Recoverable Robustness for Railway Rolling Stock Planning

Valentina Cacchiani[1], Alberto Caprara[1], Laura Galli[1],
Leo Kroon[2,3], Gábor Maróti[3], and Paolo Toth[1]

[1] D.E.I.S., University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy.
{alberto.caprara,valentina.cacchiani,l.galli,paolo.toth}@unibo.it
[2] Netherlands Railways, Utrecht, The Netherlands.
[3] Rotterdam School of Management, Erasmus University Rotterdam, P.O. Box 1738, NL-3000 DR, Rotterdam, The Netherlands. {gmaroti,lkroon}@rsm.nl.

Abstract. In this paper we explore the possibility of applying the notions of Recoverable Robustness and Price of Recoverability (introduced by [5]) to railway rolling stock planning, being interested in recoverability measures that can be computed in practice, thereby evaluating the robustness of rolling stock schedules. In order to lower bound the Price of Recoverability for any set of recovery algorithms, we consider an "optimal" recovery algorithm and propose a Benders decomposition approach to assess the Price of Recoverability for this "optimal" algorithm. We evaluate the approach on real-life rolling stock planning problems of NS, the main operator of passenger trains in the Netherlands. The preliminary results show that, thanks to Benders decomposition, our lower bound can be computed within relatively short time for our case study.

1 Introduction

Recently [5] introduced the concept of Recoverable Robustness as a generic framework for modelling robustness issues in railway scheduling problems. Also, the Price of Recoverability (PoR) was defined as a measure of recoverability. However, this notion is mainly of theoretical nature, and cannot be used in a straightforward way for concrete problems. In particular, computing PoR requires, according to [5], minimisation over a set of recovery algorithms, and it is not clear how this can be carried out.

Reference [3] considers the robustness of shunting problems. It overcomes these difficulties by analysing PoR for a few *concrete* recovery algorithms and by proving lower and upper bounds.

The purpose of this paper is to investigate another way of bringing the theoretical notion of PoR closer to practice. In particular, we consider what happens if recovery is done in the best possible way: the resulting value of PoR for this unique "optimal" recovery algorithm is then a lower bound on the value of PoR for *any* set of recovery algorithms. Under mild assumptions, this lower bound can be computed by solving a single mathematical program.

ATMOS 2008
8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2008/1590

We address the practical evaluation of the lower bound above for a specific case study, concerning the medium-term rolling stock planning problem of NS, the main operator of passenger trains in the Netherlands. It arises 2–6 months before the actual train operations, and amounts to assigning the available rolling stock to the trips in a given timetable. The objectives of the problem that is traditionally solved, call nominal problem, are related to service quality, efficiency, and – to a limited extent – to robustness. Reference [4] describes a Mixed Integer Linear Programming (MILP) model for this nominal problem. Using commercial MILP software, the solution times on real-life problems of NS are quite low, ranging from a few minutes (on most instances) to a couple of hours (on some particularly complex instances). A software tool based on this model has been in operation within NS since 2004.

The solutions of the nominal problem are optimal under undisrupted circumstances only. However, infrastructure failures, bad weather, and engine breakdowns often lead to disruptions where the nominal solution cannot be carried out any more. In such cases, disruption management must take place to come up with an adjusted rolling stock schedule. In this re-scheduling process, the original objective criteria are of marginal importance, the goal being to quickly find a feasible solution that is "close" to the nominal one and can be implemented in practice.

The computation of the lower bound on PoR mentioned above for our case study requires the solution of a very-large Linear Programming (LP) model, in which several possible disruption scenarios are considered. We propose a Benders decomposition approach for the solution of this LP, leading to a subproblem for each scenario. Our preliminary computational results on a real-life rolling stock planning instance of NS indicate that the method takes relatively short time, and widely outperforms the straightforward solution of the whole LP by a state-of-the-art solver.

This paper is structured as follows. In Section 2 we quote the definition of the Recoverable Robustness and Price of Recoverability from [5]. In Section 3 we describe the lower bound that we consider, based on a "best possible recovery" policy, and the associated mathematical programming problem. Section 4 describes the railway rolling stock scheduling problem of NS. Section 5 is devoted to our preliminary computational results. Finally, Section 6 outlines our plans for further research.

2 The Price of Recoverability

In this section we give a short summary of the definition of the Price of Recoverability by [5]. The main idea is to compute a solution to an optimisation problem and at the same time to analyse the recovery costs in case of disturbed input data. More concretely, one considers a (limited) set of scenarios with their own feasible regions, as well as a set of admissible recovery algorithms. The objective is to find a solution of the original (nominal) problem *and* a recovery algorithm in the given set. The requirement is that, using the recovery algorithm, the solution of the original optimisation problem can be transformed to a feasible solution of each scenario at "low cost". The Price of Recoverability measures both the objective function of the original problem and the recovery costs.

The set of admissible recovery algorithms can be chosen in several ways. One may consider algorithms with limited (e.g. linear) running time, or algorithms that are obtained from a particular heuristic framework (e.g. a crew re-scheduling algorithm based on iterated crew duty swaps).

The notions of Recoverable Robustness and Price of Recoverability are defined formally as follows. First of all, we are given a *Nominal Problem* of the form:

$$\mathrm{NP} = \min\{c(x) \mid x \in K\}, \tag{1}$$

where $x \in \mathbb{R}^n$ is the variable vector, $K \subseteq \mathbb{R}^n$ is the feasible region and $c : K \to \mathbb{R}_+$ is the cost function.

Moreover, we are given a set $\mathcal{S}$ of *scenarios*; each scenario $s \in \mathcal{S}$ having its own feasible region K_s. (For example, a scenario may refer to the case of cancelling some trains due to infrastructure failure, thereby requiring some kind of recovery action.) Furthermore, we are given a set $\mathcal{A}$ of *recovery algorithms*: a recovery algorithm $A \in \mathcal{A}$ takes on input a nominal solution $x \in K$ and a scenario $s \in \mathcal{S}$ and produces a solution $A(x,s) \in K_s$ which is feasible in scenario s. Finally, we are given, for $s \in \mathcal{S}$, a function $d_s : K \times K_s \to \mathbb{R}_+$ measuring the *deviation* $d_s(x, x_s)$ of a solution x_s for scenario s from a nominal solution x, and a monotone non-decreasing function $f : \mathbb{R}_+^{\mathcal{S}} \to \mathbb{R}_+$ penalising the deviation over all scenarios.

The *Recovery-Robust Optimisation Problem* defined in [5] is then:

$$\mathrm{RPOP}_{\mathcal{A}} = \min\{c(x) + f(z) \mid x \in K,\ A \in \mathcal{A},\ z_s = d_s(x, A(x,s))\ \ (s \in \mathcal{S})\}, \tag{2}$$

where $z = (z_{s_1}, z_{s_2}, \ldots) \in \mathbb{R}_+^{\mathcal{S}}$ is a vector of auxiliary variables representing the deviations.

Reference [5] chooses $f(z) = \max_{s \in \mathcal{S}} z_s$, i.e. penalises the *maximum deviation* in the objective function. A stochastic-programming approach would be to define a probability p_s for each scenario $s \in \mathcal{S}$, and to consider $f(z) = \sum_{s \in \mathcal{S}} p_s z_s$, penalising the *expected (average) deviation*.

The Price of Recoverability (PoR) is then defined as the ratio between the optimal values of the Recovery-Robust and the Nominal Problem:

$$\mathrm{PoR}_{\mathcal{A}} = \frac{\mathrm{RPOP}_{\mathcal{A}}}{\mathrm{NP}}. \tag{3}$$

Reference [5] also compares Recoverable Robustness to well-known concepts such as stochastic programming (see e.g. [2]) or robust optimisation ([1]), and discusses the similarities and differences of the approaches to capture robustness.

2.1 Reformulation of PoR

Define the function

$$\Phi_A(x) = c(x) + f\left(d_{s_1}(x, A(x, s_1)), d_{s_2}(x, A(x, s_2)), \ldots\right).$$

Then $\mathrm{RPOP}_{\mathcal{A}}$ corresponds to the minimisation of the function Φ:

$$\mathrm{RPOP}_{\mathcal{A}} = \min_{A \in \mathcal{A}} \min_{x \in K} \Phi_A(x). \tag{4}$$

In later sections of this paper we shall consider a simplified version for the case in which $\mathcal{A}$ contains a single algorithm A only:

$$\mathrm{RPOP}_{\{A\}} = \min_{x \in K} \Phi_A(x). \tag{5}$$

2.2 How to Compute PoR?

The definition (3) (via the definition (2)) requires minimisation over the set $\mathcal{A}$ of recovery algorithms. How (and if) this can be done clearly depends on how the set $\mathcal{A}$ is specified. In any case, one can follow (at least) two approaches to compute (or approximate) PoR.

In the first approach, one considers a class of small and well-behaved problems together with a small set of recovery algorithms. Then one proves worst-case bounds on PoR by an appropriate theoretical analysis. Reference [3] reports such results for the shunting problem. With our notation, such an approach essentially amounts to deriving bounds on the minimum of the function Φ_A for each $A \in \mathcal{A}$. Note that this approach is likely to succeed on fairly simplified test problems; real-life (railway) scheduling problems often have features that cannot be handled easily in theoretical worst-case proofs.

In this paper we follow a second approach, namely we restrict attention to the best possible recovery algorithm, observe that the computation of PoR for this single algorithm leads to a lower bound on PoR for each set $\mathcal{A}$, and numerically solve a mathematical programming problem to compute the value of this lower bound for a particular real-life railway resource scheduling problem. Therefore the results that we obtain are of empirical nature.

3 PoR with an Optimal Recovery Algorithm

Let $\mathcal{A}_{\mathrm{all}}$ be the set of *all* recovery algorithms and define

$$A_{\mathrm{opt}}(x, s) = \arg\min\{d_s(x, x_s) \mid x_s \in K_s\}. \tag{6}$$

In words, for each scenario s and for each $x \in K$, A_{opt} determines the solution in K_s with the smallest possible deviation from x. That is, A_{opt} represents the best possible recovery action. This is formalised in the following proposition.

Proposition 1 $\mathrm{RPOP}_{\mathcal{A}_{\mathrm{all}}} = \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$.

Proof. Clearly, $\mathrm{RPOP}_{\mathcal{A}_{\mathrm{all}}} \leq \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$. On the other hand, for each $x \in K$, $A \in \mathcal{A}$ and $s \in \mathcal{S}$ we have

$$d_s(x, A(x,s)) \geq d_s(x, A_{\mathrm{opt}}(x,s))\,.$$

Therefore

$$\min_{A \in \mathcal{A}_{\mathrm{all}}} \Phi_A(x) \geq \Phi_{A_{\mathrm{opt}}}(x)$$

and (4) yields

$$\mathrm{RPOP}_{\mathcal{A}_{\mathrm{all}}} \geq \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}.$$

In other words, the minimum of (2) if $\mathcal{A} = \mathcal{A}_{\mathrm{all}}$ is attained at A_{opt}. This of course implies that the minimum of (2) for a generic $\mathcal{A}$ cannot be better than $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$, as stated in the following corollary.

Corollary 2 $\mathrm{RPOP}_{\mathcal{A}} \geq \mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ *for every set* $\mathcal{A}$ *of algorithms.*

This implies that the computation of $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ yields a lower bound on $\mathrm{RPOP}_{\mathcal{A}}$, and therefore $\mathrm{PoR}_{A_{\mathrm{opt}}}$ a lower bound on $\mathrm{PoR}_{\mathcal{A}}$, for a generic set of recovery algorithms $\mathcal{A}$. Moreover,

$$\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}} = \min\{c(x) + f(z) \mid x \in K,\ x_s \in K_s\ (s \in \mathcal{S}),\ z_s = d_s(x, x_s)\ (s \in \mathcal{S})\}. \tag{7}$$

That is, $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ is the optimum value of a mathematical program, which is not the case for $\mathrm{RPOP}_{\mathcal{A}}$ for a generic $\mathcal{A}$. This is the reason why in this paper we focus our attention on the practical computation of the former.

3.1 Solution Methodology

For the sake of concreteness, we will restrict our attention to the case of $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ in which the following hold:

- $f(z) = \max_{s \in \mathcal{S}} z_s$, i.e. only the largest deviation is penalised in the objective function;
- $c(x) = c^{\mathsf{T}} x$ for a given $c \in \mathbb{R}^n$, i.e. the objective function is linear;
- $x \in K$ can be expressed as $Ax \geq b$ for given $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, i.e. feasibility of a nominal solution can be expressed by linear constraints (and possibly by the integrality of some components of x, see below);
- for each $S \in \mathcal{S}$, $x_s \in K_s$ can be expressed as $A_s x_s \geq b_s$ for given $A_s \in \mathbb{R}^{m_s \times n}$ and $b_s \in \mathbb{R}^{m_s}$;
- for each $S \in \mathcal{S}$, $z_s = d_s(x, x_s)$ can be expressed as $z_s = d_s^{\mathsf{T}} x + e_s^{\mathsf{T}} x_s + g_s$ for given $d_s, e_s \in \mathbb{R}^n$ and $g_s \in \mathbb{R}$, i.e. the deviation is a linear function of the nominal and the recovered solution.

If we include the possible integrality restriction on some of the x and x_s components, the above assumptions are not really restrictive, since they amount to require that the nominal problem, the feasibility of a recovered solution, and the value of the deviation can be expressed as a MILP. In the computational experiments carried over in this paper, we will restrict attention to the case in which such integrality restriction is not imposed. Depending on the specific application, this may be the case, or it may lead to solution of the LP relaxation of the actual MILP, which yields a lower bound on $\mathrm{RPOP}_{\{A_{\mathrm{opt}}\}}$ and therefore on $\mathrm{RPOP}_{\mathcal{A}}$ for each $\mathcal{A}$. In any case, the Benders decomposition approach that we illustrate can easily be modified to have integrality restrictions on the x variables. Since the purpose of this paper is to study the possibility to practically compute lower bounds on PoR for real-world instances, it is natural to restrict attention to LP relaxations.

Given the above assumptions, (7) can be formulated as follows, where λ is an auxiliary variable expressing the deviation penalty:

$$\begin{aligned}
\min \quad & c^{\mathsf{T}}x && + \lambda && && (8)\\
\text{s.t.} \quad & Ax && && \geq b, && (9)\\
& & A_s x_s & && \geq b_s, \quad \forall s \in \mathcal{S}, && (10)\\
& -d_s^{\mathsf{T}}x - & e_s^{\mathsf{T}}x_s & + \lambda && \geq g_s, \quad \forall s \in \mathcal{S}. && (11)
\end{aligned}$$

For solving (8) – (11) one can apply various mathematical programming techniques. In this paper we focus on Benders decomposition (also known as L-shaped method) (see e.g. [7]), a cutting plane method that exploits the block-diagonal structure of the problem. This is an approach widely used for such problems (such as for stochastic programming).

Briefly, the Benders decomposition approach keeps solving the (gradually extended) nominal problem (8) – (9). Based on the current optimal solution, the feasibility of the subproblem (10) – (11) is checked. The procedure terminates if the subproblem is feasible, in which case the current optimal solution is optimal also for (8) – (11). In case of infeasibility, inequalities in terms of x and λ are derived and added to the nominal problem, and the updated nominal problem is re-optimised.

Benders decomposition is applicable if the subproblems are LPs, i.e. if the x_s variables are continuous, whereas integrality on the x variables can be handled, although (as already mentioned) it will be relaxed in our computational experiments.

4 The Test Problem: Rolling Stock Re-scheduling

This section is devoted to the description of the specific real-world case study on which we focused our attention.

4.1 The Nominal Problem

We consider the medium-term railway rolling stock scheduling problem of NS. It arises 2–6 months before the actual railway operations, and has the task of assigning the available rolling stock to the *trips* in a given timetable. In this section we give a brief problem description. Further details about the problem can be found in [4] and in [6].

The rolling stock consists of *units*. Each unit has driver's seats at both ends and an own engine. It is composed of a number of carriages, and cannot be split up in every-day operations. Units are available in different *types* and can be combined with each other to form *compositions*. This allows a fine adjustment of the seat capacity to the passenger demand.

The timetable of NS is quite dense, and the turning time of the trains is short, often less then 20 minutes. The rolling stock connections are explicitly given in the input timetable by the *successor trips*: The units that serve in a trip go over to the successor trip, even though certain *composition changes* can take place. Due to the short turning times, the composition change possibilities are limited to *coupling* or *uncoupling* of one or two units at the appropriate side of the train.

The objective is three-fold. *Service quality* is measured by seat shortage kilometres. It is computed by comparing the assigned seat capacity to the a priori given expected number of passengers; by multiplying the number of unseated passengers by the length of the trip; and finally by summing these values over all trips. *Efficiency* is expressed by the carriage-kilometres which is roughly proportional both to the electricity or fuel consumption and to the maintenance costs. *Robustness* is taken into account by counting the number of composition changes. Indeed, coupling or uncoupling of units causes additional traffic through the railway nodes, and thereby may lead to delay propagation if some passing trains are late.

We note again that the the nominal problem is solved several months before the operations. This leaves enough time to plan the low-level train operations at the railway nodes. In particular, shunting drivers are scheduled to carry out the coupling and uncoupling operations. Moreover, the end-of-day rolling stock balances are such that the units are at the right place for the next day's operations.

In order to define a MILP for the problem, the set of rolling stock types is denoted by M, the set of trips by T, the set of compositions by P, and the set of stations by S. For any $m \in M$, a_m denotes the number of available rolling stock units of type m.

The main binary decision variables are $x_{t,p}$, expressing whether composition p is assigned to trip t. Moreover, we have the binary variables $z_{t,p,p'}$ whose value is 1 if trip t has composition p and if the successor of t has composition p'. The z variables are only defined for those triples (t, p, p') where the composition change from p to p' is allowed after trip t, i.e. the constraints on the composition changes are implicitly represented by these variables.

The stations are modelled by the *inventories*. The inventory of a station at a certain time instant consist of all units that are located there. The basic rule is that units to be coupled to a train are pulled from the inventory immediately upon departure, while uncoupled units are added to the inventory a certain time (say 30 minutes) after arrival. This ensures enough time for necessary shunting operations.

The integer variables $y_{t,m}$ count the inventories of the units of type m at the departure station of trip t right after the departure of trip t. The beginning-of-day and end-of-day inventories of station s of type m are represented by the variables $y^0_{s,m}$ and $y^\infty_{s,m}$.

Letting the successor of trip t be denoted by $\sigma(t)$, the departure station of t be denoted by $d(t)$, c, d be appropriate objective function coefficients, and α, β, γ be appropriate inventory coefficients, a MILP formulation is the following.

$$\min \sum_{t\in T}\sum_{p\in P} c_{t,p}x_{t,p} + \sum_{t\in T}\sum_{p\in P}\sum_{p'\in P} d_{t,p,p'}z_{t,p,p'} \tag{12}$$

$$\text{s.t.} \sum_{p\in P} x_{t,p} = 1, \qquad \forall t \in T, \tag{13}$$

$$x_{t,p} = \sum_{p'\in P} z_{t,p,p'}, \qquad \forall t \in T, p \in P, \tag{14}$$

$$x_{\sigma(t),p'} = \sum_{p\in P} z_{t,p,p'}, \qquad \forall t \in T, p' \in P, \tag{15}$$

$$\begin{aligned} y_{t,m} = y^0_{d(t),m} &+ \sum_{t'\in T}\sum_{p\in P}\sum_{p'\in P} \alpha_{t,t',p,p',m} z_{t',p,p'} \\ &+ \sum_{t'\in T}\sum_{p\in P} \beta_{t,t',p,m} x_{t',p}, \qquad \forall t \in T, m \in M, \end{aligned} \tag{16}$$

$$y^\infty_{t,m} = y^0_{d(t),m} + \sum_{t'\in T}\sum_{p\in P}\sum_{p'\in P} \gamma_{t,t',p,m} x_{t',p}, \quad \forall t \in T, m \in M, \tag{17}$$

$$\sum_{s\in S} y^0_{s,m} = a_m, \qquad \forall m \in M, \tag{18}$$

$$x_{t,p}, z_{t,p,p'} \text{ binary}, \qquad \forall t \in T, p \in P, p' \in P, \tag{19}$$

$$y_{t,m}, y^0_{s,m}, y^\infty_{s,m} \geq 0, \text{ integer}, \qquad \forall t \in T, s \in S, m \in M. \tag{20}$$

The objective (12) takes into account the trip assignments and the compositions of consecutive trips. Constraints (13) state that each trip gets exactly one composition. Constraints (14) and (15) link the z variables to the x variables. Constraints (16) and (17) compute the inventories with appropriate coefficients α, β and γ. Constraints (18) specify the available rolling stock.

The objective function can incorporate a wide variety of objective criteria related to service quality, efficiency and robustness. Experience shows that, for the practically meaningful objective coefficients, the LP relaxation of the model above is very tight, the associated lower bound being always within a few percents of the MILP optimum. The MILP model can be solved for medium-sized instances of NS within a few seconds to optimality. Simple LP rounding heuristics turned out to be powerful for the most challenging problem instances.

4.2 The Scenarios and the Associated Deviations

In our robustness framework, the solutions of the nominal problem are to be operated subject to disruption scenarios. Each scenario is obtained by assuming that a certain part of the network is blocked for a certain time interval of several hours. All the trips that interfere with the infrastructure blockage are removed. Such disruptions are quite common in practice. These are the ones that require significant resource re-scheduling.

It is worthwhile to note that the timetabling and resource scheduling decisions are strictly separated. In the Netherlands, for example, an independent infrastructure managing authority is responsible for the timetable adjustments, while the railway operators themselves are responsible for resource re-scheduling. Therefore from the resource planning's point of view, the adjusted timetable that takes care of the disruption is to be considered as input.

We assume that a disruption becomes known at the beginning of the blockage. The task is then to re-schedule the rolling stock from that point on till the end of the day. The solution has to fulfil the same requirements as the nominal problem, the only additional option being to cancel a trip.

In this research we also assume that the exact duration of the disruption is known at its beginning. Admittedly, this assumption is very optimistic for practical purposes. On the other hand, it simplifies the mathematical model, and still enables one to gain insight of the recovery capacity of rolling stock schedules.

The three main criteria in re-scheduling are as follows (in decreasing order of importance): (*i*) minimise the number of cancelled trips; (*ii*) minimise the number of newly introduced couplings and uncouplings; (*iii*) minimise the deviation of the planned end-of-day rolling stock balance. The first criterion limits the passenger inconvenience. The second criterion aims at keeping the schedule of the shunting drivers intact. The third criterion tries to restrict the consequences of the disruption on a single day.

Although the model (12) – (20) was originally developed for the nominal problem, it can be adjusted for rescheduling as well. That is, the feasibility of a recovered solution and the associated recovery costs can be computed as a variant of the model above. We express the model for a single scenario, omitting the index s that represents the scenario and noting that here s stands for the index of a station.

First of all, constraints (13) – (20) with variables $\widetilde{x}$, $\widetilde{z}$, $\widetilde{y}^0$ and $\widetilde{y}^\infty$ are to be stated for the trips of each scenario. In this case, as anticipated, we also allow the empty composition $\emptyset$, where $\widetilde{x}_{t,\emptyset} = 1$ means that trip t is cancelled. Then, one has to impose constraints that the rolling stock schedule is not changed until the beginning of the disruption, adding the constraints $\widetilde{x}_{t,p} = x_{t,p}$ for each $p \in P$ and trip $t \in T$ ending before the disruption. Finally, the model is extended to express the recovery costs:

$$\lambda \geq c_1 \sum_{t} \widetilde{x}_{t,\emptyset} + c_2 \sum_{t \in T} \widetilde{e}_t + \sum_{s \in S} \sum_{m \in M} \widetilde{d}_{s,m}, \tag{21}$$

$$\widetilde{d}_{s,m} \geq y^{\infty}_{s,m} - \widetilde{y}^{\infty}_{s,m}, \qquad \forall s \in S, m \in M, \tag{22}$$

$$\widetilde{d}_{s,m} \geq \widetilde{y}^{\infty}_{s,m} - y^{\infty}_{s,m}, \qquad \forall s \in S, m \in M, \tag{23}$$

$$w_t = \sum \left(z_{t,p,p'} \mid \ p \to p' \text{ is coupling or uncoupling} \right), \quad \forall t \in T, \tag{24}$$

$$\widetilde{w}_t = \sum \left(\widetilde{z}_{t,p,p'} \mid \ p \to p' \text{ is coupling or uncoupling} \right), \quad \forall t \in T, \tag{25}$$

$$\widetilde{e}_t \geq \widetilde{w}_t - w_t, \qquad \forall t \in T, \tag{26}$$

$$\widetilde{d}_{s,m} \geq 0, \qquad \forall s \in S, m \in M, \tag{27}$$

$$\widetilde{e}_t \geq 0, \qquad \forall t \in T. \tag{28}$$

The auxiliary variables $\widetilde{d}$ measure the deviation of the planned end-of-day rolling stock inventories (i.e. that of the nominal solution) from the realised end-of-day rolling stock inventories (i.e. those in the scenario). The value of the auxiliary variable w_t is 0 or 1 depending on whether the nominal solution has a composition change (i.e. coupling or uncoupling of units) after trip t. Similar role is played by $\widetilde{w}_t$ in the scenario. The auxiliary variable $\widetilde{e}_t$ has a value at least 1 if a new shunting is introduced after trip t, i.e. if there was no composition change after trip t in the nominal solution whereas there is one in the recovered solution. The objective penalises the variables $\widetilde{d}$ and $\widetilde{e}$ as well as all variables $\widetilde{x}$ that assign an empty composition to a trip.

5 Computational Results

We implemented the robust scheduling problem (8) – (11) with the rolling stock (re-)scheduling model described in Section 4 for the so called 3000 line of NS. This is an Inter-City line with a closed rolling stock circulation. The instance contains about 400 trips connecting 8 stations, and is served by two rolling stock types with 11 and 24 units, respectively. The 3000 line is one of the medium-sized rolling stock instances of NS.

The nominal problem is based on the actual timetable of NS. The scenarios have been generated artificially using a program of [8], which simulates the decisions of the infrastructure manager about train cancellations, including the successors of the trips after disruption. In that respect, the input data of the scenarios follow the same rules and assumptions as the nominal problem.

As already discussed, the goal of our preliminary computational tests is to investigate whether the suggested optimisation framework can be used at all to assess PoR for our rolling stock scheduling problem. Therefore we restricted ourselves to the solution of LP relaxations.

We implemented two solution methods: (*i*) solving (8) – (11) directly as a single LP; (*ii*) applying a canonical Benders decomposition Approach. Our computer codes are written in C and run on a personal computer, solving the LPs by ILOG CPLEX 10.0. The master problem has about 14,500 variables, 8,600 constraints and 310,000 non-zeros in the matrix.

The solution approaches have been tested with 2–20 scenarios, implying that the LPs solved by method (i) feature 43,000–305,000 variables, 25,000–180,000 constraints and 950,000–6,500,000 non-zeros.

For each number of scenarios, we solved two variants of the problem: Test-I and Test-II. They share the same constraint matrix but differ in the objective function. The cost coefficients are given in Table 1. Test-I focuses on service quality (by penalising seat shortages more heavily) while Test-II emphasises efficiency (by penalising carriage kilometres more heavily).

Table 1. Coefficients for the nominal objective function as well as for the recovery costs.

Criterion in nominal problem	Test-I	Test-II
seat shortage km	100	50
carriage km	9	100
composition change	5	10
Criterion for recovery	**Test-I**	**Test-II**
cancellation	1,000,000	1,000,000
inventory deviation	20,000	20,000
new shunting	10,000	10,000

The computational results with the two solution approaches are summarised in Table 2. It turns out that the huge LPs in method (i) are barely solvable. For more than 6 or 7 scenarios, the solution time exceeds our time limit of 1800 seconds. The cases with 10 or more scenarios appear to be far from being solved after several hours of CPU time. The Benders decomposition approach, on the other hand, is able to cope with the problems. After applying 24–640 and 30–360 Benders cuts for Test-I and Test-II, respectively, optimality was reached within the time limit.

The above results prove that, at least for our case study, the general lower bound on PoR that we propose can be computed within reasonable time.

6 Summary and Future Research

In this paper we summarised our understanding of the Price of Recoverability. In addition, we proposed a mathematical programming approach to compute a lower bound on the Price of Recoverability for real-life railway scheduling problems.

A Benders decomposition approach has been implemented for a medium-sized rolling stock scheduling problem of NS. The preliminary computational results indicate the problem is widely tractable with up to 20 disruption scenarios. Note

Table 2. The number of applied Benders cuts as well as the running times in seconds for the Benders decomposition approach and for the direct solution of the whole LP (referred to as 'CPLEX') on Test-I and Test-II. A dash indicates the running time of CPLEX exceeding 1800 seconds.

	Test-I			Test-II		
	Benders		CPLEX	Benders		CPLEX
# scens.	# cuts	CPU time	CPU time	# cuts	CPU time	CPU time
2	24	22	122	32	30	75
3	27	27	366	24	25	372
4	140	224	795	100	141	880
5	175	264	1,055	125	175	1,078
6	168	277	1,962	150	209	—
7	210	319	—	140	205	—
8	248	454	—	152	223	—
9	288	603	—	171	278	—
10	320	688	—	190	332	—
11	352	734	—	209	364	—
12	384	798	—	228	400	—
13	325	662	—	234	439	—
14	420	1,014	—	252	499	—
15	450	1,090	—	300	653	—
16	480	1,163	—	320	698	—
17	595	1,710	—	306	642	—
18	540	1,380	—	324	701	—
19	570	1,459	—	342	730	—
20	640	1,840	—	360	816	—

that the whole problem (a single LP) cannot be solved within several hours even with just 10 scenarios.

In order to improve the proposed method, we will go on with more thorough computational tests. First, the preliminary computations concern the LP relaxation of the rolling stock scheduling problem; we are going to study the original MILP models as well. Second, the current way of selecting the Benders cuts is very simple; we are going to evaluate the effect of more sophisticated cut selection methods. Third, Benders decomposition is not the only possible approach for solving (8) – (11). In our future research we are going to explore other mathematical programming techniques, such as convex optimisation (e.g. through the subgradient algorithm). Last but not least, we are going to investigate implications of the Price of Recoverability to railway practice.

Acknowledgment

This work was partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

References

1. D. Bertsimas and M. Sim. The Price of Robustness. *Operations Research*, 52(1): 35–53, 2004.
2. J. Birge and F. Louveaux. *Introduction to Stochastic Programming.* Springer, New York, 1997.
3. S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust Algorithms and Price of Robustness in Shunting Problems. In *Proceedings of the 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS)*, 2007.
4. P. Fioole, L. Kroon, G. Maróti, and A. Schrijver. A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174:1281–1297, 2006.
5. C. Liebchen, R. Möhring, M. Lübbecke, and S. Stiller. Recoverable robustness. Technical Report ARRIVAL-TR-0066, ARRIVAL Project, 2007.
6. G. Maróti. *Operations Research Models for Railway Rolling Stock Planning.* PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2006.
7. G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.
8. L. Nielsen. Disruption Generator for Railway Rolling Stock Re-scheduling, 2008. Software.