Fast Approaches to Robust Railway Timetabling

Matteo Fischetti, Domenico Salvagnin, and Arrigo Zanette

DEI, University of Padova, Italy

Abstract. The Train Timetabling Problem (TTP) consists in finding a train schedule on a railway network that satisfies some operational constraints and maximizes a 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 find robust TTP solutions for the aperiodic (non cyclic) case, that combine Mixed Integer Programming (MIP) 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 many situations, the efficiency is measured as the distance of the solution from an input "ideal schedule" that optimally satisfies the network demands.

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.

The purpose of the present paper is to propose and evaluate new methods to find robust and efficient solutions to the TTP, in its aperiodic (non cyclic) version described in [2]. Our approach combines Mixed Integer Programming (MIP) with Stochastic Programming (SP) and Robust Optimization techniques. We developed a solution framework whose main building blocks are: (1) a *solver*, used to obtain a tentative timetable by solving an event-based MIP model; (2) a (local) *trainer* that uses Stochastic Programming or Robust Optimization techniques to improve the robustness of the tentative solution by changing the train departure/arrival times without altering the combinatorial structure of the tentative timetable (train precedences being preserved); and (3) a black-box *validation tool*, used to quantify the robustness of the solutions found by different approaches.

The paper is organized as follows. In Section 2 we present the TTP in detail and give a natural event-based MIP formulation. In Section 3 we present our overall solution framework, whose two main building blocks are described in Sections 4 and 5. 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.

2 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 [2], the aperiodic TTP can be formulated 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 the description of 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 tracks connecting them.

trains: a train is 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 departure times 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.

(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

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

i.e., the train profit decreases if the actual timetable diverges from the ideal one; trains with negative profit are intended to remain unscheduled and do not contribute to the overall profit.

Operational constraints include:

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

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

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

Although one is allowed to leave some trains unscheduled, to simplify our presentation we consider first a non-congested network 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 [11] can be sketched as follows:

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

$$t_{i+1}^h - t_i^h \geq d_{i,i+1}^h \quad \forall h \in T, i = 1, \dots, len(h) - 1 \tag{1}$$

$$|t_i^h - t_j^k| \geq \varDelta_a \quad \forall t_i^h, t_j^k \in A_S, \forall S \in \mathcal{S} \tag{2}$$

$$|t_i^h - t_j^k| \geq \varDelta_d \quad \forall t_i^h, t_j^k \in D_S, \forall S \in \mathcal{S} \tag{3}$$

$$t_{i+1}^h < t_{j+1}^k \Leftrightarrow t_i^h < t_j^k \quad \forall t_i^h, t_j^k \in D_S, \forall S \tag{4}$$

$$\rho_h = \pi_h - \sigma_h |t_1^h - \overline{t}_1^h| - \theta_h((t_{len(h)}^h - t_1^h) - (\overline{t}_{len(h)}^h - \overline{t}_1^h)) \quad \forall h \in T \tag{5}$$

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

where $\overline{t}$ denotes the ideal time of event t.

Constraints (1) impose a minimum time difference $d_{i,i+1}$ between two consecutive events of the same train, thus imposing minimum trip durations (trains are supposed to travel always at the maximum allowed speed for the track) and minimum rests 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, we need to introduce a set of binary variables $x_{i,j}^{h,k}$ to be set to 1 iff $t_i^h \leq t_j^k$ along with big-M coefficients M, so that conditions

$$|t_i^h - t_j^k| \geq \Delta$$

can be translated to

$$t_i^h - t_j^k \geq \Delta - M x_{i,j}^{h,k}$$

$$t_j^k - t_i^h \geq \Delta - M x_{j,i}^{k,h}$$

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

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

$$t_i^h < t_j^k \Leftrightarrow t_{i+1}^h < t_{j+1}^k$$

as

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

Constraints (5) define the profits of the trains.

Finally, constraints (6) correspond to 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 of 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. An easy procedure is 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 travellers as a small (less than one minute) delay.

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

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

gives the shift s_h of train h and can be easily linearized as

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

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

$$s_h \geq 0$$

If we are given a congested network we have to choose which trains to schedule in order to maximize the overall profit. This requires the introduction of new binary variables z_h such that

$$z_h = 1 \Leftrightarrow \text{train } h \text{ is scheduled}$$

and the modification of constraints (2)-(3) linking different trains in order to make them active only if both involved trains are scheduled. In particular

$$|t_i^h - t_j^k| \geq \Delta$$

becomes

$$|t_i^h - t_j^k| \geq \Delta(z_h + z_k - 1)$$

Notice that these modifications do not introduce further big-M coefficients. Moreover, we need to modify the definition of the profit variables in order to only count scheduled trains. Constraints (5) become

$$\rho_h \leq \pi_h - \sigma_h|t_1^h - \bar{t}_1^h| - \theta_h((t_{len(h)}^h - t_1^h) - (\bar{t}_{len(h)}^h - \bar{t}_1^h)) + M(1 - z_h)$$

and we add constraints

$$\rho_h \leq \pi_h z_h$$

3 The Overall Framework

In the nominal model, train travel times are always assumed to be minimal with respect to the safety operational constraints. However this is unlikely to happen in practice as travel times are often affected by delays. Therefore, safety operational constraints are too optimistic and one needs to address robustness issues, i.e., to modify the model in some way that allows one to gain a certain amount of robustness against delays while retaining an acceptable timetable efficiency.

In order to solve the robust problem we designed the following general framework:

nominal problem solution: we start by formulating the model in a mathematically tractable way and solve it (not necessarily to optimality) with an appropriate solver.

robustness training: borrowing an expression typical of AI field, starting from the nominal problem solution we "train" the model to robustness, typically by exploiting a restricted set of samples (scenarios). This crucial step can be implemented in different ways, and will be described in the sequel.

robustness validation: once we have obtained a robust solution, we evaluate its actual robustness by using a validation tool, thus allowing a fair comparison of different training methods.

4 Validation Model

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 railways system to the occurrence of delays, by introducing small adjustments to the planned timetable (received as an input parameter).

The guidelines used in designing the validation tool can be summarized as follows:

- *limited adjustability* in response to delays with respect to the given timetable. It is our belief that timetabling robustness is *not* concerned with major disruptions (which are 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 (4) 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:

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

$$t_i^\omega - t_j^\omega \geq d_{i,j} + \delta_{i,j}^\omega \;\forall (i,j) \in \mathcal{P} \tag{7}$$
$$t_i^\omega \geq t_i \quad \forall i \in E \tag{8}$$

Constraints (7) correspond to linear inequalities just explained, in which the nominal right-hand-side value $\delta_{i,j}$ is updated by adding the (possibly zero) extra-time $\delta_{i,j}^\omega$ from the current scenario ω.

Constraints (8) 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. As far as the upper bounds are concerned, we impose none, since we allow 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.

5 Finding Robust Solutions

In this section we present three different approaches to cope with robustness. In order to have tractable models, we introduced two simplifying hypotheses: (1) all input trains have to be scheduled; (2) all event precedences are fixed "in a clever way". This can be achieved by freezing the x and z variables in the MIP model of Section 2 according to an efficient heuristic solution.

5.1 A Fat Stochastic Model

Our first attempt to solve the robust version of the TTP was to use a standard scenario-based SP formulation akin to the one proposed by Kroon, Dekker, and Vromans [6] for the periodic TTP. The model can be outlined as:

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

$$\sum_{h \in T} \rho_h \geq (1-\alpha) z^* \tag{9}$$
$$t_i^\omega - t_j^\omega \geq d_{i,j} + \delta_{i,j}^\omega \;\forall (i,j) \in \mathcal{P}, \forall \omega \in \Omega \tag{10}$$
$$t_i^\omega \geq t_i \quad \forall i \in E, \forall \omega \in \Omega \tag{11}$$
$$t_i - t_j \geq d_{i,j} \qquad \forall (i,j) \in \mathcal{P} \tag{12}$$
$$l \leq t \leq u \tag{13}$$

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 decision variables to be optimized–their optimal value will define the final timetable to be published. The model keeps a copy of the original (linear) model with a modified right hand side for each scenario, along with the original model; 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 (9), 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 will be used for benchmark purposes.

5.2 A Slim Stochastic Model

Given the computing time required by the full stochastic model, we looked for an alternative model to solve, which is simpler yet meaningful for our problem. In particular, we propose the following recourse-based formulation:

$$\min \sum_{(i,j)\in\mathcal{P},\omega\in\Omega} w_{i,j}^{\omega} s_{i,j}^{\omega}$$

$$\sum_{h\in T} \rho_h \geq (1-\alpha)z^* \tag{14}$$

$$t_i - t_j + s_{i,j}^{\omega} \geq d_{i,j} + \delta_{i,j}^{\omega} \ \forall (i,j)\in\mathcal{P}, \forall\omega\in\Omega \tag{15}$$

$$s_{i,j}^{\omega} \geq 0 \ \forall (i,j)\in\mathcal{P}, \forall\omega\in\Omega \tag{16}$$

$$t_i - t_j \geq d_{i,j} \qquad \forall (i,j)\in\mathcal{P} \tag{17}$$

$$l \leq t \leq u \tag{18}$$

In this model we have just one copy of the original variables, plus the recourse variables $s_{i,j}^{\omega}$ which account for the unabsorbed extra times $\delta_{i,j}^{\omega}$. 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 (15) with $\delta_{i,j}^{\omega} = 0$, a situation that occurs very frequently in practice since most extra-times in a given scenario are zero.

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

5.3 Light Robustness

A different way to produce robust solutions is to use the *Light Robustness* approach proposed recently by Fischetti and Monaci [3]. This method is based on the consideration that, in essence, 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

$$\begin{aligned} \min f(\gamma) & \qquad (19)\\ Ax + \beta - \gamma \quad & \leq b \qquad (20)\\ c^T x & \leq (1+\alpha) z^\star \qquad (21)\\ x & \geq 0 \qquad (22)\\ 0 \leq \gamma & \leq \beta \qquad (23) \end{aligned}$$

where β_i is a positive 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 there is a bound (controlled by α) on the efficiency loss due to the increased robustness of the 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.

6 Solution of stochastic models

The stochastic models were solved using the SAA method (see [1],[10],[12] and [7]).

Sampling of delays has been carried out by 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 has its own probability P_h of arriving in the last line station with some amount of delay. The actual delay incurred by train h operating on section (i,j) in time interval $[l,k]$ is computed using the following formula:

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

where we normalize section delay probabilities in order to distribute the cumulative delay incurred by train T operating on line $\mathcal{L}$ through each line section.

We also implemented *latin hypercube* variance reduction technique when sampling from each distribution $P_{(i,j)}$, $P_{[l,k]}$ and P_h; see [8].

7 Computational Results

We carried our tests on four single-line medium-size TTP instances provided by the Italian railway company, Trenitalia. The main characteristics of the instances are outlined in Table 1.

An almost-optimal heuristic solutions for each of these instances was computed by P. Toth and his group using the algorithm described in [2], and used as a reference solution to freeze the event precedences and to select the trains to schedule.

We implemented our framework in C++ and carried out our tests on a AMD Athlon64 X2 4200+ computer with 4GB of RAM running Linux 2.6. The MIP solver used was ILOG CPLEX 10.1 (see [4]).

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

Table 1. Instance characteristics

As far as scenarios are concerned, for each train on the line and for each scenario we generated a corresponding 5% (on average) extra-time, drawn from an exponential distribution, and distributed it proportionally to its train segments.

Given this setting, the first test we performed was aimed at comparing the different training methods for each reference solution with different values of the tradeoff parameter α, namely 1%, 5%, 10%, 20% and 40%. In particular, we compared the following alternative methods:

- *fat*: fat stochastic model (50 scenarios)
- *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 the objective is set to $k+1$. The idea beyond this weighing policy is that early extra-times in a train sequence are likely to propagate to the next ones, so they are more important.

– *LR*: light robustness model, with objective function as in *slim2* and protection level parameters 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 of such distribution with probability $\frac{1}{2}$.

The results are shown in Table 2 and graphical representations (for two instances) are given in Figure 1.

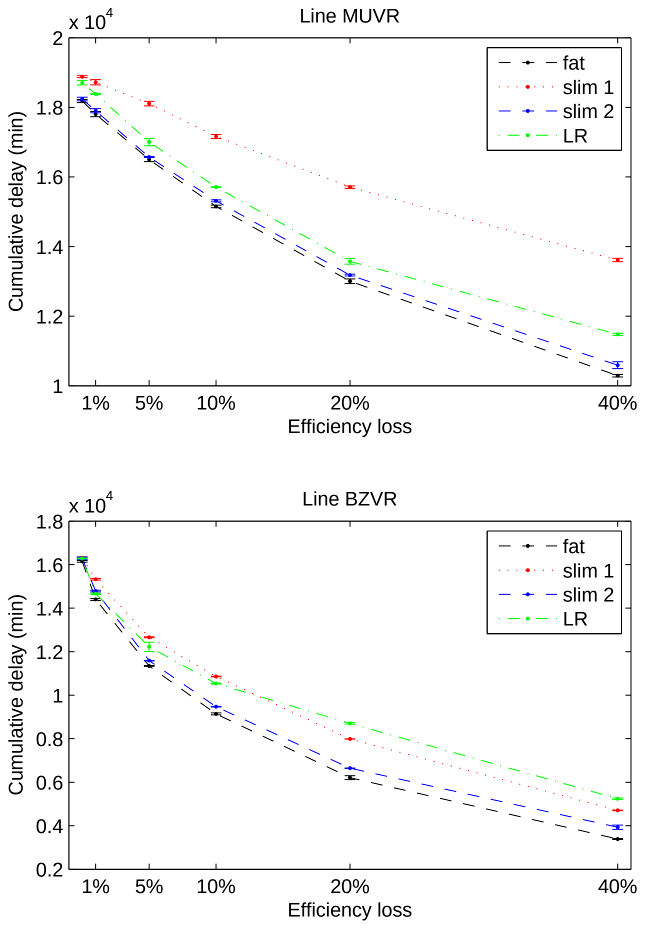

Fig. 1. Comparison of different training models applied to the best reference solution for each instance. The x-axis gives the efficiency loss (α) while the y-axis reproduces the confidence intervals of the validation figure (run with 500 scenarios).

	Fat			Slim1			Slim2			LR		
Line	Delay	WAD	Time (s)	Delay	WAD	Time (s)	Delay	WAD	Time (s)	Delay	WAD	Time (s)
BZVR α=0%	16149	–	9667	16316	–	532	16294	–	994	16286	–	2.27
BZVR α=1%	14399	16.4	10265	15325	45	549	14787	17	1087	14662	18	2.13
BZVR α=5%	11345	15.9	9003	12663	48	601	11588	19	982	12220	22	1.99
BZVR α=10%	9142	21.4	9650	10862	50	596	9469	24	979	10532	33	2.01
BZVR α=20%	6210	28.5	9072	7986	50	538	6643	31	1019	8707	52	2.04
BZVR α=40%	3389	35.4	10486	4707	50	578	3931	37	998	5241	51	2.31
BrBO α=0%	12156	–	384	12238	–	128	12214	–	173	12216	–	0.49
BrBO α=1%	11423	21.6	351	11646	42	134	11472	21	156	11499	23	0.48
BrBO α=5%	9782	18.9	357	11000	50	146	9842	22	164	10021	23	0.51
BrBO α=10%	8496	19.1	387	10179	51	132	8552	20	157	8842	23	0.51
BrBO α=20%	6664	22.1	375	8672	53	127	6763	23	153	7410	30	0.52
BrBO α=40%	4491	27.7	410	6212	52	130	4544	29	166	6221	52	0.53
MUVR α=0%	18182	–	377	18879	–	88	18240	–	117	18707	–	0.43
MUVR α=1%	17808	12.9	391	18721	37	96	17903	12	120	18386	8	0.48
MUVR α=5%	16502	14.5	385	18106	41	86	16574	13	107	17003	11	0.45
MUVR α=10%	15153	14.7	343	17163	49	84	15315	15	114	15710	13	0.43
MUVR α=20%	13004	17.1	384	15708	52	91	13180	18	116	13576	19	0.42
MUVR α=40%	10289	21.8	376	13613	52	95	10592	25	108	11479	34	0.45
PDBO α=0%	3141	–	257	3144	–	52	3139	–	63	3137	–	0.25
PDBO α=1%	2907	15.6	250	3026	51	57	2954	11	60	2954	13	0.27
PDBO α=5%	2412	14.7	223	2610	44	49	2508	20	57	2521	19	0.28
PDBO α=10%	1971	19.9	229	2244	49	50	2062	27	55	2314	37	0.25
PDBO α=20%	1357	28.4	230	1653	49	55	1486	34	60	1736	53	0.28
PDBO α=40%	676	37.1	262	879	49	55	776	41	57	1010	52	0.28
Tot:	198879	–	53246	219020	–	4293	201761	–	6960	209307	–	17

Table 2. Comparison of different training methods w.r.t. computing time, WAD and validation function (cumulative delay in minutes), for different lines and tradeoff α.

According to the figure, *slim2* always yields a very tight approximation of *fat*, while *slim1* is often poorer. As to *LR*, it usually produces good results (although not as good as *slim2*) when the tradeoff parameter α is small–which is the most relevant situation in practice.

As to computing times, the *fat* model is one order of magnitude slower than *slim1* and *slim2*, although it uses only 50 scenarios instead of 400. *LR* is extremely faster than any other method, more than two orders of magnitude w.r.t the fast stochastic models.

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: the weighted average distance (WAD) (see [6]) of the allocated buffer from the starting point. The WAD of the single train h is calculated as

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

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 buffer is 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 buffer 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.

A comparison of the various WADs for two instances is reported in Figure 2. 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 1.

Figure 3 illustrates how the buffers are distributed along the line for a sample instance. 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.

8 Conclusions

In this paper we have introduced and compared different methods to obtain robust train timetabling solutions. While the standard *fat* stochastic model is, as expected, too slow (if not intractable) for practical instances, two approximated models, namely the *slim* stochastic and *light robustness*, provide very good results in a short amount of time.

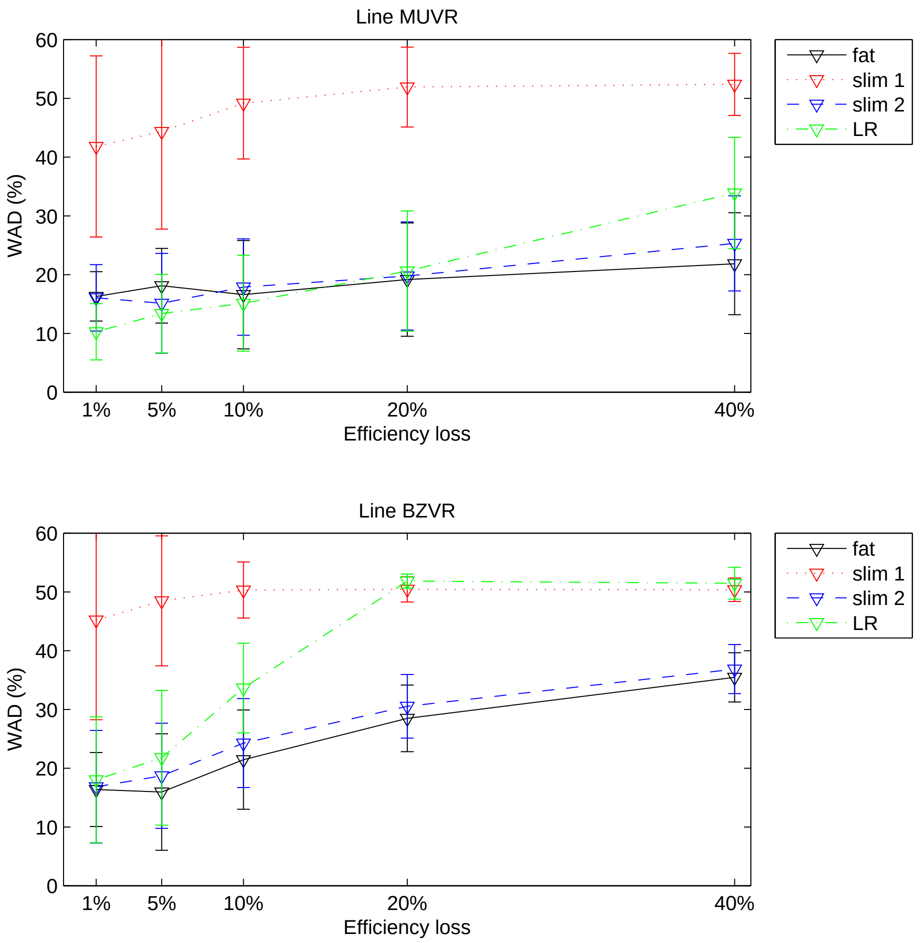

Fig. 2. Comparison of different training models from the WAD point of view (WAD is given within its confidence intervals).

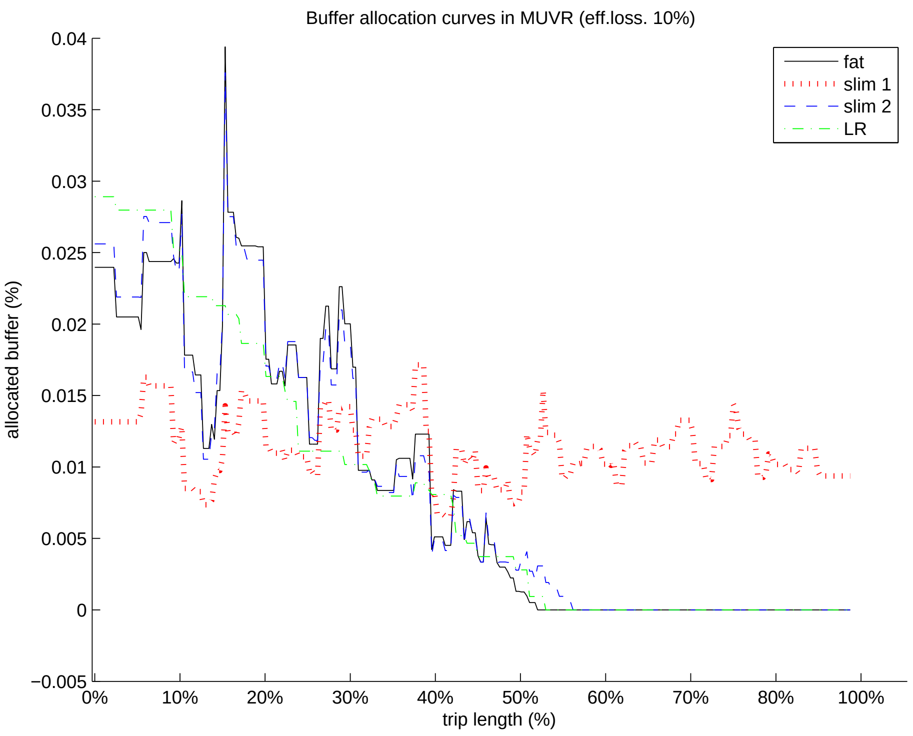

Fig. 3. Comparison of different training models from the allocated-buffer point of view.

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 (PRIN project).

References

1. 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.
2. A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.
3. M. Fischetti and M. Monaci. Robust optimization through branch-and-price. In *AIRO Proceedings*, Cesena, September 12-15 2006.
4. ILOG Inc. *ILOG CPLEX 10.1 User's Manual*, 2007.
5. A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM J. on Optimization*, 12(2):479–502, 2002.
6. L. Kroon, R. Dekker, and M. Vromans. Cyclic railway timetabling: a stochastic optimization approach. Research Paper ERS; ERS-2005-051-LIS, Erasmus Research Institute of Management (ERIM), RSM Erasmus University, Oct. 2005. available at http://ideas.repec.org/p/dgr/eureri/30007581.html.
7. 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.
8. W. L. Loh. On latin hypercube sampling. *The Annals of Statistics*, 24(5), 1996.
9. 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.
10. A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming (Hanbooks in Operations Research and Management Series)*. Elsevier Publishing Company, 2003.
11. P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIJDM: SIAM Journal on Discrete Mathematics*, 2, 1989.
12. A. Shapiro. Monte carlo sampling approach to stochastic programming. In *ESAIM: Proceedings*, volume 13, pages 65–73, December 2003.
13. 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. *Comput. Optim. Appl.*, 24, 2003.