UNIVERSITÀ DEGLI STUDI DI PADOVA

Sede Amministrativa: Università degli Studi di Padova
Dipartimento di Matematica Pura e Applicata

Scuola di Dottorato di Ricerca in Scienze Matematiche

Indirizzo di Matematica Computazionale

XXI Ciclo

Three Topics in Mixed Integer Programming

Direttore della Scuola: Ch.mo Prof. Paolo Dai Pra

Supervisore: Ch.mo Prof. Matteo Fischetti

Dottorando: Arrigo Zanette

Padova, 3 Novembre 2008

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA PURA E APPLICATA

Three Topics in Mixed Integer Programming

Ph.D. THESIS

Author: Arrigo Zanette
Coordinator: Ch.mo Prof. Paolo Dai Pra
Supervisor: Ch.mo Prof. Matteo Fishetti

2008–2009

Scuola di Dottorato di Ricerca in Matematica Computazionale – XXI CICLO

Abstract

In the present thesis we describe our contributions on three topics in Mixed Integer Programming (MIP). In chapter entitled “Lexicography and degeneracy: Can a pure cutting plane algorithm work?”, we discuss an implementation of the lexicographic version of Gomory’s fractional cutting plane method for Integer Linear Programming (ILP) problems and of two heuristics mimicking the latter. In computational testing on a battery of MIPLIB problems we compare the performance of these variants with that of the standard Gomory algorithm, both in the single-cut and in the multi-cut (rounds of cuts) version, and show that they provide a radical improvement over the standard procedure. We also offer an explanation for this surprising phenomenon. In chapter entitled “Minimal Infeasible Subsystems and Benders cuts”, taking inspiration from general cutting plane methods for MIP, we propose alternative selection criteria for Benders cuts, and analyze them computationally. Our approach is based on the correspondence between minimal infeasible subsystems of an infeasible Linear Program, and the vertices of the so-called alternative polyhedron. The choice of the "most effective" violated Benders cut then corresponds to the selection of a suitable vertex of the alternative polyhedron, hence a clever choice of the dual objective function is crucial–whereas the textbook Benders approach uses a completely random selection policy, at least when the so-called feasibility cuts are generated. Computational results on a testbed of MIPLIB instances are presented. We show that the proposed methods allow for a speedup of 1 to 2 orders of magnitude with respect to a textbook implementation. In chapter entitled “Fast Approaches to Improve the Robustness of a Railway Timetable”, we address the problem of finding a robust train timetable. The Train Timetabling Problem (TTP) consists in finding a train schedule on a railway network that satisfies some operational constraints and maximizes some profit function which counts for the efficiency of the infrastructure usage. In practical cases, however, the maximization of the objective function is not enough and one calls for a robust solution that is capable of absorbing as much as possible delays/disturbances on the network. We propose and analyze computationally four different methods to improve the robustness of a given TTP solution. The approaches combine Linear Programming (LP) and ad-hoc Stochastic Programming/Robust Optimization techniques. The computational results on real-world instances show 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.

Sommario

Nel presente lavoro di tesi descriviamo i nostri contributi su tre argomenti di Mixed Integer Programming (MIP). Nel capitolo intitolato "Lexicography and degeneracy: Can a pure cutting plane algorithm work?" discutiamo una implementazione della versione lessicografica del metodo dei piani di taglio di Gomory per problemi di Integer Linear Programming (ILP) e due euristiche. Nei test computazionali su una batteria di istanze della libreria MIPLIB confrontiamo la performance dei metodi implementati col l'algoritmo standard di Gomory, sia nella versione a singolo taglio che nella versione multi taglio (round di tagli), e mostriamo che le nostre implementazioni producono un miglioramento radicale sulla procedura standard. Inoltre forniamo un'interpretazione di questo sorprendente fenomeno. Nel capitolo intitolato "Minimal Infeasible Subsystems and Benders cuts", prendendo ispirazione dai metodi cutting plane generalmente usati in MIP, proponiamo nuovi criteri di selezione per i tagli di Benders, e li analizziamo computazionalmente. Il nostro approccio si basa sulla corrispondenza tra sistemi minimamente infeasible di un problema lineare infeasible, e i vertici del così detto poliedro delle alternative. La scelta del taglio di Benders violato più efficace corrisponde quindi alla selezione di un vertice opportuno nel poliedro delle alternative, da cui segue che è cruciale una scelta intelligente della funzione obiettivo duale–mentre l'approccio di Benders textbook usa una politica di scelta completamente casuale. Nei test computazionali mostriamo che il metodo proposto consente uno speedup da 1 a 2 ordini di grandezza rispetto all'implementazione textbook. Nel capitolo intitolato "Fast Approaches to Improve the Robustness of a Railway Timetable", ci occupiamo del problema di trovare una tabella oraria robusta in ambito ferroviario. Il Train Timetabling Problem (TTP) consiste nel trovare uno schedule dei treni di una rete ferroviaria che soddisfi dei vincoli operativi e massimizzi una funzione di profitto che stimi l'efficienza dell'uso dell'infrastruttura. Tuttavia la massimizzazione della funzione obiettivo può non essere sufficiente e si può voler trovare una soluzione robusta, ovvero capace di assorbire il più possibile i ritardi/disturbi sulla rete. A tal fine, proponiamo e analizziamo computazionalmente quattro diversi metodi per migliorare la robustezza di una data soluzione di TTP. Gli approcci combinano Programmazione Lineare e tecniche ad-hoc di Stochastic Programming/Robust Optimization. I risultati computazionali su istanze reali mostrano che due delle tecniche proposte sono molto veloci e forniscono soluzioni robuste di qualità comparabile ad un approccio di Stochastic Programming standard, ma computazionalmente molto più oneroso.

Acknowledgments

During the last three years many people have encouraged me and contributed to this thesis: I want to thank all of them without whom I would not have reached this goal.

Firstly, I have to thank my advisor, Prof. Matteo Fischetti, who supported me with his deep scientific understanding and enthusiasm for research, and made this experience really enjoyable by creating an uncommonly friendly environment.

Many thanks also to the guys of the OR group in Padova and Bologna: Domenico Salvagnin, who shared with me this adventure as a sincere friend and co-author of two chapters of this thesis, and Michele Monaci and Valentina Cacchiani, for their friendship and the help provided with the railway data. A special thought goes to Lorenzo Brunetta, whose helpfulness and kindness we miss so much.

I am deeply indebted to Prof. Egon Balas, co-author of the second chapter, for having given me the opportunity of visiting him at Carnegie Mellon and for having shared his outstanding personality with me during many inspiring conversations.

I cannot forget to thank the technical staff at the department of Mathematics and at DEI for their kind and prompt assistance.

Finally, I am sincerely grateful to whom have always unconditioned supported my choices: my family and Valentina.

Contents

Chapter 1

Mixed Integer Programming

Since the last half of the past century, Operations Research has played a fundamental role in a broad range of military and civilian applications, including logistics, management, transportation, finance, biology, and physics.

Among Operations Research techniques, Linear Programming (LP) is perhaps the most known and the basic brick for many other approaches. It has been introduced during the Second World War, to model planning (or programming, in the military jargon) problems, using linear constraints. This kind of modelling turned out to be efficiently solvable in practice by using the Simplex Method introduced by Dantzig in 1947, and also in theory, since it has been proved by Khachiyan [42] using the Ellipsoid Method, and later by Karmarkar [41] using the so-called Barrier Method, that LP problems can be solved in polynomial time.

A generalization of LP is Mixed Integer Programming (MIP). MIP can be considered the swiss army knife in Operations Research, due to its modelling expressiveness derived by mixing (hence the name) in the same mathematical model variables defined in the reals, as for LP, and in the integers. The need of integer variables to model planning decisions arose quite soon after the introduction of the simplex method. E.g., one can easily model the decision making process by introducing binary variables, that is variables that take value 1, if a decision as to be taken, 0 otherwise.

However, unlike LP, no polynomial time algorithm is known for solving MIP problems in general; indeed it has been proved that MIP problems belong to the class of $\mathcal{NP}$-hard problems (see Garey and Johnson [29], for example).

The vast practical impact and the intrinsic difficulty, make MIP problems one of the most intriguing topics in applied mathematics. Since the introduction of MIP modelling, big steps have been done in dealing with its solution: from small problems of dozens of binaries variables and constraints, we are currently able to solve instances of thousands of variables and constraints. The reason of the success is twofold: first, the increased computing power availability, that makes manageable problems of bigger and bigger size, and opens up the way to new approaches, once unpractical. The second, and probably the most important reason, is the advance in algorithms, exact and heuristic, used to solve MIP problems.

In the sequel, we will briefly introduce the main algorithmic ideas developed for attacking MIP problems.

1.1 The mathematical model

A *Mixed Integer linear Program* (MIP) is defined as follows:

$$\begin{array}{rrl} z_{MIP} = & \min & c^T x \\ & \text{subject to} & Ax \leq b \\ & & l \leq x \leq u \\ & & x \in \mathbb{R}^n \\ & & x_j \in \mathbb{Z} \quad \forall j \in I \end{array} \tag{1.1}$$

where $c \in \mathbb{Q}^n$ is the cost vector, $A \in \mathbb{Q}^{m \times n}$ is the constraints matrix, $b \in \mathbb{Q}^m$ the righ-hand side, $l \in (\mathbb{Q} \cup \{-\infty\})^m$ and $u \in (\mathbb{Q} \cup \{\infty\})^m$ the bounds on variables, typically treated apart from the rest of constraints. For simplicity sake, in the sequel we will consider $l = 0$ and and the upper bound $x \leq u$ included in the constraint matrix. Variables constrained to integrality are indexed in $I \subseteq N = \{1, \ldots, n\}$. Related to the matrix notation in which it is written, constraints are also called *rows* and variables *columns*. Integer variables x_j with bounds $0 \leq x_j \leq 1$ are called *binary* variables and typically have a different semantic than other integer variables, called *general integer*. Indeed, binary variables are typically used to model the common situation in which yes-or-no decisions arise. Moreover, from a theoretical viewpoint, any integer variable can be expressed only in terms of binary variables through the binary expansion of interval $[l, u]$ (e.g., a variable x_g ranging in the interval $[0, 7]$ can be substituted, without loss of generality, by using 3 different binary variables representing its binary expansion: $x_g = x_{b0} + 2x_{b1} + 4x_{b2}$). Any vector x satisfying constraints in (1.1) is called *feasible* solution. We will denote by z^*_{MIP} the optimal integer solution of the MIP problem.

Geometrically speaking, constraints (1.1) define the set X obtained intersecting the polyhedron $P_{LP} = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ and $\mathbb{Z}^{|I|}$. The *integer hull* of P_{LP} is the convex hull of points in X, $\text{conv}(X)$. A fundamental result of Meyer [58] is that the integer hull of the rational polyhedron P_{LP} is again a rational polyhedron, P_{MIP}. Namely, $\text{conv}(X)$ is given by the intersection of a finite number of inequalities defining P_{MIP}. So, provided that the polyhedral description of $\text{conv}(X)$ is known, integrality constraints become redundant and the problem simple [67]. However, the polyhedral description is hard to find in general. This fact supports the theoretical importance of cutting plane algorithms (see Section 1.3), in which $\text{conv}(X)$ is iteratively constructed by adding valid inequalities for $\text{conv}(X)$ to P_{LP}.

1.2 Linear programming

Discarding the integrality constraints in (1.1), the MIP model becomes a Linear Programming (LP) model.

The solutions of LP is a relaxation of MIP. That is, the optimal value of LP, z^*_{LP} is a lower bound of the optimal value of MIP, say z^*_{MIP}. The percentage difference $(z^*_{MIP} - z^*_{LP})/z^*_{LP}$ is called *integrality gap*. If this gap is 0, then the solution of the MIP model resorts to the solution of its LP relaxation. There exist entire classes of constraint matrices A that guarantee this desirable property [67].

LP can be solved efficiently, for example, by using the Simplex Method invented by Dantzig in 1947.

1.2.1 The Simplex Method

The simplex method requires to put the LP model in *standard form*. This means that inequality constraints are to be transformed into equality constraints by introducing so-called *slack* variables (e.g., for a "$\leq$" constraint a positive slack variable is added to the left-hand-side of the constraint). Moreover, applying a linear transformation on variables, lower bounds are set to zeros; on the other hand, upper bounds are treated implicitly by complementing the respective variables as needed, during the execution of the algorithm. All data is arranged for the computation in a matrix, called *tableau*. The first row of the tableau, row 0, stores the coefficients of the objective function. Other rows store the constraints matrix A. The first column, column 0, is reserved for the right-hand-side values.

A *basis* B is a square invertible matrix obtained by selecting a subset of m columns (i.e., *basic* variables). The unique solution of the linear system obtained by setting to zero all non-basic variables is called *basic solution*.

The simplex algorithm iterates through basic solutions, by exchanging one column in the basis (*leaving variable*) with another outside the basis (*entering variable*): this operation is called *pivot*.

The choice of entering and leaving variables can be done by following different measures of improvement. Indeed, the algorithm comes into two main variants. On one hand, the *primal* simplex method visits only feasible solutions by iteratively improving the objective function value, until the optimum is eventually reached. On the other hand, the *dual* simplex method visits only more-than-optimal solutions until it eventually finds a feasible solution.

The twofold nature of the algorithms relies on the LP *strong duality*, that is, there exists a dual version of the original (primal) problem, defined in a dual space, which will attain the same optimal value as the primal problem—provided such optimum exists and is finite (refer to [5] for a treatment of linear programming duality). Actually, the dual simplex corresponds to the primal simplex applied to the dual of the given problem.

When the measure of improvement fails the algorithm can cycle. This is due to either primal or dual degeneracy. In case of primal degeneracy, there are many different bases corresponding to the same basic solution; in case of dual degeneracy, there are different basic solutions with the same cost. In both the situations, some countermeasure is required to avoid cycling. The first proposed solution was to use a lexicographic simplex algorithm [33]. Heuristic alternatives were then developed to address the computational

burden of the lexicographic simplex. The reader is referred to [71] for more details on these techniques, and on the practical implementation of the simplex method as well.

The simplex method performs very well in practice, even if it has an exponential worst case running time, and its effectiveness together with improved implementations through years, are among the reasons of success in the solution of MIP problems of increasing size.

In addition to the simplex method, other algorithms have been proposed to solve LPs. In [42], Khachiyan first constructively proved the polynomial complexity of LP. The ellipsoid algorithm proposed by Khachiyan has the remarkable property of being independent on the number of constraints in input. The only requirement is that one should be able to provide efficiently, if any, a constraint violated by a given point x^*. The problem of finding a constraint violated by a given point x^* is called *separation problem*. In a sense, we can think of the ellipsoid method as a cutting plane algorithm, like the ones we will describe in the next section. Moreover, this peculiar feature has a profound theoretical consequence, when going back to MIP problems. Geometrically speaking, a MIP is a polyhedron with exponentially many defining constraints. This is a problem if one writes down the overall model; but this is not an issue anymore by using the ellipsoid method. Now the point is that if we were able to answer a separation query in polynomial time, we would also be able to solve (1.1) in polynomial time by using the ability of the ellipsoid method of dealing implicitly with constraints. Of course this is very unlikely to be the case: indeed, it has been proved [67] that the separation problem is polynomially solvable if and only if the related optimization problem is polynomially solvable as well.

In [41], Karmarkar proposed another polynomial time algorithm for solving LPs. Unlike the ellipsoid method, this algorithm has emerged in the recent years as a valid alternative to the Simplex Method, at least for huge models. E.g., the application studied in Chapter 4, we found it much faster than the simplex method for solving the very-large-scale LP models involved.

1.3 Cutting planes

If the LP relaxation does not suffice to solve the MIP problem, one may try to strengthen the relaxation by adding new inequalities and solving the LP relaxation again. As mentioned in Section 1.1, theory [58] suggests that iterating this process one could eventually find the MIP optimum, since MIP is a polyhedron. This technique is called *cutting planes method* and a pseudocode of the general scheme is given in Figure 1.1.

A *cutting plane* is an inequality valid for P_{MIP} but violated by at least one point of P_{LP}. A way of deriving a valid inequality for P_{MIP} is by aggregating rows in P_{LP} with some non-negative multipliers $u \in \mathbb{R}^n_+$:

$$u^T Ax \leq u^T b$$

Since x is non-negative the inequality obtained by rounding down the coefficients of the

input : MIP problem $\mathcal{P}$
output: optimal solution x^*_{MIP} of $\mathcal{P}$
1 **repeat**
2 Solve the current LP relaxation of $\mathcal{P}$
3 **if** *the solution* x^*_{LP} *is integer* **then**
4 **return** x^*_{LP}
5 **else**
6 Solve the separation problem: find (π, π_0) valid for X violated by x^*_{LP}
7 Add the cut $\pi^T x \leq \pi_0$ to the current LP relaxation
8 **end**
9 **until** stopping criterion reached

Figure 1.1: General cutting planes algorithm.

left-hand-side

$$\lfloor u^T A \rfloor x \leq u^T b \tag{1.2}$$

is again a valid cut for P_{LP}. Now, since $x \in X$ is integer the strengthened cut:

$$\lfloor u^T A \rfloor x \leq \lfloor u^T b \rfloor \tag{1.3}$$

is a valid inequality for X. Cuts derived in this way are called Chvátal-Gomory cuts. Chvátal proved in [19] that taking successive *closures* of this simple kind of cuts, i.e., the (undominated) cuts derived from all the possible multipliers u in the separation phase (6) of Algorithm 1.1, suffices to ensure convergence. However, it is not clear how to derive a violated cut, given a point x^*.

Gomory addressed that issue many years before, in his seminal work [35], showing that violated Chvátal-Gomory cuts can be read directly from the fractional rows of the current LP optimal tableau. In such a case, u appearing in (1.2) corresponds to the entries of a row of the optimal basis inverse. The Gomory procedure is the following. Starting from the Tableau row i

$$x_i + \sum_{j \in N \setminus B} a_{ij} x_j = a_{i0} \tag{1.4}$$

we can relax it to inequality $x_i + \sum_{j \in N \setminus B} a_{ij} x_j \leq a_{i0}$ Applying the Chvátal-Gomory rounding procedure we get the *Integer Gomory* cut:

$$x_i + \sum_{j \in N \setminus B} \lfloor a_{ij} \rfloor x_j \leq \lfloor a_{i0} \rfloor \tag{1.5}$$

Subtracting (1.5) from (1.4), and denoting by f_{ij} the fractional part of a_{ij} yields

$$\sum_{j \in N \setminus B} f_{ij} x \geq f_{i0} \tag{1.6}$$

(1.6) is called *Fractional Gomory* cut (FGC).

Provided that the MIP model has an integer objective function, that is the vector c^T has only integer coefficient, Gomory gave two alternative proofs of convergence. They both rely on the lexicographic simplex method for getting rid of degeneracy in the LP relaxations. The lexicographic simplex objective is to minimize (or maximize) the solution vector formed by the objective function z, interpreted as the most important lexicographic variable, followed by the other variables. This is equivalent to solve the original LP amended with the following objective function:

$$\min M^n x_0 + M^{n-1} x_1 \ldots + M x_{n-1} + x_n \tag{1.7}$$

where M is a sufficiently large positive integer, and $x_0 = z$. Now let us suppose x_j is the first fractional variable, according to the lexicographic order. The key point of the first proof is that a Gomory cut derived from the row of the current lexicographically optimal tableau where x_j is basic, will increment the value of the objective function (1.7) by, at least, $M^{n-j}(\lceil x_j \rceil - x_j)$. Thus, assuming without loss of generality that (1.7) is bounded, the number of iterations of Algorithm 1.1 is finite. The second proof relaxes the requirements of algorithm 1.1, allowing cuts to be derived after every lex-simplex pivot. The risk, with respect to the first proof, is that we start separating shallow cuts that do not ensure a significant improvement of (1.7). Shallow cuts, as we will see in Chapter 2, are intimately connected to the determinant of the basis. Intuitively, shallow cuts require small fractions to justify their shallowness, and small fractions require big determinant to be represented. But shallow cuts can be avoided ensuring the determinant D of the current basis is upper bounded by some integer $\bar{D}$. Indeed, it is always possible to limit the value of D, e.g., by introducing a finite number of valid inequalities (Gomory cuts) with fractional coefficients on which pivoting for decreasing D magnitude. So the algorithm will terminate in a finite number of steps. An application of this latter proof are all-integer cutting planes methods [30], where $\bar{D} = 1$.

1.3.1 Geometry of Chvátal-Gomory cuts

Let first introduce some definitions. A *supporting hyperplane* of a polyhedron $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ is an hyperplane $c^T x = c_0$ such that $c_0 = \max\{c^T x : x \in P\}$. The intersection $F = P \cap \{x : c^T x = c_0\}$ is a polyhedron by itself and is called *face* of P. The dimension of a polyhedron P is the dimension of the smallest affine subspace containing F. A polyhedron P is *full-dimensional* if its dimension is n or, equivalently, the system of inequalities $\{Ax \leq b\}$ does not contain any implicit equality. A *facet* is a proper face of maximum dimension, e.g., if the polyhedron is full-dimensional a facet is a face of dimension $n - 1$.

A cut $(\pi^1)^T x \leq \pi^1_0$ for $P = \{x : Ax \leq b\}$ *dominates* another cut $(\pi^2)^T x \leq \pi^1_0$ if $\pi^1 \geq \pi^2$ and $\pi^1_0 \leq \pi^2_0$. It is easy to prove by using duality (see [62], for example) that all the undominated Chvátal-Gomory cuts are obtainable by using dual coefficients $c_B B^{-1}$ for a certain basis B of P_{LP}. Indeed, starting from any Chvátal-Gomory cut, one can maximize in P_{LP} the left-hand-side of the cut and then apply the rounding procedure

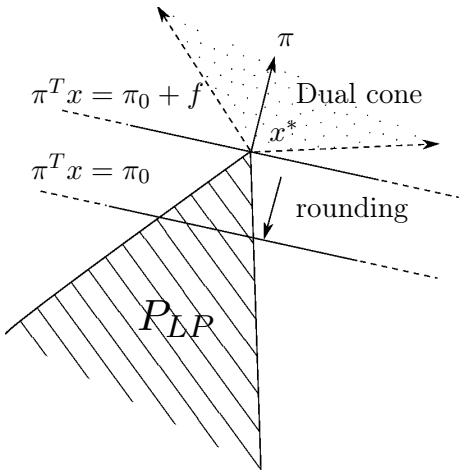

Figure 1.2: Cut geometry.

as in (1.2) in order to strengthen the initial cut and to produce an undominated one. This fact open up an interpretation of Chvátal-Gomory cuts (see Figure 1.2). Namely, any undominated Chvátal-Gomory cut can be derived by taking a vertex x^* of P_{LP} and looking for an integer objective function π^T maximized in x^* (that is, in the cone of dual multipliers u). Applying the rounding procedure in (1.3) yields the cut.

In [16] the above interpretation is applied to derive useful cuts. Indeed, under some conditions it is easy (polynomial) to find an appropriate objective function violated by a fractional basic solution. However, the general solution of the problem, addressed for example in [53], is NP hard.

1.3.2 Stronger cuts

Let us now review the derivation of Chvátal-Gomory cuts in order to show how they can be strengthened. Let our starting polyhedron be the following single inequality

$$ax \leq b,\ x \in \mathbb{Z}_+ \tag{1.8}$$

Then, denoting by f the fractional part $(a - \lfloor a \rfloor)$ of a and by f_0 the fractional part of b, we can rewrite (1.8) as follows:

$$\lfloor a \rfloor x + fx \leq b,\ x \in \mathbb{Z}_+ \tag{1.9}$$

Let $z \in \mathbb{Z}$ be equal to $\lfloor a \rfloor x$ and $y \in \mathbb{R}_+$ be equal to fx. Then we can restate (1.9) in terms of z and y:

$$z + y \leq b,\ z \in \mathbb{Z},\ y \in \mathbb{R}_+ \tag{1.10}$$

It is easy to show that a valid cut for (1.10) is the following:

$$z \leq \lfloor b \rfloor \tag{1.11}$$

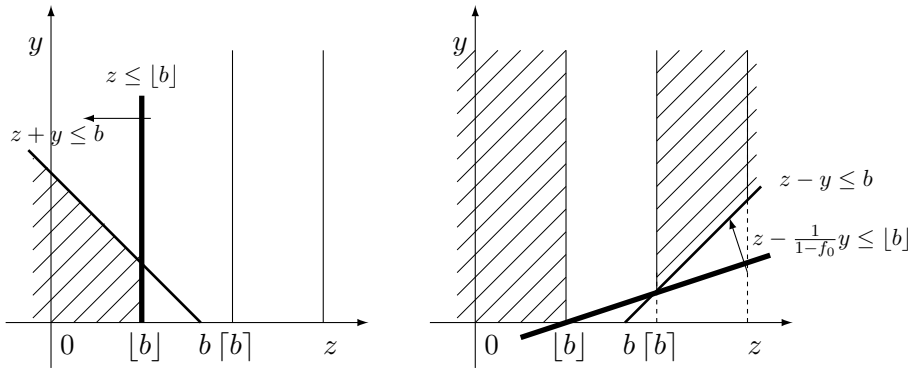

Figure 1.3: Mixed integer rounding.

which is exactly the Chvátal-Gomory cut (1.3), and is depicted in the left chart of Figure 1.3.

However, instead of (1.9) we can equivalently write:

$$\lceil a \rceil x - (1 - f)x \le b,\ x \in \mathbb{Z}_+ \tag{1.12}$$

hence:

$$z - y \le b,\ z \in \mathbb{Z},\ y \in \mathbb{R}_+ \tag{1.13}$$

for which a valid cut is given by:

$$z - \frac{1}{1 - f_0} y \le \lfloor b \rfloor \tag{1.14}$$

or

$$\lfloor a \rfloor x + \frac{f - f_0}{1 - f_0} x \le \lfloor b \rfloor \tag{1.15}$$

as depicted in the right chart of Figure 1.3. If $f - f_0 > 0$ cut (1.15) is actually a strengthening of the Chvátal-Gomory cut (1.11). This idea can be applied to strengthen any Chvátal-Gomory cut. Indeed, given the Chvátal-Gomory cut $\sum_j \lfloor a_j \rfloor x_j \le \lfloor b \rfloor$, first relax the generating row $\sum_j a_j x_j \le b$

$$\sum_{j:\ f_j \le f_0} \lfloor a_j \rfloor x_j + \sum_{j:\ f_j > f_0} a_j x_j \le b \tag{1.16}$$

i.e.,

$$\sum_{j:\ f_j \le f_0} \lfloor a_j \rfloor x_j + \sum_{j:\ f_j > f_0} \lceil a_j \rceil x_j - \sum_{j:\ f_j > f_0} (1 - f_j) x_j \le b \tag{1.17}$$

Applying (1.14) with $z = \sum_{j:\ f_j \le f_0} \lfloor a_j \rfloor x_j + \sum_{j:\ f_j > f_0} \lceil a_j \rceil x_j$ and $y = \sum_{j:\ f_j > f_0} (1 -$

$f_j)x_j$, yields the following cut:

$$\sum_j \left(\lfloor a_j \rfloor + \frac{(f_j - f_0)^+}{1 - f_0} \right) x_j \leq \lfloor b \rfloor \tag{1.18}$$

Cut (1.18) was studied by Marchand et al. in [56] and is called Mixed Integer Rounding (MIR) cut. Starting from a Tableau row, transforming it into an inequality constraint and applying the above procedures yields the Gomory Mixed Integer (GMI) cut, invented by Gomory in [36]. As for (1.5), GMI cuts can be given also in fractional form:

$$\sum_{j \in N \setminus B:\ f_{ij} \leq f_{i0}} f_{ij} x_j + \sum_{j \in N \setminus B:\ f_{ij} > f_{i0}} \frac{f_{i0}(1 - f_{ij})}{1 - f_{i0}} x_j \geq f_{i0} \tag{1.19}$$

Indeed cuts (1.18) and (1.19), despite being a strengthening of (1.3) and (1.6) respectively, can be used to deal with MIP problems involving rational (or continuous) variables.

Note that it is natural to use a *disjunctive* argument to derive and prove (1.14) and (1.11), even if for the latter the argument is less evident. That is, we want to separate a cut valid for the convex hull of the union of the two polyhedra obtained by intersecting the initial polyhedron with the disjunction $x \leq \lfloor b \rfloor \cup x \geq \lfloor b \rfloor + 1$ on the integer variable x. The *disjunctive programming* approach was pioneered in 1974 by Balas [6] and lead to the so-called *split cuts*.

A comprehensive description of MIR cuts, GMI cuts, split cuts and other families of general cutting planes for MIP problems is given in [23]. There, a hierarchy of families of cuts is given: it turns out that the most important families of cuts are, at most, as strong as GMI cuts are.

Recently Letchford et al. [44] reviewed strengthening procedures for Chvátal-Gomory cuts and proposed a new one, that does not involve the introduction of continuous variables, as in the GMI case.

1.3.3 Fortune of Gomory cutting planes

After the first successes (see [11], for example) it was observed quite soon [38] that a pure cutting plane algorithm, as in Figure 1.1, suffers from an early *tailing off* phenomenon: large sequence of cuts are added without any significant improvement towards integrality. Due to this observation, pure cutting plane methods were abandoned in practice in favor of enumerative techniques. In Chapter 2 we will see what are the causes of such an early tailing off, and how to mitigate them.

Nowadays, Gomory cutting planes are commonly used inside MIP solvers to strengthen the problem formulation. Their revival is due to Balas et al. [8], in which they report a breakthrough in the solution of binary MIP problems using rounds of GMI cuts embedded in a B&B code. Since then, GMI cuts have become a common feature in all MIP solvers.

In addition to Gomory cutting planes, other cuts are among the arsenal of state-

of-the-art MIP solvers. These cutting planes do not have a general applicability, but were developed by studying the polyhedral structure of specific problems such as knapsack, fixed-charge flow, vertex packing/covering. Solvers try to automatically identify these common substructures in general instance and to add the appropriate cuts. A comprehensive description can be found in [55].

1.4 Branch and Bound

Branch and bound (B&B) is an efficient way of enumerating all the possible integer solutions of a MIP. The outline of the algorithm is given in Figure 1.4.

The algorithm procedes by partitioning the initial problem $\mathcal{P}$ into subproblems $\mathcal{P}_i$ (*branching* phase). Subproblems $\mathcal{P}_i$ are then solved recursively by applying B&B. Usually branching is implemented by using a search tree data structure (hence the name): $\mathcal{P}$ is called the *root node* and $\mathcal{P}_i$ are child nodes. The current best integer feasible solution found is the *incumbent*. A node, and the corresponding subproblem $\mathcal{P}_i$, is said to be *fathomed* when one of the following conditions holds:

1. $\mathcal{P}_i$ has an integer feasible solution, then we update the incumbent.
2. the constraints defining $\mathcal{P}_i$ make it infeasible
3. a lower bound on the cost of $\mathcal{P}_i$ is greater than the current incumbent cost: this condition is called *bounding*

The last two conditions are typically implemented using a relaxation of problem $\mathcal{P}$, for which it is simple to verify feasibility and to find the optimal value. In fact, if $\mathcal{P}$ is feasible then all its relaxations will be feasible as well, and will have an optimal cost not greater (if minimizing) than $\mathcal{P}$. The most used relaxation is the LP relaxation, obtained by relaxing integrality constraints of (1.1); another practically important relaxation is Lagrangian relaxation [31].

The branching phase is usually carried out by selecting a fractional variable x_j and splitting current node problem Q into the two subproblems $Q_1 = \{Q \cap x_j \leq \lfloor x_j \rfloor\}$ and $Q_2 = \{Q \cap x_j \geq \lceil x_j \rceil\}$. The choice of the variable to branch on turns out to be crucial. The simplest choice strategies simply look at the values attained by variables in the current optimum relaxation, e.g., select the one with a fractional part as close as possible to 0.5. Actually, the best selection strategy would be to branch on the variable that will improve the lower bound most. However, this problem can be as hard as solving the original problem, so one can reason again on the estimate of the bound given by the relaxation, testing what happens to the lower bound after the next branching step (*strong branching*). Recently, another kind of branching has been proposed, based on *general disjunctions*. That is, instead of branching on a single variable disjunction, one may consider a general disjunction $\{\pi^T x \leq \pi_0 - 1 \cup \pi^T x \geq \pi_0\}$, e.g., the one derived from a cut as in [24], or one that tends to minimize the number of required branching nodes, as in [13].

input : MIP problem $\mathcal{P}$
output: optimal solution (x^*_{MIP}, z^*_{MIP}) of $\mathcal{P}$
let $\bar{z} = \infty$
let $L = \{\mathcal{P}\}$
while $L \neq \emptyset$ **do**
    Node Selection: Take $\mathcal{Q}$ out of L : $L = L \setminus \mathcal{Q}$
    Solve relaxation $\underline{\mathcal{Q}}$ of $\mathcal{Q}$
    if $\underline{\mathcal{Q}}$ *is feasible* **then**
        let $\underline{x}$ an optimal solution of $\underline{\mathcal{Q}}$
        let $\underline{z} = c^T \underline{x}$ its cost
        if $\underline{x}$ *is integer and* $\underline{z} < \bar{z}$ **then**
            $\bar{z} = \underline{z}$
            $\bar{x} = \underline{x}$
            Bounding: Remove every $\mathcal{Q}$ from L with bound greater than $\bar{z}$
        Branching: else if $\underline{z} < \bar{z}$ **then**
            Partition $\mathcal{Q}$ into subproblems $\mathcal{P}_1 \dots \mathcal{P}_n$
            Add $\mathcal{P}_1 \dots \mathcal{P}_n$ to L with bound $\bar{z}$
        end
    end
end
return $x^*_{MIP} = \bar{x}$ *and* $z^*_{MIP} = \bar{z}$

Figure 1.4: General Branch and Bound algorithm.

Another important choice is the order in which nodes inserted into list L (Figure 1.4) should be processed. The *best first* strategy is to select the node with the lowest bound, hoping it is closer to the optimum. A possible drawback of this approach is that it can take a long time before the first incumbent is found, since the algorithm will tend to stay close to the root node without going deep into the search tree. On the contrary, the *depth-first* always select the deepest node for processing. This minimizes memory consumption and can produce an incumbent faster. However it may spend a lot of time enumerating solution regions that have few chances of producing good integer solutions. Actual solvers mix these two strategies in order to get the best from both of them.

1.5 Branch and Cut

We have seen how to use cutting planes in a pure cutting plane algorithm. Unfortunately, these kinds of algorithms tend to stall quite early, and/or to run into numerical difficulties. A way of obviating this problem is pairing cutting planes algorithms with enumeration. The obvious way of doing it is by running the cutting plane algorithm to strengthen the formulation until tailing off or some other stopping criterion is met, and then give the new strengthen formulation in input to a B&B. This approach is called cut-and-branch.

In [1] Padberg and Rinaldi introduced the Branch-and-Cut (B&C) algorithm. According to B&C, globally valid cuts (if possible, facet defining) can be derived at every

B&B node. However there is a risk that the number of cuts becomes soon too large to be handled. Thus cuts are typically stored in a separate data structure, called *cut pool*, where they can be efficiently evaluated for violation and then inserted into the model, as needed. Also an efficient purging policy to remove cuts is important to keep the model small and the solution of the relaxation fast.

A drawback of B&C is that the LP relaxation can become harder to solve, due to the increased size of the model and to the possibly denser structure of the cuts with respect to the original system of constraints. On the other hand, often cuts help reducing the number of node quite substantially.

The approach in [1] was not intended to use general cutting planes, such as Chvátal-Gomory cutting planes, because it was not clear how to separate globally valid cuts. Only in [8] it was shown that one can indeed derive globally valid GMI cuts for 0-1 programs, using a straightforward lifting procedure. Basically, the idea is that for 0-1 problems all integer solutions are vertices of the LP relaxation, so for every assignment of variables there exists an LP basis of the original problem that yields that assignment. From that basis we can separate a cut that is valid for the original set of constraints (not including branching fixings), i.e., globally valid.

Chapter 2

Lexicography and degeneracy: Can a pure cutting plane algorithm work?

2.1 Introduction

Modern Branch-and-Cut (B&C) methods for mixed or pure Integer Linear Programs (ILPs) are heavily based on general-purpose cutting planes such as Gomory cuts, that are used to reduce the number of branching nodes needed to reach optimality. On the other hand, pure cutting plane methods based on Gomory cuts alone are typically not used in practice, due to their poor convergence properties.

In a sense, branching can be viewed as just a "symptomatic cure" to the well-known drawbacks of Gomory cuts—saturation, bad numerical behavior, etc. From the cutting plane point of view, however, the cure is even worse than the disease, in that it hides the real source of the troubles. In this respect, it is instructive to observe that a main piece of information about the performance of Gomory cuts (namely, that they perform much better if generated in rounds) was discovered only in 1996 (Balas, Ceria, Cornuéjols, and Natraj [8]), i.e., about 40 years after their introduction [35].

The purpose of our project, whose scope extends well beyond the present chapter, is to try to come up with a viable pure cutting plane method (i.e., one that is not knocked out by numerical difficulties), even if on most problems it will not be competitive with the branch-and-bound based methods.

As a first step, we chose to test our ideas on Gomory's fractional cuts, for two reasons: they are the simplest to generate, and they have the property that when expressed in the structural variables, all their coefficients are integer (which makes it easier to work with them and to assess how nice or weird they are). In particular, we addressed the following questions:

i) Given an ILP, which is the most effective way to generate fractional Gomory cuts from the optimal LP tableaux so as to push the LP bound as close as possible to the optimal integer value?

ii) What is the role of degeneracy in Gomory's method?

iii) How can we try to counteract the numerical instability associated with the iterated use of Gomory cuts?

iv) Is the classical polyhedral paradigm "the stronger the cut, the better" still applicable in the context of Gomory cuts read from the tableau? The question is not at all naive, as one has to take into account the negative effects that a stronger yet denser (or numerically less accurate) cut has on the next tableaux, and hence on the next cuts.

As we were in the process of testing various ways of keeping the basis determinant and/or condition number within reasonable limits, we decided to implement the lexicographic dual simplex algorithm used in one of Gomory's two finite convergence proofs. Gomory himself never advocated the practical use of this method; on the contrary, he stressed that its sole purpose was to simplify one of the two proofs, and that in practice other choice criteria in the pivoting sequence were likely to work better. Actually, we have no information on anybody ever having tried extensively this method in practice.

The lexicographic method has two basic ingredients: (a) the starting tableau is not just optimal, i.e., dual feasible, but lexicographically dual-feasible, and the method of reoptimization after adding a cut is the lexicographic dual simplex method; and (b) at least after every k iterations for some fixed k, the row with the first fractional basic variable is chosen as source row for the next cut.

The implementation of this method produced a huge surprise: the lexicographic method produces a dramatic improvement not only in gap closure (see Figure 2.1), but also in determinant and cut coefficient size.

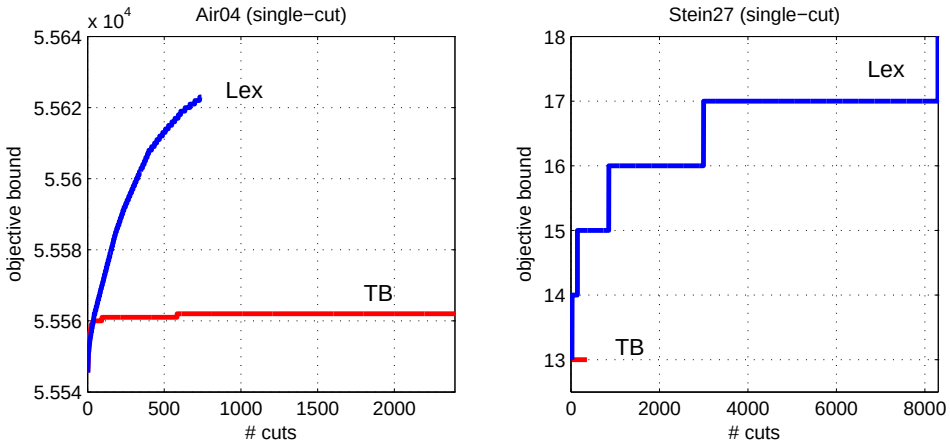

Figure 2.1: Comparison between the textbook and lexicographic implementations of single-cut Gomory's algorithm on `air04` *and* `stein27`.

It is well known that cutting plane methods work better if the cuts are generated in rounds rather than individually (i.e., if cuts from all fractional variables are added before reoptimization, rather than reoptimizing after every cut). Now it seems that if

we are generating rounds of cuts rather than individual cuts, the use of the lexicographic rule would make much less sense, in particular because (b) is automatically satisfied—so the lexicographic rule plays a role only in shaping the pivoting sequence in the reoptimization process. So we did not expect it to make much of a difference. Here came our second great surprize: as illustrated in Figure 2.2, even more strikingly than when using single cuts, comparing the standard and lexicographic methods with rounds of cuts shows a huge difference not only in terms of gap closed (which for the lexicographic version is 100% for more than half the instances in our testbed), but also of determinant size and coefficient size (not shown in the figure).

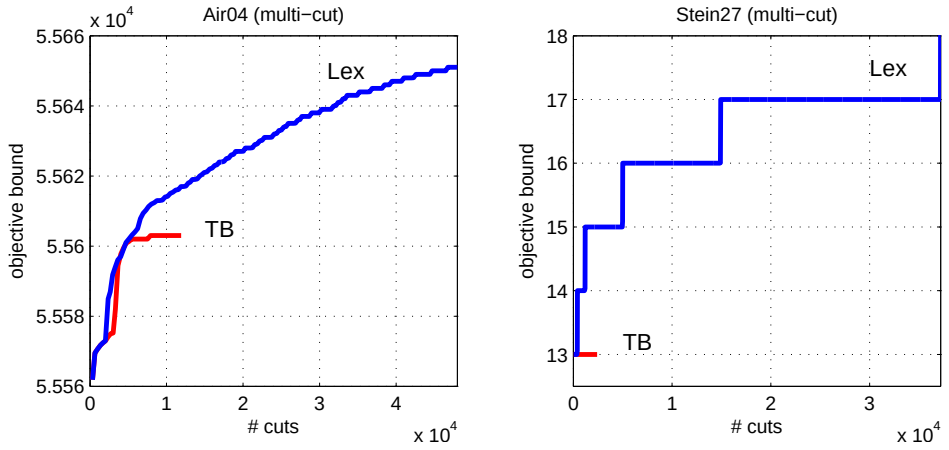

Figure 2.2: Comparison between the textbook and lexicographic implementations of multi-cut Gomory's algorithm on `air04` *and* `stein27`.

In this chapter we discuss ad evaluate computationally an implementation of the lexicographic version of Gomory's fractional cutting plane method and of two heuristics mimicking the latter one, and offer an interpretation of the outcome of our experiments.

We also describe a way to round the tableau coefficients when computing FGC coefficients, which turns out to be very effective (together with the lexicographic simplex) in producing numerically stable cuts. The integration of Gomory Mixed-Integer cuts within a lexicographic cutting plane method for pure integer programs is finally addressed.

2.2 Gomory cuts

In this chapter we focus on pure cutting plane methods applied to solving ILPs of the the form:

$$\begin{aligned} \min\ & c^T x \\ & Ax = b \\ & x \geq 0 \text{ integer} \end{aligned}$$

where $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{Z}^m$ and $c \in \mathbb{Z}^n$. Let $P := \{x \in \Re^n : Ax = b, x \geq 0\}$ denote the LP relaxation polyhedron, that we assume be bounded.

The cut generation is of course a crucial step in any cutting plane method, as one is interested in easily-computable yet effective cuts.

In 1958, Gomory [35] (see also [37]) gave a simple and elegant way to generate violated cuts, showing that x^* can always be separated by means of a cut easily derived from a row of the LP-relaxation optimal tableau. The cut derivation is based on a rounding argument: given any equation $\sum_{j=1}^n \gamma_j x_j = \gamma_0$ valid for P, if x is constrained to be nonnegative and integer then $\sum_{j=1}^n \lfloor\gamma_j\rfloor x_j \leq \lfloor\gamma_0\rfloor$ and $\sum_{j=1}^n \lceil\gamma_j\rceil x_j \geq \lceil\gamma_0\rceil$ are valid cuts. According to Gomory's proposal, the equation is the one associated with a row of the LP optimal tableau whose basic variable is fractional: we will refer to this row as the *cut generating* row, and to the corresponding basic variable as the *cut generating* variable.

The resulting cut, called *Fractional Gomory Cut* (FGC) or Chvátal-Gomory cut, has important theoretical and practical properties. First of all, one can use FGCs read from the LP tableau to derive a finitely-convergent cutting plane method. Secondly, because of the integrality of all the cut coefficients, the associated slack variable can be assumed to be integer, so the addition of FGCs does not introduce continuous variables that could make the rounding argument inapplicable in the next iterations. Moreover, the fact that the cut coefficients are integer ensures a certain "confidence level" about the numerical accuracy of the generated cuts. Indeed, once a cut is generated, small fractionalities in the computed coefficients can be removed safely so as to reduce error propagation, whereas FGCs with significant fractionalities are likely to be invalid (because of numerical issues) and hence can be skipped.

In 1960, Gomory [36] introduced the *Mixed Integer Gomory* (MIG) cuts to deal with the mixed-integer case. In case of pure ILPs, MIG cuts are applicable as well. Actually, MIG cuts turn out to *dominate* FGCs in that each variable x_j receives a coefficient increased by a fractional quantity $\theta_j \in [0, 1)$ with respect to the FGCs (writing the MIG in its $\leq$ form, with the same right-hand-side value as in its FGC counterpart). E.g, a FGC cut of the type $2x_1 - x_2 + 3x_3 \leq 5$ may correspond to the MIG $2.27272727x_1 - x_2 + 3.18181818x_3 \leq 5$. So, from a strictly polyhedral point of view, there is no apparent reason to insist on FGCs when a stronger replacement is readily available at no extra computational effort. However, as shown in the example above, the coefficient integrality of MIG cuts is no longer guaranteed, and the nice numerical properties of FGCs are lost. Even more importantly, as discussed in the sequel, the introduction of "weird fractionalities" in the cut coefficients may have uncontrollable effects on the fractionality of the next LP solution and hence of the associated LP tableau. Finally, MIG cuts introduce continuous slack variables that may receive overweak coefficients in the next iterations, leading to weaker and weaker MIG cuts in the long run. As a result, it is unclear whether FGC or MIG cuts are better suited for a pure cutting plane method for pure integer programs based on tableau cuts—a topic that we are going to investigate in the near future.

It is important to stress that the requirement of reading (essentially for free) the cuts

directly from the optimal LP tableau makes the Gomory method intrinsically different from a method that works solely with the original polyhedron where the cut separation is decoupled from the LP reoptimization, as in the recent work of Fischetti and Lodi [26] on FGCs or Balas and Saxena [25] on MIG (split) cuts. Actually, only the first round of cuts generated by the Gomory method (those read from the very first optimal tableau) work on the original polyhedron, subsequent rounds are generated from a polyhedron truncated by previously generated cuts.

We face here a very fundamental issue in the design of pure cutting plane methods based of (mixed-integer or fractional) Gomory cuts read from the LP optimal tableau. Since we expect to generate a long sequence of cuts that eventually lead to an optimal integer solution, we have to take into account side effects of the cuts that are typically underestimated when just a few cuts are used (within an enumeration scheme) to improve the LP bound. In particular, one should try to maintain a "clean" optimal tableau so as to favor the generation of "clean" cuts in the next iterations. To this end, it is important to avoid as much as possible generating (and hence cutting) LP optimal vertices with a "weird fractionality"—the main source of numerical inaccuracy. This is because the corresponding optimal LP basis necessarily has a large determinant (needed to describe the fractionality), hence the tableau contains weird entries that lead to weaker and weaker Gomory cuts.

In this respect, dual degeneracy (that is notoriously massive in cutting plane methods) can play an important role and actually can *favor* the practical convergence of the method, provided that it is exploited to choose the *cleanest* LP solution (and tableau) among the equivalent optimal ones—the sequence of pivots performed by a generic LP solver during the tableau reoptimization is aimed at restoring primal feasibility as quickly as possible, and leads invariably to an uncontrolled growth of the basis determinant, so the method gets out of control after few iterations.

2.3 Degeneracy and the lexicographic dual simplex

As already mentioned, massive dual degeneracy occurs almost invariably when solving ILPs by means of cutting plane algorithms. Indeed, cutting planes tend to introduce a huge number of cuts that are almost parallel to the objective function, whose main goal is to prove or to disprove the existence of an integer point with a certain value of the objective function.

In one of his two proofs of convergence, Gomory used the lexicographic dual simplex to cope with degeneracy. The lexicographic dual simplex is a generalized version of the simplex algorithm where, instead of considering the minimization of the objective function, viewed without loss of generality as an additional integer variable $x_0 = c^T x$, one is interested in the minimization of the entire solution vector $(x_0, x_1, \ldots, x_n)$, where $(x_0, x_1, \ldots, x_n) <_{LEX} (y_0, y_1, \ldots, y_n)$ means that there exists an index k such that $x_i = y_i$ for all $i = 1, \ldots, k-1$ and $x_k < y_k$. In the lexicographic, as opposed to the usual, dual simplex method the ratio test does not only involve two scalars (reduced cost and pivot candidate) but a column and a scalar. So, its implementation is straightforward,

at least in theory. In practice, however, there are a number of major concerns that limit this approach:

1. the ratio test may be quite time consuming;
2. the ratio test may fail in selecting the right column to preserve lex-optimality, due to round-off errors;
3. the algorithm rigidly prescribes the pivot choice, which excludes the possibility of applying much more effective pivot-selection criteria.

The last point is maybe the most important. As a clever approach should not interfere too much with the black-box LP solver used, one could think of using a perturbed linear objective function $x_0 + \epsilon_1 x_1 + \epsilon_2 x_2 \ldots$, where x_0 is the actual objective and $1 \gg \epsilon_1 \gg \epsilon_2 \gg \ldots$. Though this approach is numerically unacceptable, one can mimic it by using the following method which resembles the iterative procedure used in the construction of the so-called Balinsky–Tucker tableau [9], and is akin to the slack fixing used in sequential solution of preemptive linear goal programming (see [4] and [70]).

Starting from the optimal solution $(x_0^\star, x_1^\star, \ldots, x_n^\star)$, we want to find another basic solution for which $x_0 = x_0^\star$ but $x_1 < x_1^\star$ (if any), by exploiting dual degeneracy. So, we fix the variables that are nonbasic (at their bound) and have a nonzero reduced cost. This fixing implies the fixing of the objective function value to $x_0^\star$, but has a major advantage: since we fix only variables at their bounds, the fixed variables will remain out of the basis in all the subsequent steps. Then we reoptimize the LP by using x_1 as the objective function (to be minimized), fix other nonbasic variables, and repeat. The method then keeps optimizing subsequent variables, in lexicographic order, over smaller and smaller dual-degenerate subspaces, until either no degeneracy remains, or all variables are fixed. At this point we can unfix all the fixed variables and restore the original objective function, the lex-optimal basis being associated with the non-fixed variables.

This approach proved to be quite effective (and stable) in practice: even for large problems, where the classical algorithm is painfully slow or even fails, our alternative method requires short computing time to convert the optimal basis into a lexicographically-minimal one. We have to admit however that our current implementation is not perfect, as it requires deciding whether a reduced cost is zero or not: in some (rare) cases, numerical errors lead to a wrong decision that does not yield a lexicographically dual-feasible final tableau. We are confident however that a tighter integration with the underlying LP solver could solve most of the difficulties in our present implementation.

2.3.1 Lexicographic simplex and Gomory cuts: an entangled pair

FGCs and the dual lexicographic simplex are intimately related one each other, in the sense that FGCs are precisely the kind of cuts that allow for a significant lexicographic improvement at each step. It is therefore not surprising that Gomory's (first) proof of convergence strictly relies on the use of the lexicographic dual simplex [35].

Indeed, the lexicographic dual simplex method starts with a lexicographically optimal tableau, which means that all columns are lexicographically positive or lexicographically negative (depending on whether one minimizes or maximizes, and on the sign rule one follows in representing the columns), and preserves this property throughout the pivoting procedure. To fix our ideas, let's opt for minimization and the sign rule that requires all columns to be lexicographically negative, which means that the first entry is the negative of what usually goes under the name of reduced cost. Thus at any stage of the procedure, the first nonzero entry of each column is negative. It is this sign pattern that guarantees a certain property of the sequence of cuts generated under the lexicographic rule, provided that the "right" rounding operation is used in generating the cuts.

As a matter of fact, the Gomory method using the lexicographic simplex can be proved to be convergent only in case the kind of rounding used is consistent with the lexicographic objective. We next discuss very briefly the FGC properties that lead to a convergent method. Let the ith row of the current tableau be

$$x_h + \sum_{j \in J^-} \overline{a}_{ij} x_j + \sum_{j \in J^+} \overline{a}_{ij} x_j = \overline{a}_{i0}$$

where x_h is the basic variable in row i, J^- is the set of indices of nonbasic variables such that $a_{ij} \leq 0$, and J^+ is the set of indices of nonbasic variables such that $\overline{a}_{ij} > 0$. Moreover, let us suppose h is the first index such that $x_h^* (= \overline{a}_{i0})$ is fractional. To simplify notation, assume $h \neq 0$, i.e., the optimal objective value is not fractional.

A key observation is that, due to the lexicographic sign pattern, for all $i < h$ the first nonzero entry $\overline{a}_{ij}$ in any column $j \in J^+$ is negative.

The Gomory rounding procedure can be used to obtain the following FGC, in integer form:

$$x_h + \sum_{j \in J^-} \lceil \overline{a}_{ij} \rceil x_j + \sum_{j \in J^+} \lceil \overline{a}_{ij} \rceil x_j \geq \lceil \overline{a}_{i0} \rceil \tag{2.1}$$

Note that we round the coefficients of the original row upward. The choice is motivated by the fact that, for a minimization problem, we expect to lexicographically minimize the solution vector of the linear relaxation, hence the cut is intended to contribute in the opposite direction, namely, to increase lexicographically the solution vector.

Clearly, the round-up operation maintains the nonpositiveness of the coefficients in J^- and the positiveness of those in J^+.

In case no x_j with $j \in J^+$ becomes strictly positive after the lexicographic reoptimization, cut (2.1) requires

$$x_h \geq \lceil \overline{a}_{i0} \rceil - \sum_{j \in J^-} \lceil \overline{a}_{ij} \rceil x_j \geq \lceil \overline{a}_{i0} \rceil.$$

Otherwise, due to the particular tableau sign pattern, the increase of some x_j with $j \in J^+$ implies the increase of some higher lex-ranked basic variable x_r (the objective function included) by a positive amount. In both cases, a significant lexicographic step

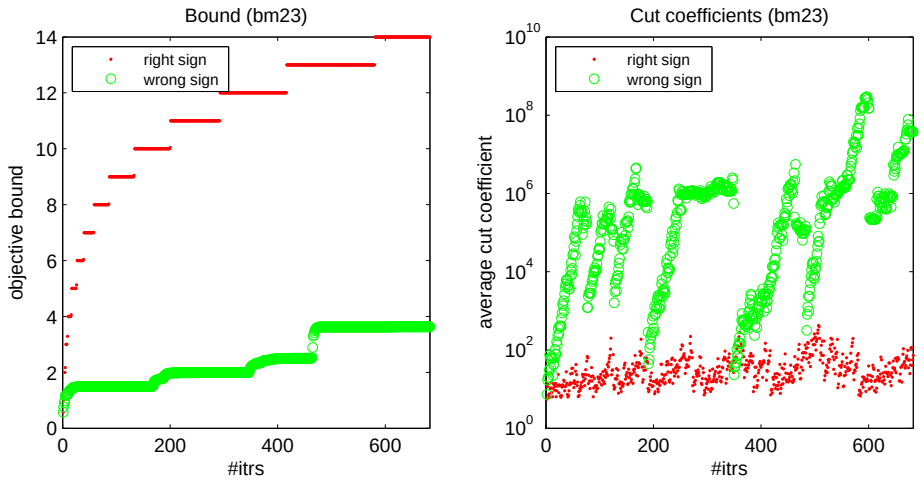

Figure 2.3: Impact of rounding direction on FGCs read from the tableau rows.

is performed: either the cut-generating variable x_h jumps, at least, to its upper integer value $\lceil x_h^* \rceil$, or some higher lex-ranked variable increases by a positive amount. These considerations allow one to conclude that the method converges after a finite number of steps; see [35] for more details.

To show the practical impact of reading the "right" FGC from the tableau rows, in Figure 2.3 we plot the behavior of two variants of the lexicographic method (in its multi-cut version using rounds of cuts) applied to instance `bm23`: one variant exploits the right FGCs, while the other uses their wrong counterpart. The figure shows a huge difference not only in terms of gap closed, but also of numerical stability (coefficient size).

2.4 Heuristics variants

While the lexicographic simplex method gives an exact solution to the problem of degeneracy, simple heuristics can be devised that mimic the behavior of lexicographic dual simplex. The scope of these heuristics is to try to highlight the crucial properties that allow the lexicographic method to produce stable Gomory cuts.

As already mentioned, a lex-optimal solution can in principle be reached by using an appropriate perturbation of the objective function, namely $x_0 + \epsilon_1 x_1 + \ldots + \epsilon_n x_n$ with $1 \gg \epsilon_1 \gg \ldots \gg \epsilon_n$. Although this approach is actually impractical, one can use a 1-level approximation where the perturbation affects a single variable only, say x_h, leading to the new objective function $\min x_0 + \epsilon x_h$. The perturbation term is intended to favor the choice of an equivalent optimal basis closer to the lexicographically optimal one, where the chosen variable x_h is moved towards its lower bound—and hopefully becomes integer.

In our first heuristic, *Heur1*, when the objective function is degenerate we swap our focus to the candidate cut generating variable, i.e., the variable x_i to be perturbed is chosen as the most lex-significant fractional variable. The idea is that each new cut

should guarantee a significant lex-decrease in the solution vector by either moving to a new vertex where the cut generating variables becomes integer, or else some other more lex-significant variables becomes fractional and can be cut.

A second perturbation heuristic, *Heur2*, can be designed along the following lines. Consider the addition of a single FGC and the subsequent tableau reoptimization performed by a standard dual simplex method. After the first pivot operation, the slack variable associated with the new cut goes to zero and leaves the basis, and it is unlikely that it will re-enter it in a subsequent step. This however turns out to be undesirable in the long run, since it increases the chances that the FGC generated in the next iterations will involve the slack variables of the previously-generated FGCs, and hence it favors the generation of cuts of higher rank and the propagation of their undesirable characteristics (density, numerical inaccuracy, etc.). By exploiting dual degeneracy, however, one could try to select an equivalent optimal basis that includes the slack variables of the FGCs. This can be achieved by simply giving a small negative cost to the FGC slack variables.

Both the heuristics above involve the use of a small perturbation in the objective function coefficients, that however can produce numerical troubles that interfere with our study. So we handled perturbation in a way similar to that used in our implementation of the lexicographic dual simplex, that requires the solution of two LPs—one with the standard objective function, and the second with the second-level objective function and all nonbasic variables having nonzero reduced cost fixed at their bound.

2.5 Cut validity

Margot [57] addressed the possibility that cutting planes generators produce invalid cuts due to numerical errors, and performed very interesting experiments on this important issue. The outcome of the experiments is that invalid cuts are separated more frequently than expected, so validity should be a major concern in the development of cut generators.

An important step towards a numerical accurate cut generator has been recently done by Cook et al. [22] for the case of MIG cuts. Their approach is based upon a common feature of the floating point unit of modern computers, namely, the ability to decide if the mantissa of a floating point operation should be rounded downward or upward. Taking advantage of this feature it is possible to derive lower and upper bounds on every floating point computation carried out. E.g., in order to derive a valid tableau-based MIG cut, first we have to start from a "valid" tableau row. A tableau row is obtained by left-multiplying matrix A, assumed to be known without uncertainty, by an approximation u of a row of the inverse of the basis matrix. The result should be an equation, say $u^T A = u^T b$, that however can turn out to be slightly invalid due to limited numerical precision. Hence one can weaken the equation into a certainly valid inequality of a given sense ($\leq$, say) by setting the floating point rounding accordingly. Similarly, by paying attention to the sense of the floating point rounding in the subsequent steps of the MIG separation algorithm, it is possible to come up with a MIG cut guaranteed to

be valid. The approach has however a potential drawback when used in a pure cutting plane context, as it can produce slightly-modified cuts that do not preserve the nice properties of the original "theoretical" cuts. Indeed, in an pure cutting plane framework valid but weaker cuts can lead to small changes in the LP optimal solution. To represent small changes, very precise tableau entries (and hence very large basis determinants) are required, so numerical inaccuracy tends to accumulate and the algorithm might stall.

When FGCs are considered, validity can be certified by just solving an LP, since these cuts are have Chvátal rank 1 with respect to the current formulation. To be more specific, the validity of a FGC of the form $\alpha^T x \leq \alpha_0$ with $(\alpha, \alpha_0) \in \mathbb{Z}^{n+1}$ requires checking whether the maximum value of the LP relaxation with objective function $s := \alpha^T x - \alpha_0$ is strictly less than 1. (FGCs in $\geq$ form can be handled in a similar way.)

In our context, a generic FGC read from the optimal tableau has the form $\sum_j \lfloor \overline{a}_{ij} \rfloor x_j \leq \lfloor \overline{a}_{i0} \rfloor$. It is easy to show that the current basis B maximizes the objective function $s := \sum_j \lfloor \overline{a}_{ij} \rfloor x_j - \lfloor \overline{a}_{i0} \rfloor$ as well. Indeed, by subtracting from this latter equation the i-th tableau row written as $0 = \sum_j \overline{a}_{ij} x_j - \overline{a}_{i0}$ one can easily project out the basic x variables and obtains the reduced-cost equation

$$s = f_{i0} - \sum_j f_{ij} x_j \tag{2.2}$$

where, as customary, $f_{ij} \in [0, 1)$ denotes the fractional part of $\overline{a}_{ij}$, and $f_{ij} = 0$ for basic variables x_j. Being $f_{ij} \geq 0$ for all j, the current basic solution x^* is guaranteed to be optimal for the maximization of s, the corresponding optimal value being $s^* = f_{i0} < 1$, which proves the validity of the FGC cut.

In practice, the solution of the maximization problem above will be carried out by a finite-precision solver whose optimality check depends on a certain threshold ϵ used to verify reduced cost signs. In this perspective, it makes sense to assert the validity of a FGC by using the same optimality threshold as in the LP solver, in the sense that cut invalidity would come into play only when the optimality test itself is invalidated by numerical problems. This observations motivates the following definition.

Definition 1 *Given a polyhedron* $P = \{(x, y) \in \mathbb{R}^{n+m}_+ : \; Ax + y = b\}$ *with* $A \in \mathbb{Z}^{m \times n}$ *and* $b \in \mathbb{Z}^m$*, a basis* B *of* (A, I)*, an integer vector* $(\alpha, \alpha_0) \in \mathbb{Z}^{n+1}$ *and a threshold* $\epsilon > 0$*, we say that cut* $\alpha^T x \leq \alpha_0$ *is a* ϵ-valid (rank 1) cut *if the objective function* $\alpha^T x$ *is* ϵ-maximized *by* B*, according to the classical reduced-cost test* $\alpha^T - \alpha_B^T B^{-1} A \leq \epsilon 1^T$*, and* $\alpha_0 \geq \lfloor \alpha_B^T B^{-1} b + \epsilon \rfloor$*.*

Accordingly, the ϵ-validity of a FGC cut is still guaranteed if one performs integer roundings by using a small positive threshold ϵ, through operator

$$\text{floor}_\epsilon(\overline{a}_{ij}) \quad = \lfloor \overline{a}_{ij} + \epsilon \rfloor \tag{2.3}$$

Indeed, with the above redefinition of the *floor* operator the FGC cut becomes $\sum_j \text{floor}_\epsilon(\overline{a}_{ij}) x_j \leq \text{floor}_\epsilon(\overline{a}_{i0})$ and the reduced costs $f'_{ij} := \overline{a}_{ij} - floor_\epsilon(\overline{a}_{ij}) = f_{ij} -$

$\lfloor f_{ij} + \epsilon \rfloor$ used in (2.2) to certify FGC validity remain greater or equal to $-\epsilon$, hence within the tolerance that certifies optimality.

In our implementation we use the above $floor_\epsilon$ operator, with ϵ set in a conservative way with respect to the optimality tolerance of the LP solver in use.

2.6 Computational results

Our set of pure ILP instances mainly comes from MIPLIB 2003 and MIPLIB 3; see Table 2.1. It is worth noting that, to our knowledge, even very small instances of these libraries (such as `stein15`, `bm23`, etc.) have never been solved by a pure cutting plane method based on FGC or MIG cuts read from the LP tableau.

Problem	Cons	Vars	LP opt	Opt	Source
air04	823	8904	55535.44	56137	MIPLIB 3.0
air05	426	7195	25877.61	26374	MIPLIB 3.0
bm23	20	27	20.57	34	MIPLIB
cap6000	2176	6000	-2451537.33	-2451377	MIPLIB 3.0
hard_ks100	1	100	-227303.66	-226649	Single knapsack
hard_ks9	1	9	-20112.98	-19516	Single knapsack
krob200	200	19900	27347	27768	2 matching
l152lav	97	1989	4656.36	4722	MIPLIB
lin318	318	50403	38963.5	39266	2 matching
lseu	28	89	834.68	1120	MIPLIB
manna81	6480	3321	-13297	-13164	MIPLIB 3.0
mitre	2054	9958	114740.52	115155	MIPLIB 3.0
mzzv11	9499	10240	-22945.24	-21718	MIPLIB 3.0
mzzv42z	10460	11717	-21623	-20540	MIPLIB 3.0
p0033	16	33	2520.57	3089	MIPLIB
p0201	133	201	6875	7615	MIPLIB 3.0
p0548	176	548	315.29	8691	MIPLIB 3.0
p2756	755	2756	2688.75	3124	MIPLIB 3.0
pipex	2	48	773751.06	788263	MIPLIB
protfold	2112	1835	-41.96	-31	MIPLIB 3.0
sentoy	30	60	-7839.28	-7772	MIPLIB
seymour	4944	1372	403.85	423	MIPLIB 3.0
stein15	35	15	5	9	MIPLIB
stein27	118	27	13	18	MIPLIB 3.0
timtab	171	397	28694	764772	MIPLIB 3.0

Table 2.1: Our test bed

Input data is assumed to be integer. All problems are preprocessed by adding an integer variable x_0 that accounts for the original objective function, from which we can derive valid cuts, as Gomory's proof of convergence prescribes. FGC cuts are derived in integer form. Then slack are substituted to bring the cut back to the space of original variables. Since all computations deal with integers, this avoids round-off errors, and the cut are ϵ-valid, according to Definition (1), where $\epsilon = 1e^{-6}$.

We carried out our experiments in a Intel Core 2 Q6600, 2.40GHz, with a time limit of 1 hour of CPU time and a memory limit of 2GB for each instance.

Our first set of experiments addressed the *single-cut* version of Gomory's algorithm. Actually, at each iteration we decided to generate *two* FGCs from the selected cut generating row—one from the tableau row itself, and one from the same row multiplied by -1.

The choice of the cut generation row in case of the lexicographic method is governed

by the rule that prescribes the selection of the least-index variable. As to the other methods under comparison, the cut generation row is chosen with a random policy giving a higher probability of selecting the cut-generating variable from those with fractional part closer to 0.5 (alternative rules produced comparable results).

A very important implementation choice concerns the cut purging criterion. The lexicographic algorithm ensures the lexicographic improvement of the solution vector after each reoptimization, thus allowing one to remove cuts as soon as they become slack at the new optimum. As far as other methods are concerned, however, we can safely remove cuts only when the objective function improves. Indeed, if the objective function remains unchanged a removed cut can be generated again in a subsequent iteration, and the entire algorithm can loop—a situation that we actually encountered during our experiments. We therefore decided to remove the slack cuts only when it is mathematically correct, i.e. after a nonzero change in the objective function value, though this policy can lead to an out-of-memory status after a long stalling phase.

Table 2.2 compares results on the textbook implementation of Gomory's algorithm (TB) and the lexicographic one (Lex). Besides the percentage of closed gap *(ClGap)*, we report 2 tightly correlated parameters to better measure the performance of each method. The first parameter is the cut coefficients size *(Coeff.)*: large coefficients, besides increasing the likelihood of numerical errors, can be a symptom of cut ineffectiveness since they are required to represent very small angles in the space of structural variables. The second parameter is the condition number κ of the optimal basis, which gives a measure of the inaccuracy of the finite-precision representation of a solution x to the linear system $Bx = b$ (the smaller the more accurate the representation). In the table, only the maximum value of the two indicators above during the run is reported. The first column reports one of the following exit-status codes: `(O)` integer optimum, `(T)` time limit, `(M)` out of memory, `(N)` no suitable cuts found as all available cuts where discarded because of their large ($> 10^{10}$) coefficients.

Problem	Textbook							Lex						
		Itrs	Cuts	Time	ClGap	Coeff.	κ		Itrs	Cuts	Time	ClGap	Coeff.	κ
air04	N	267	515	44.95	11.73	8.4e+07	4.6e+12	T	5446	10886	3601.15	**36.83**	1.8e+06	2e+14
air05	N	1237	2391	902.94	2.70	1.5e+08	9.2e+18	T	15859	31712	3601.13	**44.60**	3.2e+05	3.1e+12
bm23	N	313	618	0.23	18.09	1.4e+09	9.2e+18	O	713	1424	0.97	**100.00**	2.4e+02	2.7e+06
cap6000	N	575	1117	55.15	5.36	4.6e+07	2.7e+13	N	12606	25130	2112.15	**26.53**	4.3e+07	8.3e+18
hard_ks100	N	1485	2892	149.03	100.00	2e+08	9.3e+15	O	217	431	0.36	**100.00**	4.5e+05	1.3e+12
hard_ks9	N	164	289	0.18	98.83	2.2e+09	1.2e+13	O	889	1776	0.53	**100.00**	4.8e+04	5.3e+09
krob200	O	41	49	1.40	**100.00**	2.5	1.3e+07	O	1168	2199	256.95	100.00	8.7e+02	4.4e+08
l152lav	N	1099	2160	77.89	37.54	4.2e+08	9.2e+18	O	1122	2219	30.21	**100.00**	1.6e+04	2.1e+09
lin318	O	97	134	8.53	**100.00**	20	5.4e+08	T	2028	3702	3604.78	63.64	2.4e+04	9.7e+10
lseu	N	391	761	0.49	60.75	8.8e+08	9.2e+18	O	15120	30217	33.37	**100.00**	1.9e+04	7.3e+08
manna81	O	271	270	93.41	**100.00**	1	9.2e+18	O	879	878	742.08	100.00	1	9.7e+05
mitre	T	11722	23373	3600.96	**22.76**	1.1e+09	9.2e+18	T	5026	10047	3601.78	14.11	1.3e+06	9.8e+14
mzzv11	T	509	978	3624.01	6.17	6.2e+07	9.2e+18	T	437	870	3604.00	**24.80**	1.2e+02	2.3e+11
mzzv42z	T	1229	2413	3601.25	13.29	1.1e+08	9.2e+18	T	819	1629	3603.23	**43.67**	6.3e+02	5.5e+11
p0033	N	219	434	0.24	10.28	5.1e+08	9.2e+18	O	1496	2957	1.69	**100.00**	1.8e+03	7.6e+06
p0201	N	95	186	0.28	8.11	5.1e+08	2.2e+14	N	188144	373279	2165.83	**85.27**	5.1e+08	1.5e+21
p0548z	T	279514	535298	3601.16	**2.18**	6.5e+08	9.2e+18	T	333934	667812	3601.01	0.03	6.4e+05	1.7e+14
p2756	T	5384	10572	3601.03	0.29	2.1e+09	9.2e+18	T	56113	112222	3601.00	**0.52**	1.2e+04	6.8e+12
pipex	N	3888	7538	1.75	17.01	9.7e+08	2e+14	O	767681	1514866	1402.65	**100.00**	2.7e+06	7.3e+12
protfold	T	112	216	3866.35	5.68	2.5e+08	9.2e+18	T	505	1002	4460.44	**54.38**	1.6e+08	5.6e+14
sentoy	N	21775	36984	9.53	10.82	4.4e+08	9.2e+18	O	5729	11456	14.11	**100.00**	6.5e+04	1.3e+08
seymour	N	210	378	430.49	**21.67**	6.5e+07	9.2e+18	T	779	1538	3605.73	11.23	1.5e+03	2.2e+09
stein15	N	74	123	0.06	25.00	1.9e+09	2.7e+15	O	66	121	0.10	**100.00**	6	2.4e+03
stein27	N	46	87	0.06	0.00	1.1e+09	5.6e+17	O	4283	8254	9.26	**100.00**	76	6.5e+04
timtab1-int	T	99530	172894	3594.30	**22.18**	1.8e+09	9.2e+18	T	785007	1569491	3601.00	4.00	4.9e+08	3.4e+15

Table 2.2: Comparison between textbook and lexicographic implementation of Gomory's algorithm (single-cut version)

Table 2.2 shows clearly that in most cases the TB version has huge coefficient sizes and condition numbers, while in Lex all these values remain relatively small along the entire run. Moreover, Lex could solve to proven optimality 9 of the 25 instances of our testbed—some of these instances being notoriously hard for pure cutting plane methods.

For illustration purposes, Figure 2.4 gives a representation of the trajectory of the LP optimal vertices to be cut (along with a plot of the basis determinant) when the textbook and the lexicographic methods are used for instance `stein15`. In Figures 2.4(a) and (b), the vertical axis represents the objective function value. As to the XY space, it is a projection of the original 15-dimensional variable space. The projection is obtained by using a standard procedure available e.g. in *MATLAB* (namely, multidimensional scaling [15]) with the aim of preserving the metric of the original 15-dimensional space as much as possible. In particular, the original Euclidean distances tend to be preserved, so points that look close one to each other in the figure are likely to be also close in the original space.

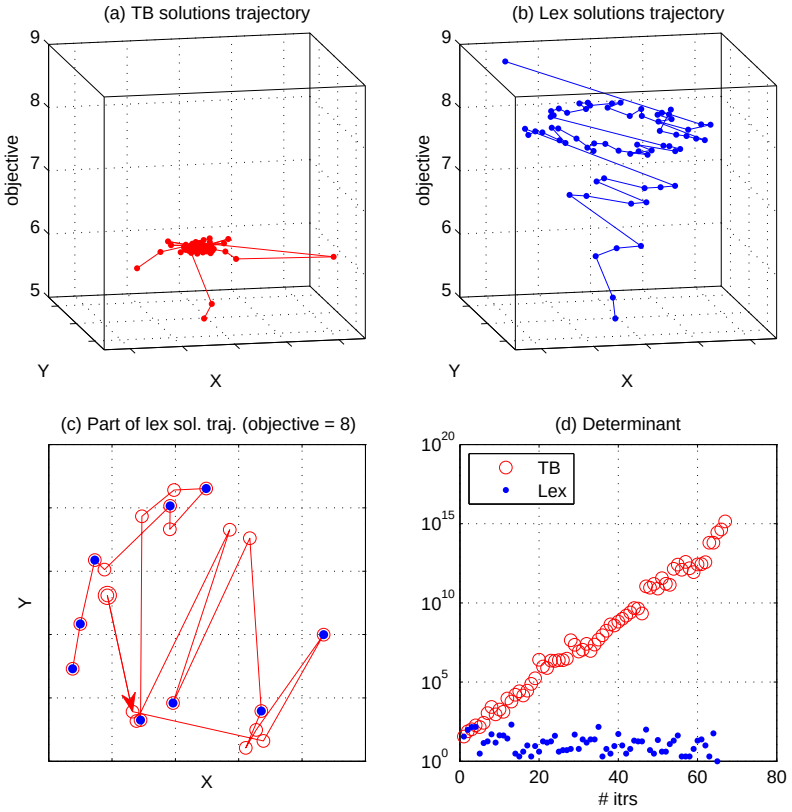

Figure 2.4: Problem stein15 (single cut). (a)-(b) Solution trajectories for TB and Lex, resp.; (c) Lower dimensional representation of the the Lex solution trajectory; the filled circles are lexicographic optima used for cut separation; their immediate next circles are optima given by the black-box dual-simplex solver, whereas the other points correspond to the equivalent solutions visited during lexicographic reoptimization; the double circle highlights the trajectory starting point. (d) Growth of determinants in TB and Lex (logarithmic scale).

According to Figure 2.4(a), the textbook method concentrates on cutting points

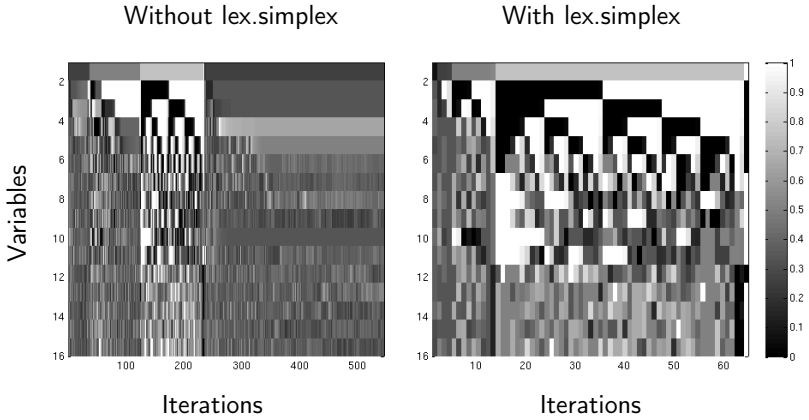

Figure 2.5: Fractional spectrography *of the sequence of solutions provided by the Gomory cutting plane method (one cut at a time).*

belonging to a small region. This behavior is in a sense a consequence of the efficiency of the underlying LP solver, that has no reason to change the LP solution once it becomes optimal with respect to the original objective function—the standard dual simplex will stop as soon as a feasible point (typically very close to the previous optimal vertex) is reached. As new degenerate vertices are created by the cuts themselves, the textbook method enters a feedback loop that is responsible for the exponential growth of the determinant of the current basis, as reported in Figure 2.4(d).

On the contrary, as shown in Figures 2.4(b), the lexicographic method prevents this by always moving the fractional vertex to be cut as far as possible (in the lex-sense) from the previous one. Note that, in principle, this property does not guarantee that there will be no numerical problems, but the method seems to be work pretty well in practice.

Finally, Figure 2.4(c) offers a closer look at the effect of lexicographic reoptimization. Recall that our implementation of the lexicographic dual simplex method involves a sequence of reoptimizations, each of which produces an alternative optimal vertex possibly different from the previous one. As a result, between two consecutive cuts our method internally traces a trajectory of equivalent solutions, hence in the trajectory plotted in Figure 2.4(b) we can distinguish between two contributions to the movement of x^* after the addition of a new cut: the one due to the black-box optimizer, an the one due to lex-reoptimization. Figure 2.4(c) concentrates on the slice objective=8 of the Lex trajectory. Each lexicographic optimal vertex used for cut separation is depicted as a filled circle. The immediate next point in the trajectory is the optimal vertex found by the standard black-box dual simplex, whereas the next ones are those contributed by the lexicographic reoptimization. The figure shows that lexicographic reoptimization has a significant effect in moving the points to be cut, that in some cases are very far from those returned by the black-box dual simplex.

Figure 2.5 gives an alternative representation of the fractional solutions visited during the optimization of `stein15`. We call it *fractional spectrography*, since it depicts

the spectrum of fractionalities along the execution of the algorithm. Fractional spectrography is a pseudo-color plot of the 3-dimensional data set formed by iterations, variables, and variable values. Iterations are represented in the x-axis, variables in the y-axis, while the z-axis, representing variable values, is a grayscale color. Variables are represented in their lexicographic order: variable 0 is the objective function, variable 1 is the lexicographically first variable, x_1, and so on. For a problem with only binary variables, the white color encodes a 1 and the black color a 0. Nonbinaries variables as, for example, the objective function, are normalized into the interval $[0, 1]$. All shadings between black and white are fractional values. This coding helps giving a visualization of the degree of fractionality of a solution. In Figure 2.5, the left chart shows the behavior of the textbook implementation of Gomory cutting planes. It is clear that, since around iteration 250, the method starts visiting fractional solutions that are closer and closer one to each other, until the chart assumes an almost uniform gray shading indicating heavy fractionality persistency. On the contrary, the right-hand-side part of Figure 2.5 shows the much crisper look of the lexicographic version, where the fractional components tend to flip between integer values and fractionalities are quickly repaired.

To support the interpretation above even further, we performed the experiment of just restarting the LP solver from scratch after having generated the FGCs, so that it is more likely that a "substantially different" optimal solution is found. This small change had a significant impact on the performance of the textbook method (though not comparable to that derived from the use of the lexicographic method), showing the importance of breaking the correlation of the optimal LP bases.

A second set of experiments was carried out on the *multi-cut* version of Gomory's algorithm, where cuts are generated in rounds. To be specific, after each LP reoptimization we consider all the tableau rows with fractional basic variable, and generate two FGCs from each row—one from the row itself, and one from the same row multiplied by -1.

The corresponding results are reported in Table 2.3: the multi-cut version of Lex performed even better than in the single-cut mode: in 13 out of the 26 instances the method reached the optimum.

Table 2.4 reports the results of our two heuristics, *Heur1* and *Heur2*. A comparison with the previous table shows that both heuristics are effective in controlling the coefficient size, determinant, and condition number. The average closed gap is significantly better than in TB, but clearly worse than in Lex.

Problem	Textbook							Lex						
		Itrs	Cuts	Time	ClGap	Coeff.	κ		Itrs	Cuts	Time	ClGap	Coeff.	κ
air04	N	29	9277	711.79	9.07	8.8e+02	2.8e+09	T	413	133647	3604.53	**24.86**	6.1e+04	9.1e+11
air05	N	43	13342	1061.33	5.32	2.1e+03	2.4e+09	T	776	224190	3600.25	**24.66**	8.9e+03	7.2e+09
bm23	N	107	1348	0.48	18.09	1.6e+09	1.4e+12	O	658	8290	1.11	**100.00**	3.4e+02	2.6e+06
cap6000	N	339	3378	200.75	8.47	3.9e+07	6.9e+15	N	5200	36832	1436.66	**30.27**	3.9e+07	2.7e+18
hard_ks100	N	1752	10689	1037.46	100.00	1.9e+08	4.5e+15	O	105	519	0.29	**100.00**	3.6e+05	1.7e+12
hard_ks9	N	265	1830	0.18	100.00	9.6e+04	1.8e+10	O	139	601	0.11	**100.00**	3e+04	4e+09
krob200	O	101	5029	339.18	100.00	2.4e+02	4.2e+08	O	39	1640	62.86	**100.00**	1.4e+02	1.7e+07
l152lav	N	440	25793	1638.96	32.97	1.1e+03	3.6e+09	O	742	25091	116.45	**100.00**	1.5e+04	1e+09
lin318	O	18	467	22.51	**100.00**	9.3	3.8e+05	O	26	951	106.25	100.00	21	7.9e+06
lseu	N	116	2543	2.78	46.03	6.3e+08	7e+14	O	15662	219216	85.02	**100.00**	1.1e+05	4.7e+10
manna81	O	1	271	4.24	**100.00**	1	1.3e+05	O	10	279	9.68	100.00	1	2.7e+05
mitre	T	154	47919	3798.87	84.80	2.2e+08	1e+18	T	225	71317	3659.66	**87.70**	1.7e+08	1.2e+16
mzzv11	T	20	14379	1940.34	**33.59**	1.2e+04	1.7e+10	T	16	14440	3603.24	31.07	7.2e+05	3.3e+13
mzzv42z	T	20	6887	426.10	**22.62**	1.4e+06	3.7e+12	T	20	15550	3602.99	20.17	3e+07	3.5e+13
p0033	N	368	4511	3.51	95.60	7.8e+08	9.7e+13	O	499	4419	0.94	**100.00**	2.1e+03	1.8e+07
p0201	N	211	5876	23.34	24.19	4.2e+08	1.9e+14	T	176477	4083975	3601.02	**98.24**	3.4e+06	1.5e+15
p0548	T	781	42639	217.06	**52.02**	5.5e+08	2.4e+20	N	1000	51456	46.07	51.78	2.2e+07	1.2e+15
p2756	T	231	9369	274.22	78.63	8.4e+08	1.2e+17	T	22936	326045	3601.11	**79.09**	3.6e+07	1.2e+17
pipex	N	2680	41857	11.67	50.23	6.3e+08	1.9e+13	O	355901	3170373	665.78	**100.00**	4.3e+06	6.2e+13
protfold	T	8	4808	19687.73	8.76	1.3e+08	1.5e+14	T	149	59515	3613.59	**45.26**	15	2.5e+07
sentoy	N	52	707	0.18	3.39	4.7e+08	2.5e+14	O	5348	69208	14.33	**100.00**	9.1e+04	1.4e+10
seymour	N	12	7950	1626.75	16.45	5.1e+07	5.9e+13	T	107	62266	3604.82	**26.89**	3.3e+02	1.2e+08
stein15	N	51	1261	0.37	50.00	1.6e+09	3e+11	O	66	688	0.12	**100.00**	6.3	1.7e+03
stein27	N	40	1821	1.06	0.00	1.2e+09	4.8e+12	O	3132	35859	9.72	**100.00**	77	1.7e+05
timtab1-int	T	148	44479	14.09	28.92	1.4e+08	1.1e+15	N	2982	921030	353.47	**44.20**	1.2e+08	1.1e+16

Table 2.3: Comparison between textbook and lexicographic implementation of Gomory's algorithm (multi-cut version)

	Heur1							Heur2						
Problem		Itrs	Cuts	Time	ClGap	Coeff.	κ		Itrs	Cuts	Time	ClGap	Coeff.	κ
air04	M	29	9277	711.79	9.07	8.8e+02	2.8e+08	M	26	7671	359.91	**11.90**	2.5e+03	9.1e+10
air05	M	43	13342	1061.33	5.32	2.1e+03	7.5e+09	M	26	7632	399.20	5.32	1.1e+03	3.1e+09
bm23	N	107	1348	0.48	18.09	1.6e+09	1.4e+18	M	660	13828	331.01	**25.54**	2.3e+06	6.9e+14
cap6000	N	339	3378	200.75	8.47	3.9e+07	2.6e+12	M	437	5478	1884.77	**9.10**	1.4e+06	4.5e+14
hard_ks100	N	1752	10689	1037.46	100.00	1.9e+08	1.1e+13	O	3234	6806	432.71	**100.00**	7.1e+03	9.9e+09
hard_ks9	O	265	1830	0.18	**100.00**	9.6e+04	2.3e+05	O	281	2138	0.41	100.00	2.4e+05	5.8e+10
krob200	O	101	5029	339.18	**100.00**	2.4e+02	1.3e+07	O	105	6371	442.80	100.00	2.6e+02	9.6e+07
l152lav	M	440	25793	1638.96	32.97	1.1e+03	1.7e+05	M	200	13912	1118.15	**34.49**	1.2e+04	7.7e+10
lin318	O	18	467	22.51	**100.00**	9.3	5.4e+08	M	29	1559	262.08	99.01	24	1.6e+08
lseu	N	116	2543	2.78	46.03	6.3e+08	9.2e+18	M	520	16394	333.86	**47.78**	2.2e+05	8.3e+12
manna81	O	1	271	4.24	**100.00**	1	9.2e+18	O	1	271	4.39	100.00	1	9.8e+04
mitre	T	154	47919	3798.87	84.80	2.2e+08	9.2e+18	O	153	22616	885.86	**100.00**	5.8e+04	8.1e+13
mzzv11	M	20	14379	1940.34	33.59	1.2e+04	9.2e+18	M	50	21526	2188.58	**40.21**	1.9e+02	1.5e+11
mzzv42z	M	20	6887	426.10	22.62	1.4e+06	9.2e+18	M	70	15777	1593.59	**38.97**	1.3e+03	1.5e+12
p0033	N	368	4511	3.51	95.60	7.8e+08	3.4e+18	O	201	2222	0.41	**100.00**	1.1e+04	6.1e+08
p0201	N	211	5876	23.34	24.19	4.2e+08	9.2e+18	M	670	25540	1462.61	**36.08**	1.1e+06	8.2e+13
p0548	N	781	42639	217.06	**52.02**	5.5e+08	9.2e+18	M	350	22082	1360.48	47.03	3.5e+04	1e+15
p2756	N	231	9369	274.22	**78.63**	8.4e+08	9.2e+18	M	250	15315	1127.85	78.17	5.7e+03	5.7e+12
pipex	N	2680	41857	11.67	**50.23**	6.3e+08	9.2e+18	M	2240	40953	390.15	49.08	1.2e+08	6.7e+14
protfold	N	8	4808	19687.73	8.76	1.3e+08	9.2e+18	T	21	6994	3788.34	8.76	1.6	3.5e+06
sentoy	N	52	707	0.18	3.39	4.7e+08	9.2e+18	M	780	15678	346.80	**19.74**	3.5e+05	9.8e+13
seymour	N	12	7950	1626.75	16.45	5.1e+07	9.2e+18	M	20	9993	2336.60	**21.67**	3.7	2.6e+06
stein15	N	51	1261	0.37	50.00	1.6e+09	1.1e+17	M	500	13772	242.22	50.00	54	6.5e+05
stein27	N	40	1821	1.06	0.00	1.2e+09	9.2e+18	M	280	13604	151.34	0.00	5.8	1.3e+04
timtab1-int	N	148	44479	14.09	28.92	1.4e+08	9.2e+18	M	1320	413451	747.77	**31.95**	1.8e+06	7.3e+14

Table 2.4: The two heuristics compared (multi-cut version).

Figures 2.6 and 2.7 give some illustrative plots for instance `sentoy`. The figures clearly show the typical degenerate behavior of TB, with instable phases of rapid growth of determint/coefficients/κ exploring small space regions with shallow cuts. It is worth observing the striking difference in the plots of the *average cut depth*, computed as the geometric distance of the cut from the separated vertex, averaged over all the cuts in a round. Even more interesting, the TB and Lex have a completely different behavior as far as the *optima distance* (computed as the Euclidean distance between two consecutive fractional vertices to be cut) is concerned. As a matter of fact, as already shown by Figure 2.4, lexicographic reoptimization is quite successful in amplifying the dynamic (and diversity) of the fractional solutions.

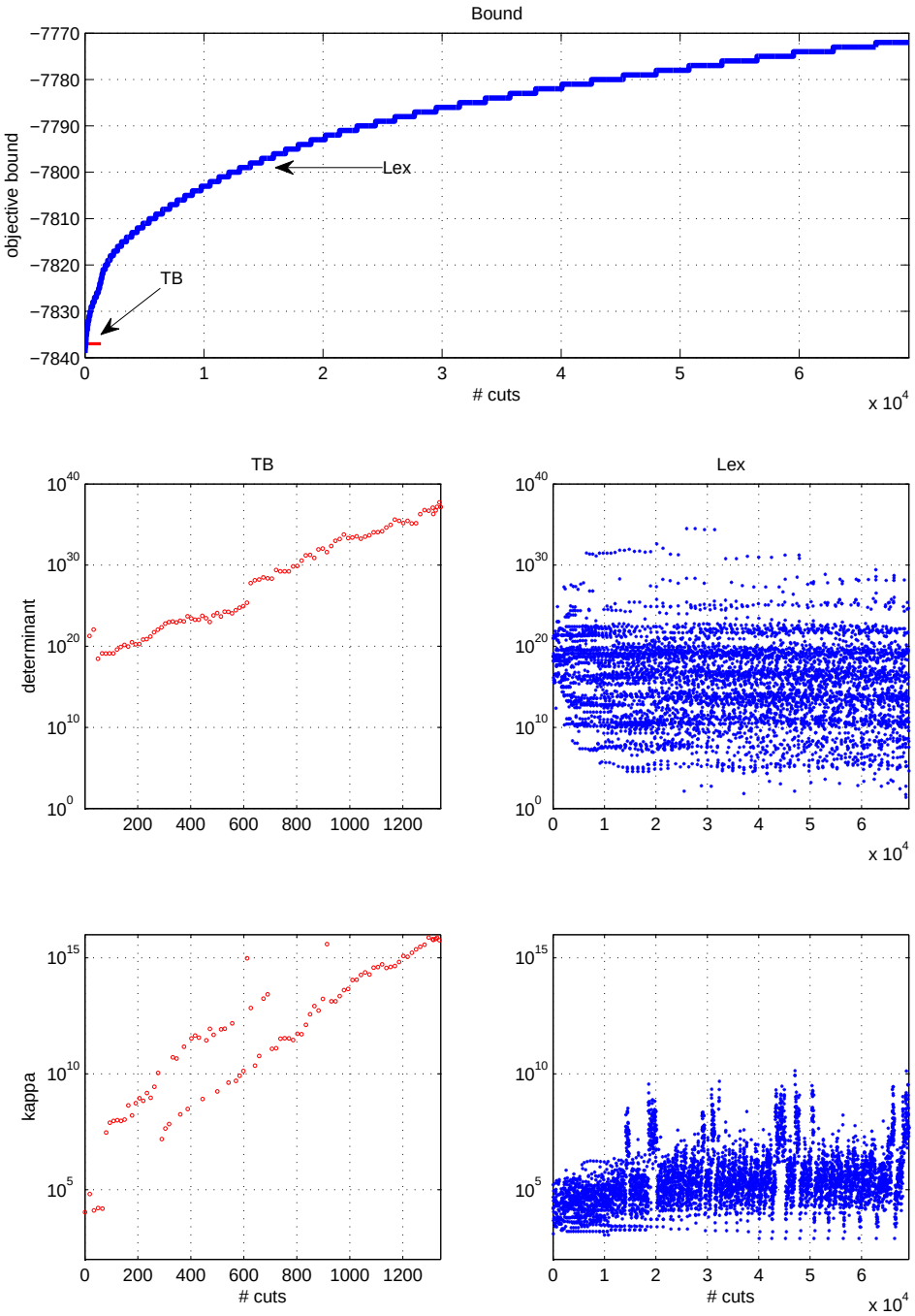

Figure 2.6: Comparison between the textbook and lexicographic implementations of the multi-cut algorithm on `sentoy`.

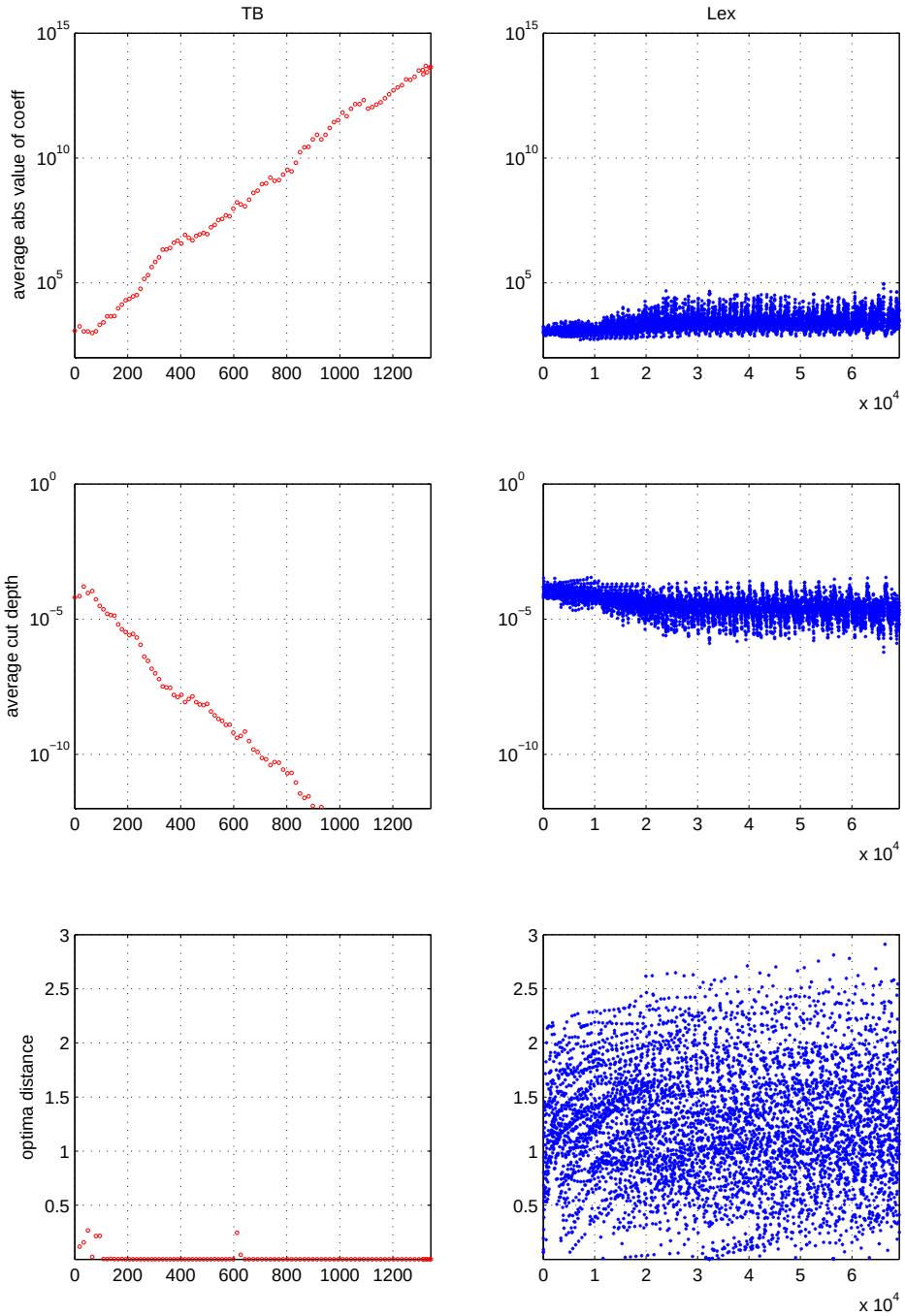

Figure 2.7: Comparison between the textbook and lexicographic implementations of the multi-cut algorithm on `sentoy`

2.7 Approximating MIG cuts for pure integer programs

Using MIG cuts in an all-integer context has the undesired feature of introducing continuous slack variables. This fact can be a problem for two main reasons. First, observations in Section 2.5 do not hold any more for MIG cuts, since Definition 1 relies on the Chvátal-Gomory rounding argument. Second, fractional values affected by round off errors are introduced in the all-integer initial formulation, and can significantly contribute to the numerical degradation of the method.

State-of-the-art cut generators use various heuristics to try to ensure the validity of a cut. For example, they try to guess the rational representation of the tableau row and round the coefficients accordingly, in a conservative way. This is important to reduce the propagation of numerical errors during iterations, even if it is unlikely to be applicable for hard problems or after a certain number of iterations. Often, cut generators even weaken the right-hand-side of the derived MIG cuts. However, as already discussed, cut weakening can be a very poor choice in an iterative framework intended to produce long sequences of stable cuts (see Section 2.5).

A common practice in B&C codes consists in discarding cuts that have a certain probability of being invalid, or are numerically unpleasant. Typical criteria used to discard cuts are the size of coefficients, the dynamism (ratio between the largest and the smallest absolute value of coefficients), and the density. Discarding cuts is quite successful in limiting numerical in B&C codes, but it is badly suited for a pure cutting plane algorithm since it can lead to early termination.

In an early stage of our work, we tested the COIN-OR [52] MIG cut generator embedded in the textbook cutting plane framework, and compared its performance to that of our lexicographic implementation using FGCs. Results are reported in Table 2.5, showing that the FGC lexicographic method outperforms the textbook implementation even though the latter uses the stronger MIG cuts.

Problem	TB (with MIGs)							Lex (with FGCs)						
		Itrs	Cuts	Time	ClGap	Coeff.	κ		Itrs	Cuts	Time	ClGap	Coeff.	κ
air04	T	190	55919	3602.92	16.25	6.9e+08	8.5e+15	T	413	133647	3604.53	**24.86**	6.1e+04	9.1e+11
air05	M	112	26087	2609.82	8.33	2e+08	1.1e+14	T	776	224190	3600.25	**24.66**	8.9e+03	7.2e+09
bm23	T	5697	51954	3602.98	31.21	1.9e+11	6.5e+18	O	658	8290	1.11	**100.00**	3.4e+02	2.6e+06
cap6000	E	29	267	3.23	**62.68**	2.7e+07	7.1e+11	N	5200	36832	1436.66	30.27	3.9e+07	2.7e+18
hard_ks100	E	42	143	0.05	—	1.5e+07	5.4e+15	O	105	519	0.29	**100.00**	3.6e+05	1.7e+12
hard_ks9	E	1096	6654	17.40	—	6.7e+09	7.3e+19	O	139	601	0.11	**100.00**	3e+04	4e+09
krob200	E	248	15879	1615.20	92.83	1.1e+09	8.4e+12	O	39	1640	62.86	**100.00**	1.4e+02	1.7e+07
l152lav	E	145	10626	703.54	35.28	4.1e+08	2.9e+14	O	742	25091	116.45	**100.00**	1.5e+04	1e+09
lin318	E	88	10082	1761.61	96.38	1e+09	5.8e+14	O	26	951	106.25	**100.00**	21	7.9e+06
lseu	E	43	631	0.12	69.96	6.8e+08	4.9e+12	O	15662	219216	85.02	**100.00**	1.1e+05	4.7e+10
manna81	O	1	870	0.41	100.00	1	3.6e+06	O	10	279	9.68	**100.00**	1	2.7e+05
mitre	E	11	1235	2.03	**100.00**	4.6e+08	1.6e+16	T	225	71317	3659.66	87.70	1.7e+08	1.2e+16
mzzv11	E	93	40159	2551.53	**60.95**	1.3e+08	8.5e+15	T	16	14440	3603.24	31.07	7.2e+05	3.3e+13
mzzv42z	E	87	38099	1277.54	**42.28**	3.7e+08	2.6e+14	T	20	15550	3602.99	20.17	3e+07	3.5e+13
p0033	E	466	6316	21.57	79.69	1.5e+10	9.4e+15	O	499	4419	0.94	**100.00**	2.1e+03	1.8e+07
p0201	E	113	6987	22.27	67.92	2.3e+09	1.1e+13	T	176477	4083975	3601.02	**98.24**	3.4e+06	1.5e+15
p0548	E	191	5590	35.18	**94.84**	1.5e+09	7.5e+15	N	1000	51456	46.07	51.78	2.2e+07	1.2e+15
p2756	E	48	812	4.19	**98.41**	1.3e+07	9.9e+09	T	22936	326045	3601.11	79.09	3.6e+07	1.2e+17
pipex	E	334	3814	4.29	48.95	2.9e+10	9.4e+19	O	355901	3170373	665.78	**100.00**	4.3e+06	6.2e+13
protfold	T	52	22482	3625.47	21.73	8.5e+04	1e+09	T	149	59515	3613.59	**45.26**	15	2.5e+07
sentoy	E	56	476	0.11	28.24	5.9e+08	1.5e+12	O	5348	69208	14.33	**100.00**	9.1e+04	1.4e+10
seymour	T	109	53009	3611.00	26.89	1.4e+06	3.6e+13	T	107	62266	3604.82	26.89	3.3e+02	1.2e+08
stein15	T	6793	99742	3601.70	75.00	2.1e+09	1.7e+21	O	66	688	0.12	**100.00**	6.3	1.7e+03
stein27	T	2439	62996	3601.19	0.00	1.8e+06	4.1e+14	O	3132	35859	9.72	**100.00**	77	1.7e+05
timtab1-int	E	158	26804	521.49	38.77	6.9e+08	1.2e+17	N	2982	921030	353.47	**44.20**	1.2e+08	1.1e+16

Table 2.5: Comparison between MIG cuts in a textbook implementation of Gomory's algorithm, and FGC cuts with the lexicographic simplex (multi-cut version)

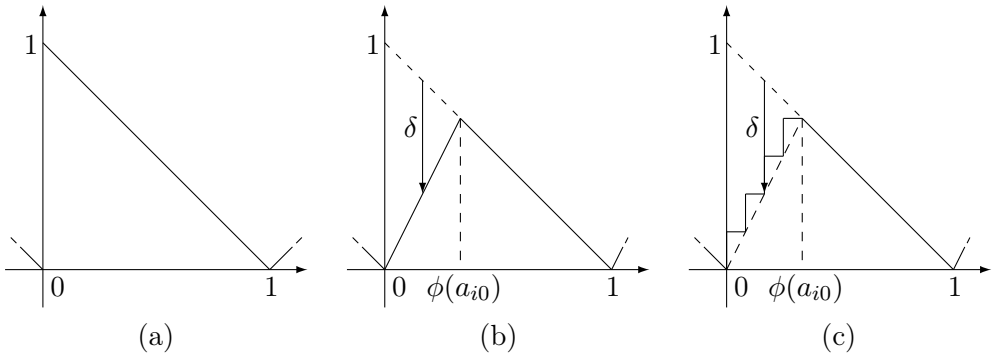

Figure 2.8: Different subadditive functions: (a) *FGC*, (b) *MIG*, (c) *d-MIG*

We also tried to embed the COIN-OR MIG separator in our lexicographic framework. Unfortunately, due to the aforementioned sophisticated policies applied during separation, it is difficult to ensure that all the theoretical requirements of the Gomory's convergence proof are fulfilled, and the performance turned out to be very poor also in practice. To take control over all aspects of MIG separation, we therefore decided to implement our own version. First we tested a naïve implementation, without any particular strategy for protecting against possible numerical errors. As expected, invalid cuts appeared very soon, after just few iterations.

Our goal was to weaken MIG cuts in very controlled way, so as to obtain an all-integer counterpart that ensures cut validity (under the assumptions of Section 2.5) and preserves the strong properties leading to a finitely-convergent cutting plane method. To this end, it is well known [62] that a FGC, in its fractional form, can be obtained by applying the subadditive function $\phi(a_{ij}) = \lceil a_{ij} \rceil - a_{ij} = 1 - f_{ij}$ to the coefficients of row i. Similarly, MIG cuts can be obtained using the subadditive function

$$\phi_{MIG}(a_{ij}) = \phi(a_{ij}) - \delta_j$$

where

$$\delta_j = max \left\{ 0,\ \frac{\phi(a_{ij}) - \phi(a_{i0})}{1 - \phi(a_{i0})} \right\}$$

See Figure 2.8 for an illustration. So let us rewrite the MIG cut in the following form:

$$\sum_{j \in N \setminus B} (\phi(a_{ij}) - \delta_j) x_j \geq \phi(a_{i0}) \tag{2.4}$$

Note that this latter cut shares with (2.1) the sign pattern used (together with the lexicographic property) to prove convergence; see Section 2.3.1.

Now, given $\alpha \in \mathbb{Z}_+$ we can approximate (from below) δ_j by replacing it with its discretized counterpart k_j/α, where k_j is the largest positive integer such that $k_j/\alpha \leq \delta_j$. By construction, the resulting d-MIG (d for discretized) cut

$$\sum_{j \in N \setminus B} (\phi(a_{ij}) - \frac{k_j}{\alpha}) x_j \geq \phi(a_{i0}) \tag{2.5}$$

is a weakening of the MIG cut (hence its validity) that dominates the FGC. In other words, d-MIG cuts play an intermediate role between MIG cuts and FGCs, and they get closer and closer to MIG cuts as parameter α increases. Our procedure might resemble the strengthening procedure of Letchford and Lodi [45]. However the cuts in [45] are substantially different from ours, since d-MIG cuts are a dominated (but hopefully more stable) version of MIGs while there is no strict dominance relationship between MIGs and cuts in [45].

Since α is chosen beforehand, all computations above can be done in integers. In order to derive the cut we can therefore use the numerically more stable rounding function defined in (2.3). Note that the slack substitution phase needed to add the cut to the previous LP, may involve fractional slacks variables from other cuts. There are two ways of handling this potential source of inaccuracy. The first is to multiply (2.5) by α so that the cuts added to the problem are all-integer. The second possibility is based on the fact that all slacks are integer multiples of the common fraction $1/\alpha$, so it is easy to compute their rational representation when needed for the slack substitution. In a preliminary computation, the second method exhibited a slightly better numerical stability, so it is the one implemented in our code.

Figure 2.9 illustrates, for a small problem (`bm23`), the effect of separating d-MIG cuts of different approximation levels within a lexicographic framework, till proved optimality is reached. In the figure on the right-hand side, the the x-axis reports approximation error $1/\alpha$ for $\alpha = 1$, 2, 4, 8, and 16, namely 1, 0.5, 0.25, 0.125, and 0.0625 respectively. Tighter approximations were tried but turned out to be numerically unstable, and the run was aborted before convergence. The y-axis reports the number of bits required during the overall process to represent cut coefficients (e.g., to obtain an approximation error of 0.0625, we need approximately 29 bits). It can be seen that, as the approximation gets closer to the real MIG (from left to right in the x-axis), the size of coefficients first grows exponentially, and then tends to saturate along the MIG asymptote. Figure 2.9 (left) illustrates the evolution of the coefficient sizes when moving from FGCs (approximation error 1) to cuts closer and closer to MIG cuts (approximation error 0.25 and 0.0625). On the x-axis we report the iteration (i.e., cut) number, whereas the y-axis gives the maximum number of bits required to represent cut coefficients in each iteration. According to the figure, the three versions with approximation error 1, 0.25 and 0.0625 converge to the optimal solution in 651, 663, and 714 iterations, respectively. This is quite unsatisfactory, as the increased strength of d-MIG cuts reduces the overall number of iterations by, at most, 9%. Moreover, the coefficient fluctuation is highly amplified when the approximation error becomes smaller and smaller. These results were confirmed on other instances, and seem to indicate that the size of coefficients of d-MIG cuts grows exponentially with α, thus making their usage impractical even when cuts are embedded in a lexicographic framework. This also suggests that MIG cuts themselves can be very complex to manage by a pure cutting plane method.

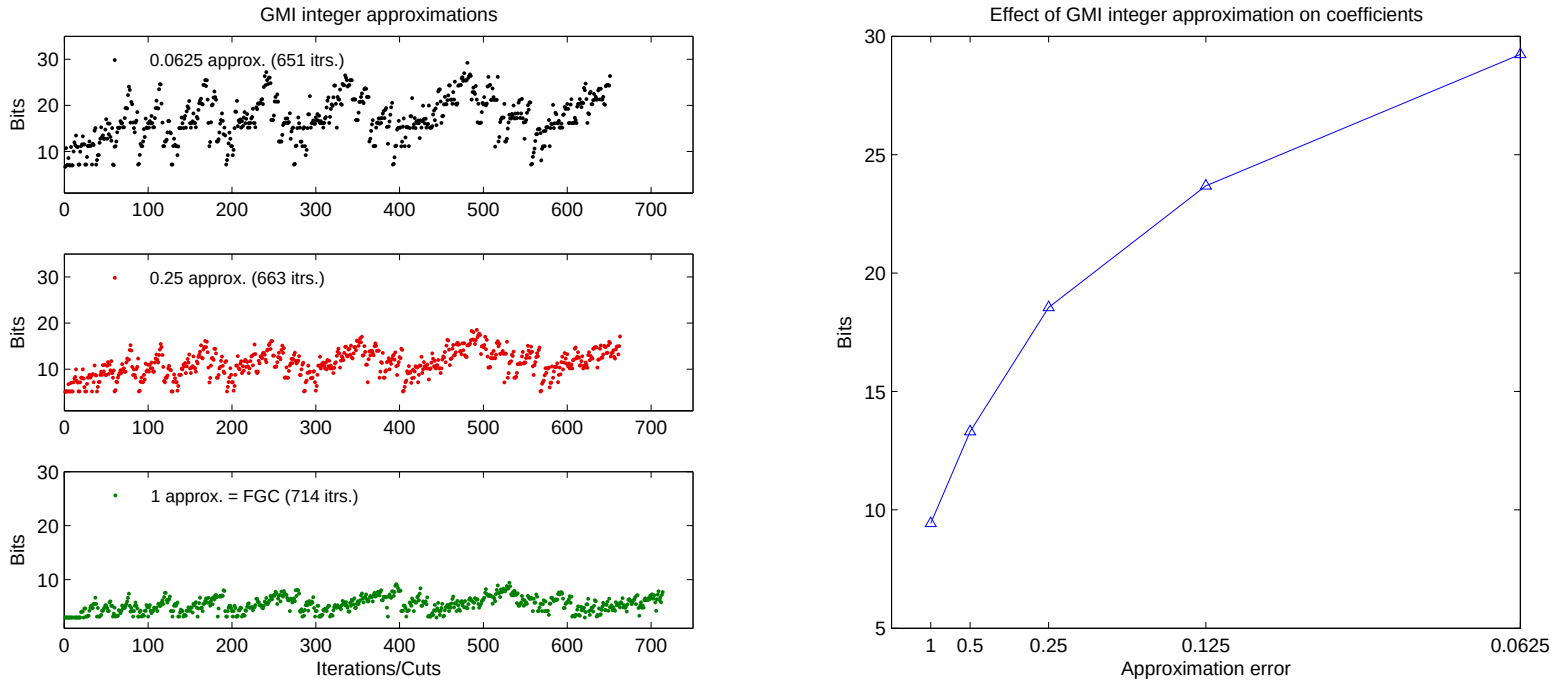

Figure 2.9: Gomory's lexicographic method (one cut at a time) using d-MIG cuts on instance `bm23`

2.8 Conclusions and future work

Pure cutting plane algorithms have been found not to work in practice because of numerical problems due to the cuts becoming increasingly parallel (a phenomenon accompanied by dual degeneracy), increasing determinant size and condition number, etc. For these reasons, cutting planes are in practice used in cut-and-branch or branch-and-cut mode.

In this chapter we have discussed an implementation of the lexicographic version of Gomory's fractional cutting plane method and of two heuristics mimicking the latter one. In computational testing on a battery of MIPLIB problems, we compared the performance of these variants with that of the standard Gomory algorithm, both in the single-cut and in the multi-cut (rounds of cuts) version, and showed that they provide a radical improvement over the standard procedure. In particular, we reported the exact solution of ILP instances from MIPLIB such as `stein15`, `stein27`, and `bm23`, for which the standard Gomory cutting plane algorithm is not able to close more than a tiny fraction of the integrality gap.

We have identified the right choice of direction in rounding the tableau coefficients when generating cuts, which turned out to be very effective (together with the lexicographic simplex) in producing numerically stable cuts.

Future work should address the integration of lexicographic simplex with other kinds of cuts, including MIG cuts. We made a first step in this direction, by introducing a numerically more stable discretized version of MIG cuts. Our preliminary computational results seem however to indicate that these cuts (and hence MIG cuts) are intrinsically more difficult to handle than FGCs, at least within a cutting plane method for pure ILPs.

Chapter 3

Minimal Infeasible Subsystems and Benders cuts

3.1 Introduction

There are many situations in mathematical programming where cutting planes can be generated by solving a certain Cut Generation Linear Program (CGLP) whose feasible solutions define a family of valid inequalities for the problem at hand. Disjunctive cuts and Benders cuts are two familiar examples.

Benders cuts were originally proposed in [12] as a machinery to convert a generic mixed-integer program involving integer variables x and continuous variable y into an integer program involving the x variables only, possibly plus a single continuous variable η taking into account the overall contribution to the objective function of the continuous variables (say $d^T y$). The continuous y variables are projected away by a standard projection technique based on dynamic cutting-plane generation. At each iteration, one solves the current *master problem* relaxation in the (x, η) space, and sends the optimal solution (x^*, η^*) to the so-called *slave problem*. This is an LP in the y space that tries to define suitable y-variables y^* such that (x^*, y^*) is feasible for the original problem, and $\eta^* = d^T y^*$. If the slave problem is feasible, we are done. Otherwise, a so-called (feasibility or optimality) *Benders cut* in the (x, η) space is generated by using Farkas' characterization of infeasible LPs, the cut is added to the master problem, and the method is iterated.

The definition of the CGLP is however only the first step for the effective use of the associated cuts, as three main topics need to be addressed:

i) *When to cut?* Possible answers range from "only when an integer super-optimal solution is available" (as in the original proposal of Benders, where cuts are applied only to cut the optimal solution of the current master problem) to "whenever a fractional (or integer infeasible) solution is available" (as in modern branch-and-cut frameworks).

ii) *What to cut?* The usual choice in integer programming is to cut the optimal solution of an LP relaxation. However this may lead to unstable behavior and

slow convergence, so stabilization through box constraints or quadratic penalty functions may be needed–this is not usually done in standard branch-and-cut algorithms, but it is common practice e.g. in bundle methods.

iii) *How to choose the cut?* Given the point x^* to be separated, choose the "best possible" cut(s) among the violated ones.

All three points above play an important role in the design of an effective solution method. In this chapter we focus on (iii), and in particular we address the topic of selecting in an effective way Benders cuts for general Mixed-Integer Linear Programs (MIPs). As we aim at understanding the properties that make a single Benders cut "a good cut" in a branch-and-cut context, in the present study we do not address alternative solution approaches that generate rounds of Benders cuts (as opposed to just one or two cuts) at each separation call—though we believe that the idea of generating a large number of simultaneous cuts can play an important role in speeding up the overall convergence of any cutting plane method, including the Benders' one.

We propose alternative selection criteria for Benders cuts, and analyze them computationally. As customary in mixed-integer programming, the effectiveness of the generated cuts is measured by the quality of the root node bound.

Our approach is based on the correspondence between minimal infeasible subsystems of an infeasible LP, and the vertices of the so-called *alternative polyhedron*. The choice of the "most effective" violated Benders cut then corresponds to the selection of a suitable vertex of the alternative polyhedron, hence a clever choice of the dual objective function is crucial—whereas the textbook Benders approach uses a completely random selection policy, at least when feasibility cuts are generated.

Computational results on a testbed of MIPLIB instances are presented, where the quality of Benders cuts is measured in terms of "percentage of gap closed" at the root node, as customary in cutting plane methods. We show that the proposed methods allow for a speedup of 1 to 2 orders of magnitude with respect to the textbook one.

3.2 Benders cuts: theory ...

Suppose we are given a MIP problem

$$\begin{aligned} \min c^T x + d^T y \\ Ax \geq b \\ Tx + Qy \geq r \\ x \geq 0,\ x \text{ integer} \\ y \geq 0 \end{aligned} \tag{3.1}$$

where $x \in \Re^n$, $y \in \Re^t$, and matrix Q has m rows.

Classical Benders decomposition states that solving such a problem is equivalent to

solving

$$\begin{aligned} \min c^T x + \eta \\ Ax \geq b \\ \eta \geq u^T(r - Tx), \quad u \in \text{VERT} \\ v^T(r - Tx) \leq 0, \quad v \in \text{RAY} \\ x \geq 0,\ x \text{ integer} \end{aligned} \tag{3.2}$$

where the additional variable η takes into account the objective function term $d^T y$, while sets VERT and RAY contain the vertices and extreme rays (respectively) of the polyhedron D defined by:

$$\begin{aligned} \pi^T Q \leq d^T \\ \pi \geq 0 \end{aligned} \tag{3.3}$$

The above formulation has exponentially many inequalities, so an iterative solution approach based on cutting planes is needed, that can be outlined as follows.

1. Solve the so-called *master problem*:

$$\begin{aligned} \min c^T x + \eta \\ Ax \geq b \\ \{\text{previously generated Benders cuts}\} \\ x \geq 0,\ x \text{ integer} \end{aligned} \tag{3.4}$$

   including (some of) the Benders cuts generated so far (none at the very beginning). Let (x^*, η^*) be an optimal solution of the master problem.

2. Solve the so-called *dual slave problem*:

$$\begin{aligned} \max \pi^T(r - Tx^*) \\ \pi^T Q \leq d^T \\ \pi \geq 0 \end{aligned} \tag{3.5}$$

3. If the dual slave problem is unbounded, choose any unbounded extreme ray $\overline{v}$, and add the so-called *Benders feasibility cut*

$$\overline{v}^T(r - Tx) \leq 0$$

   to the master and go to Step 1. Otherwise, let the optimal value and an optimal vertex be z^* and $\overline{u}$ respectively. If $z^* \leq \eta^*$ then stop. Otherwise, add the so-called *Benders optimality cut*

$$\eta \geq \overline{u}^T(r - Tx)$$

   to the master problem, and go to Step 1.

The distinction between *optimality cuts* (involving the η variable) and *feasibility cuts* (that assert some property of the feasible x vector) is very important in practice, and will be analyzed in greater detail in the sequel.

As already noted by other authors, but seldom applied in practice, Benders cuts can be generated to separate any solution (integer or not) of the master problem. As a consequence, these cuts can easily be embedded into a modern branch-and-cut scheme where Benders cuts (among others) are generated at each node of the branching tree.

Note that:

- Although presented for the MIP case, the Benders framework is by no means limited to it. In particular, any problem of the form

$$\begin{aligned} \min c(x) + d^T y \\ g(x) \geq 0 \\ F(x) + Qy \geq r \\ y \geq 0 \end{aligned} \tag{3.6}$$

  with arbitrary $c()$, $g()$ and $F()$ is suitable to be solved with this method, provided that we have a solver for the master problem (see [32]). This also means that, given any arbitrary partition of the variables, any linear programming problem can be casted into the Benders framework, by projecting away a subset of the variables. This is indeed done in practice with problems that simplify considerably (e.g., decompose) after fixing a subset of their decision variables—this is the case, e.g., in Stochastic Linear Programs (SLPs).

- The Benders method is in fact a pure cutting plane approach in which, given a solution (x^*, η^*) of a problem relaxation (the master), we look for a violated valid inequality. In particular, the search for such an inequality is done by solving an LP problem (the dual slave), which acts as a *Cut Generating LP* akin to the one used in disjunctive programming (as a matter of fact, disjunctive cuts can be viewed as Benders cuts derived from a compact extended formulation).

- The set of Benders cuts corresponds to the vertices and extreme rays of D and is independent of the current master solution (x^*, η^*), which is used only to decide which is next cut to add. For this purpose a suboptimal (or even infeasible) master solution can be used as well, as e.g. in the recent proposals by Rei e al. [65] and by Poojari and Beasley [64].

Given the considerations above, in the following we focus on a generic LP of the form

$$\begin{aligned} \min c^T x + d^T y \\ Ax \geq b \\ Tx + Qy \geq r \\ x \geq 0 \\ y \geq 0 \end{aligned} \tag{3.7}$$

This LP may be the root relaxation of a MIP problem, or just a large-scale LP problem suitable for Benders decomposition (e.g., a SLP problem).

3.3 ... and practice

The first question we asked ourselves was: What can be considered a modern, yet classical, implementation of Benders decomposition to be used for benchmarking purposes? As a matter of fact, any implementation of the Benders approach has to face a number of implementation issues that affect heavily the overall performance of the method, and many authors using Benders cuts tend to classify their methods as just "standard implementations" without giving sufficient details.

A first issue is how to obtain a good, yet easily computable, initial lower bound on η, so as to prevent the generation of several dominated (and thus useless) optimality cuts. From a theoretical point of view, we are interested in the best-possible optimality cut of the form

$$\eta \geq \pi^T r - 0^T x$$

so $\pi^T r$ can be obtained by just solving the LP:

$$\begin{aligned} \max \pi^T r \\ \pi^T Q \leq d^T \\ \pi^T T = 0^T \\ \pi \geq 0 \end{aligned} \tag{3.8}$$

However, if the slave problem does not have a special structure (i.e., if it does not decompose nicely), the introduction of the coupling matrix T yields an LP problem of the same size as the original LP, so this approach is not always viable computationally. Therefore, in our tests we prefer to calculate a trivial bound on $d^T y$ based only on the lower and upper bounds on the y variables (if no bounds are given, we just write $\eta \geq -M$ for a suitably large M).

Then we addressed the relative contribution of optimality and feasibility cuts to the convergence of the method. Indeed, according to our computational experience these two classes of cuts behave quite differently in many important respects:

- For many problems where term $d^T y$ gives a significant contribution to the overall optimal value, optimality cuts can be much more effective in moving the bound

than feasibility cuts, because they involve the η variable explicitly.

- Optimality cuts are typically quite bad from a numerical point view. In particular, optimality cuts tend to exhibit an higher *dynamism* than feasibility cuts, i.e., a higher ratio between the maximum and minimum absolute value of the cut coefficients. This was somewhat expectable, because optimality cuts have to take into account the objective function, which may be of a completely different magnitude (and precision) with respect to the constraints.

- Optimality cuts tend to be much denser than the feasibility ones. Again, this is not surprising since the role of optimality cuts is to provide a lower bound on the objective function term η that is based on the value of the variables x of the master problem, and it is unlikely that just a few master variables can succeed in producing a tight bound.

As a consequence, it is important to have some control on the kind (and quality) of Benders cuts generated at each iteration. Unfortunately, Benders decomposition—as it is typically implemented in the literature—is heavily biased toward feasibility cuts. As a matter of fact, as long as a violated feasibility cut exists, the dual slave is unbounded and hence no optimality cut is generated. As noted by Benders himself [12], however, if we solve the dual slave with the primal simplex method, then when we discover an unbounded ray we are "sitting on a vertex" of polyhedron D, and thus we can generate also an optimality cut with no additional computational effort. A main drawback of this approach is that optimality cut produced is not guaranteed to be violated, and in any case its discovery was quite "random" as the corresponding vertex is by no mean a one maximizing a certain quality index such as cut violation, depth, etc.

The lack of control on the *quality* of the Benders cuts is even more striking when feasibility cuts are generated, since the textbook method does not give any rule to choose among the unbounded rays. To illustrate this important (and often underestimated) point, suppose that we want to apply a textbook Benders decomposition approach to the well-known Asymmetric Traveling Salesman Problem (ATSP). Our compact MIP formulation then involves binary variables x_{ij} associated with the arcs of digraph $G = (V, A)$, and continuous flow variables y^k_{ij} that describe a flow of value 1 from a fixed source node (say node 1) to sink node k, for all $k \in V \setminus \{1\}$. In this example, system $Ax \geq b$ corresponds to in- and out-degree restrictions, whereas system $Tx + Qy \geq r$ is made by $|V| - 1$ independent blocks corresponding to the flow-conservation equations for each k, plus the coupling constraints $y^k_{ij} \leq x_{ij}$ for all $k \in V \setminus \{1\}$ and $(i, j) \in A$. It is not hard to see that, in this case, Benders cuts are of the feasibility type only, and correspond to the classical Subtour Elimination Constraints (SECs) of the form $\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq 1$. These cuts are known to be facet-defining (assuming G is complete digraph), hence they are very strong in practice—so we can conclude that "Benders cuts make a wonderful job". What is clearly inefficient is instead the way these cuts would be handled by the standard Benders method. First of all, SECs would be generated only after having solved to proven optimality the current master,

and used to cut integer points only. This is clearly inefficient, since SECs should be generated at each node of the branching tree, or at least whenever the incumbent solution is updated (as in the old-day method by Miliotis [59, 60]). But even if SECs were generated within a modern branch-and-cut framework, what is completely missing in the Benders method is a sensible cut selection criterion—once a violated SEC exists, the dual slave becomes unbounded and *any* violated SEC can be returned by the separation procedure—whereas we know that SEC density (among other characteristics) plays a crucial role in speeding-up convergence.

The considerations above prompted us to introduce an effective criterion for choosing among violated (optimality or feasibility) Benders cuts, very much in the spirit of disjunctive cut generation that is also based on CGLPs (see Balas, Ceria, and Cornuéjols [7], and also Fischetti, Lodi and Tramontani [27]). As far as we know, no research effort was devoted to this particular topic in the literature, with one notable exception—the acceleration procedure by Magnanti and Wong [54]. This procedure provides a criterion to choose, among equivalent optimal vertices of the dual slave polyhedron, a "Pareto-optimal" one that corresponds to a maximally-violated optimality cut that is not strictly dominated (within the master feasible solution set) by any other maximally-violated cut. The procedure has however some drawbacks:

- According to its original definition, the procedure would require the dual slave to have a bounded optimal value, hence it could not be applied in a completely general context involving feasibility cuts—this drawback can however be partially overcome by introducing artificial dual bounds.

- The user has to provide a point in the relative interior of the master feasible set. This is quite a simple task if the the master has a very special structure, as in the cases addressed by Magnanti and Wong in their study, but is NP-hard in general if the master is a MIP, since we need a point in the relative interior of the convex hull of the integer feasible points, which is usually not known. Moreover, the outcome of the procedure depends on the choice of the interior point.

- The method may be computationally heavy, as it requires to solve two LPs to generate a single cut, the second LP being often quite time-consuming due to the presence of an additional equation that fixes the degree of violation to the cut—this equation is in fact very dense and numerically unstable.

- The Magnanti-Wong criterion benefits from the existence of several equivalent optimal solutions of the dual slave problem (i.e., several maximally-violated optimality cuts), which is however not very frequent when fractional (as opposed to integer) points of the master are cut.

3.4 Benders cuts and Minimal Infeasible Subsystems

The CGLP of a Benders cut can always be seen as a feasibility problem: given a master solution (x^*, η^*), it is possible to generate a violated cut if and only if the following

primal slave problem is infeasible:

$$\begin{aligned} d^T y &\leq \eta^* \\ Qy &\geq r - Tx^* \\ y &\geq 0 \end{aligned} \tag{3.9}$$

or equivalently, by LP duality, if the following dual slave problem is unbounded:

$$\begin{aligned} \max \pi^T (r - Tx^*) - \pi_0 \eta^* \\ \pi^T Q \leq \pi_0 d^T \\ \pi, \pi_0 \geq 0 \end{aligned} \tag{3.10}$$

If the separation is successful, given the dual solution (extreme ray) $(\overline{\pi}, \overline{\pi}_0)$ the generated cut is

$$\overline{\pi}^T (r - Tx) - \overline{\pi}_0 \eta \leq 0$$

In practice, one is interested in detecting a "minimal source of infeasibility" of (3.9), so as to detect a small set of rows that allow to cut the master solution. According to Gleeson and Ryan [34], the rows of any *Minimal* (with respect to set inclusion) *Infeasible Subsystem* (MIS) of (3.9) are indexed by the support of the vertices of the following polyhedron, sometimes called the *alternative polyhedron*:

$$\begin{aligned} \pi^T Q &\leq \pi_0 d^T \\ \pi^T (r - Tx^*) - \pi_0 \eta^* &= 1 \\ \pi, \pi_0 &\geq 0 \end{aligned} \tag{3.11}$$

where the unbounded objective function—namely, the cut violation to be maximized—has been fixed to a normalization positive value (if the alternative polyhedron is empty, we are done). By choosing an appropriate objective function it is therefore possible to optimize over the alternative polyhedron, thus selecting a violated cut corresponding to a MIS of (3.9) with certain useful properties. Therefore, the choice of the objective function is a main issue to be addressed when designing a separation procedure based on a CGLP, as in the Benders method.

A natural objective function whose purpose is to try to minimize the cardinality of the support of the optimal vertex (and hence to find a small-cardinality MIS [1]) is

$$\min \sum_{i=1}^{m} \pi_i + \pi_0 \tag{3.12}$$

As we are only interested in solutions with a positive cut violation, and since $\{(\pi, \pi_0) \geq 0 : \pi^T Q \leq \pi_0 d^T\}$ is a cone, we can swap the role of the objective function (3.12) and of the normalization condition in (3.11), yielding the following equivalent

[1] Finding a minimum-cardinality MIS is an NP-hard problem in general; see, e.g., Amaldi et al. [3]

CGLP akin to the one used for disjunctive cuts by Balas, Ceria, and Cornuéjols [7]:

$$\begin{aligned} \max \pi^T(r - Tx^*) - \pi_0\eta^* \\ \pi^T Q \leq \pi_0 d^T \\ \sum_{i=1}^{m} \pi_i + \pi_0 = 1 \\ \pi, \pi_0 \geq 0 \end{aligned} \tag{3.13}$$

It is worth noting that the feasible solution set of the above CGLP is never empty nor unbounded, so a violated cut can be generated if and only if the CGLP has a strictly positive optimal value. The latter formulation is preferable from a computational point because the normalization constraint $\sum_{i=1}^{m} \pi_i + \pi_0 = 1$, though very dense, is numerically more stable than its "cut violation" counterpart $\pi^T(r - Tx^*) - \pi_0\eta^* = 1$. Moreover, at each iteration only the CGLP objective function is affected by the change in the master solution, hence its re-optimization with the primal simplex method is usually quite fast.

A geometric interpretation of (3.13) is as follows. The CGLP feasible set is now defined as the intersection of the homogenization of the dual polyhedron D with the normalization hyperplane $\sum_{i=1}^{m} \pi_i + \pi_0 = 1$. It is not difficult to see that there is a one-to-one correspondence between the vertices of this feasible set and the extreme rays (if $\pi_0 = 0$) and vertices (if $\pi_0 \neq 0$) of D. Therefore, the reformulation does not actually change the set of Benders cuts that can be generated, but it is nevertheless useful in that it allows for a more clever choice of the violated cut to be separated.

3.5 Computational results

The effectiveness of our CGLP formulation has been tested on a collection of problems from the MIPLIB 2003 library [2]. Among the instances in this testbed, we have chosen the mixed-integer cases with a meaningful number of integer and continuous variables. Moreover, we discarded some instances with numerical instability and which, after the variables were partitioned, were too easy to solve even by the classical Benders method [2]. Table 3.1 shows our final testbed with the main characteristics of each instance.

Standard variable partitioning has been applied—integer (and binary) variables are viewed as master variables x, and the continuous variables are viewed as slave variables y.

We implemented two variants of the classical (textbook) Benders method, as well as two variants of our MIS-based CGLP, namely:

tb: This is the original method as proposed by Benders [12]. If the dual slave problem is bounded, we generate one optimality cut, otherwise we generate both a feasibility and an optimality cut (the optimality cut being added to the master problem only if it is violated by the current master solution).

[2] A couple of instances exhibit a block structure of the slave problem and just a few iterations where enough to terminate the method.

Problem	# variables	# integer	# continuous	# constraints
10teams	2025	1800	225	230
a1c1s1	3648	192	3456	3312
aflow40b	2728	1364	1364	1442
danoint	521	56	465	664
fixnet6	878	378	500	478
modglob	422	98	324	291
momentum1	5174	2349	2825	42680
pp08a	240	64	176	136
timtab1	397	171	226	171
timtab2	675	294	381	294
tr12-30	1080	360	720	750

Table 3.1: Testbed characteristics

tb_noopt: This is a standard Benders implementation method as often seen on textbooks. This method always generate only one cut per iteration—in case of unboundedness, only the feasibility cut associated with the unbounded dual-slave ray detected by the LP solver is added to the master.

mis: This is our basic MIS-based method. It uses the CGLP (3.13) to solve the separation problem, hence it generates only one cut per iteration.

mis2: This is a modified version of *mis*: after having solved the CGLP, if the generated cut is an optimality one, we enforce the generation of an additional feasibility cut by imposing the condition $\pi_0 = 0$.

In our experiments, we handled the equations in the MIP model (if any) explicitly, without replacing them with pairs of inequalities; this implies the presence of free dual multipliers and the use of their absolute value in the normalization condition.

The implementation was done in C++ on a Linux 2.6 platform and all tests were performed on an Intel Core2 Quad CPU Q6600 with 4GB of RAM. We used ILOG Cplex 11.0 as the black-box LP solver; we disabled the LP presolver and forced the use of the primal simplex method for the solution of the dual slaves so as to be able to get a meaningful output even in case of unbounded problems. Before solving an instance, we performed a standard bound shifting in order to reduce the number of slave variable bounds to dualize. For this reason, the optimal LP value reported in our tables may differ from the value reported in the literature.

The quality of the generated Benders cuts is measured in terms of "percentage gap closed" at the root node, as customary in cutting plane methods. The results are shown in Tables 3.2 and 3.3. Results with *tb_noopt* are not reported since this method was never better (and often much worse) than *tb*: a typical behavior is illustrated in Figures 3.1 and 3.2.

		Time				Iterations							
Problem	Method	80%	90%	95%	99%	80%	90%	95%	99%	bestBound	optimum	totTime	totIter
10teams	tb	0.08	0.21	0.22	0.26	24	35	38	43	897.00	897.00	0.51	71
	mis	0.04	0.05	0.06	0.07	9	11	14	19	897.00	897.00	0.21	66
	mis2	0.04	0.05	0.05	0.07	9	11	14	19	897.00	897.00	0.22	66
a1c1s1	tb	-	-	-	-	-	-	-	-	707.61	997.53	1000.45	3714
	mis	3.68	6.01	10.62	19.39	144	218	296	482	997.53	997.53	39.95	914
	mis2	86.58	189.93	280.78	-	62	118	173	-	982.38	997.53	451.98	296
aflow40b	tb	0.03	0.03	0.04	0.09	1	2	5	13	1005.66	1005.66	0.24	44
	mis	0.05	0.05	0.07	0.16	1	1	2	7	1005.66	1005.66	0.89	44
	mis2	0.04	0.04	0.07	0.17	1	1	2	7	1005.66	1005.66	0.90	44
danoint	tb	21.22	24.30	29.16	29.16	595	654	766	766	62.64	62.64	36.15	1251
	mis	0.18	0.18	0.18	1.03	43	43	43	87	62.64	62.64	1.74	186
	mis2	3.00	3.00	3.00	5.42	43	43	43	72	62.64	62.64	12.46	167
fixnet6	tb	0.75	1.19	1.61	2.14	183	254	310	368	1200.88	1200.88	3.20	523
	mis	0.05	0.12	0.16	0.34	39	65	83	139	1200.88	1200.88	0.70	230
	mis2	0.28	0.43	0.64	1.02	26	39	56	87	1200.88	1200.88	1.79	161
modglob*	tb	-	-	-	-	-	-	-	-	-	-	-	-
	mis	0.34	0.34	1.38	2.01	62	62	303	473	20430900.00	20430947.62	50.31	3573
	mis2	0.58	0.87	3.00	6.06	34	61	274	613	20430900.00	20430947.62	44.83	3079
momentum1*	tb	-	-	-	-	-	-	-	-	-	-	-	-
	mis	0.35	0.59	0.73	1.60	0	3	5	18	72793.30	72793.35	26.21	207
	mis2	-	-	-	-	-	-	-	-	-	-	-	-
pp08a	tb	0.01	0.01	0.01	1.03	9	14	16	339	2748.35	2748.35	4.11	825
	mis	0.13	0.28	0.33	0.52	125	195	213	280	2748.35	2748.35	1.71	696
	mis2	0.03	0.04	0.04	0.68	9	13	14	179	2748.35	2748.35	2.20	540
timtab1	tb	60.15	61.70	67.09	77.27	676	705	778	963	28655.10	28694.00	83.70	1046
	mis	1.51	2.33	3.08	5.03	601	831	978	1294	28694.00	28694.00	6.13	1431
	mis2	2.81	3.48	4.13	5.09	362	433	494	575	28694.00	28694.00	5.83	635
timtab2	tb	444.26	517.35	663.03	898.50	1162	1388	1731	2165	83269.00	83592.00	1003.04	2327
	mis	17.96	35.51	52.70	119.58	1091	1493	1812	2965	83592.00	83592.00	204.90	4080
	mis2	14.36	21.33	29.74	46.28	536	682	827	1131	83592.00	83592.00	64.14	1395
tr12-30	tb	1.22	1.95	3.14	19.14	146	188	222	254	14210.43	14210.43	357.80	518
	mis	18.83	36.41	59.49	88.98	547	669	768	860	14210.43	14210.43	123.18	1015
	mis2	0.63	0.66	0.66	12.69	17	18	18	249	14210.43	14210.43	13.42	272

Table 3.2: Comparison of the effectiveness of various separation methods in moving the lower bound at the root node. We report the computing time and number of iterations needed to reach 80%, 90%, 95% and 99% of the optimal root relaxation value, as well as the total running times and number of iterations needed for convergence (within a time limit of 2,000 seconds). Times are given in CPU seconds. (*) indicates failed cut generation due to numerical problems.

Problem	Method	# cuts	# opt.	# feas.	Avg Dens.	Master Rate	Avg T. Sep (s)
10teams	tb	107	36	71	383	1.14E-04	1.94E-04
	mis	66	0	66	53	3.97E-05	1.94E-04
	mis2	66	0	66	53	3.44E-05	2.39E-04
a1c1s1	tb	5893	3714	2179	76	1.65E-04	6.01E-03
	mis	914	906	8	26	2.63E-05	3.05E-02
	mis2	577	296	281	14	4.13E-05	1.52E+00
aflow40b	tb	44	0	44	252	4.74E-06	4.24E-03
	mis	44	0	44	242	-5.93E-06	1.86E-02
	mis2	44	0	44	242	1.04E-05	1.89E-02
danoint	tb	1412	1251	161	48	3.09E-06	2.02E-02
	mis	186	186	0	37	5.25E-06	8.72E-03
	mis2	180	167	13	35	4.99E-06	7.38E-02
fixnet6	tb	806	523	283	46	1.05E-05	1.24E-03
	mis	230	210	20	22	9.38E-06	2.01E-03
	mis2	321	160	161	24	1.60E-05	9.59E-03
modglob*	tb	-	-	-	-	-	-
	mis	3573	3557	16	31	6.74E-06	2.30E-03
	mis2	3088	3077	11	29	5.84E-06	6.25E-03
momentum1*	tb	-	-	-	-	-	-
	mis	414	383	31	143	3.18E-04	2.61E-02
	mis2	-	-	-	-	-	-
pp08a	tb	901	825	76	40	7.97E-06	3.14E-04
	mis	696	688	8	17	3.52E-06	5.91E-04
	mis2	613	540	73	16	3.45E-06	2.46E-03
timtab1	tb	2083	1042	1041	56	2.52E-06	4.37E-04
	mis	1431	1354	77	17	4.31E-06	1.10E-03
	mis2	1268	633	635	10	8.82E-06	4.88E-03
timtab2	tb	4609	2316	2293	103	8.98E-05	5.98E-04
	mis	4080	3918	162	45	1.90E-05	3.29E-03
	mis2	2783	1388	1395	23	3.79E-05	1.31E-02
tr12-30	tb	1026	513	513	144	4.13E-03	7.69E-04
	mis	1015	999	16	44	3.20E-04	4.55E-03
	mis2	544	272	272	19	6.75E-05	4.02E-02

Table 3.3: Statistics on the Benders cuts generated by the different methods. We report the number of generated (optimality and feasibility) cuts, their average density, the rate of growth of the master solution time as a function of the number of iterations (standard linear regression on the master-problem running times vs. iterations), and the average separation time in CPU seconds. (*) indicates failed cut generation due to numerical problems.

As reported in Table 3.2 *tb* is the most efficient method only in 1 out of 11 instances, namely `aflow40`, and only with little advantage over the competitors. On the other hand, *mis* and *mis2* are much more effective on 10 out of 11 instances, with speedups of 1 to 2 orders of magnitude. As expected, the average density of the cuts generated by *mis* and *mis2* is considerably smaller than *tb*, see Table 3.3. This has a positive effect on the rate of growth of the master solution time as a function of the number of iterations, as reported in column *Master Rate* in the table.

A closer analysis of instance `a1c1s1` provides some insights on the strength of the proposed methods: at each iteration, while *tb* generates weak feasibility and optimality cuts, with no selection criteria for both, *mis* is able to cut the current master solution with just a good optimality cut. This is however not always the best strategy: for example, in `timtab1`, `timtab2` and `tr12-30`, feasibility cuts are really crucial for the effectiveness of the method and should be preferred—hence *mis2* becomes the leading method.

A comparison between *mis* and *mis2* shows that *mis* candidates as the method of choice, as it is usually faster due to the extra computing time that *mis2* spends in generating the additional feasibility cut (at least, in our present implementation); see Table 3.3. Nevertheless, as already mentioned, there are instances such that `timtab2` and `tr12-30` where the extra separation effort is rewarded by a significant improvement of the overall performance.

3.6 Conclusions

We have investigated alternative cut selection criteria for Benders cuts. By using the correspondence between minimal infeasible subsystems of an infeasible LP and the vertices of a so-called *alternative polyhedron*, we were able to define a simple yet effective cut-generation LP allowing for the selection of strong Benders cuts. Computational results on a set of MIPLIB instances show that the proposed method allows for a speedup of 1 to 2 orders of magnitude with respect to the textbook one.

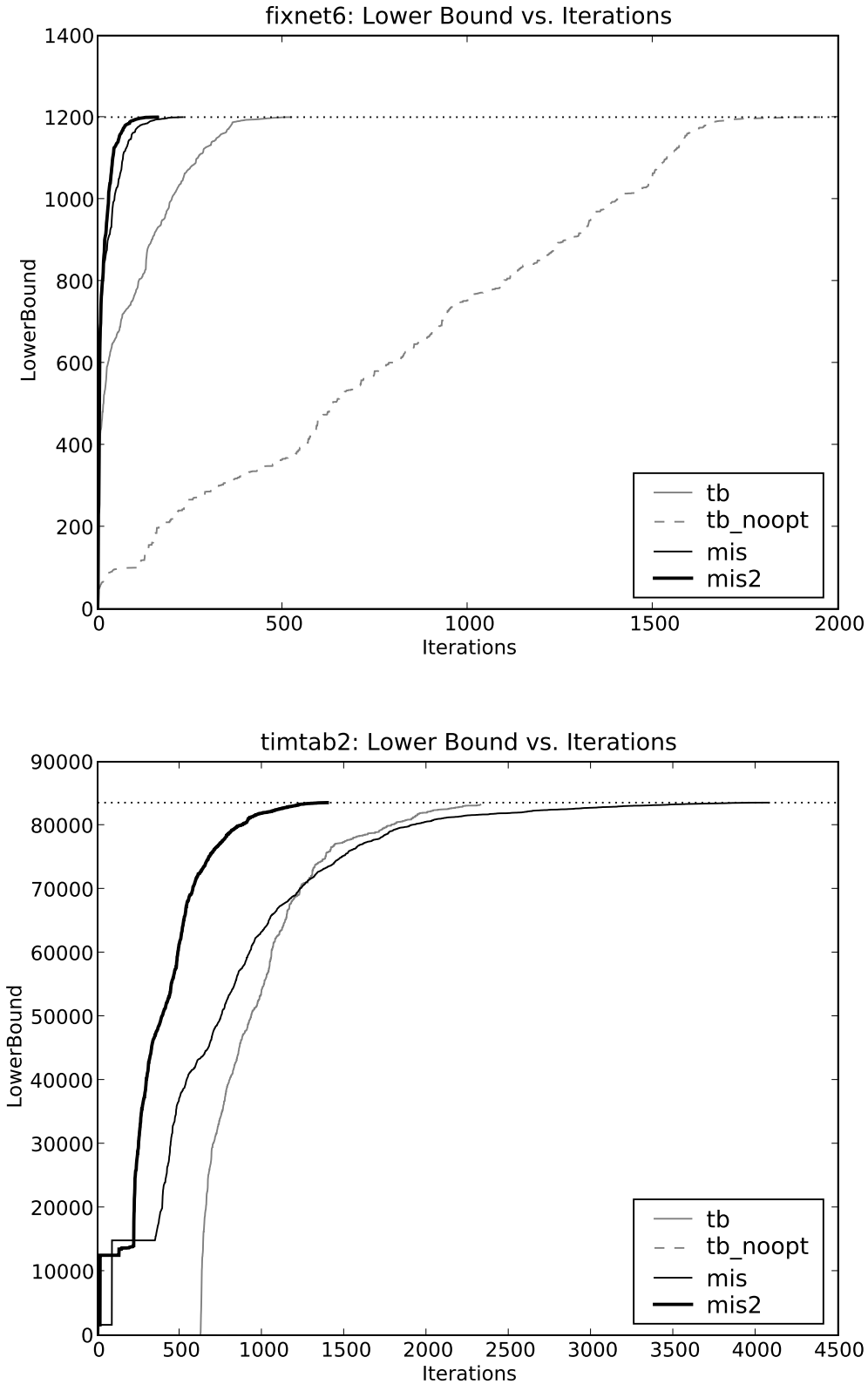

Figure 3.1: Lower bound growth vs. iterations with different separation methods. The dotted line is the known optimal value. For timtab2, tb_noopt *was not able to improve its initial null lower bound.*

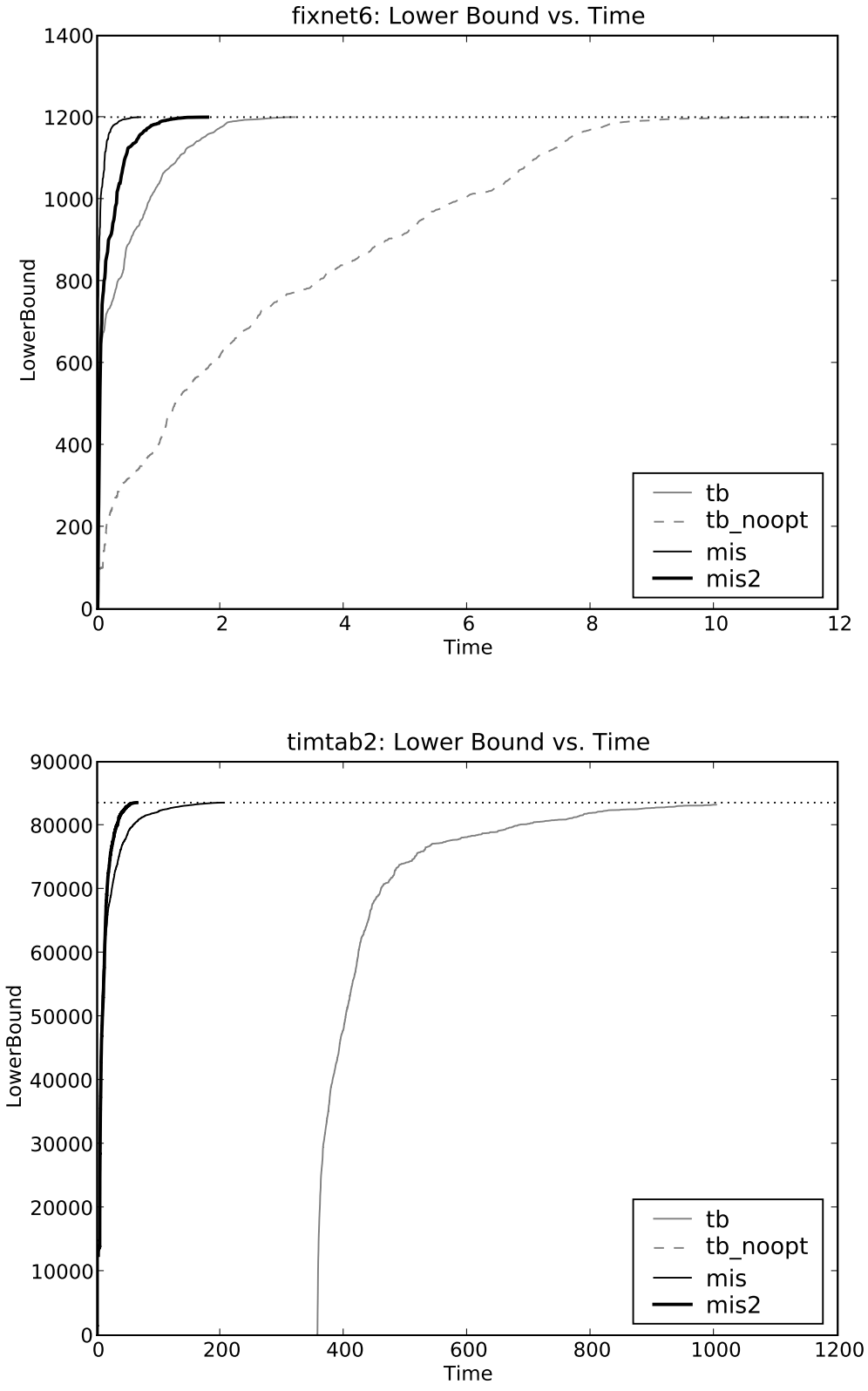

Figure 3.2: Lower bound growth vs. time with different separation methods. The dotted line is the known optimal value. For timtab2, tb_noopt *was not able to improve its initial null lower bound.*

Chapter 4

Fast Approaches to Improve the Robustness of a Railway Timetable

4.1 Introduction

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

In practice, however, the maximization of some objective function is not enough: the solution is also required to be robust against delays/disturbances along the network. Very often, the robustness of optimal solutions of the original problem turns out to be not enough for their practical applicability, whereas easy-to-compute robust solutions tend to be too conservative and thus unnecessarily inefficient. As a result, practitioners call for a fast yet accurate method to find the most robust timetable whose efficiency is only slightly smaller than the theoretical optimal one. This is typically obtained by adding "buffer times" to the schedule according to certain simple rules (see §2.2 in [43])

The purpose of the present chapter is to propose and evaluate new methods to improve the robustness of a given TTP solution. In particular, we address the aperiodic (non cyclic) TTP version described in [17]. Our approach combines Linear Programming (LP) with Stochastic Programming (SP) and Robust Optimization techniques.

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

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

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

stage 3) robustness validation: the robustness of the final solution found in stage

2 is evaluated by using a validation tool, thus allowing for a fair comparison of different training methods.

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

4.2 Literature review

The TTP problem has two main variants: the *periodic* and *aperiodic* versions. The periodic TTP's goal is to design a timetable that is operated cyclically after a (small) period of time; this is a typical requirement for passenger trains in order to come up with an easy-to-remember timetable. The first authors who developed a model for generating periodic timetables were Serafini and Ukovic [68], who introduced a mathematical model called *Periodic Event Scheduling Problem* (PESP). In PESP, a set of repetitive events is scheduled under periodic time-window constraints. Consequently, the events are scheduled for one cycle in such a way that the cycle can be repeated. Most models for periodic TTP are based on PESP. A natural LP formulation of PESP is quite weak, due to kind of big-M constraints (where M is the period). An alternative stronger formulation is treated in Nachtigall [61] and Liebchen and Peeters [46, 63] among others, and is based on cycle bases and separation of valid inequalities.

As to robustness, Kroon et al. [43] describe a stochastic optimization variant of PESP. Their model explicitly takes into account stochastic disturbances of the railway processes, distinguishing between a planned timetable and several realizations of the timetable under pre-determined stochastic disturbances. The model can be used to allocate time supplements and buffer times to the processes in the planned timetable in such a way that the average delay of the realizations of the trains is minimized. In order to keep the computation times within an acceptable bound, they start with an existing timetable and fix the precedences among trains. They show that a substantial increase in robustness can be achieved by taking into account stochastic disturbances in the design of the timetable. For the case of one trip serving 10 stations, Liebchen and Stiller [47] provide a theoretical explanation for the effects observed empirically in Kroon et al. [43]. Very recently, a new notion of robustness, called *recoverable robustness*, has been proposed in [48], which integrates the notion of robustness and recoverability into a common framework. Applications to integrated timetabling/delay management in railway systems are described and evaluated in [48, 49, 20].

The aperiodic TTP is especially relevant on heavy-traffic, long-distance corridors, where the capacity of the infrastructure is limited due to greater traffic densities, and competitive pressure among the train operators is expected to increase in the near future. In the Caprara et al. [17] setting, a train operator applies for allocating its trains on the infrastructure, and specify a profit for the "ideal timetable" they are asking for. Then the infrastructure manager collects all requests from train operators and computes a feasible timetable maximizing the overall profit, i.e., the difference

between the profits of the scheduled trains and a cost-penalty function, which takes into account the deviations of the final timetables with respect to the ideal ones (possibly leaving some trains unscheduled).

Different ILP models based on a graph representation of the problem were presented in [17, 18]. In these papers the problem is modeled by means of a directed acyclic multi-graph, in which nodes correspond to arrival and departure events from the stations and arise at some discretized time instants, and arcs correspond to train stops within a station or to train trips. A Lagrangian relaxation method is used to derive bounds on the optimal solution value as well as to drive a heuristic procedure.

4.3 The Nominal Model

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

Following [17], the aperiodic TTP can be described as follows: given a railway network, described as a set of stations connected by tracks, and an ideal train timetable, find an actual train schedule satisfying all the operational constraints and having a minimum distance from the ideal timetable.

The entities involved in modelling the problem are the following:

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

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

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

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

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

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

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

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

where

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

and

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

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

Operational constraints include:

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

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

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

As already mentioned, in the present chapter we do not address the solution of the nominal TTP problem explicitly, in that a nominal solution is assumed to be provided in input. Nevertheless, we next outline the structure of a MIP formulation for the nominal TTP problem, since a relaxed version of it is at the basis of the LP models used in Sections 4.5 and 4.6.

Although in the nominal problem one is allowed to leave some trains unscheduled, to simplify our presentation we consider the situation where one is required to schedule all the trains. A natural event-based model in the spirit of the Periodic Event Scheduling Problem (PESP) formulation used in the periodic (cyclic) case [68] can be sketched as follows:

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

$$t_{i+1}^h - t_i^h \geq d_{i,i+1}^h \quad \forall h \in T, i = 1, \ldots, len(h) - 1 \tag{4.1}$$

$$|t_i^h - t_j^k| \geq \Delta_a \quad \forall t_i^h, t_j^k \in A_S, \forall S \in \mathcal{S} \tag{4.2}$$

$$|t_i^h - t_j^k| \geq \Delta_d \quad \forall t_i^h, t_j^k \in D_S, \forall S \in \mathcal{S} \tag{4.3}$$

$$t_{i+1}^h < t_{j+1}^k \Leftrightarrow t_i^h < t_j^k \quad \forall t_i^h, t_j^k \in D_S, \forall S \in \mathcal{S} \tag{4.4}$$

$$\rho_h = \pi_h - \sigma_h|t_1^h - \bar{t}_1^h| - \theta_h((t_{len(h)}^h - t_1^h) - (\bar{t}_{len(h)}^h - \bar{t}_1^h)) \quad \forall h \in T \tag{4.5}$$

$$l \leq t \leq u \quad \forall t \in E \tag{4.6}$$

Constraints (4.1) impose a minimum time difference $d_{i,i+1}$ between two consecutive events of the same train, thus imposing minimum trip duration (trains are supposed to travel always at the maximum allowed speed for the track) and minimum stop time at the stations.

Constraints (4.2)-(4.3) model the headway times between two consecutive arrival or departure events in the same station (Δ_d and Δ_a being the minimum departure and arrival headway, respectively). Since these constraints are nonlinear and we do not know in advance the order in which events occur at the stations, in our MIP model we introduce a set of binary variables $x_{i,j}^{h,k}$ to be set to 1 iff $t_i^h \leq t_j^k$ along with a big-M coefficient M, so that conditions

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

can be translated to

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

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

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

Given the linearization of constraints (4.2)-(4.3), it is easy to translate the track capacity constraints (4.4) as

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

Constraints (4.5) define the profits of the trains, whereas constraints (4.6) model the user-defined time windows of each event.

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

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

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

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

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

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

4.4 The Stochastic Programming Paradigm

Having stated the problem as a MIP, a well known tool to find robust solutions is the (two-stage) *Stochastic Programming* approach; see [14],[66] for an introduction to the SP methodology. In SP, the set of constraints is decomposed in *structural* constraints, which represent the deterministic part of the model, and *control* constraints which have a stochastic nature and whose coefficients depend on the particular scenario. Roughly speaking, the approach allows one to take decisions in the first stage by ignoring the stochastic part of the model, but enforces some costly *recourse* action when indeterminacy will eventually occur. Thus a natural optimization objective for this two-stage strategy is to minimize the total expected cost:

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

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

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

$$w^* = \min_{x \in X} c^T x + \sum_{k=1}^{|\Omega|} p_k q_k^T r_k, \quad r_k \in Y_k, \forall k = 1 \ldots |\Omega| \tag{4.7}$$

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

As $|\Omega|$ is often very large, various sampling-based approaches have been proposed to estimate the second-stage objective function. *Interior sampling* methods use samples during the algorithm execution to estimate, from time to time, algorithmic parameters such as function values, gradients, optimality cuts. *Exterior sampling* methods, instead, use the *Sample Average Approximation* (SAA) algorithm to estimate the optimal objective. We have chosen exterior sampling, since it has some advantages over interior sampling [69]: ease of numerical implementation, good theoretical convergence prop-

erties [72], well developed statistical inference (validation and error analysis, stopping rules). Furthermore, it is easily amenable to variance reduction techniques, ideal for parallel computations.

4.4.1 The Sample Average Approximation Method

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

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

and it satisfies the following central limit theorem:

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

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

The SAA approximation of (4.7) reads:

$$w_N^* = \min_{x \in X} c^T x + \frac{1}{N}\sum_{k=1}^{N} q_k^T r_k, \quad r_k \in Y_k, \forall k = 1 \ldots N \tag{4.8}$$

Under mild assumptions it was proved that the optimal value of SAA problem (4.8) converges with probability 1 to w^* (the optimal value of the stochastic problem) as N tends to infinity (see [72]).

Also, it is possible to use SAA to estimate the optimality gap by deriving lower and upper bounds on w^*. These bounds will be used to quantify the confidence intervals of the optimal solution of the stochastic model (see Section 4.7, Figure 4.9). Indeed, an *upper bound* on w^* is

$$c^T \hat{x} + \mathbb{E}[Q(\hat{x}, \xi(\omega))] = c^T \hat{x} + \mathbb{E}[\bar{Q}_N(\hat{x}, \xi(\omega_i))] \tag{4.9}$$

where $\hat{x}$ is a given feasible, yet suboptimal, first-stage decision vector. The expectation in the right hand side of (4.9), by its own, can be estimated as the mean of $N' \ll N$ (say) independent SSA $\bar{Q}_N^j(\hat{x}, \xi(\omega_i^j))$, obtaining:

$$\overline{UB} = \frac{1}{N'}\sum_{j=1}^{N'} \bar{Q}_N^j(\hat{x}, \xi(\omega_i^j)) \tag{4.10}$$

$$\sigma_u^2 = \text{var } \bar{Q}_N(\hat{x}, \xi(\omega_i)) = \frac{1}{(N'-1)N'}\sum_{j=1}^{N'} (\bar{Q}_N^j(\hat{x}, \xi(\omega_i^j)) - \bar{UB}) \tag{4.11}$$

It is easy to show (see [50]) that a lower bound on w^* is given by $\mathbb{E}[w_N^*]$. Again, we can compute this expectation by sampling:

$$\overline{LB} = \frac{1}{N'} \sum_{j=1}^{N'} w_N^{*j} \tag{4.12}$$

$$\sigma_l^2 = \text{var } \bar{w}_N^* = \frac{1}{(N'-1)N'} \sum_{j=1}^{N'} (w_N^{*j} - \overline{LB}) \tag{4.13}$$

4.4.2 Sampling

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

$$\delta^h_{(i,j)}([l,k]) = \delta^h P_{[l,k]} \frac{P_{(i,j)}}{\sum_{(i,j)\in\mathcal{L}} P_{(i,j)}} \tag{4.14}$$

where we normalize sections delay probabilities in order to distribute the cumulative delay $\delta^T P_{[l,k]}$ incurred by train T operating on line $\mathcal{L}$ through each line section (i, j). Note that $P_{(i,j)}$ and $P_{[l,k]}$ could be deterministic numbers between 0 and 1, but typically they are treated as random variables.

When using random sampling, the outcome can be affected by a large variance, making it difficult to interpret. So we decided to use *LatinHypercube* (LH) variance reduction technique when sampling from each distribution of $P_{(i,j)}$, $P_{[l,k]}$ and δ^h. Other techniques such as, e.g., Importance Sampling [21] can in principle fit our simulation setting as well, but they are quite involved. On the contrary, LH sampling is of general applicability and comes with a straightforward implementation. The standard approach to get a sample from a particular probability distribution is to apply the inverse of the desired Cumulative Distribution Function (CDF) to a sample drawn from a uniform distribution. The process is then repeated until the required number of samples N is collected. Using LH sampling, instead, we first subdivide the $[0, 1]$ interval in N equal subintervals, and from each of them we draw a sample from a uniform distribution spanning the subinterval. Then the obtained sample vector is inverted through the CDF and randomly permuted. For more theoretical insights on LH sampling, the interested reader is referred to [51].

LH sampling proved to be quite effective in our application. Figure 4.1 shows the reduction in variance σ when sampling from an exponential distribution with or without LH sampling. In our computational testing, we observed an even larger variance

reduction (one order of magnitude or more).

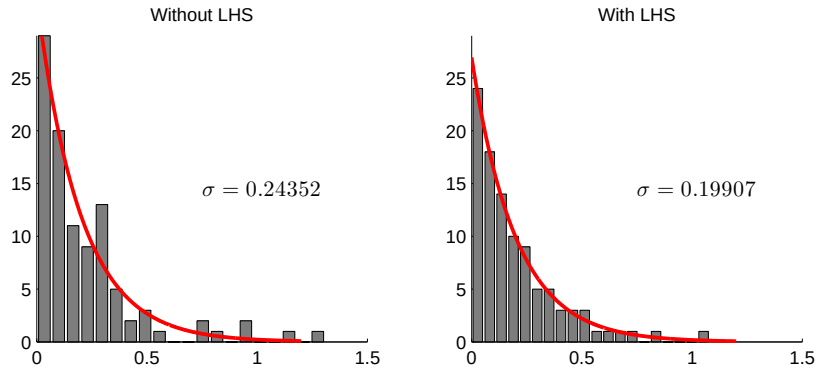

Figure 4.1: Reduction of variance σ with LH when approximating, through sampling, the exponential probability distribution function (solid line).

4.5 Validation Model

An important component in our framework is robustness validation. Validation is often carried out inside the model itself, as is the case when a SP approach is used. However, we decided to implement an external simulation-based validation module that is independent from the optimization model itself, so that it can be of general applicability and allows one to compare solutions coming from different methods. The module is required to simulate the reaction of the railway system to the occurrence of delays, by introducing small adjustments to the planned timetable received as an input parameter. Validation is a major topic on its own. Indeed, the set of actions the railway system can make to react to disruptions is quite large—see for example [39]—and the decision making process is often complicated by strict real-time requirements and complex business rules. Validation can be carried out by optimization methods, as proposed in [48, 49, 20]. However, the complexity of such models grows rapidly as soon as we allow complex decisions to be made. Thus, simulation methods are often used to measure empirically the robustness of a given timetable—see, for example, [10]. For the purpose of the present chapter, we decided to implement a simple LP-based validation tool based on the following simplified assumptions.

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

- Speed of validation: The validation tool should be able to analyze quickly the

behavior of the timetable under many different scenarios.

Given these guidelines, we designed a validation model which analyzes a single delay scenario ω at a time. As all precedences are fixed according to the input solution to be evaluated, constraints (4.1-4.3) 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 (4.1) can be written. The problem of adjusting the given timetable t under a certain delay scenario ω can thus be rephrased as the following simple linear programming model with decision variables t^ω:

$$\min \sum_{j \in E} w_j \left(t_j^\omega - t_j\right) \tag{4.15}$$

$$t_i^\omega - t_j^\omega \geq d_{i,j} + \delta_{i,j}^\omega \quad \forall (i,j) \in \mathcal{P} \tag{4.16}$$

$$t_i^\omega \geq t_i \qquad \forall i \in E \tag{4.17}$$

Constraints (4.16) correspond to the linear inequalities just explained, in which the nominal right-hand-side value $d_{i,j}$ is updated by adding the (possibly zero) extra-time $\delta_{i,j}^\omega$ from the current scenario ω. Weights w_j appearing in objective function (4.15) are related to the relative importance of the events, and typically depend on the number of passengers affected.

Constraints (4.17) are non-anticipatory constraints stating the obvious condition that one is not allowed to anticipate any event with respect to its published value in the timetable. Since these values are known, these constraints act as simple lower bounds on the decision variables. Instead, we impose no upper bounds since we allow for an unlimited stretch of the timetable to recover from delays, i.e., a feasible timetable is always achievable.

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

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

4.6 Finding Robust Solutions

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

4.6.1 A Fat Stochastic Model

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

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

$$\sum_{h \in T} \rho_h \geq (1-\alpha) z^* \tag{4.18}$$

$$t_i^\omega - t_j^\omega \geq d_{i,j} + \delta_{i,j}^\omega \quad \forall (i,j) \in \mathcal{P}, \forall \omega \in \Omega \tag{4.19}$$

$$t_i^\omega \geq t_i \quad \forall i \in E, \forall \omega \in \Omega \tag{4.20}$$

$$t_i - t_j \geq d_{i,j} \quad \forall (i,j) \in \mathcal{P} \tag{4.21}$$

$$l \leq t \leq u \tag{4.22}$$

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

The objective is to minimize the cumulative delay over all events and scenarios. The original objective function $\sum \rho_h$ is taken into account through constraint (4.18), where $\alpha \geq 0$ is a tradeoff parameter and z^* is the objective value of the reference solution.

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

4.6.2 A Slim Stochastic Model

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

$$\min \sum_{(i,j) \in \mathcal{P}, \omega \in \Omega} w_{i,j} s_{i,j}^\omega \tag{4.23}$$

$$\sum_{h \in T} \rho_h \geq (1-\alpha) z^* \tag{4.24}$$

$$t_i - t_j + s^{\omega}_{i,j} \geq d_{i,j} + \delta^{\omega}_{i,j} \quad \forall (i,j) \in \mathcal{P}, \forall \omega \in \Omega \tag{4.25}$$

$$s^{\omega}_{i,j} \geq 0 \quad \forall (i,j) \in \mathcal{P}, \forall \omega \in \Omega \tag{4.26}$$

$$t_i - t_j \geq d_{i,j} \qquad \forall (i,j) \in \mathcal{P} \tag{4.27}$$

$$l \leq t \leq u \tag{4.28}$$

In this model we have just one copy of the original variables, plus the recourse variables $s^{\omega}_{i,j}$ which account for the unabsorbed extra times $\delta^{\omega}_{i,j}$ with respect to the minimum train trip times. It is worth noting that the above "slim" model is inherently smaller than the fat one. Moreover, one can drop all the constraints of type (4.25) with $\delta^{\omega}_{i,j} = 0$, a situation that occurs very frequently in practice since most extra-times in a given scenario are zero.

As to objective function, it involves a weighted sum of the recourse variables. Finding meaningful values for weights $w_{i,j}$ turns out to be very important. Indeed, we will show in Section 4.7 how to define these weights so as to produce solutions whose robustness is comparable with that obtainable by solving the (much more time consuming) fat model.

4.6.3 Light Robustness

A different way to produce robust solutions is to use the *Light Robustness* (LR) approach proposed recently by Fischetti and Monaci [28]. This method is based on the consideration that, roughly speaking, robustness is about putting enough slack on the constraints of the problem. In its simpler version, the LR counterpart of the LP model

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

reads

$$\min f(\gamma) \tag{4.29}$$

$$Ax - \gamma \leq b - \beta \tag{4.30}$$

$$c^T x \leq (1+\alpha) z^\star \tag{4.31}$$

$$x \geq 0 \tag{4.32}$$

$$0 \leq \gamma \leq \beta \tag{4.33}$$

where β_i is a parameter giving the desired *protection level* (or slack) on constraint i, and $\gamma_i \geq 0$ is a decision variable giving the corresponding *unsatisfied slack*. The objective is to minimize a given function f of the γ variables (typically, a linear or quadratic expression). Moreover, (4.31) gives a bound (controlled by α) on the efficiency loss due to the increased robustness of the solution , where $z^\star$ is the value of the input nominal solution.

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

Table 4.1: Nominal solution characteristics

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.

4.7 Computational Results

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

An almost-optimal heuristic solution for each of these instances was computed by using the algorithm described in [17]. The algorithm is a Lagrangian heuristic based on the computation of paths on a time-expanded network, whose computing time was in the order of minutes on a Pentium IV, 2.4 GHz PC. The corresponding solutions were used as the input nominal solutions to freeze the event precedences and to select the trains to schedule. Solution characteristics are given in Table 4.1.

We implemented our framework in C++ and carried out our tests on a AMD Athlon64 X2 4200+ computer with 4GB of RAM . ILOG CPLEX 10.1 [40] was used as MIP solver.

According to the sampling model described in Section 4.4.2, we generated an extra time $\delta^h(\omega)$ corresponding to each train h and to each scenario ω, drawing them from an exponential distribution with mean $\mu = 5\%$. In lack of more detailed data from the Italian railways operator about the actual distribution of delays in line sections, we assume a proportional distribution of delays along line segments. Accordingly, probabilities $P_{(i,j)}$ in (4.14) are proportional to the length of train segments, barring a small additive white Gaussian noise (standard deviation $\sigma = 0.01$, i.e., a random adjustment of 1-2%), and probabilities $P_{[l,k]}$ are deterministically set to 1.

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

methods:

- *fat*: fat stochastic model (50 scenarios only)
- *slim1*: slim stochastic model with uniform objective function–all weights equal (400 scenarios)
- *slim2*: slim stochastic model with enhanced objective function (400 scenarios), where events arising earlier in each train sequence receive a larger weight in the objective function. More specifically, if the i-th event of train h is followed by k events, its weight in(4.23) is set to $k+1$. The idea behind this weighing policy is thatunabsorbed disturbances $s^{\omega}_{i,j}$ in a train sequence are likely to propagate to the next ones, so the first ones in the line are the most important to minimize.
- *LR*: Light Robustness model with objective function as in *slim2* (using the *slim1* objective function produces significantly worse results).Protection level parameters are set to $\Delta = -\mu \ln \frac{1}{2}$, where μ is the mean of the exponential distribution. This is the protection level required to absorb a delay drawn from such a distribution with probability $\frac{1}{2}$. For example, setting a buffer of 1 minute we can absorb half of the times an exponentially distributed disturbance of mean 1.44 minutes.

As to the validation model, weights w_j appearing in objective function (4.15) are assumed to be equal to 1, i.e., all events are considered equally important.

The results are shown in Table 4.2, while graphical representations are given in Figures 4.2 and 4.3.

According to the figures, *slim2* always yields a very tight approximation of *fat*, while *slim1* is often poorer. As to *LR*, it usually produces good results that are only slightly worse than *slim2*, mainly in the most-realistic cases where the tradeoff parameter α is small. As to computing times (reported in Table 4.2), the *fat* model is one order of magnitude slower than *slim1* and *slim2*, although it uses only 50 scenarios instead of 400. *LR* is much faster than any other method—more than two orders of magnitude w.r.t the fat stochastic models. Therefore, *LR* qualifies as the method of choice for addressing large-scale real cases, as it guarantees good levels of robustness and requires very short computing times.

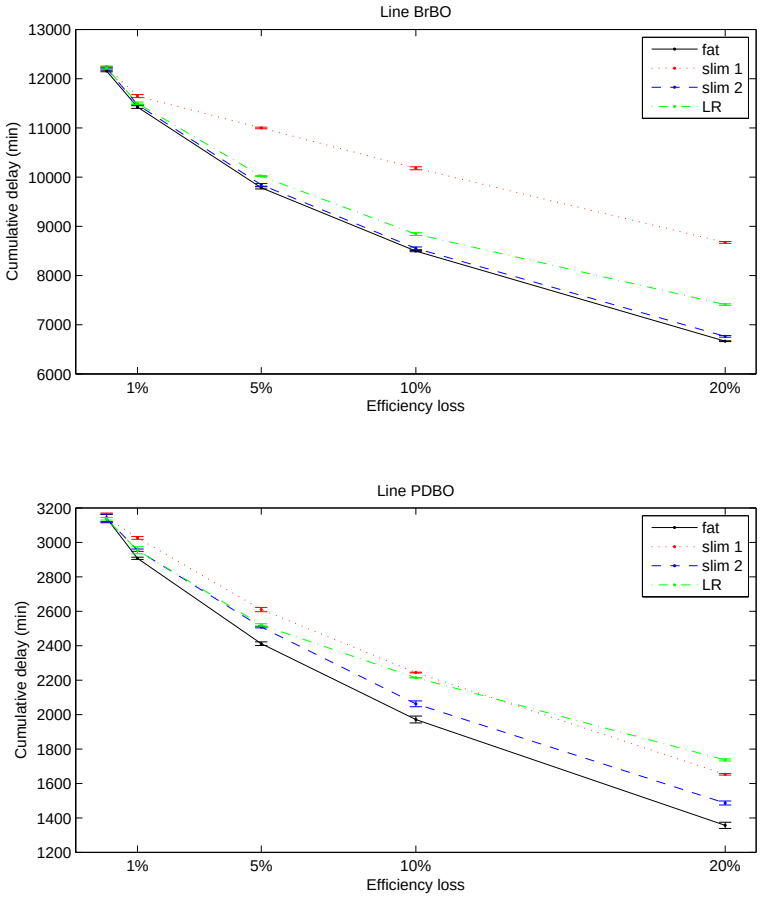

Figure 4.2: Comparison of different training models applied to the best reference solution for each instance. On the x-axis there is the efficiency loss (α) while the y-axis reproduces the confidence intervals of the validation figure (run with 500 scenarios).

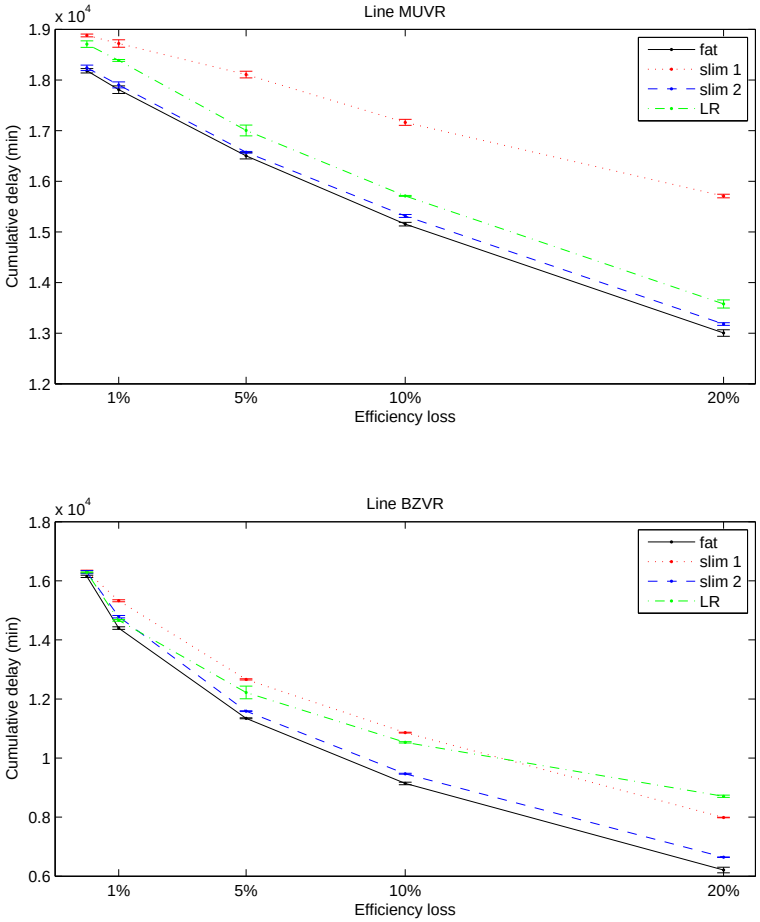

Figure 4.3: Comparison of different training models applied to the best reference solution for each instance. On the x-axis there is the efficiency loss (α) while the y-axis reproduces the confidence intervals of the validation figure (run with 500 scenarios).

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

Table 4.2: Comparison of different training methods with respect to computing time, percentage WAD and validation function (cumulative delay in minutes), for different lines and tradeoff α.

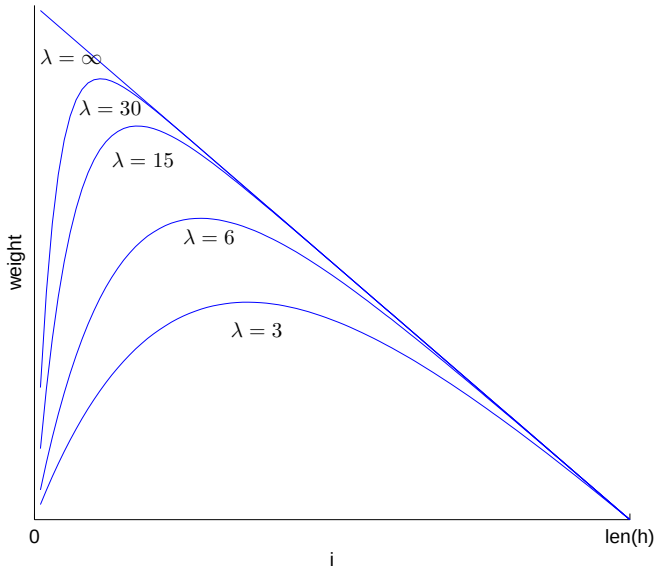

Figure 4.4: Alternative weighing functions for slim2 *and* LR*, giving weight w_{ij} as a function of position i in the line.*

We also tried a variation of the *slim2* (and *LR*) objective function. The variation is motivated by observations in [43] about the optimal distribution of buffers on a single corridor. There, it was observed that buffers that are placed too early risk to be left unused, because the probability to face any delay at this early position is too small. As a consequence, it might be worthwhile to lower the weights $w_{i,j}$ arising in the early sections of the line. Figure 4.4 plots different parametric variants of the *slim2* objective function. All of the them obey a common formula, namely:

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

parametrized in λ ($\lambda = \infty$ gives the original *slim2* weighing scheme). Table 4.3 reports the percentage improvements with respect to case $\lambda = \infty$ for *slim2* and *LR*, respectively. It turns out that the new objective function typically produces slightly worse results for *LR*, while *slim2* takes advantage of it for large values of λ. In any case, the improvement is not substantial (up to 3-4%).

One might also wonder what is the effect of the input nominal solution to the subsequent robustness improving phase. To answer this question, we performed the following experiment. We took the scheduled trains in the heuristic nominal solutions of [17] used in the previous experiments, and we constructed the MIP model described in Section 4.3, where the choice of precedences is left open. Then we collected a series of heuristic nominal solutions of increasing efficiency for that model. This was obtained by running the MIP solver with a 5-minute time limit and by storing all the incumbent solutions produced during the run. Moreover, we ran the solver with a 1-hour time limit so as to produce an almost optimal solution of value, say, z_{ref}. (For all instances,

Slim2

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

LR

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

Table 4.3: Percentage robustness improvement with respect to $\lambda = \infty$ for the different weighing functions plotted in Figure 4.4; a negative value corresponds to a worse robustness.

the optimality gap after 1 hour was less than 4%.) Then, we compared the robustness achieved by our *fat* model when starting from these solutions, by allowing for a relative efficiency loss α with respect to z_{ref}. The left-hand side part of Table 4.4 gives the outcome of the experiment, for two instances (BrBO and MUVR). Columns correspond to the 10 best solutions obtained within the 5-minute time limit, sorted from left to right by increasing efficiency. E.g., for instance BrBO, the 10 solutions have a loss of efficiency ranging from 5.5% to 0.4% with respect to z_{ref}. Rows correspond to the different thresholds α used (1%, 5%, 10%, and 20%). The table entries then give the percentage increase in robustness (as measured by the validation tool) with respect to robustness measured when starting from the almost optimal solution of value z_{ref}. E.g., for BrBO, if we allow for a 10% efficiency loss with respect to z_{ref} and start from a nominal solution which is already 4.5% worse, we lose 13.9% in terms of robustness achievable through the *fat* training method. Missing entries correspond to infeasible cases.

As expected, starting from a worse nominal solution reduces the degree of freedom in the subsequent training phase, leading to a robustness loss. This negative effect could in principle be counterbalanced by the different precedence structure of the solution, in the sense that a less-efficient solution could involve precedence patterns leading to improved robustness. However, our experiments seem to indicate that the precedence structure of the solutions plays only a secondary role. This support the viability of our approach, where only the most-efficient nominal solution available is “trained" for robustness.

To better quantify the effect of fixing all precedences when improving robustness of the nominal solution, we performed a second experiment consisting of solving the MIP version of the *LR* model where all precedences are left unfixed. Note that this is only viable for *LR*, since the other models are too large to be attacked by a general-purpose

MIP solver. As in our previous experiments, we considered a loss of efficiency α ranging from 1 to 20% with respect to the almost-optimal solution value z_{ref}. The solution of value z_{ref} was also used to provide a first incumbent to the MIP solver. In these runs the MIP solver performed quite well, in that the optimality gap after 1 hour of computing time was less than 1% for all instances. (Note however that the model does not take into account the possibility of leaving some trains unscheduled.)

The results of our second experiment are reported in the right-hand side part of Table 4.4. For PDBO and BZVR, the MIP model did not find any better solution than the incumbent, so these cases are not reported in the table. The last column of Table 4.4 reports the percentage robustness improvement of the MIP *LR* model described above, over the linear *LR* model described in Section 4.6.3. E.g., for case BrBO with a threshold α of 10% with respect to z_{ref}, the MIP version of *LR* is able to improve by only 4.3% over the simple linear *LR*. Furthermore, the second-last column of Table 4.4 reports, for comparison sake, the percentage difference between the solution robustness obtained by the MIP *LR* and the robustness obtained by using *fat* on the same almost optimal solution z_{ref}. Results show that the new scheme produces only marginal robustness improvements with respect to the simple linear *LR*. This confirm that, for the cases in our testbed, the precedence structure of the solutions is not really important, efficiency being the key figure in determining the maximum achievable robustness. However, this may be no longer the case for more complex network topologies.

BrBO

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

MUVR

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

Table 4.4: Effects of nominal input solution on robustness.

A simple yet often used in practice policy to enforce robustness in a timetable is to allocate a buffer that is just proportional to the train duration. Figure 4.5 gives the results of this simple policy on a sample instance, where we first compute the maximum total amount of buffer we can allocate for a given efficiency loss, and then distribute it proportionally along the line. According to the figure, the proportional buffer allocation policy and *slim1* behave quite similarly. This is not surprising, since model *slim1*

actually favors a proportional buffer allocation—this is confirmed in the other instances as well (not shown in the figure). On the other hand, much better results are obtained by applying more clever optimization methods, showing the practical relevance of the optimization approaches.

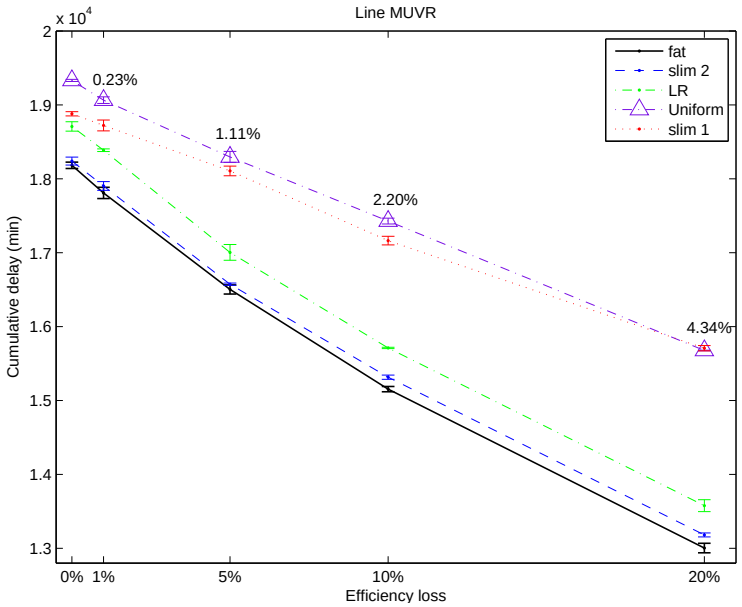

Figure 4.5: Comparison of a simple "proportional" buffer allocation strategy against the proposed methods. The percentages shown are the total amount of buffer it was possible to allocate within a given tradeoff.

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

$$WAD_h = \frac{1}{\sum_{i=1}^{len(h)-1} s_{i,i+1}} \sum_{i=1}^{len(h)-1} \frac{s_{i,i+1}(t^h_{i+1} + t^h_i)/2}{t^h_{len(h)} - t^h_1} \tag{4.34}$$

where $s_{i,i+1}$ is the amount of buffer allocated from t_i to t_{i+1}. The WAD is a number between 0 and 1 which measures how the buffers are distributed along the train trip. For example, a value of 0.5 means that the same amount of buffers were allocated in the first half and in the second half of the trip; values smaller or bigger than 0.5 relate to a shift in buffers distribution towards the begin or the end of the trip, respectively. The WAD of an entire line is calculated as the mean of all the WADs of the trains of the line. The reader is referred to [43] for a more detailed discussion.

A comparison of the various WADs is reported in Table 4.2 and illustrated in Figures 4.6 and 4.7. It can be seen that there is a significative correlation between the

degree of approximation of the various WADs with respect to "perfect WAD" (WAD_{fat}) and the robustness of the solution–as computed by the validation tool and reported in Figure 4.2 and 4.3. In Figures 4.6 and 4.7, *slim1* WAD is almost always 50%, meaning a uniform allocation of buffers. On the other hand, the other methods tend to allocate buffers earlier in the line, resulting in a lower value of the WAD. Moreover, as the allowed efficiency loss increases (x axis), the WAD increases as well, meaning that uniform allocation becomes a good choice. We can also note that *LR* behaves better for small efficiency losses. Indeed, *LR* uses a fixed buffer β to deal with disturbances. When the problem is less constrained in efficiency, these buffers can become too small, and the LP solver will start to distribute the buffer excess, in a somehow unpredictable way, so as to meet the increased degree of freedom, thus degrading the performance of the method. E.g., this is the case of lines BZVR and PDBO. Moreover, BZVR and PDBO are more congested than other two instances, which also explains the better performance of the uniform allocation strategy.

Figure 4.8 reports how the buffers are distributed along the line. The figure is obtained by normalizing each line by the length of the corridor, and averaging the buffers allocated in each normalized line section. The averages are then normalized by the total amount of allocated buffer, so that the area of each chart approximately sums up to 1. E.g., *slim1* allocates buffers almost uniformly along the line—the particular structure of the timetable being responsible of local fluctuations. It is clear that *slim2* produces a very tight approximation of *fat*, while *slim1* does not. It is worth noting that *LR* uses a smoother allocation of buffers, while *slim1* yields a better approximation of their oscillations, but misses the global allocation policy. In this respect, *slim2* performs quite well instead. This is due to the fact that *LR* does not exploit directly the scenario information, thus it has to cope with very little information. Again, note that the poorest method (*slim1*) produces an almost uniform distribution of the buffers, whereas the best ones tend to allocate them earlier. This confirms the findings reported in [43].

Finally, given the intrinsic approximation of the stochastic methods due to the evaluation of the expectation, we have computed lower and upper bounds on the optimal solutions of the stochastic models, as described in Section 4.4. A typical plot obtained for the slim stochastic model is reported in Figure 4.9, showing very narrow estimation gaps. Similar results are obtained with the other models, except *fat* that behaves a little worse due the reduced number of scenarios.

4.8 Conclusions and future work

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

We examined different robustness improving models. The best performing, in terms of validated cumulative delay, is a "fat" stochastic reformulation of the nominal TTP problem. However, the solution of this model turned out to be very hard (if not impos-

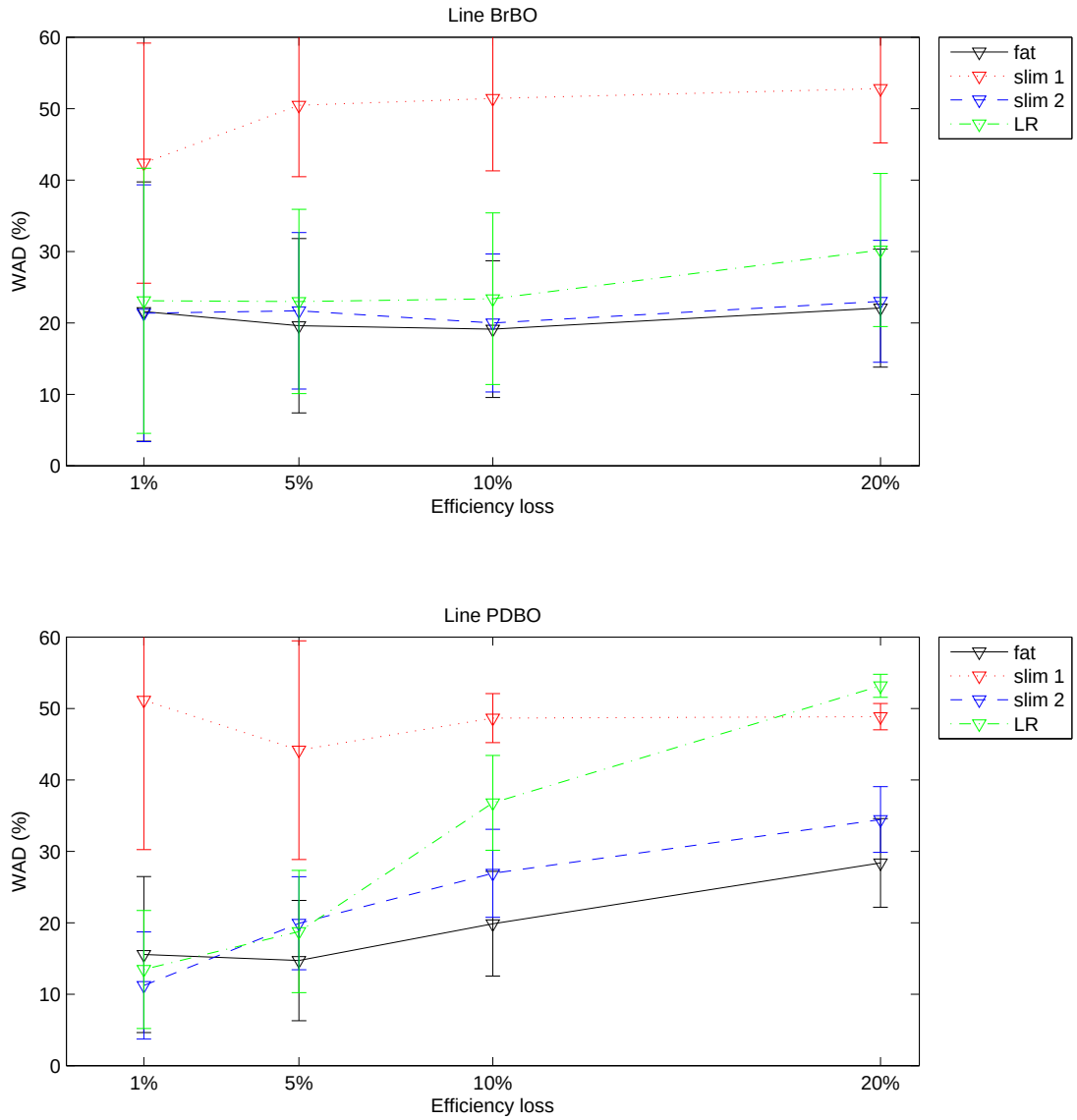

Figure 4.6: Comparison of different training models from the WAD point of view (WAD is given within its confidence intervals).

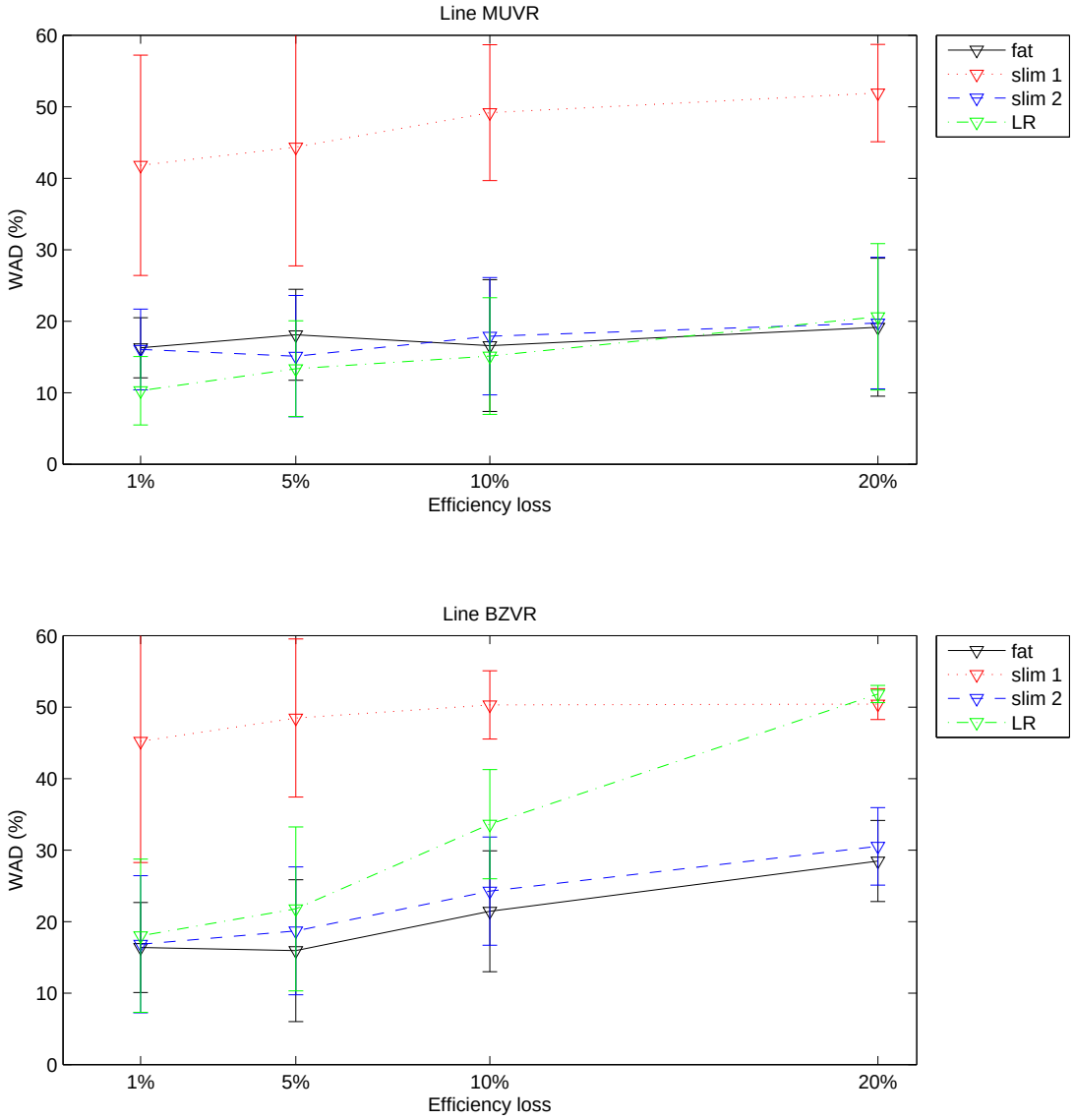

Figure 4.7: Comparison of different training models from the WAD point of view (WAD is given within its confidence intervals).

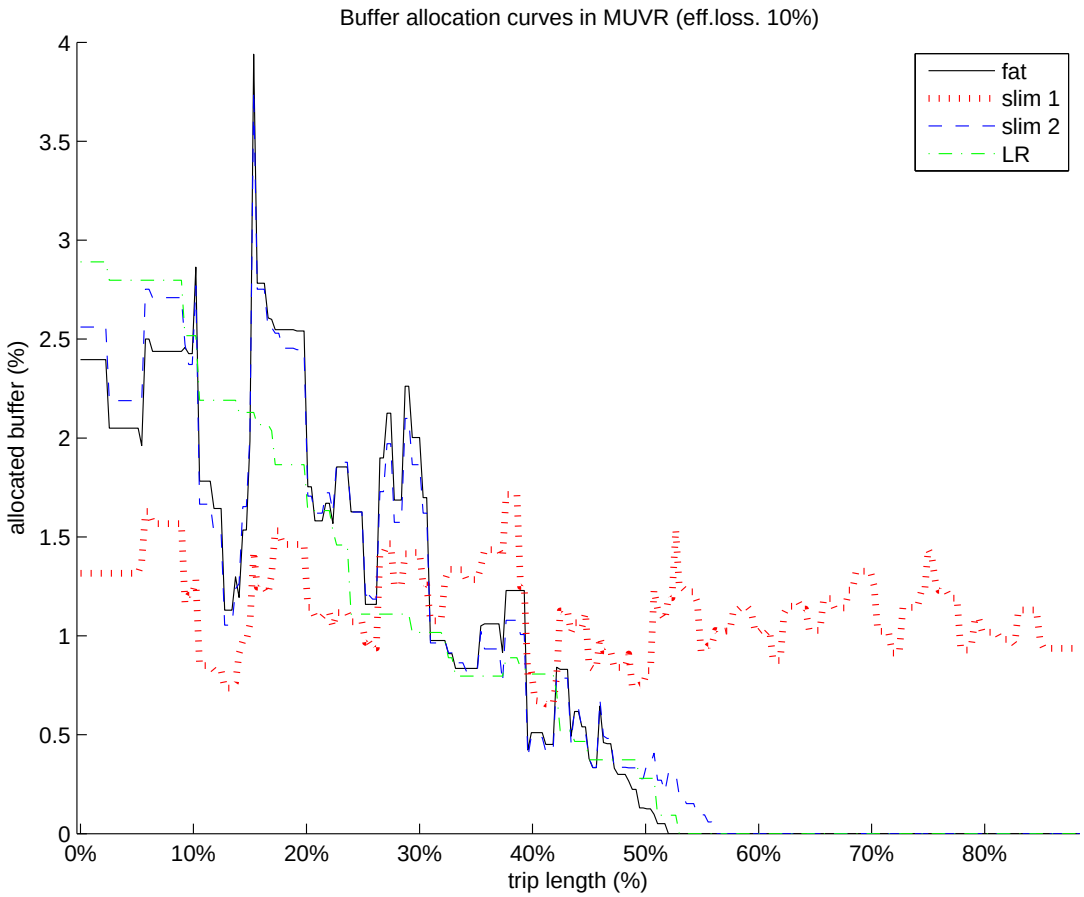

Figure 4.8: Comparison of different training models from the allocated-buffer point of view.

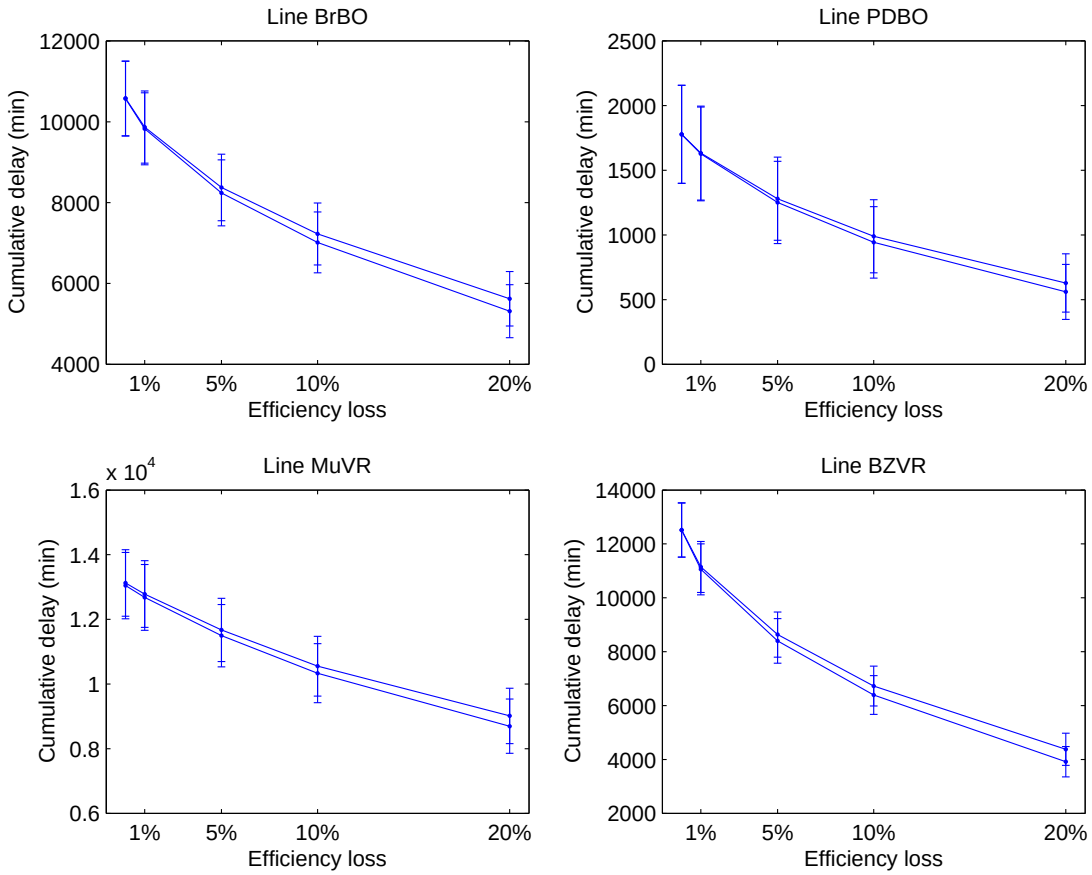

Figure 4.9: Confidence intervals of upper and lower bounds of the optimal solution of stochastic model slim2

sible) for practical instances. A "slim" version performed much better, provided that a clever objective function is used. The fastest method, Light Robustness (LR), proved to be quite accurate when dealing with a reasonable robustness–efficiency tradeoff, allowing for a fast solution of large instances. On the whole, Light Robustness qualifies as a suitable tool for addressing large-scale real scenarios, and can even be embedded in the nominal solver to find optimized train-precedence patterns leading to more robust timetables.

Future direction of research should address the important topics below.

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

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

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

Bibliography

[1] A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.*, 33:60?100, 1991.

[2] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):1–12, 2006.

[3] E. Amaldi, M. E. Pfetsch, and L.E. Trotter Jr. On the maximum feasible subsystem problem, IISs and IIS-hypergraphs. *Mathematical Programming*, 95(3):533–554, 2003.

[4] J. L. Arthur and A. Ravindran. PAGP, a partitioning algorithm for (linear) goal programming problems. *ACM Trans. Math. Softw.*, 6(3):378–386, 1980.

[5] A. Bachem and W. Kern. *Linear Programming Duality. An Introduction to Oriented Matroids*. Number 074 in Universitext. Springer, 1992.

[6] E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89(1-3):3–44, December 1998.

[7] E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42:1229–1246, 1996.

[8] E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19:1–9, 1996.

[9] M. L. Balinski and A. W. Tucker. Duality theory of linear programs: A constructive approach with applications. *SIAM Review*, 11(3):347–377, July 1969.

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

[11] E. M. L. Beale. Survey of integer programming. *OR*, 16(2):219–228, jun 1965.

[12] J.F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, Dec. 1962.

[13] L. Bertacco. *Exact and Heuristic Methods for Mixed Integer Linear Programs*. PhD thesis, University of Padova, 2005.

[14] John R. Birge and Francois Louveaux. *Introduction to Stochastic Programming (Springer Series in Operations Research and Financial Engineering)*. Springer, 1st ed. 1997. corr. 2nd printing edition, 2000.

[15] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[16] A. Caprara, M. Fischetti, and A. N. Letchford. On the separation of maximally violated mod-k cuts. In *Proceedings of the 7th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 87–98, London, UK, 1999. Springer-Verlag.

[17] A. Caprara, M. Fischetti, and P. Toth. Modeling and solving the train timetabling problem. *Operations Research*, 50(5):851–861, 2002.

[18] A. Caprara, M. Monaci, P. Toth, and P.L. Guida. A lagrangian heuristic algorithm for a real-world train timetabling problem. *Discrete Applied Mathematics*, 154(5):738–753, 2006.

[19] V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

[20] Serafino Cicerone, Gianlorenzo D'Angelo, Gabriele Di Stefano, Daniele Frigioni, and Alfredo Navarra. On the interaction between robust timetable planning and delay management. Technical Report ARRIVAL-TR-0116, ARRIVAL project, 2007.

[21] Charles E. Clark. Importance sampling in Monte Carlo analyses. *Operations Research*, 9(5):603–620, 1961.

[22] W. Cook, S. Dash, R. Fukasawa, and M. Goycoolea. Numerically safe gomory mixed-integer cuts. 2008.

[23] G. Cornuéjols. Valid inequalities for mixed integer linear programs. *Math. Program.*, 112(1):3–44, 2007.

[24] G. Cornuéjols, L. Liberti, and G. Nannicini. Improved strategies for branching on general disjunctions. *Optimization Online*, 2008.

[25] Balas E. and A. Saxena. Optimizing over the split closure. *Mathematical Programming*, DOI 10.1007/s10107-006-0049-5, 2006.

[26] M. Fischetti and Lodi A. Optimizing over the first Chvátal closure. *Mathematical Programming B*, 110(1):3–20, 2007.

[27] M. Fischetti, A. Lodi, and A. Tramontani. Experiments with disjunctive cuts. Technical report, November 2007. In preparation.

[28] M. Fischetti and M. Monaci. Light robustness. Technical Report ARRIVAL-TR-0119, ARRIVAL Project, 2008.

[29] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[30] R.S. Garfinkel and G.L. Nemhauser. *Integer Programming.* New York: John Wiley and Sons, 1972.

[31] A. M. GEOFFRION. Lagrangian relaxation and its uses in integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[32] Arthur M. Geoffrion. Generalized benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[33] Alex Orden George B. Dantzig and Philip Wolfe. The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific J. Math.*, 5(2):183–195, 1955.

[34] J. Gleeson and J. Ryan. Identifying minimally infeasible subsystems of inequalities. *ORSA Journal on Computing*, 2(1):61–63, 1990.

[35] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society*, 64:275–278, 1958.

[36] R. E. Gomory. An algorithm for the mixed integer problem. Technical Report RM-2597, The RAND Cooperation, 1960.

[37] R. E. Gomory. An algorithm for integer solutions to linear programming. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302, New York, 1963. McGraw-Hill.

[38] R. E. Gomory. Early integer programming. *Operations Research*, 50:78–81, Jan. - Feb. 2002.

[39] Mads Hofman, Line Madsen, Julie Jespersen Groth, Jens Clausen, and Jesper Larsen. Robustness and recovery in train scheduling - a case study from DSB S-tog a/s". In Riko Jacob and Matthias Müller-Hannemann, editors, *ATMOS 2006 - 6th Workshop on Algorithmic Methods and Models for Optimization of Railways.* Internationales Begegnungs- und Forschungszentrum f"ur Informatik (IBFI), Schloss Dagstuhl, Germany, 2006.

[40] ILOG Inc. *ILOG CPLEX 10.1 User's Manual*, 2007.

[41] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[42] L. Khachian. A polynomial algorithm for linear programming. *Doklady Akad Nauk USSR*, 244(5):1093–1096, 1979.

[43] L.G. Kroon, R. Dekker, and M.J.C.M. Vromans. Cyclic railway timetabling: a stochastic optimization approach. In *Algorithmic Methods for Railway Optimization*, Lecture Notes in Computer Science, pages 41–66. Springer Berlin / Heidelberg, 2007.

[44] Adam N. Letchford and Andrea Lodi. Strengthening chvátal-gomory cuts and gomory fractional cuts. *Operations Research Letters*, 30:74–82, 2002.

[45] A.N. Letchford and A. Lodi. Strengthening chvátal-gomory cuts and gomory fractional cuts. *Oper. Res. Lett.*, 30(2):74–82, 2002.

[46] C. Liebchen and L. W.P. Peeters. On cyclic timetabling and cycles in graphs. Technical Report 761-2002, TU Berlin, Mathematical Institute, 2002.

[47] C. Liebchen and S.Stiller. Delay resistant timetabling. Technical Report ARRIVAL-TR-0056, ARRIVAL Project, 2006.

[48] Christian Liebchen, Marco Lübbecke, Rolf H. Möhring, and Sebastian Stiller. Recoverable robustness. Technical Report ARRIVAL-TR-0066, ARRIVAL-Project, 2007.

[49] Christian Liebchen, Michael Schachtebeck, Anita Schöbel, Sebastian Stiller, and André Prigge. Computing delay resistant railway timetables. Technical Report ARRIVAL-TR-0071, ARRIVAL Project, October 2007.

[50] Jeff Linderoth, Alexander Shapiro, and Stephen Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, February 2006.

[51] W. L. Loh. On latin hypercube sampling. *The Annals of Statistics*, 24(5), 1996.

[52] R. Lougee-Heimer. The common optimization interface for operations research: Promoting open-source software in the operations research community. *IBM J. Res. Dev.*, 47(1):57–66, 2003.

[53] A. Lodi M. Fischetti. Optimizing over the first Chvàtal closure. In M. Juenger and V. Kaibel, editors, *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science 3509, pages 12–22. Springer-Verlag Berlin Heidelberg, 2005.

[54] T.L. Magnanti and R.T. Wong. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria. *Operations Research*, 29:464–484, 1981.

[55] H. Marchand, A. Martin, R. Weismantel, and L. Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397–446, November 2002.

[56] H. Marchand and L.A. Wolsey. Aggregation and mixed integer rounding to solve mips. Papers 9839, Catholique de Louvain - Center for Operations Research and Economics, 1998.

[57] F. Margot. Testing cut generators for mixed-integer linear programming. *Optima*, 77, 2008.

[58] R. R. Meyer. On the existence of optimal solutions to integer and mixed integer programming problems. *Mathematical Programming*, 7:223–235, 1974.

[59] P. Miliotis. Integer programming approaches to the travelling salesman problem. *Mathematical Programming*, 10:367–378, 1976.

[60] P. Miliotis. Using cutting planes to solve the symmetric travelling salesman problem. *Mathematical Programming*, 15:177–178, 1978.

[61] K. Nachtigall. Periodic network optimization and fixed interval timetables. Habilitation Thesis, Deutsches Zentrum für Luft-und Raumfahrt, Braunschweig, 1999.

[62] G. Nemhauser and L. Wolsey. *Integer and combinatorial optimization.* Wiley, 1988.

[63] L. W. P. Peeters. *Cyclic Railway Timetable Optimization.* PhD thesis, Erasmus University Rotterdam, 2003.

[64] C. Poojari and J. Beasley. Improving Benders decomposition using a genetic algorithm. Technical report, 2006. Submitted to INFORMS Journal on Computing.

[65] W. Rei, J. F. Cordeau, M. Gendreau, and P. Soriano. Accelerating Benders decomposition by local branching. Technical report, January 2006. To appear in INFORMS Journal on Computing, January 2006.

[66] A. Ruszczynski and A. Shapiro, editors. *Stochastic Programming (Hanbooks in Operations Research and Management Series).* Elsevier Publishing Company, 2003.

[67] A. Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, Inc., New York, NY, USA, 1986.

[68] P. Serafini and W. Ukovich. A mathematical model for periodic scheduling problems. *SIJDM: SIAM Journal on Discrete Mathematics*, 2, 1989.

[69] A. Shapiro. Monte carlo sampling approach to stochastic programming. In *ESAIM: Proceedings*, volume 13, pages 65–73, December 2003.

[70] M. Tamiz, D. F. Jones, and E. El-Darzi. A review of goal programming and its applications. *Annals of Operations Research*, (1):39–53, 1995.

[71] T. Terlaky. Book review: Computational techniques of the simplex method. *Computational Optimization and Applications*, 26(2):209–210, November 2003.

[72] B. Verweij, S. Ahmed, A. J. Kleywegt, G. Nemhauser, and A. Shapiro. The sample average approximation method applied to stochastic routing problems: A computational study. *Computational and Applied Optimization*, 24, 2003.