improvement of the MIP *LR* model described above, over the linear *LR* model described in Section 6.3. E.g., for case BrBO with a threshold α of 10% with respect to z_{ref}, the MIP version of *LR* is able to improve by only 4.3% over the simple linear *LR*. Furthermore, the second-last column of Table 4 reports, for comparison sake, the percentage difference between the solution robustness obtained by the MIP *LR* and the robustness obtained by using *fat* on the same almost optimal solution z_{ref}. Results show that the new scheme produces only marginal robustness improvements with respect to the simple linear *LR*. This confirm that, for the cases in our testbed, the precedence structure of the solutions is not really important, efficiency being the key figure in determining the maximum achievable robustness. However, this may be no longer the case for more complex network topologies.

A simple yet often used in practice policy to enforce robustness in a timetable is to allocate a buffer that is just proportional to the train duration. Figure 5 gives the results of this simple policy on a sample instance, where we first compute the maximum total amount of buffer we can allocate for a given efficiency loss, and then distribute it proportionally along the line. According to the figure, the proportional buffer allocation policy and *slim1* behave quite similarly. This is not surprising, since model *slim1* actually favors a proportional buffer allocation—this is confirmed in the other instances as well (not shown in the figure). On the other hand, much better results are obtained by applying more clever optimization methods, showing the practical relevance of the optimization approaches.

While the validation output gives a reliable measure of how robust a solution is against delays, other figures exist that summarize somehow the "static" structure of a solution. These figures are useful to get insights into the structure of the solutions obtained with different training methods. In particular, we used

BrBO

	Fat											LR-MIP	
α												vs Fat	vs LR
eff(%)=	-5.5	-4.5	-3.9	-2.7	-2.2	-1.7	-1.3	-1.2	-0.8	-0.4	0.0	0.0	0.0
1%	–	–	–	–	–	–	–	–	-4.1	-2.7	0.0	-0.4	-0.1
5%	–	-20.3	-18.2	-8.1	-7.4	-4.4	-2.8	-3.1	-1.6	-2.2	0.0	1.7	4.1
10%	-23.9	-13.9	-15.2	-5.2	-5.6	-2.9	-1.9	-2.8	-1.4	-2.6	0.0	-1.0	4.3
20%	-22.2	-11.9	-14.9	-4.6	-4.5	-3.1	-2.1	-2.8	-2.4	-3.0	0.0	-11.8	2.7

MUVR

	Fat											LR-MIP	
α												vs Fat	vs LR
eff(%)=	-27.4	-14.9	-9.9	-9.2	-7.6	-6.8	-2.7	-1.6	-1.6	-1.3	0.0	0.0	0.0
1%	–	–	–	–	–	–	–	–	–	–	0.0	-0.6	0.0
5%	–	–	–	–	–	–	-3.7	-1.7	-1.2	-1.7	0.0	-1.2	-0.1
10%	–	–	-19.2	-16.5	-12.6	-10.1	-1.6	-0.8	-0.3	0.2	0.0	-1.4	1.1
20%	–	-25.5	-13.1	-12.7	-9.1	-8.6	-2.1	-0.9	0.1	-0.8	0.0	-4.3	1.4

Table 4. Effects of nominal input solution on robustness.

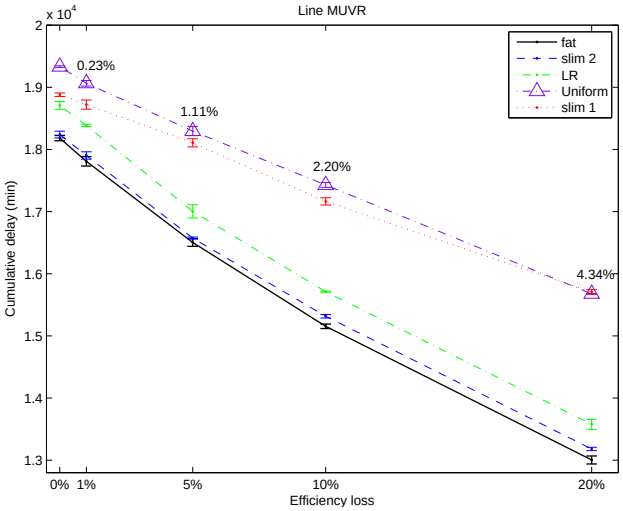

Fig. 5. Comparison of a simple "proportional" buffer allocation strategy against the proposed methods. The percentages shown are the total amount of buffer it was possible to allocate within a given tradeoff.

the weighted average distance (WAD) of the allocated buffer from the starting point. The WAD of the single train h is calculated as

$$WAD_h = \frac{1}{\sum_{i=1}^{len(h)-1} s_{i,i+1}} \sum_{i=1}^{len(h)-1} \frac{s_{i,i+1}(t^h_{i+1} + t^h_i)/2}{t^h_{len(h)} - t^h_1} \tag{34}$$

where $s_{i,i+1}$ is the amount of buffer allocated from t_i to t_{i+1}. The WAD is a number between 0 and 1 which measures how the buffers are distributed along the train trip. For example, a value of 0.5 means that the same amount of buffers were allocated in the first half and in the second half of the trip; values smaller or bigger than 0.5 relate to a shift in buffers distribution towards the begin or the end of the trip, respectively. The WAD of an entire line is calculated as the mean of all the WADs of the trains of the line. The reader is referred to [12] for a more detailed discussion.

A comparison of the various WADs is reported in Table 2 and illustrated in Figures 6 and 7. It can be seen that there is a significative correlation between the degree of approximation of the various WADs with respect to "perfect WAD" (WAD_{fat}) and the robustness of the solution–as computed by the validation tool and reported in Figure 2 and 3. In Figures 6 and 7, *slim1* WAD is almost always 50%, meaning a uniform allocation of buffers. On the other hand, the other methods tend to allocate buffers earlier in the line, resulting in a lower value of the WAD. Moreover, as the allowed efficiency loss increases (x axis), the WAD increases as well, meaning that uniform allocation becomes a good choice. We can also note that LR behaves better for small efficiency losses. Indeed, LR uses a fixed buffer β to deal with disturbances. When the problem is less constrained in efficiency, these buffers can become too small, and the LP solver will start to distribute the buffer excess, in a somehow unpredictable way, so as to meet the increased degree of freedom, thus degrading the performance of the method. E.g., this is the case of lines BZVR and PDBO. Moreover, BZVR and PDBO are more congested than other two instances, which also explains the better performance of the uniform allocation strategy.

Figure 8 reports how the buffers are distributed along the line. The figure is obtained by normalizing each line by the length of the corridor, and averaging the buffers allocated in each normalized line section. The averages are then normalized by the total amount of allocated buffer, so that the area of each chart approximately sums up to 1. E.g., *slim1* allocates buffers almost uniformly along the line—the particular structure of the timetable being responsible of local fluctuations. It is clear that *slim2* produces a very tight approximation of *fat*, while *slim1* does not. It is worth noting that LR uses a smoother allocation of buffers, while *slim1* yields a better approximation of their oscillations, but misses the global allocation policy. In this respect, *slim2* performs quite well instead. This is due to the fact that LR does not exploit directly the scenario information, thus it has to cope with very little information. Again, note that the poorest method (*slim1*) produces an almost uniform distribution of the buffers, whereas the best ones tend to allocate them earlier. This confirms the findings reported in [12].

Finally, given the intrinsic approximation of the stochastic methods due to the evaluation of the expectation, we have computed lower and upper bounds on the optimal solutions of the stochastic models, as described in Section 4. A typical plot obtained for the slim stochastic model is reported in Figure 9, showing very narrow estimation gaps. Similar results are obtained with the other models, except *fat* that behaves a little worse due the reduced number of scenarios.

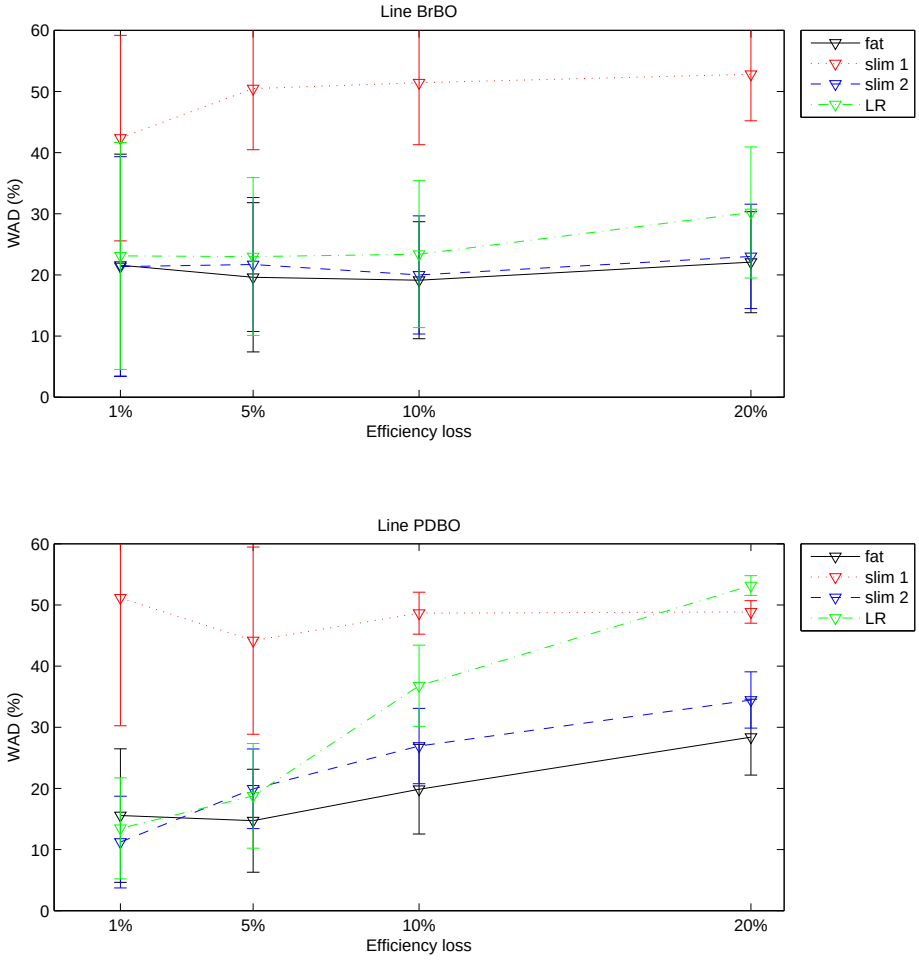

Fig. 6. Comparison of different training models from the WAD point of view (WAD is given within its confidence intervals).

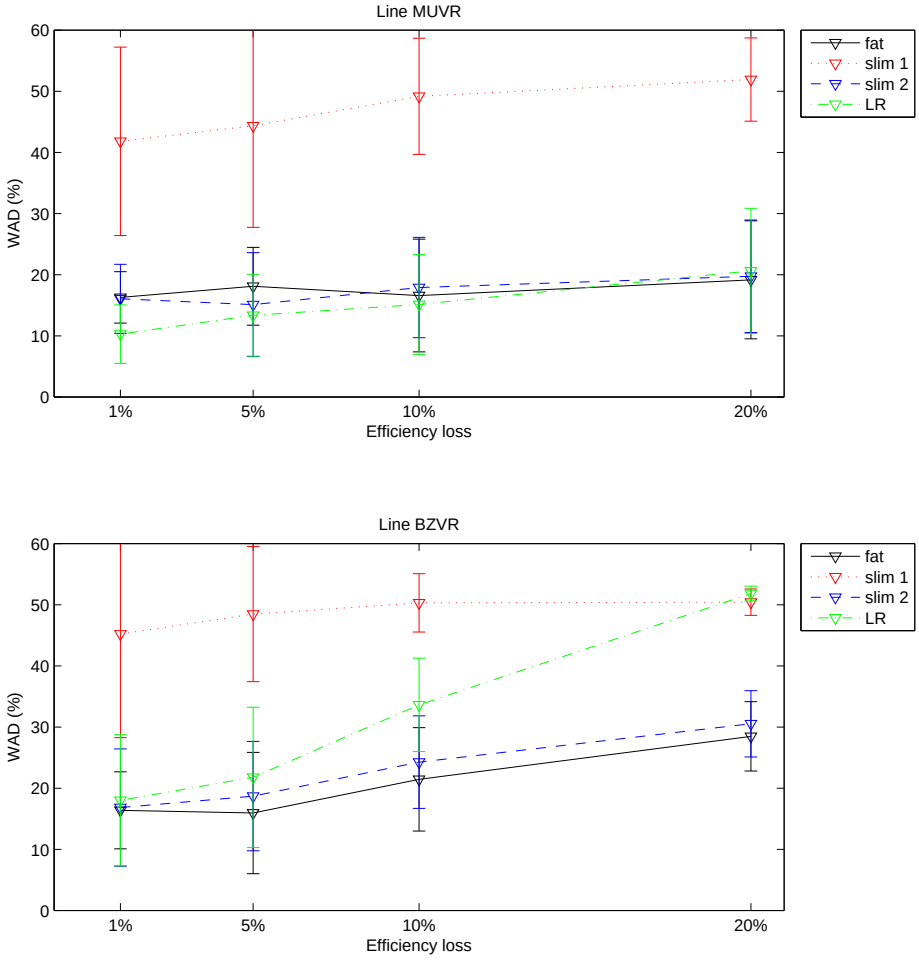

Fig. 7. Comparison of different training models from the WAD point of view (WAD is given within its confidence intervals).

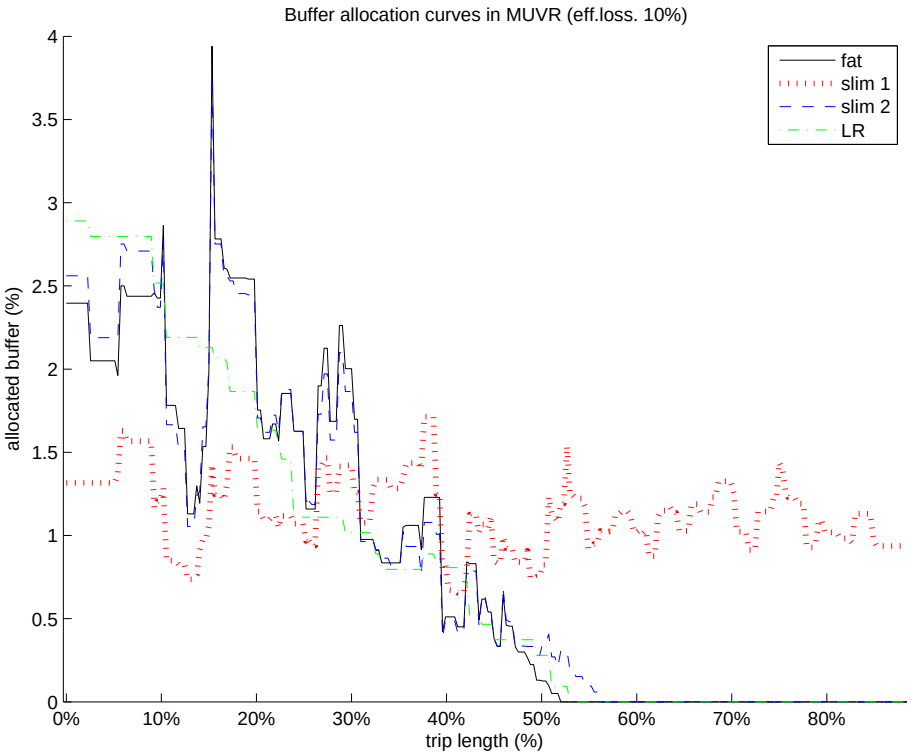

Fig. 8. Comparison of different training models from the allocated-buffer point of view.

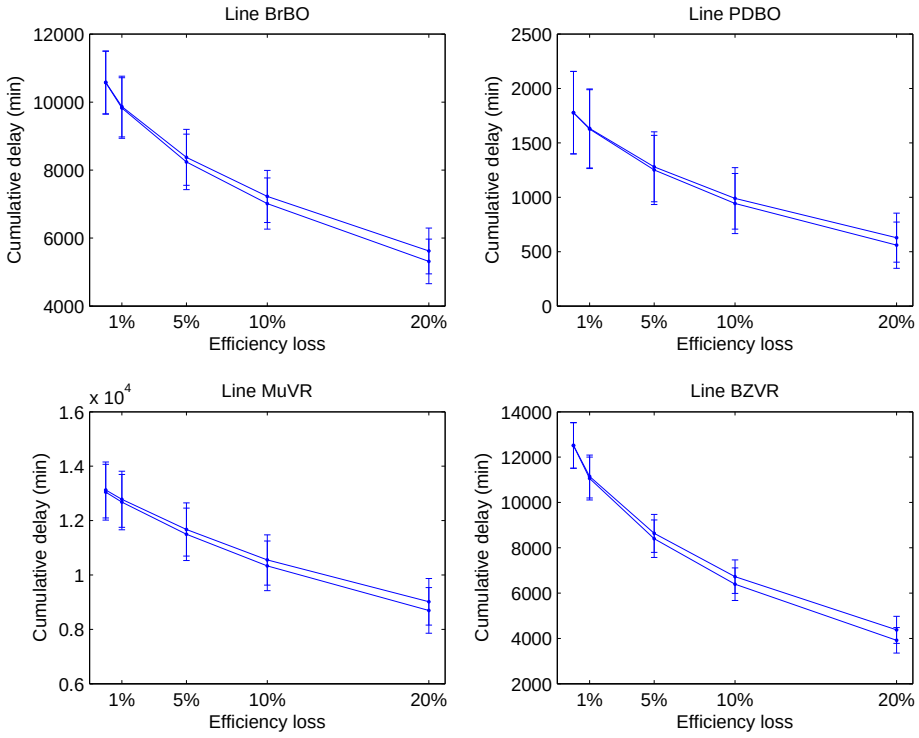

Fig. 9. Confidence intervals of upper and lower bounds of the optimal solution of stochastic model *slim2*

8 Conclusions and future work

In this paper we have described a three-stage framework as a practical tool for building and testing robust solutions for the Train Timetabling Problem. We mainly focused on robustness improvement of a given nominal solution. Robustness was then validated in terms of the total cumulative delay, computed by solving an LP model.

We examined different robustness improving models. The best performing, in terms of validated cumulative delay, is a "fat" stochastic reformulation of the nominal TTP problem. However, the solution of this model turned out to be very hard (if not impossible) for practical instances. A "slim" version performed much better, provided that a clever objective function is used. The fastest method, Light Robustness (LR), proved to be quite accurate when dealing with a reasonable robustness–efficiency tradeoff, allowing for a fast solution of large instances. On the whole, Light Robustness qualifies as a suitable tool for addressing large-scale real scenarios, and can even be embedded in the nominal solver to find optimized train-precedence patterns leading to more robust timetables.

Future direction of research should address the important topics below.

In the present paper, we quantified (for the *LR* model) the gain in terms of robustness resulting from relaxing the requirement that all precedences in the nominal solution must be preserved. It would be interesting to extend this analysis to the (much more difficult to solve) *slim2* model.

We performed our computations on real-world unidirectional corridors operated by the Italian railways operator; it would be interesting to address more complex network topologies.

Finally, in our study we used a simplified LP-based validation tool to estimate the cumulative delay in a set of random scenarios. An interesting research topic would be to measure the actual price required to recover a delayed timetable by using the same strategies used in real-world delay management.

Acknowledgments

This work was supported by the Future and Emerging Technologies unit of the EC (IST priority), under contract no. FP6-021235-2 (project "ARRIVAL") and by MiUR, Italy (PRIN 2006 project "Models and algorithms for robust network optimization"). We thank Paolo Toth, Alberto Caprara and Valentina Cacchiani for providing us with the nominal TTP solutions used in our computational study. Thanks are also due to three anonymous referees for their constructive comments.

References

1. F. Barber, S. Cicerone, D. Delling, G. D. Stefano, M. Fischetti, L. Kroon, D. Salvagnin, P. Tormos, C. Weber, and A. Zanette. New frameworks for the interaction between robust and online timetable planning, and for monitoring the status quo of the system. Technical Report ARRIVAL-D3.4, ARRIVAL Project, 2008.

2. J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming (Springer Series in Operations Research and Financial Engineering)*. Springer, 1st ed. 1997. corr. 2nd printing edition, 2000.
3. A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.
4. A. Caprara, M. Monaci, P. Toth, and P. Guida. A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, 154(5):738–753, 2006.
5. S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. On the interaction between robust timetable planning and delay management. Technical Report ARRIVAL-TR-0116, ARRIVAL project, 2007.
6. C. E. Clark. Importance sampling in Monte Carlo analyses. *Operations Research*, 9(5):603–620, 1961.
7. M. Fischetti and M. Monaci. Light robustness. Technical Report ARRIVAL-TR-0119, ARRIVAL Project, 2008.
8. M. Fischetti, A. Zanette, and D. Salvagnin. Fast approaches to robust railway timetabling. In C. Liebchen, R. K. Ahuja, and J. A. Mesa, editors, *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum f"ur Informatik (IBFI), Schloss Dagstuhl, Germany.
9. M. Hofman, L. Madsen, J. J. Groth, J. Clausen, and J. Larsen. Robustness and recovery in train scheduling - a case study from DSB S-tog a/s". In R. Jacob and M. Müller-Hannemann, editors, *ATMOS 2006 - 6th Workshop on Algorithmic Methods and Models for Optimization of Railways*. Internationales Begegnungs- und Forschungszentrum f"ur Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.
10. ILOG Inc. *ILOG CPLEX 10.1 User's Manual*, 2007.
11. A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.
12. L. Kroon, R. Dekker, and M. Vromans. Cyclic railway timetabling: a stochastic optimization approach. In *Algorithmic Methods for Railway Optimization*, Lecture Notes in Computer Science, pages 41–66. Springer Berlin / Heidelberg, 2007.
13. C. Liebchen, M. Lübbecke, R. H. Möhring, and S. Stiller. Recoverable robustness. Technical Report ARRIVAL-TR-0066, ARRIVAL-Project, 2007.
14. C. Liebchen and L. W. Peeters. On cyclic timetabling and cycles in graphs. Technical Report 761-2002, TU Berlin, Mathematical Institute, 2002.
15. C. Liebchen, M. Schachtebeck, A. Schöbel, S. Stiller, and A. Prigge. Computing delay resistant railway timetables. Technical Report ARRIVAL-TR-0071, ARRIVAL Project, October 2007.
16. C. Liebchen and S.Stiller. Delay resistant timetabling. Technical Report ARRIVAL-TR-0056, ARRIVAL Project, 2006.
17. J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, February 2006.
18. W. L. Loh. On latin hypercube sampling. *The Annals of Statistics*, 24(5), 1996.
19. W. K. Mak, D. P. Morton, and R. K. Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(24):10, February 1999.
20. K. Nachtigall. Periodic network optimization and fixed interval timetables. Habilitation Thesis, Deutsches Zentrum für Luft-und Raumfahrt, Braunschweig, 1999.

21. L. W. P. Peeters. *Cyclic Railway Timetable Optimization*. PhD thesis, Erasmus University Rotterdam, 2003.
22. A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming (Hanbooks in Operations Research and Management Series)*. Elsevier Publishing Company, 2003.
23. P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIJDM: SIAM Journal on Discrete Mathematics*, 2, 1989.
24. A. Shapiro. Monte carlo sampling approach to stochastic programming. In *ESAIM: Proceedings*, volume 13, pages 65–73, December 2003.
25. B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational and Applied Optimization*, 24, 2003.