A Lagrangian Heuristic for Robustness, with an Application to Train Timetabling

Valentina Cacchiani[1], Alberto Caprara[1], Matteo Fischetti[2]

[1]DEIS, University of Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy
e-mail:{valentina.cacchiani,alberto.caprara}@unibo.it
[2]DEI, University of Padova, Via Gradenigo 6/A, I-35131 Padova, Italy
e-mail:matteo.fischetti@unipd.it

Abstract

Finding robust yet efficient solutions to optimization problems is a major practical issue that received large attention in recent years. Starting with stochastic programming, many of the approaches to robustness lead to a significant change in the problem formulation with respect to the non-robust (nominal) case. Besides requiring a much larger computational effort, this often results into major changes of the associated software.

Lagrangian heuristics form a wide family of methods that work well in finding efficient (i.e., low cost) solutions for many problems. These methods approximately solve a relaxation of the problem at hand through an iterative Lagrangian optimization scheme, and apply several times a basic heuristic driven by the Lagrangian dual information (typically, the current Lagrangian costs) so as to hopefully update the current best feasible solution. In this context, the underlying Lagrangian optimization iterative method has the main purpose of producing "increasingly reliable" Lagrangian costs, while diversifying the search in the last iterations, when the Lagrangian bound is very close to convergence.

The purpose of this paper is to propose, for the first time to the best of our knowledge, a very simple modification of the Lagrangian optimization scheme capable of dealing with robustness. This modification is based on two simple features: (a) we modify the problem formulation by introducing artificial parameters intended to "control" the solution robustness; and (b) during Lagrangian optimization, we dynamically change the weight of the control parameters so as to produce subproblems where robustness becomes more and more important. In this way, during the process we can easily collect a set of, roughly speaking, "Pareto optimal" heuristic solutions that have a different tradeoff between robustness and efficiency, and leave the final user the choice of the ones to analyze in more details—e.g., through a time-consuming validation tool.

As a proof-of-concept, our approach is applied to the well-known aperiodic Train Timetabling Problem (TTP) on a corridor, and is computationally analyzed on real-world test cases from the Italian Railways, showing that it produces within much shorter computing times solutions whose quality is comparable with those produced by existing approaches to robustness. This proves the effectiveness for the specific application, suggesting that a simple modification of existing Lagrangian heuristics is a very promising way to deal with robustness in many other cases.

Keywords: Lagrangian Heuristics, Robustness, Train Timetabling, Railways Optimization, Computational Analysis.

1 Introduction

Finding robust yet efficient solutions to optimization problems is a major practical issue that received large attention in recent years. The first approach in this direction is the use of stochastic programming, in which not only the so-called nominal scenario, but also all the alternative ones,

along with an associated probability, are explicitly considered in the problem formulation. Of course, this approach results in a much larger model than in the nominal case, whose solution is unavoidably way more time consuming. Moreover, any solution software for the nominal problem requires major modifications (and possibly to be rewritten from scratch) to be adapted to the robust case according to stochastic programming. Although alternative approaches to robustness exist, that tend to be less cumbersome than stochastic programming, for many of them these two aspects remain, namely the computing times are much higher and the software modifications are major with respect to the nominal case.

Lagrangian heuristics form a wide family of methods that work well in finding efficient solutions for many problems. These methods approximately solve a relaxation of the problem at hand through an iterative Lagrangian optimization scheme. During execution, a basic heuristic driven by the Lagrangian dual information (typically, the current Lagrangian costs) is applied several times so as to hopefully update the current best feasible solution. In this context, the underlying Lagrangian optimization iterative method has the main purpose of producing "increasingly reliable" Lagrangian costs, while diversifying the search in the last iterations, when the Lagrangian bound is very close to convergence.

The main goal of the present paper is to investigate the performance of a simple modification of a given Lagrangian heuristic, that is able to cope with robustness. In doing so, we expect to obtain a rather cheap (in terms of programming effort) yet effective (in terms of both solution cost and robustness) heuristic, that can be competitive with other approaches to robustness from the literature. To the best of our knowledge, this approach is new. On the other hand, note that our aim is not to introduce a new formal concept of robustness, but rather to borrow from exiting ones, adapting them to be dealt with within the Lagrangian framework.

Our starting observation was that the Lagrangian optimization scheme is "natively" suited to deal with robustness, provided that two simple features are embedded into it: (a) the problem formulation is modified by introducing artificial parameters as "control variables" for solution robustness; and (b) during Lagrangian optimization, the weight of the control variables is gradually increased so as to produce subproblems where robustness becomes more and more important. In this way, during the process we can easily collect a set of, roughly speaking, "Pareto optimal" heuristic solutions that have a different tradeoff between robustness and efficiency, and leave the final user the choice of the ones to analyze at a later time in more details—e.g., through a time-consuming validation tool.

To test our method, we focused on a railways problem that is often attacked through Lagrangian heuristics, namely, the Train Timetabling Problem (TTP). This problem has been widely studied both in its periodic (i.e., cyclic) and aperiodic versions. Most of the work from the literature deals with the *nominal* problem, where uncertainty is not taken into account. Here, we are given a set of "ideal timetables" for a set of trains, that however cannot be implemented due to capacity restrictions. The goal is to change these ideal timetables as little as possible, while satisfying the track capacity constraints (trains can be cancelled if needed).

As briefly reviewed in Section 2, different definitions (and measures) of robustness have been applied to TTP. For the purpose of the present work, we decided to build on the recent approach of Fischetti, Salvagnin and Zanette [11], using the same robustness concepts and measures. The proof-of-concept here is to show how a relatively minor modification of a "nominal" Lagrangian heuristic is able to produce solutions with a comparable (or even better) robustness than in [11], but within a significantly shorter computing time. Our method is tested on real-world instances of

the Italian Railways. The robustness of the heuristic solutions found is measured, *a posteriori*, by using the same simulation-based validation tool as in [11]. In spite of its simplicity, the proposed approach turns out to be rather effective and (at least) competitive with other more sophisticated methods. As expected, one of the main advantages of our method is that robust solutions require only a small computational overhead with respect to the nominal problem, so large scale instances can be attacked. In addition, the method produces several robust solutions with different efficiency and robustness levels, among which one is allowed to choose.

Note that the adaptation of the Lagrangian optimization scheme to deal with robustness had to be carefully defined for the specific problem at hand. On the other hand, we are convinced that analogous, specific, but still minor adaptations can be found for other relevant problems, although this is out the scope of the present paper.

The paper is organized as follows. The main literature on robust TTP is briefly reviewed in Section 2. In Section 3 we formally describe the nominal version of our TTP, and illustrate a known time-space graph representation of the problem along with an associated ILP formulation. In Section 4 we modify the ILP formulation by adding the "control variables" to deal with robustness, whereas in Section 5 we describe our Lagrangian heuristic algorithm. Computational experiments on real-world instances from Trenitalia (the main Italian railways operator for passengers) are presented in Section 6. Conclusions and future research are finally discussed in Section 7.

2 Literature on Robust TTP

In this section, we review the main papers investigating the concept of *robustness* related to TTP. The reader interested in the various notions of robustness for railway optimization in general is referred to, e.g., [1], whereas surveys on the nominal TTP, in both periodic and aperiodic variants, can be found in [1, 6, 8].

Roughly speaking, a TTP solution is considered to be robust if it avoids delay propagation as much as possible. A common way to avoid delay propagation is to introduce *buffer times* in the planning phase. Buffer times correspond to empty time slots in the time schedule of the train, used to absorb a possible delay. The question is how much and where the buffer times have to be inserted, so as to guarantee a good tradeoff between the nominal quality (efficiency) and the delay resistance (robustness).

A stochastic programming approach was used by Kroon et al. in [12] for the periodic version of TTP, and tested on real-world instances from NS (the main operator of passenger trains in the Netherlands). Computational experiments showed that the robustness of a timetable can greatly be improved by only minor modifications of the timetable, obtaining rather stable results under variation of the intensity of the disturbance distributions.

Fischetti and Monaci [10] later proposed a general heuristic scheme for robustness called *Light Robustness*. A set of slack variables was used to measure (an estimate of) the solution robustness, and their sum was minimized in the objective function. In addition, a maximum worsening of the solution efficiency with respect to the best nominal value was imposed as a hard constraint. For TTP, this approach was implemented by Fischetti et al. in [11] and compared with stochastic programming methods. In their work, TTP was modelled as a Periodic Event Scheduling Problem (PESP) (see [15]), adapted for the aperiodic case, and a validation tool was applied to assess (off-line) the quality of the final robust solution in terms of delay absorption.

The general notion of *Recoverable Robustness* was introduced by Liebchen et al. in [13] and

applied in the TTP context. This framework combines recovery policies and robust planning in the same concept. It deals with a limited set of scenarios (arising, e.g., from uncertainty in driving and stopping times) and of recovery algorithms. The new notion was also used by D'Angelo et al. in [9] for the case of TTP on single line corridors. These authors proposed an algorithm that solves the problem of planning robust timetables when the input event activity network topology is a tree. The algorithm was tested on real-world instances of the Italian Railways and the "price of robustness" was computed with respect to different scenarios.

Liebchen et al. in [14] also addressed the robust version of the periodic TTP. Delay resistant timetables are computed by means of an objective function that lies in between the traditional timetabling objective and the delay management objective. The authors evaluated the delay resistance of a timetable under several delay scenarios, and tested their method on real-world data of a part of the German railway network of Deutsche Bahn AG.

A genetic robust TTP algorithm was developed by Tormos et al. in [17] and tested on real-world instances of the Spanish Rail Network. It focused on an efficient allocation of time buffers so as to obtain timetables that are less sensitive to disturbances, while keeping the total travel time at satisfactory values.

3 Our nominal TTP

In this section we describe the nominal TTP that we consider in this paper and briefly recall, for the sake of clarity, a formulation based on a time-space graph representation of the problem. We refer the reader to Caprara et al. [7] for a detailed description.

In what follows we focus on the study of TTP on a corridor, i.e., on a single one-way line connecting two major stations. Our method easily generalizes to railway networks of arbitrary topology, as illustrated in [4], but here we restrict to the corridor case in order to be able to compare with the method of [11] and to use the associated validation tool. We are given a sequence of stations $S = \{1, \ldots, s\}$ along the corridor, 1 being the first station and s the last one, and a set of trains T, each having to travel along a subsequence of stations (i.e., not necessarily from 1 to s). For each train an ideal timetable (i.e., the timetable suggested by a train operator) is also given. In the nominal TTP the aim is to change the ideal timetable for the trains "as little as possible", while satisfying the track capacity constraints, imposing a minimum headway time between two consecutive departures from a station and between two consecutive arrivals at a station, and allowing overtaking only within stations. To obtain a feasible timetable one is allowed to change the departure of any train from its first station (*shift*) and/or to increase the stopping time in one or more of the visited stations (*stretch*). Each train is assigned an ideal (positive) profit which is gained if it is scheduled according to its ideal timetable. The profit values, assigned by the train operators, depend on the type of the train (intercity, regional, etc.): a higher profit means that the train operator is willing to schedule the corresponding train according to the suggested ideal timetable (without any change). The profit is decreased (according to a linear function) if shift and/or stretch are applied; if the profit becomes null or negative, the train is cancelled. Shift and stretch penalty functions are given on input, defined according to the suggestions of the train operators.

A common approach to deal with the nominal TTP [7] is to discretize the time horizon (e.g., by considering one time instant for each minute in a day) and to formulate the problem on a time-space directed graph $G = (V, A)$. Each node corresponds to a possible time instant where a train can

depart from or arrive at a station along the line. In addition, we consider a dummy source node σ and a dummy sink node τ. Each arc in A represents the travel of a train from a station to the next one (segment arc), or the stop of a train at a station (station arc). Node σ is connected to every node corresponding to a possible departure of a train from its first station, while every node corresponding to a possible arrival of a train at its last station is connected to node τ. Given the graph representation above, each feasible timetable for a train corresponds to a suitable path from σ to τ in G. In Figure 1, copied from [7], we show an example of the time-space graph G, for a corridor with 4 stations and 3 trains. Time is represented on the horizontal axis. Each station is represented by two lines, except from the first station and the last station. Nodes on the station lines correspond to departure and arrival time instants. Train t^1 departs from station 2, stops in station 3 and arrives in station 4; train t^2 departs from station 1, stops in stations 2 and 3 and arrives in station 4; train t^3 departs from station 1, stops in station 2 and arrives in station 3. In Figure 1, we show three feasible paths (i.e., timetables) from σ to τ for the three trains.

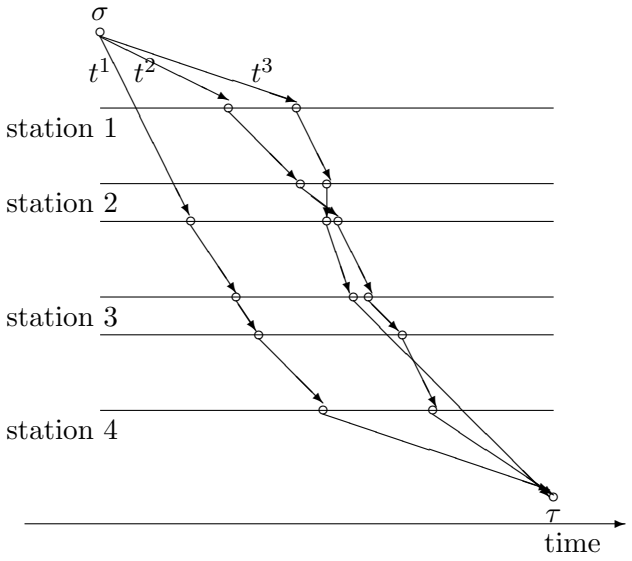

Figure 1: An example of time-space graph G (with $s = 4$, $|T| = 3$).

We next introduce some further notation.

- $A^t \subseteq A$ is the set of the arcs in A that may be used by paths for train $t \in T$;
- x_a is a binary variable that has value 1 if arc $a \in A^t$ is selected in the solution for train $t \in T$, and 0 otherwise;
- p_a is the profit gained if arc $a \in A^t$ is selected in the solution: if arc a connects σ to a node of the first station of the train, then p_a corresponds to the ideal profit of the train minus the shift penalty; for any station arc a, p_a is the stretch penalty; p_a is null for any other arc; notice that p_a can be positive, null or negative for any arc a that connects σ to a node of the first station of the train, depending on the shift penalty; as to station arcs, p_a is always negative because it represents the penalty for increasing the stopping time at the station;
- $\mathcal{C}$ is the collection of all the cliques of pairwise incompatible arcs, due to track capacity constraints.

The ILP formulation then reads:

$$\max \sum_{t \in T} \sum_{a \in A^t} p_a x_a, \tag{1}$$

$$\sum_{a \in \delta^+(\sigma) \cap A^t} x_a \leq 1, \quad t \in T, \tag{2}$$

$$\sum_{a \in \delta^+(v) \cap A^t} x_a - \sum_{a \in \delta^-(v) \cap A^t} x_a = 0, \quad t \in T, v \in V \setminus \{\sigma, \tau\}, \tag{3}$$

$$\sum_{a \in C} x_a \leq 1, \quad C \in \mathcal{C}, \tag{4}$$

$$x_a \in \{0,1\}, \quad a \in A. \tag{5}$$

Objective function (1) maximizes the sum of the profits of the arcs associated with each path in the solution. Constraints (2)-(3) impose to have at most one path from source σ to sink τ in G (i.e., a feasible timetable) for each train $t \in T$. Finally, the exponentially many clique constraints (4) avoid the selection of incompatible arcs, thus enforcing the track capacity restrictions.

The resulting Lagrangian problem then calls for a set of maximum Lagrangian profit paths for the trains, taking into account the original profit (i.e., the ideal profit decreased according to the shift and/or stretch that occurs) and the penalties for the relaxed constraints. If the maximum Lagrangian profit of a path for a train is nonpositive, then the train is not scheduled in the solution.

It is worth mentioning that an alternative equivalent formulation is used in [7] and in the subsequent [4, 5], as well as in our implementation, where node (instead of arc) variables are introduced to express the track capacity constraints (4) to be relaxed in a Lagrangian way. For the purposes of the current presentation, we will stick to the formulation with (4) since it is much more convenient to illustrate, and everything adapts in an obvious way to the model actually used.

4 Our robust TTP

A main drawback of the nominal TTP illustrated in the previous section is that it does not take into account, in the planning phase, possible delays that can occur at an operational level. These delays can strongly affect the quality of the solution or even make it infeasible. For example, it may become necessary to change (or even cancel) the scheduled timetable for some trains, thus deteriorating the quality of service to the customers. On the other hand, all of the approaches to robustness listed in Section 2 tend to share the two main drawbacks mentioned in the introduction, namely their application generally requires significant changes in the existing software for the nominal TTP as well as much higher computing times.

We now describe how to modify model (1)-(5) so as to get robust solutions that take the possible presence of delays into account.

As already mentioned, delay propagation is typically reduced by means of buffer times, i.e., of idle times inserted in the timetables. In our formulation we consider fixed travel times, hence buffer times correspond to station stops lasting longer than required: "short" delays can hopefully be absorbed by the buffer time, thus retaining the feasibility of the schedule. On the other hand, allowing buffer times that can absorb even "very long" (less likely to occur) delays would be a too conservative choice and may produce inefficient solutions that are not acceptable in practice.

The aim of the robust problem is therefore a bi-objective problem: maximize the profit of the scheduled trains as in the nominal problem (*efficiency*), but also maximize buffer times (*robustness*).

Our approach borrows from Light Robustness [10] the idea of estimating the robustness of a solution through the slack variables associated with its constraints. In Light Robustness, "soft" constraints are imposed in order to have at least a certain given additional time distance (w.r.t. the minimal one) between any two events that occur at a station (i.e., two consecutive departures of trains from a station). In order to guarantee feasibility, slack variables are introduced for the new constraints and minimized in the objective function. Our approach works in a similar way, however the additional desired time distances are not fixed, but determined on the fly by the approach itself. In addition, contrarily to Light Robustness, we do not impose an explicit constraint on the maximum efficiency loss.

To be more specific, in our TTP model the need of inserting buffer times corresponds to favoring "short" stretches. So, we attach an artificial prize (parametric weight) to the station arcs representing a stopping time lasting longer than the minimum stopping time, but only within a fixed threshold (set to 15 minutes in our tests). In other words, the artificial prizes will tend to increase the stopping time in each station up to 15 minutes, while longer stops will not receive any additional prize because their utility is questionable.

Our robust version of ILP model (1)-(5) includes constraints (2)-(5) but uses the new objective function:

$$\max \sum_{t \in T} \sum_{a \in A^t} p_a x_a + F_k \sum_{t \in T} \sum_{a \in A^t} b_a^t x_a \tag{6}$$

where F_k is a dynamically-updated weighting factor (to be defined later), and $b_a^t \geq 0$ is a parameter (also to be defined later) giving an additional profit depending on the amount of buffer time associated with arc a for train t. The new objective function is of multi-objective type, that takes the *efficiency* of a solution (i.e., its nominal cost given by the first sum) and its estimated robustness (second sum) into account.

5 Lagrangian robustness

In this section we describe the heuristic algorithm that we use to solve the robust model of the previous section. The algorithm follows closely the one of [7]. In fact, as anticipated in the introduction, our aim is to design just a minor modification of an existing Lagrangian heuristic that leads to a sound framework to produce alternative solutions taking robustness into account.

We consider the Lagrangian relaxation of constraints (4) for the model defined by the new objective function (6) subject to constraints (2)-(5), and solve it within a simple subgradient optimization framework to determine near-optimal Lagrangian multipliers. The Lagrangian objective function reads:

$$\max \sum_{t \in T} \sum_{a \in A^t} (p_a + F_k b_a^t) x_a + \sum_{C \in \mathcal{C}} \lambda_C (1 - \sum_{a \in C} x_a) = \sum_{t \in T} \sum_{a \in A^t} \overline{p}_a^t x_a + \sum_{C \in \mathcal{C}} \lambda_C \tag{7}$$

where factor F_k now depends on the subgradient iteration counter k, while for $t \in T$ and $a \in A^t$ we have $\overline{p}_a^t := p_a + F_k b_a^t - \sum_{C \in \mathcal{C}_a} \lambda_C$, where $\mathcal{C}_a \subseteq \mathcal{C}$ denotes the subfamily of cliques containing arc a. The Lagrangian relaxation thus calls for a maximum profit path in G for each train. Constraints (4) are handled via a dynamic constraint generation scheme that identifies, at each iteration of the subgradient optimization procedure, some of the constraints that are violated by the current

Lagrangian solution and adds them to a "pool" containing the constraints that were violated "recently". At each iteration of the subgradient procedure, we also apply a constructive heuristic algorithm based on the Lagrangian profits, that now includes the parametric prize b_a^t given to station arcs. The resulting feasible solution is characterized by an efficiency value and a total buffer time, the latter figure being a rough estimate of its level of robustness. In addition, we implemented a simple local search procedure, which tries to refine every heuristic solution we find: in particular, it tries to reschedule any train that was subject to shift and/or stretch, while keeping the others fixed, and while taking into account the goals of efficiency and robustness.

At this point we have our Lagrangian heuristic, plus a set of "control parameters" b_a^t (and F_k) to enforce robustness. Based on the work in [12], who studied the optimal distribution of buffers on a single corridor, we apply the following formula for the definition of b_a^t:

$$b_a^t := \min\{q_a^t, M\} \cdot (1 - e^{-\lambda p})(len(t) - p) \tag{8}$$

Here, for a station arc a, q_a^t is the number of minutes of buffer when used by train t; e.g., if a connects an arrival node at time 8:00am to a departure node at time 8:15am and the minimum waiting time for train t at that station is 2 minutes, the buffer time q_a^t is 13 minutes. Moreover, for a segment arc a, $q_a^t := 0$, recalling that travel times between consecutive stations cannot be increased with respect to the ideal timetable. Finally, M and λ are parameters ($M = 15$ and $\lambda = 3$ in our tests), $len(t)$ is the number of stations visited by train t, and p is the position of arc a for train t (i.e., $p = 1$ if arc a arises at the very beginning of train t, and $p = len(t)$ if it arises at the very end). In this way, a different weight is given to the buffer times, based on the position of the stopping arc along the path. Indeed, it turns out that it is not worthwhile to give a large weight in the very beginning of the path of the train (the buffer times may be unused because the probability to face a delay in the early sections is low), nor in the very end of the path of the train (it may be too late for the buffer times to be useful). As to F_k, we start with $F_0 := F_{startEff}$ ($= 0.1$), then increase it to $F_{startRob}$ ($= 0.5$) after $K_{iterEff}$ ($= 900$) subgradient iterations, and then increase it by F_{step} ($= 0.5$) every $K_{iterRob}$ ($= 100$) subgradient iterations, until $F_k > F_{end}$ ($= 2.0$). In this way, we concentrate on efficiency in the first iterations and give a larger weight to robustness later on.

Not surprisingly, solutions with large total buffer time are not necessarily more robust than those with shorter buffers, because robustness also depends on where the buffer times are actually placed. In order to evaluate the robustness of a given solution in a more precise way, an external validation tool such as the one developed by [11] can be used. Given a TTP solution (the "published timetable"), this tool considers different random perturbation scenarios (the perturbation affecting, e.g., the duration of same train legs). For each scenario, the tool updates the published timetable to make it feasible under the occurred perturbation, and computes the total delay incurred with respect to the published one. The tool assumes that all the trains in the solution have to be scheduled, and that all train precedences are fixed (i.e., major changes of the train circulation are not taken into account). The average total delay across all scenarios is finally computed, which is a reliable measure of robustness—the lower the average total delay, the more robust the timetable. For a detailed description of how the perturbation scenarios are generated, we refer the reader to Sections 4.2 and 7 of [11].

Unfortunately, the validation tool requires considerable computing time, so it cannot be used "on the fly" to evaluate the robustness of the heuristic solutions computed at each subgradient step. We therefore determined a simple "rule of thumb" to select the best solutions in terms of efficiency and robustness among the several ones generated during subgradient optimization. On one hand,

we want to take the efficiency of the solution into account, so as to avoid a large worsening of the nominal objective. Thus, we consider different efficiency thresholds with respect to the best nominal value (99%, 95% and 90% as in [11]). On the other hand, we want to select solutions that have a good quality in terms of robustness. Thus, among the solutions that respect the efficiency threshold, we determine which are the "most robust" ones, based on the simple formula below. We observe that a solution is robust if, for each pair of trains, there is "enough time" between the departure/arrival time instants at each station that they both visit (so that delay propagation is less likely to occur). This is formalized in the objective function used in [11], which is defined as follows:

$$\sum_{(i,j)\in E} \max\{0, w_{ij}(\Delta - (\theta_j - \theta_i - m_{ij}))\}. \tag{9}$$

Here, E is the set of all the pairs of events (departures or arrivals) occurring at the same station, θ_i and θ_j are the corresponding time instants in chronological order, m_{ij} is the minimum time distance between the two events (due to the track capacity constraints), and Δ is the number of additional minutes that we require between any two events. Moreover, $w_{ij} := w_i + w_j$ is the weight of the pair of events, where $w_i := (1 - e^{-\lambda p})(len(t) - p)$, already used in (8), t being the train associated with event i and p the position of event i for t (and the same for w_j). In other words, for each station, we consider all the pairs of departure/arrival events and the additional time distance $(\theta_j - \theta_i - m_{ij})$ that is left between them with respect to m_{ij}: if this distance is less than Δ $(= 3.5)$ minutes, then we weigh the difference in the total sum according to the position of the events in the timetables of the corresponding trains. The robustness of a solution can then be evaluated by considering this sum (the smaller the better), which we call the *robustness sum*.

In [11], objective function (9) can easily be made linear in the θ_i variables used, associated with events, whereas in our case it would be quadratic in the x_a variables. Recalling that the main purpose here is to consider a simple adaptation of the nominal method in [7], in the optimization we stuck to the rougher robustness estimation provided by (6), linear in the x_a variables, and used (9) only in order to evaluate a posteriori the robustness of each solution found. In fact, we also tried simple linearizations of (9) in our optimization process, which requires the determination of an optimal path for each train both to solve the Lagrangian relaxation for the current multipliers and to compute the heuristic solutions. Given that the results were not as good as those using (6), we decided to stick to the latter.

As already mentioned, and following [11], to select the solutions to validate we adopt the following rule: we consider different efficiency thresholds with respect to the best nominal solution value (99%, 95% and 90%), and for each of them we validate only the solution whose efficiency is above the threshold and whose robustness sum is smallest. In case no solution found is above the threshold, we validate the one with the highest efficiency. In the selection process we consider both the solutions obtained before and after the simple local search procedure. Of course, many other solutions with different values of efficiency are obtained by our method and not validated: by providing these on output anyhow, a user could adopt many other criteria to choose the most desirable solution (among a large set of solutions), depending on the specific requirements.

Algorithm 1 illustrates the general structure of our robust Lagrangian approach (*RobuLag*).

Algorithm 1: Robust Lagrangian approach (`RobuLag`)—the basic scheme

$\mathcal{C} := \emptyset$ (set of active clique constraints (4));
$F := F_{start}$ (current robustness factor F_k in objective function (7));
$\mathcal{S} := \emptyset$ (set of candidate TTP feasible solutions);
$i := 0$ (sub-iteration counter);
while *($F \leq F_{end}$)* **do**
    solve relaxed model (6),(2)-(5) (i.e., determine the maximum Lagrangian profit path in G for each train) and obtain Lagrangian solution r;
    compute a heuristic solution s based on Lagrangian profits (i.e., determine the maximum Lagrangian profit path in G for each train, satisfying all constraints (4));
    add s to $\mathcal{S}$, compute its robustness sum by formula (9), and store its value;
    refine solution s by the local search procedure and obtain s';
    add s' to $\mathcal{S}$, compute its robustness sum by formula (9), and store its value;
    find and add to $\mathcal{C}$ some clique constraints violated by r (setting to zero their Lagrangian multiplier λ_C);
    update Lagrangian multipliers λ_C, $C \in \mathcal{C}$ (subgradient step);
    set $i := i + 1$;
    if *($i > K_{iterEff}$)* **then** $F := F_{startRob}$, $i := 0$;
    if *($i > K_{iterRob}$)* **then** $F := F + F_{step}$, $i := 0$;
end
foreach *efficiency threshold value* $\theta \in \{99\%, 95\%, 90\%\}$ *w.r.t the best nominal one* **do**
    select a solution $s \in \mathcal{S}$ with efficiency above θ and minimum robustness sum;
    if *no such solution exists* **then** select a solution $s \in \mathcal{S}$ with highest efficiency;
    call validation tool on s and return s
end

6 Computational results

In this section we present computational experiments on real-world instances provided by the Italian Railways. Our Lagrangian heuristic algorithm was coded in C and developed in the Microsoft Visual Studio 6.0 environment. All tests were performed on a PC with a Pentium IV 3.2 GHz processor with 2GB memory. We focused our attention on the instances listed in Table 1, all associated with main corridors where several trains transit. All CPU times reported in the tables are expressed in seconds. In each table we report the efficiency of the solution (*Effic.*) and the corresponding Average Total Delay (*ATD*), as computed by the long-run validation. In particular, for our method we report the values for the three robust solutions associated with the three efficiency thresholds. Note that the reported solutions with smallest robustness sum associated with distinct thresholds may coincide, and in fact this happens in a few cases. In addition we report the total computing time.

A comparison of our method with the method for the nominal TTP from which it was derived (see [7]) is shown in Table 2. As expected, the efficiency that we get for the robust solutions is lower than the nominal one, but the gain in robustness is quite significant (especially when we consider solutions with lower efficiency) and confirms effectiveness of the proposed approach. The computing times are acceptable since the problem has to be solved in a planning phase.

Instance	Corridor	#trains	#stations
MdMI1	Modane-Milan	100	54
MdMI2	Modane-Milan	200	54
MdMI3	Modane-Milan	300	54
MdMI4	Modane-Milan	400	54
ChMI	Chiasso-Milan	194	16
ChRo	Chiasso-Rome	41	102

Table 1: Testbed instances from the Italian Railways.

	Nominal Lagrangian Heuristic		Robust Lagrangian Heuristic	
Instance	Effic.	ATD	Effic.	ATD
MdMI1	9,316	17,027	9,240	14,657
MdMI1			8,935	12,902
MdMI1			8,621	12,743
	total CPU time = 566		total CPU time = 2,299	
MdMI2	18,542	37,365	18,357	35,094
MdMI2			17,668	26,889
MdMI2			16,723	24,419
	total CPU time = 1,830		total CPU time = 5,136	
MdMI3	24,638	45,145	24,412	44,738
MdMI3			23,410	37,733
MdMI3			22,181	32,094
	total CPU time = 3,479		total CPU time = 9,365	
MdMI4	27,259	53,059	26,989	49,798
MdMI4			25,958	44,432
MdMI4			24,540	38,095
	total CPU time = 5,227		total CPU time = 12,150	
ChMI	20,816	3,068	20,618	2,906
ChMI			20,252	2,739
ChMI			20,252	2,739
	total CPU time = 519		total CPU time = 1,471	
ChRo	5,567	39,181	5,515	35,241
ChRo			5,418	33,658
ChRo			5,418	33,658
	total CPU time = 462		total CPU time = 1,753	

Table 2: Efficiency value and total average delay for the Lagrangian nominal and robust solutions, respectively.

In Table 3 we present a comparison between our approach and the method proposed in [11], that we call FSZ in the sequel. As already mentioned in the introduction, our purpose here is not to show that our approach outperforms all the methods for robust TTP available in the literature, but rather to show that, although it is a straightforward simple-minded modification of an existing approach for the nominal case, it yields comparable (in fact, better) results with respect to a

state-of-the-art method such as FSZ (whose code was accessible to us).

FSZ works as follows. One starts by receiving on input the best nominal solution obtained by the Lagrangian heuristic of [7], and, in a first phase, tries to improve it (without caring about robustness) with a time limit of 2 hours by applying a commercial MIP solver (ILOG Cplex 11.0) to the aperiodic counterpart of the PESP model. Then, in a second phase, the best nominal solution found is provided as a first incumbent to the MIP solver, and one solves a Light Robustness model, in which the new objective function is the minimization of (9), with the explicit constraint that the efficiency be at least equal, respectively, to the three thresholds that we use in our selection, leading to three distinct FSZ solutions. The time limit for this second phase is again 2 hours. Note that, with respect to the input nominal solution, event precedences are left unfixed but train cancellations are not allowed in both phases.

With respect to FSZ, working in two separate phases, our Lagrangian approach works in an integrated way: it finds a feasible schedule for the trains by optimizing efficiency and robustness at the same time, being also allowed to cancel trains during all the process. For this comparison, we considered as best nominal efficiency value the one found by FSZ at the end of the first phase, and this is the reason why the results for our method are different from those in Table 2. According to Table 3, our Lagrangian heuristic turns out to be effective in producing robust solutions of good quality, that often dominate the FSZ ones (dominated solutions are marked by $^{\mathrm{d}}$). In particular, out of 18 solutions (6 instances times 3 efficiency thresholds), 13 are dominated by our method, sometimes by a large amount. In the remaining 5 cases, for 2 solutions the ATD values obtained by our method and by FSZ are very similar but the efficiency values that we find are higher. In addition, the computing time of our method is generally much shorter.

In Table 4 we also present a comparison of the two methods when imposing a shorter time limit of 20 minutes, so as to emphasize that our approach can be very attractive when it is necessary to have a robust solution quickly—or when attacking larger instances. In our approach, we limited to 100 (rather than 900) the number $K_{iterEff}$ of subgradient iterations to find a high-efficiency robust solution. Again, in most cases (13 out of 18) our robust solutions strictly dominate the FSZ ones, while the opposite arises only once.

In order to show how the two methods scale when the number of trains increases, we also tested them on the instances obtained by combining two or more instances of the line Modane-Milan. In particular, we considered instances having from 500 up to 1000 trains. In Table 5, the results obtained by the proposed method are compared with those obtained by applying FSZ [11]. In our case, we halved the values of $K_{iterEff}$ (= 450 here) and $K_{iterRob}$ (= 50 here) as the subgradient iterations are quite time consuming for these sizes. As it can be seen, in 14 out of 18 cases the solutions of FSZ are dominated. In addition, the computing time of the our method is notably shorter.

7 Conclusions and future research

In this paper we have shown how to modify an existing Lagrangian heuristic so as to produce robust solutions. To this end, two simple features need to be implemented: (a) introduce artificial parameters intended to "control" the solution robustness; and (b) during Lagrangian optimization, dynamically update the weight of the control parameters so as to produce subproblems where robustness becomes more and more important. In this way, during the process one can easily collect a set of, roughly speaking, "Pareto optimal" heuristic solutions that have a different tradeoff

Instance	Robust Lagrangian Heuristic Effic.	ATD	FSZ [11] Effic.	ATD
MdMI1	9,240	14,657	9,209[d]	16,683[d]
MdMI1	8,935	12,902	8,837[d]	14,070[d]
MdMI1	8,621	12,743	8,372	12,675
	total CPU time = 2,299		total CPU time = 29,366	
MdMI2	18,465	37,202	18,437	36,376
MdMI2	17,737	30,538	17,692[d]	32,355[d]
MdMI2	16,818	23,319	16,761[d]	29,716[d]
	total CPU time = 5,136		total CPU time = 30,630	
MdMI3	23,410	37,733	23,313[d]	45,465[d]
MdMI3	22,493	33,485	22,371[d]	40,433[d]
MdMI3	21,398	29,630	21,193[d]	37,673[d]
	total CPU time = 9,365		total CPU time = 32,279	
MdMI4	27,230	51,691	27,170[d]	52,202[d]
MdMI4	26,081	44,573	26,072[d]	47,527[d]
MdMI4	24,708	37,287	24,700[d]	44,258[d]
	total CPU time = 12,150		total CPU time = 34,027	
ChMI	20,252	2,739	20,041[d]	3,328[d]
ChMI	20,252	2,739	19,231	2,675
ChMI	20,252	2,739	18,219	2,703
	total CPU time = 1,471		total CPU time = 29,319	
ChRo	5,515	35,241	5,512[d]	37,332[d]
ChRo	5,418	33,658	5,289[d]	35,081[d]
ChRo	5,418	33,658	5,011	31,849
	total CPU time = 1,753		total CPU time = 29,262	

Table 3: Comparison on efficiency value and average total delay between our Lagrangian heuristic and the method in [11].

between robustness and efficiency, and leave the final user the choice of the ones to analyze in more details—e.g., through a time-consuming external validation tool.

Our approach has been applied to the well known (aperiodic) Train Timetabling Problem (TTP) on a corridor, and has been computationally analyzed on real-world test cases from the Italian Railways, with a comparison with the method recently proposed by [11]. It turned out that, in spite of its simplicity, our approach is very competitive and often obtains robust solutions of good quality in very short computing time.

Future research can be devoted to validate our approach to robustness on other classes of problems. In particular, the application of our method for the case in which the paths to be followed by trains are not fixed in advance—in a general railway network—could be investigated, with the aim of understanding whether the availability of detours with not too long travel times can lead to more robust solutions.

Still in the railways domain, another direction of research can be to address robustness for rolling stock optimization or optimization of train paths within large railway stations.

Finally, an interesting research topic is the design of fast heuristics for multi-objective optimiza-

Instance	Robust Lagrangian Heuristic Effic.	ATD	FSZ [11] Effic.	ATD
MdMI1	9,249	14,724	9,229[d]	16,652[d]
MdMI1	8,991	13,748	8,856[d]	14,088[d]
MdMI1	8,672	12,955	8,390	12,621
MdMI2	18,307[d]	38,017[d]	18,412	36,652
MdMI2	17,674	31,120	17,668[d]	32,544[d]
MdMI2	16,833	25,857	16,738[d]	29,776[d]
MdMI3	20,425	26,154	16,712[d]	44,223[d]
MdMI3	20,425	26,154	16,037[d]	40,745[d]
MdMI3	20,425	26,154	15,193[d]	40,403[d]
MdMI4	21,552	29,063	20,530[d]	53,917[d]
MdMI4	21,552	29,063	19,700[d]	51,021[d]
MdMI4	21,552	29,063	18,663[d]	48,508[d]
ChMI	20,080	2,826	20,011[d]	3,263[d]
ChMI	20,080	2,826	19,190	2,729
ChMI	20,080	2,826	18,192	2,656
ChRo	5,488	35,265	5,442[d]	51,788[d]
ChRo	5,398	33,333	5,222[d]	42,319[d]
ChRo	5,398	33,333	4,947[d]	46,857[d]

Table 4: Comparison on efficiency and average total delay between our Lagrangian heuristic and the method in [11], with shorter time limit (1,200 CPU seconds).

tion, to be obtained as simple modifications of given Lagrangian heuristics for the single-objective version of the problem—very much in the spirit of the approach proposed in the present paper.

Acknowledgements

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). We are very grateful to Domenico Salvagnin and Arrigo Zanette for having provided us with the codes implementing the method and the validation tool in [11], and with plenty of suggestions and remarks on their use.

References

[1] R.K. Ahuja, R. Möhring, C. Zaroliagis, Eds., Robust and Online Large-Scale Optimization, *Lecture Notes in Computer Science 5868*, Springer-Verlag, Berlin Heidelberg, (2009).

[2] D. Bertsimas and M. Sim, "Robust discrete optimization and network flows", *Mathematical Programming*, 98, 49-71 (2003).

[3] D. Bertsimas and M. Sim, "The price of robustness", *Operations Research*, 52, 35-53 (2004).

[4] V. Cacchiani, A. Caprara and P. Toth, "A Column Generation Approach to Train Timetabling on a Corridor", *4OR* 6 (2), 125–142 (2008).

Instance (#trains)	Robust Lagrangian Heuristic Effic.	ATD	FSZ [11] Effic.	ATD
MdMI (500)	29,570	61,025	29,911	63,184
MdMI (500)	28,793	56,816	28,702[d]	58,756[d]
MdMI (500)	27,239	47,950	27,192[d]	55,247[d]
	total CPU time = 6,612		total CPU time = 34,892	
MdMI (600)	32,476	70,241	32,776	75,690
MdMI (600)	31,494	60,659	31,452[d]	69,825[d]
MdMI (600)	29,836	51,478	29,796[d]	65,813[d]
	total CPU time = 7,398		total CPU time = 36,383	
MdMI (700)	32,392	65,756	33,570	73,186
MdMI (700)	32,308	63,148	32,214[d]	66,726[d]
MdMI (700)	30,836	57,467	30,519[d]	63,076[d]
	total CPU time = 8,090		total CPU time = 36,677	
MdMI (800)	32,610	67,533	33,675	75,052
MdMI (800)	32,610	67,533	32,315[d]	68,945[d]
MdMI (800)	30,887	58,184	30,614[d]	64,916[d]
	total CPU time = 8,486		total CPU time = 37,634	
MdMI (900)	31,870	64,617	33,144	69,508
MdMI (900)	31,870	64,617	31,805[d]	64,721[d]
MdMI (900)	30,356	58,113	30,131[d]	59,471[d]
	total CPU time = 8,968		total CPU time = 38,440	
MdMI (1000)	30,874	62,154	30,854[d]	67,606[d]
MdMI (1000)	29,630	56,307	29,607[d]	63,409[d]
MdMI (1000)	28,655	54,703	28,049[d]	59,900[d]
	total CPU time = 9,319		total CPU time = 39,300	

Table 5: Comparison on efficiency value and average total delay between our Lagrangian heuristic and the method in [11] for larger instances.

[5] V. Cacchiani, A. Caprara and P. Toth, "Scheduling extra freight trains on railway networks", *Transportation Research Part B*, 44B, Issue 2, 215–231, (2010).

[6] V. Cacchiani and P. Toth, "Nominal and Robust Train Timetabling Problems", *Technical Report OR-10-5*, University of Bologna, (2010).

[7] A. Caprara, M. Fischetti and P. Toth, "Modeling and Solving the Train Timetabling Problem", *Operations Research* 50, 851–861 (2002).

[8] A. Caprara, L. Kroon, M. Monaci, M. Peeters and P. Toth, "Passenger Railway Optimization", in C. Barnhart, G. Laporte (eds.), *Transportation*, Handbooks in Operations Research and Management Science 12, Elsevier, Amsterdam, 129–187 (2007).

[9] G. D'Angelo, G. Di Stefano and A. Navarra, "Recoverable-robust timetables for trains on single line corridors", in *Proceedings of 3rd International Seminar on Railway Operations Modelling and Analysis (RailZurich2009)* (2009).

[10] M. Fischetti and M. Monaci, "Light Robustness", in R.K. Ahuja, R. Möhring, C. Zaroliagis, Eds., Robust and Online Large-Scale Optimization, *Lecture Notes in Computer Science 5868*, Springer-Verlag, Berlin Heidelberg, 61–84 (2009).

[11] M. Fischetti, D. Salvagnin and A. Zanette, "Fast Approaches to Improve the Robustness of a Railway Timetable", *Transportation Science* 43, 321–335 (2009).

[12] L. Kroon, G. Maròti, M. R. Helmrich, M. Vromans and R. Dekker, "Stochastic improvement of cyclic railway timetables", in Algorithmoc Methods for Railway Optimization, *Transportation Research Part B*, 42 (6), 553–570 (2008).

[13] C. Liebchen, M. Lübbecke, R. Möhring and S. Stiller, "The concept of recoverable robustness, linear programming recovery, and railway applications", in R.K. Ahuja, R. Möhring, C. Zaroliagis, Eds., Robust and Online Large-Scale Optimization, *Lecture Notes in Computer Science 5868*, Springer-Verlag, Berlin Heidelberg, 1–27, (2009).

[14] C. Liebchen, M. Schachtebeck, A. Schöbel, S. Stiller and A. Prigge, "Computing delay resistant railway timetables", *Computers and Operations Research*, 37, n. 5, 857–868, (2010).

[15] P. Serafini and W. Ukovich, "A Mathematical Model for Periodic Event Scheduling Problems", *SIAM Journal on Discrete Mathematics* 2 (1989), 550–581.

[16] F. Barber, L. Ingolotti, A. Lova, P. Tormos, and M.A. Salido, "Meta-Heuristic and Constraint-based Approaches for Single-Line Railway Timetabling", in R.K. Ahuja, R. Möhring, C. Zaroliagis, Eds., Robust and Online Large-Scale Optimization, *Lecture Notes in Computer Science 5868*, Springer-Verlag, Berlin Heidelberg, 145–181, (2009).

[17] M. P. Tormos, A. L. Lova, L. P. Ingolotti and F. Barber, "A Genetic Approach to Robust Train Timetabling", Technical Report ARRIVAL-TR-0173, ARRIVAL Project (2008).