1

PM-1 3 1/2"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file / Votre référence

Our file / Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Analyzing Infeasible Mixed-Integer and Integer Linear Programs

by
Olivier Guieu

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Science
Information and Systems Science
Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario

October 5, 1995

©Copyright

1995, Olivier Guieu

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-612-09005-1

Canada

Name G. U. I. C. U.

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

PSYCHOLOGY RESEARCH

SUBJECT TERM

0 7 3 6

SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language
General 0679
Ancient 0289
Linguistics 0290
Modern 0291
Literature
General 0401
Classical 0294
Comparative 0295
Medieval 0297
Modern 0298
African 0316
American 0591
Asian 0305
Canadian (English) 0352
Canadian (French) 0355
English 0593
Germanic 0311
Latin American 0312
Middle Eastern 0315
Romance 0313
Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion
General 0318
Biblical Studies 0321
Clergy 0319
History of 0320
Philosophy of 0322
Theology 0469

SOCIAL SCIENCES

American Studies 0323
Anthropology
Archaeology 0324
Cultural 0326
Physical 0327
Business Administration
General 0310
Accounting 0272
Banking 0770
Management 0454
Marketing 0338
Canadian Studies 0385
Economics
General 0501
Agricultural 0503
Commerce-Business 0505
Finance 0508
History 0509
Labor 0510
Theory 0511
Folklore 0358
Geography 0366
Gerontology 0351
History
General 0578

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science
General 0615
International Law and Relations 0616
Public Administration 0617
Recreation 0814
Social Work 0452
Sociology
General 0626
Criminology and Penology 0627
Demography 0938
Ethnic and Racial Studies 0631
Individual and Family Studies 0628
Industrial and Labor Relations 0629
Public and Social Welfare 0630
Social Structure and Development 0700
Theory and Methods 0344
Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture
General 0473
Agronomy 0285
Animal Culture and Nutrition 0475
Animal Pathology 0476
Food Science and Technology 0359
Forestry and Wildlife 0478
Plant Culture 0479
Plant Pathology 0480
Plant Physiology 0817
Range Management 0777
Wood Technology 0746
Biology
General 0306
Anatomy 0287
Biostatistics 0308
Botany 0309
Cell 0379
Ecology 0329
Entomology 0353
Genetics 0369
Limnology 0793
Microbiology 0410
Molecular 0307
Neuroscience 0317
Oceanography 0416
Physiology 0433
Radiation 0821
Veterinary Science 0778
Zoology 0472
Biophysics
General 0786
Medical 0760

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Polynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences
General 0566
Audiology 0300
Chemotherapy 0992
Dentistry 0567
Education 0350
Hospital Management 0769
Human Development 0758
Immunology 0982
Medicine and Surgery 0564
Mental Health 0347
Nursing 0569
Nutrition 0570
Obstetrics and Gynecology 0380
Occupational Health and Therapy 0354
Ophthalmology 0381
Pathology 0571
Pharmacology 0419
Pharmacy 0572
Physical Therapy 0382
Public Health 0573
Radiology 0574
Recreation 0375

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences
Chemistry
General 0485
Agricultural 0749
Analytical 0486
Biochemistry 0487
Inorganic 0488
Nuclear 0738
Organic 0490
Pharmaceutical 0491
Physical 0494
Polymer 0495
Radiation 0754
Mathematics 0405
Physics
General 0605
Acoustics 0986
Astronomy and Astrophysics 0606
Atmospheric Science 0608
Atomic 0748
Electronics and Electricity 0607
Elementary Particles and High Energy 0798
Fluid and Plasma 0759
Molecular 0609
Nuclear 0610
Optics 0752
Radiation 0756
Solid State 0611
Statistics 0463
Applied Sciences
Applied Mechanics 0346
Computer Science 0984

Engineering
General 0537
Aerospace 0538
Agricultural 0539
Automotive 0540
Biomedical 0541
Chemical 0542
Civil 0543
Electronics and Electrical 0544
Heat and Thermodynamics 0348
Hydraulic 0545
Industrial 0546
Marine 0547
Materials Science 0794
Mechanical 0548
Metallurgy 0743
Mining 0551
Nuclear 0552
Packaging 0549
Petroleum 0765
Sanitary and Municipal System Science 0790
Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of this thesis

**ANALYZING INFEASIBLE MIXED-INTEGER AND
INTEGER LINEAR PROGRAMS**

submitted by Olivier Guieu
in partial fulfilment of the requirements for
the degree of Master of Science
Information and Systems Science

J.W.Chinneck, Thesis Supervisor

S.Mahmoud, Chair, Department of Systems and Computer Engineering

Carleton University

October 5, 1995

ABSTRACT

The infeasibility of a mixed-integer linear program (MILP) or all-integer linear program (ILP) is a difficult problem to contend with. Since the models can be quite large and complex, automated assistance in analyzing MILP/ILP infeasibility is necessary.

The identification of Irreducible Infeasible Subsets of constraints (IISs) has been used successfully for analyzing infeasible continuous-variable linear programs. This thesis extends the concept to the much more difficult non-continuous case.

Properties and characteristics of MILP/ILP IISs are presented, as are a series of algorithms for isolating IISs. Experimental results illustrating the efficiency and effectiveness of the algorithms in analyzing MILPs/ILPs of various sizes are presented.

ACKNOWLEDGEMENTS

I wish to acknowledge the guidance and help given to me by Dr. John W. Chinneck. Many valuable comments and explanations due to Ed Klotz, of CPLEX Optimization Inc., are gratefully acknowledged.

I also wish to thank my friends and fellow students for sharing with me this great experience. And last but not least, I would like to heartily thank my family for supporting me through these past two years.

Table Of Contents

Acceptance Sheet	ii
Abstract	iii
Acknowledgements	iv
Table Of Contents	v
List of Figures	x
List of Algorithms	xi
List of Tables	xiii
Chapter 1 Introduction	1
Chapter 2 Background and Notation	2
2.1 Linear Programming	2
2.1.1 Notation and Definitions	2
2.1.2 State of the Art in LP Post-Infeasibility Analysis	3
2.2 Mixed Integer Linear Programming	12
2.2.1 Preliminary	12
2.2.2 Notation and Definitions	12
2.2.3 MILP Solution Techniques: an Introduction	14
2.2.4 Cutting Plane Technique	14
2.2.5 Branch and Bound Algorithm	15
2.2.6 Branch and Cut Approach	17
2.2.7 Duality Results for MILPs and Role of the LP Relaxation	18
2.2.8 A Special Case: Linear Diophantine Equations	19

	2.2.9	Sensitivity Analysis	20
Chapter	3	State of the Art in MILP Post-Infeasibility Analysis	22
Chapter	4	Problem Statement	23
	4.1	Object of the Research	23
	4.2	Deficiencies of the State of the Art	24
	4.3	Conclusion	24
Chapter	5	Problem Characterization	25
	5.1	Introduction	25
	5.2	Desirable Features of an IIS	25
	5.3	Properties	25
	5.3.1	Assumption	25
	5.3.2	On the “Continuous” Part of IISs	27
	5.3.3	Maximum Cardinality of an IIS	27
	5.3.4	Number of IISs	28
	5.4	On the B & B Tree	29
	5.4.1	Introduction	29
	5.4.2	Node and Leaf Information	30
	5.4.3	Branch Information	32
	5.4.4	Modification of the Objective Function	33
	5.4.5	Sensitivity Analysis on IRs	33
Chapter	6	Presentation of the Experiments	35
	6.1	Test Set Description	35
	6.2	Software Prototype	37
	6.3	Contents of Tables	37
Chapter	7	Deletion Filtering Algorithms for Small MILPs	39

7.1	Overview	39
7.2	Algorithms	39
7.2.1	Deleting IRs First	40
7.2.2	Deleting LCs First	41
7.3	Validity of the Isolations	41
7.4	Time and Space Complexities	42
7.5	Example	42
7.5.1	Example Description	42
7.5.2	Algorithm 3: Deleting IRs First	43
7.5.3	Algorithm 4: Deleting LCs First	44
7.6	Computational Results	45
7.7	Discussion	46
Chapter 8	Incremental Filtering Algorithm for Small-Medium MILPs	51
8.1	Overview	51
8.2	Algorithm	51
8.3	Validity of the Isolation	52
8.4	Time and Space Complexities	54
8.5	Opportunist Incremental Filter	54
8.6	Example	55
8.6.1	Algorithm 6: Straight Incremental Filter	55
8.6.2	Algorithm 7: Opportunist Incremental Filter	58
8.7	Computational Results	61
8.8	Discussion	62
Chapter 9	Incremental/Deletion Filtering Algorithm for Small-Medium MILPs	65
9.1	Overview	65

9.2	Algorithm	65
9.3	Validity of the Isolation	65
9.4	Time and Space Complexities	66
9.5	Opportunist Incremental/Deletion Filter	67
9.6	Example	67
9.6.1	Algorithm 8: Straight Incremental/Deletion Filter	67
9.6.2	Algorithm 9: Opportunist Incremental/Deletion Filter	69
9.7	Computational Results	70
9.8	Discussion	70
Chapter	10 Speed Considerations	73
10.1	On MILPs solutions	73
10.1.1	Settings of the MILP Solver	73
10.1.2	Using Past Isolation Results	78
10.2	Grouping Constraints	81
10.2.1	Fixed-Size Blocks	82
10.2.2	Adaptive-Size Blocks	86
Chapter	11 B & B Tree Filtering for large MILPs	98
11.1	Overview	98
11.2	Algorithms	99
11.2.1	Notation	100
11.2.2	B & B Tree Filtering Algorithm	101
11.2.3	Sensitivity B & B Tree Filtering Algorithm	102
11.2.4	Heuristic/Sensitivity B & B Tree Filtering Algorithm	103
11.3	Validity of the Isolations	103
11.4	Time and Space Complexities	103
11.5	Example	105

11.5.1	Algorithm 13: B & B Tree Filtering Algorithm	105
11.5.2	Algorithm 14: Sensitivity B & B Tree Filtering Algorithm	107
11.5.3	Algorithm 15: Heuristic/Sensitivity B & B Tree Filtering Algorithm	108
11.6	Computational Results	109
11.6	Discussion	110
Chapter	12 A Framework for Analyzing Inconsistent MILPs	115
12.1	A Few Remarks	115
13.2	A Framework for Isolating IISs in Inconsistent MILPs	116
Chapter	13 Conclusions	119
13.1	Settings of B & B MILP Solver and Inconsistent MILP Formulations	119
13.2	IIS-Isolation Algorithms	119
13.3	Contributions	121
13.4	Future Research	121
Appendix	A Derivation of a Cut	123
Appendix	B Hermite Normal Form Transformation	125
References		129

List of Figures

Figure 1	Example of a Feasible Region	4
Figure 2	Example of a Cluster of IISs	6
Figure 3	Example of a B & B Tree	16
Figure 4	Example of a “difficult” MILP	18
Figure 5	Example of IISs in ILPs	26
Figure 6	An IIS with $(n + 2)$ Linear Constraints	28
Figure 7	Two Different B & B Trees	31
Figure 8	Information Gathered in a B & B Tree	105

List of Algorithms

Algorithm	1	(LC-BD) Deletion Filter	9
Algorithm	2	Incremental Filter	10
Algorithm	3	(IR-LC-BD) Deletion Filter	40
Algorithm	4	(LC-IR-BD) Deletion Filter	41
Algorithm	5	“Safety BDs” (IR-LC-BD) Deletion Filter	46
Algorithm	6	(IR-LC-BD) Incremental Filter	52–53
Algorithm	7	(IR-LC-BD) Opportunist Incremental Filter	56–57
Algorithm	8	(IR) Incremental / (LC-BD) Deletion Filter	66
Algorithm	9	(IR) Opportunist Incremental / (LC-BD) Deletion Filter	68
Algorithm	10	(IR-LC-BD) Optimistic Blocking Deletion Filter	80–81
Algorithm	11	(IR-LC-BD) Conservative Blocking Deletion Filter	85
Algorithm	12	Examples of Different Adaptive Blocking Strategies	87
Algorithm	13	B & B Tree Deletion Filter	101
Algorithm	14	Sensitivity B & B Tree Deletion Filter	102

Algorithm	15	Heuristic/Sensitivity B & B Tree Deletion Filter	104
Algorithm	16	A Framework for Isolating IISs in Inconsistent MILPs	118

List of Tables

Table	1	Test Data Sets	36
Table	2	Infeasibility Proofs	37
Table	3	Algorithm 3, Depth-First Search, Default Branching Scheme	45
Table	4	Algorithm 5, Depth-First Search, Default Branching Scheme	47
Table	5	Algorithm 4, Depth-First Search, Default Branching Scheme	48
Table	6	Rankings: Based on Tables 3 and 5	49
Table	7	Algorithm 6, Depth-First Search, Default Branching Scheme	61
Table	8	Algorithm 7, Depth-First Search, Default Branching Scheme	62
Table	9	Algorithm 8, Depth-First Search, Default Branching Scheme	71
Table	10	Algorithm 9, Depth-First Search, Default Branching Scheme	72
Table	11	Algorithm 3, Depth-First Search, Branching on Max.-Infeasibility Variables	74
Table	12	Algorithm 3, Depth-First Search, Default Branching Scheme	75
Table	13	Algorithm 3, Best-Bound Search,	

		Branching on Max.-Infeasibility Variables	76
Table	14	Algorithm 3, Best-Bound Search, Default Branching Scheme	77
Table	15	Rankings: Based on Tables 11, 12, 13, and 14	78
Table	16	Algorithm 3, Using Priority Lists	79
Table	17	Algorithm 10, Fixed Size Blocks	82
Table	18	Algorithm 6, Fixed Size Blocks	83
Table	19	Algorithm 8, Fixed Size Blocks	84
Table	20	Algorithm 11, Fixed Size Blocks	86
Table	21	Algorithm 10, Additive Adaptive Size Blocks (1)	88
Table	22	Algorithm 10, Additive Adaptive Size Blocks (2)	89
Table	23	Algorithm 10, Multiplicative Adaptive Size Blocks (1)	90
Table	24	Algorithm 10, Multiplicative Adaptive Size Blocks (2)	91
Table	25	Algorithm 11, Additive Adaptive Size Blocks (1)	92
Table	26	Algorithm 11, Additive Adaptive Size Blocks (2)	93
Table	27	Algorithm 11, Multiplicative Adaptive Size Blocks (1)	94
Table	28	Algorithm 11, Multiplicative Adaptive Size Blocks (2)	95
Table	29	Rankings: Based on Tables 21, 22, 23, and 24	96
Table	30	Rankings: Based on Tables 25, 26, 27, and 28	97
Table	31	Algorithm 13, B & B Tree Deletion Filtering	111

Table 32	Algorithm 14, Sensitivity B & B Tree Deletion Filtering	. 112
Table 33	Algorithm 15, Heuristic/Sensitivity	
	B & B Tree Deletion Filtering	. 113
Table 34	Rankings of Isolation Algorithms,	
	Based on Tables 3, 7 and 9	. 116
Table 35	Ranking of Blocking Strategies Algorithm,	
	Based on Tables 17, 20 , 22 and 26	. 117

CHAPTER 1

INTRODUCTION

Mixed-integer linear programs (MILPs) or all-integer linear programs (ILPs) appear in the mathematical formulation of most combinatorial optimization problems, such as fleet-crew scheduling, assignment problems, vehicle routing, etc. Following the increasing trend in linear program dimensions, MILPs or ILPs with thousands of constraints and hundreds of integer variables are now common. Checking that the current formulation is feasible and being able to repair an infeasible formulation has become crucial at every stage of the modeling process.

Diagnosing any large inconsistent mathematical program can be a very daunting task for the modeller and requires in most cases the use of tools derived from artificial intelligence techniques. However, preprocessing the original infeasible problem by algorithmic tools helps to focus on a substantially smaller, but still infeasible, subproblem. This can speed up the diagnosis and even allow direct explanation of the source of infeasibility.

The theoretical considerations which led to the notion of an Irreducible Infeasible Subset (IIS) as well as practical efficient implementations of isolation algorithms have been developed over the last few years for analyzing infeasible linear programs. However at present, no method has been designed to handle inconsistent MILPs and ILPs. There is therefore a genuine need for developing efficient isolation algorithms for infeasible MILPs/ILPs. Providing a similar framework for the mixed-integer and all-integer linear programs is the purpose of this thesis.

CHAPTER 2

BACKGROUND AND NOTATION

2.1 Linear Programming

2.1.1 Notation and Definitions.

Linear programming problems (LPs) are the mathematical formulation of a certain class of optimization problems for which:

- the continuous and real-valued variables of the problem (x_j) must satisfy a set of linear constraints.
- the value of a linear function of the x_j , called the *objective function* must be either maximized or minimized.

An LP can therefore be expressed as:

$$\begin{array}{ll}
 \text{Maximize} & F(x) = \sum_{j=1}^n c_j \cdot x_j \\
 \text{Subject to} & \sum_{j=1}^n a_{ij} \cdot x_j \sim b_i \quad (i = 1, 2, \dots, m) \\
 \text{where} & x \in \mathbb{R}^n \quad \text{and} \quad \sim \in \{\leq, =, \geq\}
 \end{array}$$

A linear programming problem (P) is written in *canonical form* if and only if it has the following expression:

$$\begin{array}{ll}
 P: \text{ Maximize} & F(x) = c \cdot x \\
 \text{Subject to} & A \cdot x \leq b, \quad x \geq 0 \\
 \text{where} & x \in \mathbb{R}^n \text{ is the vector of variables,} \\
 & b \in \mathbb{R}^m \text{ is the vector of the right hand sides (RHS),} \\
 & A \in \mathbb{R}^{m \times n} \text{ is the matrix of the } a_{ij}, \\
 & c \in \mathbb{R}^{1 \times n} \text{ is the row vector of the cost coefficients.}
 \end{array}$$

Equivalently a LP can be expressed in *standard form* where all less-than-or-equal-to constraints ($Ax \leq b$) are replaced by equations ($Ax = b$) through the use of additional *slack variables*.

A vector $x \in \mathbb{R}^n$, satisfying the *functional constraints* $Ax \leq b$ is a *feasible solution* of (P) if it also satisfies the nonnegativity constraint, i.e. $x \in \mathbb{R}_+^n$. Hence solving (P) consists of finding a feasible solution that maximizes the objective function. In case no feasible solution exists the LP is said to be *infeasible*. The set of all feasible solutions of (P) defines the *feasibility set* denoted by $\mathcal{F}_{LP}$ which can be geometrically seen as the *region of feasibility* of P; every inequality divides the original variable space into two parts, and the feasible region is the convex polyhedron obtained by intersecting all these “half spaces”.

However not all linear constraints play the same role in the definition of the feasible region. From Figure 1 we can see that the half spaces defined by constraints 1,2,3 and “ $x_2 \geq 0$ ” only, completely restrain the region of feasibility; they are called *necessary constraints*. On the other hand, constraints 4, 5 and “ $x_1 \geq 0$ ” are *redundant constraints*: they can be dropped from the system without modifying the feasible region [Telgen 83]. Note that constraint 4 touches the feasible region whereas constraint 5 does not. Telgen[83] subclassifies redundant constraints according to this criteria into *weakly* (4) and *strongly redundant* (5) constraints.

2.1.2 State of the Art in LP Post-Infeasibility Analysis.

As noted by Nemhauser [94] there have been tremendous improvements in the ability to solve linear programming problems over the last four decades, resulting from the conjunction of more and more powerful hardware as well as software tools. As larger models are solved, post-infeasibility analysis techniques become a crucial and necessary step in LP modeling. It is not unusual to get infeasible large LPs in the early stages of modeling. Hence a number of different approaches have been developed in the past two decades to deal with infeasible linear programming problems. As pointed

out by Chakravarti [94] three main directions have been explored so far in post-infeasibility analysis: elastic programming methods, infeasibility subset isolations and infeasible subset isolations. These techniques and others will be explained in the general framework for post-infeasibility analysis given below but first, a few terms need to be defined.

Definition: A set of constraints is *inconsistent* if any associated LP is infeasible.

Definition: An *Inconsistent System* (IS) is an inconsistent set of constraints.

Definition: An *Irreducibly Inconsistent System* (IIS) is a minimal inconsistent set of constraints [van Loon 81].

An IIS is characterized by:

- an IIS is an IS,
- all proper subsystems of an IIS are consistent.

It is also useful to subclassify IISs depending on the nature of the constraints involved:

Definition: An *Irreducibly Inconsistent System of Functional constraints* (IISF) is the complete subset of functional constraints in an IIS [Chinneck and Dravnieks 91].

We also recall the definitions specifying the possible set relationships between IISs:

Definition: An IIS is *overlapped* if it shares at least one constraint with another IIS otherwise it is *isolated*.

Definition: A *cluster* of IISs is a maximal set of IISs such that each IIS overlaps at least one other IIS of the cluster (Figure 2).

And finally,

Definition: A non-IIS constraint does not belong to any IIS.

Preprocessing Techniques

Bound tightening methods belong to the more general set of preprocessing techniques. The idea is to reformulate the LP problem so that a “better” formulation is reached (however both formulations should share the same set of feasible solutions). A reformulation of the original problem could result for instance in tighter bounds on some variables, some coefficients could even be modified, some constraints could become *tighter* (modification of the RHS value) or *relaxed* (in the case of a redundant constraint). Techniques that reformulate problems have been described by Brearley

et al.[75]. Infeasibility may be detected by bound tightening in some cases prior to the use of any solution method. For instance the computation of the lower and upper bounds of the left hand side of a constraint, by a linear combination of the individual bounds on each term of the sum, may lead to inconsistency with the RHS. Then, tracking back all constraints involved in the tightening process could isolate an IS. Such methods may fail to detect infeasibility when, for instance, it is the conjunction

of several constraints that cause infeasibility. In the following we examine a particular implementation of these techniques.

Greenberg, through the ANALYZE software [Greenberg 87,93a,93b,93c,94] provides a complete integrated environment for an in-depth analysis of linear programs. A set of powerful analysis procedures allow the experienced user to successfully achieve a clear insight of the structure of the LP. The purpose of that tool is not to solve LPs but rather to analyse them as suggested in its name. However in some cases, through the use of techniques such as bound tightening, a solution may be reached. The inner structure of the LP may be used as well and heuristics locating useful substructures are provided [Greenberg 94]. Similarly, the inconsistency of a set of constraints may be established, and an infeasible subset may be isolated by backtracking the successive steps of the bound tightening routine. There is no guarantee however that the isolated subset is going to be minimal and furthermore, ANALYZE may not always detect infeasibility.

Elastic Programming

When infeasibility arises in an LP problem, one may also consider modifying the right hand side (RHS) values of some linear constraints, converting them to goals to be satisfied. Using "sensitivity analysis" one may determine, without incurring the cost of solving another LP, the minimum changes of the RHS values required to reach feasibility. Constraints that cause infeasibility are identified by using the information available when phase I terminates proving the LP infeasible [Roodman 79]. As mentioned by Dravnieks[89], this approach only points out constraints that need to be "repaired" to reach feasibility regardless of their correctness in the original model, i.e. a typographic error in the RHS value of some constraint may lead to the identification of a different constraint as the cause of infeasibility.

IIS Isolation Techniques

Several techniques have been developed to cope with the problem of finding an IIS for an infeasible LP.

van Loon[81] adopted an *exhaustive enumerative approach* to isolate IISs in LPs based on two elements:

- at each step of the enumeration, simplex-like operations are performed to check the feasibility of an untested basis,
- IISs are identified via a necessary and sufficient characterization which is computationally cheap to verify, once the tableau manipulations have been conducted.

Hence, the algorithm enumerates exhaustively all possible bases, including some that are not desired, i.e. that do not identify any IIS, in order to find the set of IISs contained in the original problem. Computational efficiency concerns related to an implementation of this approach have been extensively discussed by Dravnieks[89].

A *directed enumeration scheme* has been designed by Gleeson and Ryan[90] for identifying all IISs in LPs, as opposed to van Loon's approach where all subsystems are enumerated. Using one alternate of Farkas' lemma, they show how to enumerate all IISs by finding all the extreme points of a related polyhedron; then IISs are given by the supports of the vertices belonging to that polytope. The practical implementation of this method is based on an efficient algorithm, designed by Dyer[83], for enumerating all those extreme points and has been implemented by Parker and Ryan[94]. However this approach has been proven theoretically inefficient by Chinneck and Dravnieks [91] when one is interested in isolating the first IIS.

Several *filtering routines* or "filters", i.e. procedures that retain constraints belonging to IISs and discard constraints that do not, have been successfully implemented and tested by Chinneck and Dravnieks[91]. These procedures have been incorporated in publicly available software packages and hence may provide useful benchmarks for new algorithms. These filtering routines do provide a positive identification of IISs in an infeasible LP. When several IISs are present in the infeasible

LP, a straight deletion filter (Algorithm 1) isolates the IIS whose first constraint is tested last. Although any ordering of the constraints is possible, LC-BD, i.e. linear constraints (LCs) first and then bounds on the variables (BDs), usually gives “better” IISs, hence the name of Algorithm 1. Sensitivity filtering of the constraints [Dravnieks 89 pp. 6–8] may speed up the deletion filtering process. Further heuristics have been developed by Chinneck [93b,94] in order to isolate the IIS whose row cardinality is minimum since these are easier to diagnose. An implicit measure of the relative quality of an IIS isolated in an infeasible LP is therefore the ratio of the number of row constraints contained in this IIS to the row cardinality of the “best” IIS, i.e. the IIS with the smallest number of row constraints.

Input	:	An <i>infeasible</i> LP $P_0 = \{LC_0, BD_0\}$
Step 0	:	SET $T := LC_0 \cup BD_0$
Step 1	:	FOR EACH $lc_k \in LC_0$ DO IF $T \setminus \{lc_k\}$ <i>inconsistent</i> THEN SET $T := T \setminus \{lc_k\}$
Step 2	:	a. SET $BD_1 = BD_0 \setminus \{BDs \text{ on variables not in } lc \in T\}$ b. SET $T := (T \setminus BD_0) \cup BD_1$ c. FOR EACH $bd_k \in BD_1$ DO IF $T \setminus \{bd_k\}$ <i>inconsistent</i> THEN SET $T := T \setminus \{bd_k\}$
Step 3	:	STOP: T is an IIS

ALGORITHM 1. (LC-BD) Deletion Filter

The infeasible LP (LP_0), pictured in Figure 2, is filtered by Algorithm 1:

Step 0: $T := \{c_1, \dots, c_5, x \geq 0, y \geq 0\}$

Step 1: $T \setminus \{c_1\}$ *FEASIBLE*

Step 1: $T \setminus \{c_2\}$ *FEASIBLE*

Step 1: $T \setminus \{c_3\}$ *INFEASIBLE;* $T := T \setminus \{c_3\}$

Step 1: $T \setminus \{c_4\}$ *INFEASIBLE;* $T := T \setminus \{c_4\}$

Step 1: $T \setminus \{c_5\}$ *FEASIBLE*
Step 2: a. & b. T not modified
Step 2: c. $T \setminus \{x \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x \geq 0\}$
Step 2: c. $T \setminus \{y \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{y \geq 0\}$
Step 3: STOP : $T = \{c_1, c_2, c_5\}$ is an IIS

Conversely, the *incremental* algorithm (Algorithm 2) [Tamiz et al. 95a, 95b] builds IISs from scratch: constraints from the original LP (C) are added one by one to a test set of constraints (T). When infeasibility is first encountered (Step 1), the last added constraint is moved to an initially empty auxiliary LP(I). All following constraints are discarded from C and the process is repeated until the temporary LP I becomes infeasible (Step 2), it is then an IIS.

C : Ordered set of constraints in original LP ($C = \{c_1, \dots, c_n\}$)
I : Current set of constraints contributing to inconsistency
T : Test set of constraints
Step 0: SET $I = \emptyset$
Step 1: SET $T = I$
 FOR EACH $c_k \in C$ DO
 SET $T := T \cup \{c_k\}$
 IF T inconsistent THEN
 SET $I := I \cup \{c_k\}$
 SET $C := \{C_1, \dots, C_k\}$
 GOTO Step 2
Step 2: IF I feasible THEN
 GOTO Step 1
 ELSE
 STOP: **I is an IIS**

ALGORITHM 2. Incremental Filter

Unlike the straight deletion filter, if the model contains several IISs, the IIS

whose last constraint is seen first is identified. This difference is illustrated by the analysis of the infeasible LP of Figure 2:

Step 0: $I = \emptyset$
Step 1: $T = \emptyset$
 $T = \{c_1\}$ *FEASIBLE*
 $T = \{c_1, c_2\}$ *FEASIBLE*
 $T = \{c_1, c_2, c_3\}$ *INFEASIBLE*
 $I = \{c_3\}$
 $C = \{c_1, c_2, c_3\}$

Step 2: $I = \{c_3\}$ is *FEASIBLE*

Step 1: $T = \{c_3\}$
 $T = \{c_3, c_1\}$ *FEASIBLE*
 $T = \{c_3, c_1, c_2\}$ *INFEASIBLE*
 $I = \{c_3, c_2\}$
 $C = \{c_1, c_2\}$

Step 2: $I = \{c_3, c_2\}$ is *FEASIBLE*

Step 1: $T = \{c_3, c_2\}$
 $T = \{c_3, c_2, c_1\}$ *INFEASIBLE*
 $I = \{c_3, c_2, c_1\}$
 $C = \{c_1\}$

Step 2: $I = \{c_3, c_2, c_1\}$ is *INFEASIBLE*
STOP : $I = \{c_3, c_2, c_1\}$ is an IIS

Naturally, the performances of the two algorithms are likely to be different even if they lead to the same IIS since different numbers of phase 1 solutions of different LPs are performed. However, no efficiency comparison between deletion filtering and incremental filtering for LPs has been performed.

IIS Set Covers

“Repairing” an IIS in an inconsistent system of linear constraints will not necessarily lead to a feasible overall system. When an infeasible system contains two or more non-overlapping IISs then all have to be “repaired” to reach feasibility. Hence another approach consists of isolating a subset of constraints that, once removed,

leaves the remaining problem feasible. Such subsets have been called *infeasibility subsets* by Chakravarti[94]. Since these subsets contain at least one constraint from every IIS, they are also called *IIS set covers*. Infeasibility subsets with low cardinality appear to be easier to analyse. Chakravarti proved that isolating a minimum-cardinality subset is equivalent to solving a certain MILP, the mixed integer elastic linear program (MIELP). Thus, it is a NP-hard problem. Sankaran[93] has shown that it is NP-hard, even with a totally unimodular coefficient matrix and an integer RHS vector. IIS set cover isolation algorithms are discussed in [Parker and Ryan 94], where constraints are weighted; minimum weight rather than minimum-cardinality IIS set covers are isolated. The algorithms incrementally isolate all the IISs of an inconsistent system. This approach has the drawback of being complex to implement. It also potentially suffers from computational overhead as it requires the solution of possibly many set covering problems. Chinneck[95] developed an easy to implement, polynomial-time heuristic solution for minimum-cardinality IIS set covers.

2.2 Mixed Integer Linear Programming

2.2.1 Preliminary.

Mixed-integer linear programming and all-integer linear programming, a special case of the former, provide a framework for formulating optimization problems such as non-linear, non-convex and combinatorial problems; both will be generically referred to as **MILPs**. Dantzig[60] notes that classical analytical approaches are most likely to fail on MILPs, knocked down by local optima on disconnected parts. A rounding integer approach is also likely to suffer from the same deficiencies [Jensen 68]. The cutting plane approach by Gomory[63] provided the first efficient solution technique for all MILPs and rendered the MILP formulation approach to these optimization problems practical.

2.2.2 Notations and Definitions.

MILPs arise from LPs where some variables are required to be integers. A general expression for a MILP is:

$$\begin{aligned}
 &\text{Maximize} && F(\mathbf{x}) = \sum_{j=1}^n c_j \cdot x_j \\
 &\text{Subject to} && \sum_{j=1}^n a_{ij} \cdot x_j \leq b_i \quad (i = 1, 2, \dots, m) \\
 &&& x_j \geq 0 \quad (j = 1, 2, \dots, n) \\
 &&& x_j \quad \text{integer for } j \in IR, IR \subseteq \{1, 2, \dots, n\}
 \end{aligned}$$

where IR is the set of the integer restrictions or “implicit constraints” [Agin 66]; i.e. the union of the binary restrictions ($x_j \in \{0, 1\}$) and of the general integer restrictions. Throughout the remainder of the thesis, we adopt the following notation “ $\overline{x_i}$ ” for the IR on the variable x_i . When all the variables are restricted to only take integer values, i.e. when $IR = \{1, 2, \dots, n\}$, the problem is an all integer linear programming problem or ILP.

Conceptually a MILP can be thought of as a couple (P, IR) where P is the *relaxed LP*: the LP obtained by relaxing all the integrality constraints in the original MILP and where IR is the initial set of integer restrictions.

Definition: The *integrality gap* is equal to the optimal value of the relaxed problem minus the optimal value of the original MILP.

Barnhart et al.[93] as well as Ashford et al.[92] lay stress on the fact that a *tight* LP relaxation of the MILP, i.e. a small integrality gap, is essential to improve the solvability of the problem.

The feasible region of a MILP is defined similarly to the LP case, i.e. it is the union of all vectors $\mathbf{x} \in \mathbb{R}^n$ associated with a feasible solution. It results from the intersection of the relaxed LP feasible region and the integer hyperplanes defined by the IR s. For instance, if x_1 is a binary variable then these hyperplanes are $\{x_1 = 0\}$ and $\{x_1 = 1\}$. Hence, when all variables are restricted to take integer values only, the MILP feasible region is the set of the integer points contained in the relaxed LP

feasible region. Naturally a MILP is said to be *infeasible* when there exists no vector $\mathbf{x} \in \mathbb{R}^n$ such that:

$$\begin{cases} \mathbf{x} \in \mathcal{F}_{LP} \\ \mathbf{x}_i \text{ integer } \forall i \in I_R \end{cases}$$

or equivalently when $\mathcal{F}_{MILP} = \emptyset$.

2.2.3 MILP Solution Techniques: an Introduction.

Several methods have been designed to cope with ILPs and MILPs; a survey and historical review can be found in [Beale 77]. However, most techniques are based on a similar schema. To solve the MILP $\mathbf{P}_0$, a sequence of auxiliary problems $\mathbf{P}_i$ ($i = 1, \dots, k, \dots$) is generated until a feasible solution of a $\mathbf{P}_k$ also satisfies $\mathbf{P}_0$. In order to achieve this, these problems must be such that:

$$\mathcal{F}_{\mathbf{P}_1} \supseteq \mathcal{F}_{\mathbf{P}_2} \supseteq \mathcal{F}_{\mathbf{P}_3} \dots$$

and for all k , $\mathcal{F}_{\mathbf{P}_0} \subseteq \mathcal{F}_{\mathbf{P}_k}$. An infeasible intermediate $\mathbf{P}_{k_0}$ establishes the inconsistency of the initial $\mathbf{P}_0$. The most common approach uses the LP-relaxation of the MILP $\mathbf{P}_0$ as starting point ($\mathbf{P}_1$) of the series of auxiliary problems.

2.2.4 Cutting Plane Technique.

At each step, the cutting plane method [Gomory 63, Hoffman et al. 85, Wolsey 89] generates one linear constraint, or cut. Then $\mathbf{P}_{k+1}$ is obtained by adding the generated constraint to $\mathbf{P}_k$. By progressively adding constraints, the region of feasibility of the subproblems $\mathbf{P}_k$ is shrunk towards the feasible region of the original MILP until a solution to it is reached, in a finite number of steps. Added constraints must reduce the set of feasible solutions of $\mathbf{P}_k$ without affecting the set of solutions of $\mathbf{P}_0$. An example of a derivation of such a constraint is given in Appendix A.

Infeasibility is proven in the cutting plane approach when the created subproblem $\mathbf{P}_k$ proves infeasible. This in turn implies that the original MILP is infeasible.

2.2.5 Branch and Bound Algorithm.

On the other hand, in the branch and bound approach, each auxiliary LP P_k generates two subproblems P_{2k} and P_{2k+1} : the branch and bound algorithm derives a binary tree structure of subproblems rather than a linear chain as described in Section 2.2.3. This binary tree structure is also referred to as the *B & B tree* (Figure 3). The root of the B & B tree is usually chosen to be the relaxed LP P_1 . An integer variable in P_0 which has a fractional value (α) in P_k is chosen as the *branching variable*. The feasible region associated with the current node is subdivided into two by the introduction of two half-space inequalities ($x_k \leq \lfloor \alpha \rfloor$ and $x_k \geq \lceil \alpha \rceil$). When all integer restrictions specified in P_0 are satisfied in P_k then a feasible solution of P_0 has been reached; the value of the objective function is therefore a lower bound on the solution since P_0 is a maximization problem. As a side effect, this also implies that the original problem P_0 is feasible. On the other hand, if no feasible subproblem P_k is encountered while developing the tree, then the original MILP is infeasible.

Several heuristics have been developed to efficiently solve MILPs by B & B trees [Benichou et al. 71, Breu et al. 74, Forrest et al. 74] Moreover, specific knowledge about the problem or the form of the solution can also be used. Common subproblem selection procedures are best-bound, depth-first searches. Branching variables may be chosen according to their value in the current node or to some estimates of the degradation of the objective function value, when branching on a particular variable. The maximum infeasibility criterion, developed by Breu et al., branches on the variable which maximizes the violation of the LCs and BDs, all other variables being at their current value.

At any intermediate node of the B & B tree [Agin 66], bounding techniques, based on estimates of the objective function value, provide upper bounds on the maximum value of the objective function in any feasible node of the subtree below the current node. This procedure allows the discarding or *pruning* of branches that are not

promising without explicitly developing them, reducing the potentially exponential size of the tree to be explored: this explains the relative efficiency of B & B techniques over complete enumeration schemes[Davis et al. 71].

However this search can be quite time and space consuming for some particular cases as shown below. In fact, the following MILP (P_α) will force the branching process to recurrently branch on the variables x and y until an integer feasible solution is reached; respective values of x and y increase from the LP-relaxation optimum solution by increments of 1. This “climbing” process of the B & B algorithm is illustrated in Figure 4 by the solution of $P_{0.1}$.

$$\begin{array}{ll}
 P_\alpha: & \text{Minimize} & F_\alpha(x,y) = 2x + y \\
 & \text{Subject to} & (1 + \alpha)x - y \geq 1.2 \\
 & & x - y \leq 1.8
 \end{array}$$

x, y non-negative integers

$P_{(\alpha=0)}$ is trivially an inconsistent system. Not surprisingly, the closer to zero α is, the more nodes are required before reaching a feasible solution (76 for $\alpha = 0.01$, 796 for $\alpha = 0.001$, 7996 for $\alpha = 0.0001, \dots$). Moreover, the B & B algorithm can not recognize the infeasibility of P_0 by itself since one branch of the tree does not terminate: branching on $x, y, x, \dots$

Inconsistency is established when all the branches of the associated B & B tree terminate in an infeasible LP. Brearley et al.[75] mentions that this detection can be quite time consuming since it requires the **complete** exploration of the tree. On the other hand, feasibility is established when the first feasible node is generated. Computational costs can therefore be radically different.

2.2.6 Branch and Cut Approach.

As noted by Savelsbergh[94], combinations of the two previous methods have shown their efficiency [Lawler et al. 85, Parker et al. 88, Wolsey 89] on combinatorial optimization problems as well as on large scale binary ILPs. This approach consists of adding cuts to some of the subproblems P_k of the B & B tree to further reduce the integrality gap and thus accelerate the solution process. The speed up can be significant. The derivation of a cut in the previous case, based on the final tableau of the relaxed LP solution [Winston 91 pp 513–514], leads to the following MILP P'_α :

$$\begin{array}{ll}
 P'_\alpha: & \text{Minimize} \quad F_\alpha(x, y) = 2x + y \\
 & \text{Subject to} \quad (1 + \alpha)x - y \geq 1.2 \\
 & \quad \quad \quad x - y \leq 1.8 \\
 & \quad \quad \quad x > \frac{0.2}{\alpha} \quad \text{CUT} \\
 & \quad \quad \quad \mathbf{x, y \text{ non-negative integers}}
 \end{array}$$

The addition of a cutting plane shifts the LP-relaxation solution to the close vicinity of the integer solution. Cutting planes could be introduced when for instance

an identical sequence of IRs is repeatedly branched on along a particular branch. The B & B solution of P'_α requires at most 2 nodes, independently of the value of α .

In case of an infeasible MILP the introduction of a cut leads to an infeasible subproblem and therefore terminates this branch. Thus, the branch and cut approach combines the efficiency of the B & B algorithm with the completeness of the cutting plane technique.

FIGURE 4. Example of a "difficult" MILP

2.2.7 Duality Results for MILPs and Role of the LP Relaxation.

Duality theories for the ILP and MILP cases can be found in [Wolsey 81, Jeroslaw 78,79]. The weak and strong duality theorems can also be stated in these contexts. However, as opposed to the LP case where optimality could easily be checked by using some information provided by the dual problem, neither of these theories provide the same practical interest since the computation of the dual objective function requires first the solution of the whole problem. Moreover, the functions generated heavily depend on the solution technique employed.

In the case of MILPs with rational coefficients, which is the general case from a computational point of view, Meyer[74] (p 234) specifies the relationship between a MILP (M) and the associated relaxed LP (M'') by the following theorem:

“If (M'') is unbounded, then (M) is either unbounded or infeasible. If, on the other hand, (M'') has an optimal solution, then (M) either has an optimal solution or is infeasible.”

It does not however establish sufficient conditions, based on the relaxed LP, for a MILP to be infeasible. Hence, in order to prove a MILP infeasible, one has to “fail” while attempting to solve it.

2.2.8 A Special Case: Linear Diophantine Equations.

Linear Diophantine equations are a special case of ILPs where all linear constraints are equations with integer coefficients. The problem lies here in finding a feasible integer solution to this set of equations. Unlike the ILP case, there is no need to optimize any objective function. Naturally, all results discussed below can be extended to problems with rational coefficients. A few terms need to be defined first [Nemhauser and Wolsey 88, pp. 189–191].

Definition: For all $A \in \mathbb{R}^{m \times n}$, $L(A) = \{y \in \mathbb{R}^m: y = A \cdot x, x \in \mathbb{Z}^n\}$ is the *lattice* generated by the columns of A .

Definition: $C \in \mathbb{Z}^{n \times n}$ is *unimodular* if and only if $|\det C| = 1$.

Definition: $H \in \mathbb{Z}^{m \times m}$ is in *Hermite normal form* if:

- a. $h_{ij} = 0, \forall i < j$
- b. $h_{ii} > 0, i = 1, \dots, m$
- c. $h_{ij} \leq 0$ and $|h_{ij}| < h_{ii}, \forall i > j.$

Solutions to $S = L(A) \cap \{b\}$ for any $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^n$ are given by:

$$x = C_1 \cdot H^{-1} \cdot b + C_2 \cdot z, \quad z \in \mathbb{Z}^{n-m}$$

where $C = (C_1, C_2)$ is unimodular,

and $A \cdot C = (H, 0)$ is in Hermite normal form.

Thus, integer solutions exist if and only if $H^{-1} \cdot b \in \mathbb{Z}^m$, an integer version of Farkas' lemma [Chvátal 83]. Polynomial-time algorithms based on elementary column operations (linear combinations and interchange of columns) on A over the ring of integers, can be used to determine C and H [Bradley 71, Kannan et al. 79], moreover modulo arithmetic operations also guarantee a polynomial upper bound on the magnitude of matrix coefficients [Domich et al. 87]. Appendix B treats an example of linear Diophantine equations based on such algorithms. Contrary to the general MILP case, the *Linear Equation Integer Feasibility Problem* possesses an overall polynomial complexity. However, Nemhauser et al. [88] point out that the construction of the Hermite normal form quickly becomes computationally impractical as the matrix of the coefficients grows.

2.2.9 Sensitivity Analysis.

For sensitivity analysis of MILPs, some extra information must be stored during the solution process. The nature of this additional information depends on the solution technique employed. Once an optimal solution has been determined, one can then, at a lesser cost, solve MILPs that differ from the original one by only relatively small perturbations of some coefficients. Modifications of RHS values are treated in

[Roodman 73, Marsten and Morin 77, Holn and Klein 78,84]; variations of the objective function coefficients are discussed in [Piper and Zoltners 76, Geoffrion and Nauss 77]. In fact, in most cases the information stored during the initial search is such that only a small, if any, amount of the initial computation is required to determine the optimal solution of the perturbed MILP. These methods were primarily designed to solve families of MILPs that share the same constraints but have different RHS values.

CHAPTER 3

STATE OF THE ART IN MILP POST-INFEASIBILITY ANALYSIS

Few methods have been designed for post-optimality analysis of MILPs. Only one method tackles the post-infeasibility case.

Specific bound tightening methods, dealing with integer and binary variables, are described in Savelsbergh[94]. The primary goal of these techniques is to reduce the integrality gap by a reformulation of the original MILP since a “tight” formulation of the MILP, i.e. with an integrality gap as close to zero as possible, is an essential element in a fast and efficient solution [Barnhart et al. 93]. Infeasibility can be detected as a side effect of these reformulation methods when, for instance, the bounds on an integer variable implied by the set of linear constraints and bounds on other variables, do not include any integer value. Then, as in the LP case, backtracking all the reformulation operations may isolate an inconsistent set of linear constraints and integer restrictions. In the case of binary variables, which often appear as decision variables (“If project A is adopted then project B must be postponed”), logical implication chains can be built, some variables may be fixed to 0 or 1 and infeasibility can also sometimes be detected too.

ANALYZE, described in Section 2.1.3, contains these functions for MILPs, embedded in the REDUCE command.

CHAPTER 4

PROBLEM STATEMENT

4.1 Object of the Research

Post-infeasibility analysis of mathematical programs involves a combination of several approaches. Tools derived from artificial intelligence techniques and the modeller's intervention are required to analyze and understand the possible sources of infeasibility. Algorithmic tools can speed up this explanation process by preprocessing the original problem prior to analysis. In most cases, these tools will help the modeller to focus attention on a relatively small subpart of the problem only, which either enables a human intervention or drastically speeds up an automated analysis.

In the LP context these considerations lead to the notion of a small Infeasible Subset (IS): the Irreducible Infeasible Subset (IIS). An IIS contains the minimal, but sufficient, amount of information from the original formulation necessary to correct an inconsistency. Chinneck and Dravnieks[89] provide a framework for efficiently isolating IISs.

The goal of this thesis is to provide efficient methods for isolating ISs and possibly IISs (if the MILP solver is able to decide feasibility in a finite number of nodes in all intermediate steps) in inconsistent MILP problems. The term "efficient" actually includes two different issues. The first is relevant to the speed of the isolation whereas the second raises the question of how useful an IS or IIS is. An IS is most useful when it contains few extraneous constraints and ideally, one would like to get the most useful IS without incurring the cost of enumerating all the possible ones.

4.2 Deficiencies of the State of the Art

Bound tightening methods can be efficient only on relatively “trivial” cases, e.g. when the inconsistency involves only a few constraints or variables bounds. In the case of binary variables, implication chains can be established to take into account relationships among variables spanning a number of constraints, provided these constraints contain binary variables only. However, this approach cannot handle the more general integer variables case because of the potentially exponential number of possible integral values. Moreover, minimality of the findings is not guaranteed.

4.3 Conclusion

Following the increasing solving capabilities for LPs and MILPs [Nemhauser 94] resulting from the conjunction of more and more powerful hardware and software tools, large LPs and MILPs are now common. But while tools have been incorporated into commercial LP solvers to isolate IISs from infeasible LPs, no similar systematic approach has been developed to facilitate the analysis of inconsistent MILPs.

Hence, adapting and developing techniques to isolate minimal infeasible substructures from inconsistent MILPs is a worthwhile undertaking.

CHAPTER 5

PROBLEM CHARACTERIZATION

5.1 Introduction

All definitions given in Section 2.1 concerning IISs in a continuous context can, without any modification, be extended to qualify IISs in the discrete, MILP case. The only difference is that the definition of a “constraint” now includes integer restrictions along with linear constraints and variable bounds.

However, a thorough study of the characteristics of IISs in the specific case of MILPs is required in order to gain insights into the structure of IISs and to understand the relationships between distinct IISs of the same infeasible MILP. This knowledge is necessary in order to develop efficient IIS-isolation algorithms.

5.2 Desirable Features of an IIS

IISs in MILPs may contain integer restrictions (**IRs**), linear constraints (**LCs**) and bounds on variables (**BDs**). Chinneck[93b,94] argues that, in the LP case, IISs with few LCs are easier to diagnose. In the MILP case, IISs having no LCs, only IRs and BDs, are possible. For example, a positive integer variable may be forced to be non-positive by the BDs. IISs consisting solely of IRs and BDs are likely to be easily discovered by bound-tightening presolvers. On the other hand, similar to the LP case, BDs on real-valued variables seem to have a lesser explanatory ability. Thus IISs with few IRs and few LCs provide better candidates for further analysis.

5.3 Properties

5.3.1 Assumption.

FIGURE 5. Examples of IISs in ILPs

As mentioned in section Section 2.2.2, the feasible region of a MILP ($\mathcal{F}_{MILP}$) is the union of all the vectors $\mathbf{x} \in \mathcal{F}_{LP}$ satisfying the integrality requirement on variable x_i , written $\overline{x_i}$, for all $i \in IR$. Thus, if the relaxed LP is infeasible, i.e. $\mathcal{F}_{LP} = \emptyset$, then the original MILP is also infeasible, since $\mathcal{F}_{MILP} \subseteq \mathcal{F}_{LP}$. In this case, prior to any treatment of the initial MILP it is necessary to analyze and correct the initial LP-relaxation. This can be done using well-established techniques for isolating IISs

in infeasible LPs.

This thesis focuses on inconsistent MILPs with feasible initial LP-relaxations only.

5.3.2 On the “Continuous” Part of IISs.

Property 1: An IIS may contain some LCs and BDs redundant in the LP-relaxation.

Proof: We exhibit an example of such a case in Figure 5A where integrality requirements are borne by the variables x and y . Constraint **D** is clearly strongly redundant in the LP-relaxation since it does not intersect the feasible region (light grey shaded area) at all, yet it appears in the IIS $\{A, B, D, \underline{y}\}$. $\square$

Property 2: In an infeasible MILP, the subsystem made of the necessary LCs and BDs of the relaxed LP, plus the original IRs contains at least one IIS.

Proof: Let $P_0 = (LP_0, IR_0)$ be any infeasible MILP; LP_0 is feasible by assumption. Let LP_1 stand for the LP made of the necessary constraints of LP_0 only. Then $P_1 = (LP_1, IR_0)$ is also inconsistent since $\mathcal{F}_{LP_1} \stackrel{Def}{=} \mathcal{F}_{LP_0} = \emptyset$. Thus P_1 contains at least one IIS. (cf Figure 5A where $(LP_1, \underline{x})$ is an IIS and so is $(LP_1, \underline{y})$) $\square$

This does not imply that an infeasible MILP contains one single cluster of IISs. For instance, the simple inconsistent MILP pictured in Figure 5B possesses two non-overlapping IISs $\{A, B, \underline{x}\}$ and $\{C, D, \underline{y}\}$. However in real-life formulations, such a case is unlikely to happen and in most cases IISs share some linear constraints, LCs or BDs.

5.3.3 Maximum Cardinality of an IIS.

Chvátal [83, pp. 145–146] notes that any IIS in an inconsistent LP of n variables has at most $(n + 1)$ constraints. This upper limit on LCs and BDs does not hold for MILPs (cf Figure 6 where 4 LCs and 2 IRs belong to the IIS), though it is possible that a different upper limit may exist.

FIGURE 6. An IIS with $(n + 2)$ Linear Constraints

5.3.4 Number of IISs.

We establish in this section an analog to Chakravarti's theorem[94] stating that an infeasible LP may possess an exponential number of IISs.

Theorem 1: An infeasible MILP may possess exponentially many IISs.

Proof: We define a set of infeasible MILPs satisfying the exponentiality criteria. Let $P_{n,k,\lambda}$ be the following MILP:

$$\begin{aligned}
 &\text{Maximize} && F(\mathbf{x}) = \sum_{i=1}^n c_i \cdot x_i \\
 &\text{Subject to} && \sum_{i=1}^n x_i \leq \frac{k\lambda + n}{\lambda} \\
 &&& x_i \geq \frac{1}{\lambda} \\
 &&& x_i \geq 0 \\
 &&& x_i \text{ integer} \quad (i = 1, 2, \dots, n)
 \end{aligned}$$

$P_{n,k,\lambda}$ is clearly infeasible for $\lambda > n > k > 0$ since the n lower bounds on the x_i ($x_i \geq \frac{1}{\lambda}$) combined with the integer restrictions “push” the sum of the x_i to n . However, if all integer restrictions ($\underline{x}_1, \dots, \underline{x}_n$) are relaxed, then the x_i can be chosen so that their sum satisfies the first constraint since $\frac{k\lambda + n}{\lambda} > \frac{n}{\lambda}$. Moreover, $P_{n,1,\lambda}$ contains at least n (i.e. $\binom{n}{1}$) IISs, each given by a different i_0 :

$$IIS(i_0) \quad \left\{ \begin{array}{l} \sum_{i=1}^n x_i \leq 1 + \frac{n}{\lambda} \\ x_{i_0} \geq \frac{1}{\lambda} \\ x_i \geq 0 \quad (i = 1, 2, \dots, i_0 - 1, i_0 + 1, \dots, n) \\ x_{i_0} \quad \text{integer} \end{array} \right.$$

Similarly, for any integer k less than n , i.e. $k \in (0, n]$, $P_{n,k,\lambda}$ contains $\binom{n}{k}$ IISs. In this case, IISs contain k IRs as well as k lower bounds ($x_i \geq \frac{1}{\lambda}$). Thus, the maximum number of IISs is attained for $k = \lfloor \frac{n}{2} \rfloor$. Using Stirling's approximation for the Γ function,

$$x! = \Gamma(x+1) \simeq x^x e^{-x} \sqrt{2\pi x}, \quad \text{for } x \in \mathcal{N}$$

we can compute its maximum asymptotic value for an even $n = 2k$. Since $\binom{n}{\lfloor \frac{n}{2} \rfloor} = \frac{(2k)!}{k!^2}$ is of order

$$O\left(\frac{2^n}{\sqrt{\pi n/2}}\right),$$

we can conclude that $P_{n, \lfloor \frac{n}{2} \rfloor, \lambda}$ contains an exponential number of IISs. $\square$

5.4 On the B & B Tree

5.4.1 Introduction.

At every node of the B & B tree, several questions have to be answered. Is it worth developing the subtree beneath the current node (bounding and fathoming)? If the answer is yes, then which variable should be branched on and in which direction

first (variable selection)? If it is no, which subproblem should be developed next (node selection)? A vast literature discusses the different strategies to efficiently solve a MILP by a B & B tree [Benichou et al. 71, Breu et al. 74, Forrest et al. 74].

Different B & B strategies are likely to generate different trees. Thus, for an infeasible MILP, several proofs of its inconsistency may exist, furthermore each one of these trees may implicate a distinct IIS (the relationship between a B & B tree and an IIS is described below). Figure 7 illustrates two different binary trees associated with the following inconsistent MILP, which has 2 IISs: $\{C_1, C_2, C_3, \bar{x}\}$ and $\{C_1, C_2, C_3, \bar{y}\}$:

$$\begin{array}{ll}
 \text{Maximize} & F(x, y) = x + 2y \\
 \text{Subject to} & (1) \quad -x + y \leq 0.1 \\
 & (2) \quad x + y \geq 2.6 \\
 & (3) \quad x \leq 1.5 \\
 & x, y \quad \text{integers.}
 \end{array}$$

The reverse property is conjectured only:

Conjecture 1: Any IIS of an infeasible MILP can be obtained from at least one B & B tree.

To prove that a MILP is infeasible, the complete B & B tree has to be developed and all its branches have to terminate on an infeasible subproblem. More than just a proof of inconsistency, one can also use the information in the B & B tree for post-infeasibility analysis.

5.4.2 Node and Leaf Information.

Definition: A *leaf* or *final node* [Agin 66] of a B & B tree or a binary tree obtained by a Branch and Cut algorithm, is either a node where all the IRs are satisfied or a node whose associated subproblem is infeasible.

Definition: An intermediate node of a B & B tree is any node that is not a leaf or final node.

For any intermediate node (N) of the binary tree derived from P , define $IR_N = \{x, y, \dots\}$ as the set of all satisfied IRs.

Property 3: The infeasible MILP P does not contain any IIS whose integer part is exactly identical to some IR_N .

Proof: For any infeasible $P = (LP, IR)$, (LP, IR_N) is feasible by definition. Thus, any (LP', IR_N) , where $LP' \subseteq LP$, is also feasible and can not therefore be an IIS.

□

If, at every leaf of the B & B tree, one marks all LCs and BDs from the original MILP which are effectively involved in the inconsistency of the LP subproblem by using sensitivity filtering, then the following property holds:

Property 4: Once the binary tree has been completed, no unmarked LC or BD belongs to any IIS associated with that B & B tree.

Proof: Let lc (bd) be an unmarked LC (BD) and (IR_0, LC_0, BD_0) any IIS associated to the B & B tree. Since lc (bd) is not marked, it is not tight in any of the leaf nodes. Thus, if we relax lc (bd) in the original MILP, one can develop the same binary tree. Therefore, one can conclude that $lc \notin LC_0$ ($bd \notin BD_0$). $\square$

5.4.3 Branch Information.

Definition: A *path* in a B & B tree is a set of branches, from the root to a leaf, where each branch is labelled by the name of the variable branched on.

Definition: The *set of active IRs* ($\mathcal{A}_T$) in a B & B tree ($\mathbf{T}$) is the union of all the IRs in all the paths associated to that tree.

For instance, the tree developed in figure 3 contains 2 distinct paths: $\{\underline{x}\}$ and $\{\underline{x}, \underline{y}\}$. The active IRs are $\mathcal{A}_T = \{\underline{x}, \underline{y}\}$.

Property 5: For any infeasible MILP $P_0 = (LC, BD, IR)$, with a set of active IRs $\mathcal{A}_T$, then $P' = (LC, BD, \mathcal{A}_T)$ is also inconsistent.

Proof: Consider the following MILP $P' = (LC, BD, \mathcal{A}_T)$. One can then derive the same B & B tree $\mathbf{T}$ and conclude that P' is infeasible as well. $\square$

Theorem 2: For any infeasible MILP $P_0 = (LC, BD, IR)$, at least one of the MILPs (LC, BD, IR_k) , $(LC, BD, \mathcal{A}_T)$, where IR_k is the set of integer restrictions for the branch label variables in a path derived from the B & B tree $\mathbf{T}$, is infeasible. $\square$

Proof: The justification of the theorem is based on the preceding property.

Remark: Practically this theorem can assist in the identification of Infeasible Sets (ISs) with few IRs: (LC, BD, IR_k) sets are tested from the smallest one to the largest, the process is stopped as soon as one (LC, BD, IR_k) proves inconsistent; if all of them are feasible then, based on the preceding property, $(LC, BD, \mathcal{A}_T)$ is inconsistent.

5.4.4 Modification of the Objective Function.

The first feasible leaf of the B & B tree reveals that the original MILP is feasible, thus other branches of the tree do not have to be explored when the goal is merely to decide the feasibility of the MILP. On the other hand, one has to develop the whole tree to prove that a MILP is infeasible. In both circumstances, the original objective function does not play any role and can therefore be replaced by an objective function that is more efficient in deciding feasibility. Moreover, this objective function can be changed before processing any new branch since it does not affect the feasibility of the MILP.

Traditionally, constraints of the form $x_k \leq \lfloor \alpha_k \rfloor$, where $\lfloor \alpha_k \rfloor$ is the largest integer smaller than α_k , (respectively $x_k \geq \lceil \alpha_k \rceil$) are introduced on left (resp. right) branches on the variable x_k . One can also use “normal form”-type constraints, of the form $x_k + s_{k_0} = \lfloor \alpha_k \rfloor$ (resp. $x_k - s_{k_1} = \lceil \alpha_k \rceil$) where s_{k_0} (resp. s_{k_1}) is a positive slack variable, without changing the search. The objective function can then be modified to take into account these slack variables as they are introduced and to ensure that the x_k 's stretch to the possible integer values $\lfloor \alpha_k \rfloor$ (resp. $\lceil \alpha_k \rceil$) whenever feasible by minimizing $F' = \sum_{k \geq 1} s_k$.

5.4.5 Sensitivity Analysis for IRs.

Some variables which are not involved in any IIS at all may have been “inadvertently” branched on in the B & B search. To identify them one can perform some sensitivity analysis on the infeasible leaves. If the current subproblem is inconsistent independently of the RHS values of the constraints introduced when branching on a particular variable, then the integer restriction on this variable is not likely to play any role in the infeasibility and is removed from the path IR_k leading to the current inconsistent subproblem. Such variables can be identified by sensitivity filtering [Dravnieks 89].

We can therefore state the following corollary:

Corollary 1: For any infeasible MILP $P_0 = (LC, BD, IR)$, at least one of the MILPs (LC, BD, IR'_k) , $(LC, BD, \mathcal{A}_T)$, where IR'_k is the sensitivity-filtered path derived from the B & B tree $\mathbf{T}$, i.e. the set of IRs involved in the sensitivity-filtered leaf LP, is infeasible.

CHAPTER 6

EXPERIMENTAL SETUP

A series of numerical experiments support the development of the algorithms presented in chapters 7 through 11. The data sets used during the experiments are described below, followed by remarks on the software prototype and the format of the tables of results.

6.1 Test Set Description

The inconsistent test problems used to benchmark the different algorithms and approaches cover a wide range of sizes and difficulties. They were derived from a library of feasible datasets: the *miplib* depository available by anonymous ftp at “softlib.cs.rice.edu” and maintained by Dr. Robert E. Bixby and Dr. E. Andrew Boyd, contains a set of real-world MILPs in MPS format, an industry standard for encoding linear programs. Excepting “dell.mps” and “p0201”, all infeasible problems were generated by adding an extra LC which preserves the feasibility of the LP-relaxation but renders the integer formulation infeasible. For instance, in a minimization problem, an upper bound on the objective function, with a RHS value within the LP-solution to MILP-solution range, satisfies this criterion. The “dell” infeasible MILP was provided by Robert F. Dell, Assistant Professor of Operations Research at the Naval Postgraduate School in Monterey CA. This MILP suffers from scaling difficulties and the detection of inconsistency requires the use of “aggressive” scaling preprocessing techniques. Ed Klotz, of CPLEX Optimization Inc., provided “p0201”.

Table 1 summarizes the characteristics of the infeasible MILPs: number of linear constraints (LCs), number of columns, number of non-zeros in the coefficient matrix,

number of bounds (BDs) on the variables (0-1 restrictions on integer variables count as 2 BDs), number of general integer restrictions and binary variables.

Forrest et al.[74] classify MILP formulations into two distinct groups: MILPs with a "small number (about 100) of 0-1 integer variables embedded in a very large and difficult linear program" and MILPs with general integer variables involved in a relatively simpler linear program. According to this classification, the first five problems belong to the latter whereas the others belong to the former class.

TABLE 1. Test Data Sets

No.	Problem	LCs	Variables	Nonzeros	BDs	Gen. IRs	0-1 IRs
1	bell3a	124	133	441	204	32	39
2	bell3b	124	133	441	204	32	39
3	bell4	106	117	385	181	30	34
4	bell5	92	104	340	162	28	35
5	dell	500	626	4580	825	6	99
6	flugpl	19	18	64	29	11	0
7	g503	41	48	144	72	0	24
8	misc01	55	83	746	164	0	82
9	misc02	40	59	414	116	0	58
10	misc03	97	160	2054	318	0	159
11	misc04	1726	4897	17253	4691	0	30
12	misc05	301	136	2946	204	0	74
13	misc06	821	1808	5860	1894	0	112
14	mod008	7	319	1562	638	0	319
15	mod013	63	96	288	144	0	48
16	p0201	134	201	1925	402	0	201
17	stein9	14	9	54	18	0	9
18	stein15	37	15	135	30	0	15
19	stein27	119	27	405	54	0	27
20	stein45	332	45	1079	9	0	45

Table 2 lists the times and number of nodes of the B & B trees and number of LP iterations (pivots) required by CPLEX 3.0, a MILP solver, to prove each model infeasible. Times were measured on a SPARC sun4 workstation with 32MB of main memory and running at 36 Mhz. The default settings of the CPLEX MILP solver were used, except for the "dell" formulation where variables were rescaled.

TABLE 2. Infeasibility Proofs

Problem	B & B Tree Nodes	LP Iterations	Sol. Time (secs.)
bell3a	12	144	0.23
bell3b	2	103	0.17
bell4	0	105	0.15
bell5	2	108	0.13
dell	6	767	3.48
flugpl	5012	3624	11.17
g503	0	45	0.05
misc01	28	410	0.95
misc02	54	501	0.55
misc03	8	299	0.80
misc04	2	2094	12.17
misc05	52	1118	3.42
misc06	38	3429	20.47
mod008	956	3949	15.42
mod013	124	490	0.79
p0201 ¹	-	-	0.03
stein9	12	32	0.07
stein15	92	340	0.40
stein27	1218	5517	10.98
stein45	1112	11298	39.05

¹ Solved by preprocessing.

6.2 Software Prototype

The Deletion, Incremental and Incremental/Deletion Filtering algorithms are written in C and use the CPLEX 3.0 callable library. A B & B algorithm is used to decide the feasibility of the test sets. Experiments on the B & B Tree Filtering approach were performed by modifying the *Mixed INTEGER Optimizer* (MINTO) 2.0 package [Nemhauser et al. 94] which is also based on the LP solver CPLEX.

6.3 Presentation of Experimental Results

Every table uses the same format. For each isolation, the characteristics of the outcome (number of IRs, LCs and BDs, where BDs include the 0-1 boundaries for binary variables) are listed as well as the number of nodes and LP iterations required

to complete the isolation and to prove the original model infeasible and, finally, the overall isolation time. As mentioned in Section 2.2.5, a B & B algorithm may not be able to determine whether a MILP is feasible or not, instead branching infinitely. So, to avoid system crashes, the maximum number of nodes in any B & B tree is limited to 10,000. Thus, some isolations may provide *infeasible subsets* only, with no guaranty of minimality of these subsets. When available, the number of dubious constraints (IRs, LCs or BDs), i.e. constraints whose removal causes the failure of the MILP solver to decide feasibility in a finite number of nodes, are listed in the tables in brackets.

The total number of nodes is a good measure of algorithm performance because it does not depend on the efficiency of the algorithm implementation, only on the algorithm itself and the MILP solver performance. Since the same MILP solver is used in all of the experiments this factor can be ignored. Hence the total number of nodes is the best available measure of algorithm performance. Ranking tables provide comparisons of the different techniques according to three different criteria: quality of the isolation (i.e. when all isolated ISs contain the same IIS, this criterion favors ISs with lower cardinality in IRs then LCs and finally BDs), total number of nodes and overall time required to complete the isolation. In the case of ties, performances rather than algorithms are ranked, i.e. if two algorithms lead to the best output, ranks will be "1,1,2,..." instead of "1,1,3,...". Average performance for each approach, according to each criterion, are also given.

CHAPTER 7

DELETION FILTERING ALGORITHMS FOR SMALL MILPS

7.1 Overview

The purpose of the deletion filtering algorithm is to identify a single IIS in an infeasible MILP or ILP. It operates in the same manner as for ordinary LPs. Constraints are sequentially “filtered”, i.e. constraints that do not belong to the IIS being identified currently are removed, one at a time, and the current test set is modified accordingly. As in the continuous case, when the inconsistent MILP contains several IISs, the ordering of the constraints determines the isolated IIS: the IIS whose first constraint is checked last is output.

Feasibility is determined by solving the current MILP until the first feasible solution is found, or until it is proven that no feasible solution exists. However, as mentioned in Section 2.2.5, a pure B & B algorithm may not be able to determine whether a MILP is feasible or not. In case of “doubt”, the constraint currently being tested is kept in the IS; minimality of the output is therefore not guaranteed.

7.2 Algorithms

According to Agin[66], MILPs contain two classes of constraints: IRs are the implicit constraints whereas LCs and BDs are the explicit ones. It is natural to take this classification into account when designing IIS isolation algorithms. Two distinct algorithms derived from Dravnieks’[89] deletion filter for LPs are discussed below. The first one takes advantage of well-formulated MILPs having small integrality gaps by deleting IRs first. The second algorithm deletes LCs first and then discards variables which have been eliminated from the model, along with any associated IRs.

In both cases, BDs are deleted last for computational reasons. In fact, if two MILPs share the same IR and LC sets, it is likely that the MILP with more (or tighter) BDs requires less computation to prove infeasible.

Both algorithms are initialized in a similar fashion:

Initialization:

Let $P_0 = (IR_0, LP_0)$ where LP_0 is the LP-relaxation of P_0 ,

i.e. $LP_0 = \{LC_0, BD_0\}$

Let $T = IR_0 \cup LC_0 \cup BD_0$ be the set of ALL the constraints

7.2.1 Deleting IRs First.

Once IRs and LCs have been filtered, some variables may have been entirely eliminated from the original infeasible MILP. Thus, only BDs on variables which still appear in the current problem need to be considered (Algorithm 3).

Input	:	An infeasible MILP or ILP P_0
Initialization		
Step 0	:	IF LP_0 inconsistent GOTO Step 2
Step 1	:	FOR EACH $ir_k \in IR_0$ DO IF $T \setminus \{ir_k\}$ inconsistent SET $T := T \setminus \{ir_k\}$
Step 2	:	FOR EACH $lc_k \in LC_0$ DO IF $T \setminus \{lc_k\}$ inconsistent SET $T := T \setminus \{lc_k\}$
Step 3	:	a. SET $BD_1 := BD_0 \setminus \{BDs \text{ on variables not in } lc \in T\}$ b. SET $T := (T \setminus BD_0) \cup BD_1$ c. FOR EACH $bd_k \in BD_1$ DO IF $T \setminus \{bd_k\}$ inconsistent SET $T := T \setminus \{bd_k\}$
Step 4	:	STOP: T is an IIS

ALGORITHM 3. (IR-LC-BD) Deletion Filter Algorithm for Isolating IISs

7.2.2 Deleting LCs First.

When deleting LCs, as previously mentioned, some variables may be eliminated from the current MILP. Thus, when LCs are deleted first, after the completion of Step 0, it is useless to consider IRs and BDs that are no longer in the current formulation (Algorithm 4).

Input : An *infeasible* MILP or ILP P_0

Initialization

Step 0 : FOR EACH $lc_k \in LC_0$ DO

IF $T \setminus \{lc_k\}$ *inconsistent* SET $T := T \setminus \{lc_k\}$

Step 1 : a. SET $IR_1 := IR_0 \setminus \{ \text{IRs on variables not in } lc \in T \}$

b. SET $T := (T \setminus IR_0) \cup IR_1$

c. FOR EACH $ir_k \in IR_1$ DO

IF $T \setminus \{ir_k\}$ *inconsistent* SET $T := T \setminus \{ir_k\}$

Step 2 : a. SET $BD_1 := BD_0 \setminus \{ \text{BDs on variables not in } lc \in T \}$

b. SET $T := (T \setminus BD_0) \cup BD_1$

c. FOR EACH $bd_k \in BD_1$ DO

IF $T \setminus \{bd_k\}$ *inconsistent* SET $T := T \setminus \{bd_k\}$

Step 3 : **STOP**: T is an **IIS**

ALGORITHM 4. (LC-IR-BD) Deletion Filter Algorithm for Isolating IISs

7.3 Validity of the Isolations

Theorem 3: Assuming feasibility is always decided correctly, both deletion filtering algorithms lead to the identification of a single IIS in an inconsistent MILP.

Proof: Since both algorithms differ by the ordering of the different steps only, we only need to prove that Algorithm 3 isolates IISs to prove that they both do.

The proof proceeds in two steps. Firstly, we can remark that constraints, IRs or LCs or BDs, are discarded if and only if their removal leaves the test set T infeasible.

Thus, the output of the algorithm, T , is also inconsistent.

Secondly, the subset isolated is minimal. We prove this assertion by assuming that one extra constraint, say the IR on variable x_i ($\underline{x_i}$), is wrongly identified as part of T . Hence, $T \setminus \{\underline{x_i}\}$ is also inconsistent as well as any set of constraints T' such that $T \setminus \{\underline{x_i}\} \subseteq T'$. This would have caused the removal of $\underline{x_i}$ in Step 1 and therefore leads to a contradiction.

The output T of Algorithm 3 is an minimal infeasible subset of the original constraints: T is an IIS. $\square$

7.4 Time and Space Complexities

Both deletion filtering algorithms require the same number of MILP solutions in the worst case: one for each constraint of the original MILP (either IR or LC or BD). However, as remarked earlier the cost of solving a MILP may vary greatly, depending on the number of IRs and on the value of the integrality gap. The cost of solving the test MILP tends to decrease as IRs are filtered out. On the other hand, when filtering LCs and BDs, the cost tends to increase as constraints are discarded from the temporary MILP $\mathbf{T}$ since the integrality gap of $\mathbf{T}$ is likely to increase.

The implementation of a deletion filter requires little memory apart from the space required for the MILP solver. In the case of infinite branches in B & B trees, an upper limit on the maximum size of the binary tree is fixed so that the solver does not consume all available memory.

7.5 Example

7.5.1 Example Description.

Let P be the following infeasible ILP:

$$\begin{array}{ll} \mathbf{P:} & \text{Minimize} \quad x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\ & \text{Subject To} \quad C_1 : \quad x_1 + x_4 \geq 1 \\ & \quad \quad \quad C_2 : \quad x_2 + x_5 \geq 1 \end{array}$$

$$\begin{aligned}
C_3 : \quad & x_3 + x_6 \geq 1 \\
C_4 : \quad & x_1 + x_5 + x_6 \geq 1 \\
C_5 : \quad & x_2 + x_4 + x_6 \geq 1 \\
C_6 : \quad & x_3 + x_4 + x_5 \geq 1 \\
C_7 : \quad & x_1 + x_2 + x_3 \leq 1.5 \\
C_8 : \quad & x_4 + x_5 + x_6 \leq 1.5
\end{aligned}$$

With $x_1, x_2, x_3, x_4, x_5, x_6$ non-negative integers

7.5.2 Algorithm 3: Deleting IRs First.

- Step 0:** LP_0 is *FEASIBLE*
- Step 1:** $T \setminus \{\underline{x}_1\}$ *INFEASIBLE*; $T := T \setminus \{\underline{x}_1\}$
- Step 1:** $T \setminus \{\underline{x}_2\}$ *INFEASIBLE*; $T := T \setminus \{\underline{x}_2\}$
- Step 1:** $T \setminus \{\underline{x}_3\}$ *INFEASIBLE*; $T := T \setminus \{\underline{x}_3\}$
- Step 1:** $T \setminus \{\underline{x}_4\}$ *INFEASIBLE*; $T := T \setminus \{\underline{x}_4\}$
- Step 1:** $T \setminus \{\underline{x}_5\}$ *FEASIBLE*
- Step 1:** $T \setminus \{\underline{x}_6\}$ *FEASIBLE*
- Step 2:** $T \setminus \{C_1\}$ *FEASIBLE*
- Step 2:** $T \setminus \{C_2\}$ *FEASIBLE*
- Step 2:** $T \setminus \{C_3\}$ *FEASIBLE*
- Step 2:** $T \setminus \{C_4\}$ *INFEASIBLE*; $T := T \setminus \{C_4\}$
- Step 2:** $T \setminus \{C_5\}$ *FEASIBLE*
- Step 2:** $T \setminus \{C_6\}$ *FEASIBLE*
- Step 2:** $T \setminus \{C_7\}$ *FEASIBLE*
- Step 2:** $T \setminus \{C_8\}$ *FEASIBLE*
- Step 3:** a. $BD_1 = \{x_1 \geq 0, \dots, x_6 \geq 0\}$
- Step 3:** b. T not modified
- Step 3:** c. $T \setminus \{x_1 \geq 0\}$ *FEASIBLE*
- Step 3:** c. $T \setminus \{x_2 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_2 \geq 0\}$
- Step 3:** c. $T \setminus \{x_3 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_3 \geq 0\}$
- Step 3:** c. $T \setminus \{x_4 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_4 \geq 0\}$
- Step 3:** c. $T \setminus \{x_5 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_5 \geq 0\}$
- Step 3:** c. $T \setminus \{x_6 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_6 \geq 0\}$
- Step 4:** STOP: $T = \{\underline{x}_5, \underline{x}_6, C_1, C_2, C_3, C_5, C_6, C_7, C_8, x_1 \geq 0\}$ is an IIS

The IIS isolated contains:

- 2 IRs out of 6 IRs in the original formulation

- 7 LCs out of 8 LCs in the original formulation
- 1 BD out of 6 BDs in the original formulation

Algorithm 3 requires 11 proofs of feasibility and 10 of inconsistency.

7.5.3 Algorithm 4: Deleting LCs First.

Step 0: $T \setminus \{C_1\}$ *INFEASIBLE*; $T := T \setminus \{C_1\}$
Step 0: $T \setminus \{C_2\}$ *FEASIBLE*
Step 0: $T \setminus \{C_3\}$ *FEASIBLE*
Step 0: $T \setminus \{C_4\}$ *INFEASIBLE*; $T := T \setminus \{C_4\}$
Step 0: $T \setminus \{C_5\}$ *FEASIBLE*
Step 0: $T \setminus \{C_6\}$ *FEASIBLE*
Step 0: $T \setminus \{C_7\}$ *FEASIBLE*
Step 0: $T \setminus \{C_8\}$ *FEASIBLE*

Step 1: a. $IR_1 = \{\underline{x}_1, \dots, \underline{x}_6\}$
Step 1: b. T not modified
Step 1: c. $T \setminus \{\underline{x}_1\}$ *INFEASIBLE*; $T := T \setminus \{\underline{x}_1\}$
Step 1: c. $T \setminus \{\underline{x}_2\}$ *INFEASIBLE*; $T := T \setminus \{\underline{x}_2\}$
Step 1: c. $T \setminus \{\underline{x}_3\}$ *INFEASIBLE*; $T := T \setminus \{\underline{x}_3\}$
Step 1: c. $T \setminus \{\underline{x}_4\}$ *FEASIBLE*
Step 1: c. $T \setminus \{\underline{x}_5\}$ *FEASIBLE*
Step 1: c. $T \setminus \{\underline{x}_6\}$ *FEASIBLE*

Step 2: a. $BD_1 = \{x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 2: b. T not modified
Step 2: c. $T \setminus \{x_1 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_1 \geq 0\}$
Step 2: c. $T \setminus \{x_2 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_2 \geq 0\}$
Step 2: c. $T \setminus \{x_3 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_3 \geq 0\}$
Step 2: c. $T \setminus \{x_4 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_4 \geq 0\}$
Step 2: c. $T \setminus \{x_5 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_5 \geq 0\}$
Step 2: c. $T \setminus \{x_6 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_6 \geq 0\}$

Step 3: STOP : $T = \{\underline{x}_4, \underline{x}_5, \underline{x}_6, C_2, C_3, C_5, C_6, C_7, C_8\}$ is an IIS

The IIS isolated contains:

- 3 IRs out of 6 IRs in the original formulation
- 6 LCs out of 8 LCs in the original formulation
- 0 BD out of 6 BDs in the original formulation

Algorithm 4 requires 9 proofs of feasibility and 11 of inconsistency.

7.6 Computational Results

TABLE 3. Results for (IR-LC-BD) Deletion Filter (Algorithm 3)

- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	7284	44847	36	190	1:55
* bell3b	8	75	77(48)	1325018	2307950	2	127	2:18:33
* bell4	15	65	43(14)	1349744	1455399	2	110	2:01:55
* bell5	6	53	78(25)	1010942	2105135	2	108	1:39:32
dell	5	381	764	7512	1133866	6	767	1:35:03
* flugpl	6	6	6(6)	414413	241286	5712	4248	11:45
* g503	1	33	15(4)	213	6180	0	45	6
* misc01	12	28(1)	65(25)	4929	41813	28	242	59
* misc02	14	30(1)	43(7)	6610	45758	34	326	55
* misc03	3	31(1)	110(40)	2982	53854	6	214	2:01
* misc04	3	1186(294)	3731(2477)	20385	8199496	2	2392	19:41:38
* misc05	7	62(4)	55(32)	9486	188309	52	1007	7:57
* misc06	11	613(26)	1017(684)	44277	3201026	58	3774	6:28:52
* mod008	73	7	313(292)	583635	2056161	926	3832	1:56:59
* mod013	15	50	50(21)	16889	74834	130	681	1:44
* p0201	1	2	6(5)	911440	6338045	224	2463	5:04:14
stein9	6	13	0	1109	2217	44	86	3
stein15	6	36	0	3937	16185	92	340	19
stein27	14	109	0	139602	625873	1218	5517	19:43
stein45	16	227	4	1035530	10126733	1112	11298	7:40:27

* IS(not necessarily IIS) isolated.

Tables 3 and 5 provide a comparison of the two approaches, (IR-LC-BD) and (LC-IR-BD) Deletion Filters, based on the test set of infeasible MILPs described in Section 6.1. Comparative rankings, when identical IISs are identified by Algorithms 3 and 4, are listed in Table 5.

Only a few IISs are identified in Table 3. Large numbers of dubious BDs indicate that infinite bounds on variables challenge the ability of the MILP solver to

Input : An *infeasible* MILP or ILP P_0
 A "Safety BD" function: $Safety_{BD}(bd_k) = bd'_k$

Initialization

Step 0 : IF LP_0 *inconsistent* GOTO **Step 2**

Step 1 : FOR EACH $ir_k \in IR_0$ DO
 IF $T \setminus \{ir_k\}$ *inconsistent* SET $T := T \setminus \{ir_k\}$

Step 2 : FOR EACH $lc_k \in LC_0$ DO
 IF $T \setminus \{lc_k\}$ *inconsistent* SET $T := T \setminus \{lc_k\}$

Step 3 : a. SET $BD_1 := BD_0 \setminus \{BDs \text{ on variables not in } lc \in T\}$
 b. SET $T := (T \setminus BD_0) \cup BD_1$
 c. FOR EACH $bd_k \in BD_1$ DO
 IF $(T \setminus \{bd_k\}) \cup Safety_{BD}(bd_k)$ *inconsistent* THEN
 SET $T := (T \setminus \{bd_k\}) \cup Safety_{BD}(bd_k)$

Step 4 : **STOP**: T is an **IIS**

ALGORITHM 5. "Safety BDs" (**IR-LC-BD**) Deletion Filter Algorithm

decide feasibility in a reasonable number of nodes, that is below the 10,000 limit. Thus, keeping finite bounds or "safety bounds" on the variables throughout deletion filtering, i.e. preserving a small integrality gap, allows the MILP solver to decide feasibility status correctly. These safety BDs can be either set by the modeller or automatically computed as a function of the BDs. Table 4 lists the performance of a slightly modified Algorithm 3 where BDs are not completely relaxed when filtered (Algorithm 5): lower bounds are for instance lowered instead by an arbitrary 100 units and upper bounds increased in a similar fashion. Also, the objective function of each model has been set equal to the sum of the integer restricted variables.

7.7 Discussion

A B & B algorithm's inability to decide feasibility in a finite number of nodes

TABLE 4. Results for (IR-LC-BD) Deletion Filter (Algorithm 5)

- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68(1)	39	24229	180017	272	1824	6:08
bell3b	1	66	19	168107	1311117	50	512	46:56
bell4	19	69	16	2099584	2623663	2	123	3:12:02
bell5	1	52	22	520739	1747975	10000	23422	1:16:36
dell	5	199	350	13896	978103	3	639	58:21
flugpl	2	2	6	460800	278300	4296	3794	12:54
g503	1	33	9	235	8645	2	57	8
misc01	12	28	44	5482	81428	22	563	1:51
misc02	14	30	34	7090	72867	38	456	1:24
misc03	3	31	116	2784	125598	8	515	4:24
* misc04	3	1186(1)	3727	31616	16439421	4	2428	32:33:47
misc05	7	62	27	9454	304841	40	1418	12:19
* misc06	11	614(1)	1371	45399	12058549	170	5176	36:54:46
mod008	73	7	39	487038	1735403	820	3250	1:42:28
mod013	15	50	19	19909	158206	160	1310	3:03
p0201	1	2	3	1083	54676	2	171	1:13
stein9	6	13	0	710	1304	36	65	3
stein15	6	36	0	3663	14153	92	305	17
stein27	14	109	0	112941	619692	1383	7053	18:06
stein45	16	227	4	1028221	11089986	1042	13633	8:36:17

* IS(not necessarily IIS) isolated.

increases as the number of finitely bounded variables shrinks during the deletion filter. This explains why so few IISs are identified by “pure” deletion filters. Note however, that in most cases a few LCs only are identified as dubious and no IR is questionable in the output. Most dubious constraints are BDs on variables. Table 4 illustrates how deletion filters based on B & B searches can identify IISs in most cases if BDs are replaced by looser BDs instead of being dropped in the filtering of the infeasible formulation. Moreover, this also emphasizes that:

- Finite BDs are vital in the original inconsistent MILP even if they lead to some redundancy.

TABLE 5. Results for (LC-IR-BD) Deletion Filter (Algorithm 4)

- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h.m:s
* bell3a	13	46	65(65)	746510	1089361	36	190	57:31
* bell3b	51	55	62(61)	733668	1212488	2	127	1:17:42
* bell4	13	51(1)	55(38)	982327	1431763	2	110	1:19:53
* bell5	45	24(1)	63(59)	692561	959428	2	108	51:21
dell	4	33	114	14747	122082	6	767	6:04
* flugpl	6	11	8(8)	355378	237514	5718	4248	10:48
* g503	11	21	25(18)	3672	12195	0	45	17
* misc01	14	25(1)	64(48)	7826	38424	28	242	51
* misc02	13	29(1)	43(9)	7784	48736	34	326	58
* misc03	1	36(1)	117(32)	25929	166925	6	214	4:40
* misc04	3	1186(305)	3731(2284)	1786804	23012352	2	2392	101:47:54
* misc05	7	62(4)	55(23)	776164	3996470	52	1007	2:31:49
* misc06	11	613(26)	1017(652)	622445	19420458	58	3774	61:05:35
* mod008	73	7	313(313)	1125475	2588166	926	3832	2:57:52
* mod013	15	50	50(22)	37383	144965	130	681	3:18
* p0201	1	2	6(5)	616176	3689674	224	2463	3:20:05
stein9	6	13	0	1074	2139	44	86	3
stein15	6	36	0	3943	16192	92	340	19
stein27	14	109	0	162993	704387	1218	5517	21:59
stein45	16	227	4	1280217	9144909	1112	11298	7:16:18

* IS(not necessarily IIS) isolated.

- If the “loose-BDs” approach fails to isolate an IIS a B & Cut algorithm can be used instead as mentioned in Section 2.2.6.
- A “good” objective function can speed up the detection of infeasibility.

Isolated ISs contain in general a small fraction of the IRs in the original formulation, almost half of the identified ISs contain less than 10% of the original IRs and roughly two-third of them less than 25%. However, unlike the continuous case, IISs may contain large numbers of LCs and BDs. Indeed, more than two-third of the ISs isolated from the test set contain more than 50% of the original LCs and BDs.

Both approaches lead to the identification of the same subset when the original

TABLE 6. Rankings

Method 1: Algorithm 3: (IR-LC-BD) Deletion Filter

Depth-First Search, Default Branching Scheme

Method 2: Algorithm 4: (LC-IR-BD) Deletion Filter

Depth-First Search, Default Branching Scheme

Infeasible MILP	Method 1			Method 2		
	Isol.	Nodes	Time	Isol.	Nodes	Time
bell3a	1	1	1	2	2	2
bell3b	1	2	2	2	1	1
bell4	2	2	2	1	1	1
bell5	1	2	2	2	1	1
dell	- *	-	-	-	-	-
flugpl	1	2	2	1	1	1
g503	1	1	1	2	2	2
misc01	1	1	2	2	2	1
misc02	2	1	1	1	2	2
misc03	2	1	1	1	2	2
misc04	1	1	1	2	2	2
misc05	1	1	1	2	2	2
misc06	1	1	1	2	2	2
mod008	1	1	1	2	2	2
mod013	1	1	2	2	2	1
p0201	1	2	2	1	1	1
stein9	1	2	1	1	1	1
stein15	1	1	1	1	2	1
stein27	1	1	1	1	2	2
stein45	- *	-	-	-	-	-
Avg	1.17	1.33	1.39	1.56	1.67	1.50

* The two methods isolated different IISs.

MILP contains only one IIS. However, some superfluous constraints may appear in the output when dubious cases are encountered, i.e. when the B & B tree is truncated because the maximum number of nodes is exceeded. Thus, since these two algorithms are based on different orderings of the constraints, different ISs, containing the same IIS, may result from the isolations. Deleting IRs first leads to ISs with fewer questionable LCs whereas deleting LCs first favors ISs with fewer dubious IRs.

When deleting LCs first, if a fair number of LCs are deleted then some variables may have been eliminated from the isolation. Thus, the number of iterations in Steps 1 and 2 (deletion of IRs and LCs) may decrease. For instance, the LC-first filtering of “dell.mps”, discards 80 IRs out of 105 and 137 BDs out of 825. Thus, even if most of the nodes are generated while deleting LCs, the completion of the isolation requires a relatively small number of nodes and may turn out to be competitive compared to the other approach.

However this advantage disappears when IISs with roughly the same characteristics (including a relatively large number of LCs) are identified in both approaches. Moreover, in this case, the size of the temporary MILPs T tend to stay about the same throughout the LC filtering phase, which explains why deleting LCs first performs so poorly on some problems.

Prohibitive isolation times tend to restrict the use of straight deletion filtering techniques to relatively “small” MILPs since the number of MILP solutions is roughly equal to the total number of constraints contained in the original MILP. However, when assessing how “small” a MILP is, another criterion has to be taken into account as well, i.e. how difficult it is to solve this MILP; unfortunately this is fairly independent of its size as shown in Table 2.

CHAPTER 8

INCREMENTAL FILTERING ALGORITHM

FOR SMALL-MEDIUM MILPS

8.1 Overview

In this Chapter we adapt the incremental filter detailed in Section 2.1.3 to the MILP framework, using the implicit-explicit constraint classification (i.e. IRs as opposed to LCs and BDs).

Unlike the deletion filtering technique a straight incremental filter isolates the IIS whose last constraint is first in the ordered list of constraints. Therefore the ordering of the constraints is important. One could consider for instance adding IRs first, then LCs and BDs or, conversely, LCs and IRs and then BDs.

The experimental evidence of Chapter 7 favors deleting IRs first versus deleting LCs first in the deletion filter algorithm since it is generally faster. The initial assumption in Section 5.3.1 tells us that the LP-relaxation of the infeasible MILP can be considered feasible. Thus, one only needs to add IRs to the relaxed LP first, identify the *integer part* of the IIS and then treat LCs and BDs in a similar fashion.

Because this algorithm starts with small test MILPs and moves towards larger ones until infeasibility is detected, it may be more suitable for large MILPs than the deletion filter, which starts with large test MILPs and moves to smaller ones.

8.2 Algorithm

Algorithm 6 is an adaptation of Algorithm 2, outlined in Chapter 2, to the MILP context.

C : Ordered Set of Constraints in Original Model

I : Current Set of Constraints Contributing to Infeasibility

T : Test Set of Constraints

Input : An infeasible MILP or ILP P_0 containing IR IRs, LC LCs and BD BDs.

Step 0 :
 a. SET $C := IR_0 \cup LC_0 \cup BD_0$
 b. SET $I := \emptyset$

Step 1 :
 a. SET $T := I \cup LC_0 \cup BD_0$
 b. FOR EACH $ir_k \in C$ DO
 $T := T \cup \{ir_k\}$
 IF T inconsistent THEN
 $I := I \cup \{ir_k\}$
 $C := C \setminus \{ir_k, ir_{k+1}, \dots, ir_{IR}\}$
 IF $I \cup LC_0 \cup BD_0$ inconsistent GOTO Step 2
 GOTO Step 1

Step 2 :
 a. IF $I \cup BD_0$ inconsistent GOTO Step 3
 b. SET $T := I \cup BD_0$
 c. FOR EACH $lc_k \in C$ DO
 $T := T \cup \{lc_k\}$
 IF T inconsistent THEN
 $I := I \cup \{lc_k\}$
 $C := C \setminus \{lc_k, lc_{k+1}, \dots, lc_{LC}\}$
 IF $I \cup BD_0$ inconsistent GOTO Step 3
 GOTO Step 2b.

ALGORITHM 6. (IR-LC-BD) Incremental Filter (part 1)

8.3 Validity of the Isolation

Theorem 4: The Incremental Filtering Algorithm leads to the identification of a single IIS in an infeasible MILP.

Proof: Let P_0 be the infeasible MILP being filtered.

We first prove that the output of the algorithm is infeasible. If P_0 is minimal, i.e. P_0 is an IIS, then all constraints have to be added through Steps 1, 2 and 3 until

```

Step 3 :      a.  IF  $I$  inconsistent GOTO Step 4
                b.  SET  $BD_1 := BD_0 \setminus \{ \text{BDs on variables not in } lc \in T \}$ 
                c.  SET  $C := (C \setminus BD_0) \cup BD_1$ 
                d.  SET  $T := I$ 
                e.  FOR EACH  $bd_k \in C$  DO
                     $T := T \cup \{bd_k\}$ 
                    IF  $T$  inconsistent THEN
                         $I := I \cup \{bd_k\}$ 
                         $C := C \setminus \{bd_k, bd_{k+1}, \dots, bd_{BD}\}$ 
                        IF  $I$  inconsistent GOTO Step 4
                        GOTO Step 3d.

Step 4 :      STOP:   $I$  is an IIS

```

ALGORITHM 6. (IR-LC-BD) Incremental Filter (part 2)

I becomes infeasible. The algorithm exits with an inconsistent set I in **Step 4** in all cases and exits with $I = LP_0$ in case of minimality of the original set of constraints.

We establish minimality of the output I , assuming that an extra constraint C belongs to I so that $I \setminus \{C\}$ is infeasible. Without loss of generality we can suppose that C is a LC and rename it lc_0 . Some test set of constraints T , containing the integer part of the IIS plus some LCs (including lc_0) and all BDs, must have been infeasible since lc_0 was added to I . But, the same T , deprived of lc_0 , is also inconsistent. Indeed, by assumption $I \setminus \{lc_0\}$ is also inconsistent. Therefore $I \setminus \{lc_0\} \subseteq T \setminus \{lc_0\}$ implies that $T \setminus \{lc_0\}$ is inconsistent as well. This contradicts the sequence of steps in the algorithm. In fact, if lc_0 was the first LC to be added, then **Step 2a.** should have been activated before, and lc_0 would not have been added to I . If other LCs were already contained in I , then one step prior to checking lc_0 , the inconsistency of T should have discarded all LCs past the current one, including lc_0 and thus lc_0 could not have been contained in I .

The incremental filter identifies a minimal and infeasible inconsistent set of con-

straints, it is therefore an IIS. $\square$

8.4 Time and Space Complexities

Setting aside considerations of the variability of the time required for the solutions of the test sets, the incremental filtering algorithm requires in the worst case about $\frac{1}{2}(ir^2 + lc^2 + bd^2)$ MILP solutions of test sets (both T and I) where ir is the number of IRs, lc of LCs and bd of BDs in the original infeasible MILP. The incremental filter will perform the most poorly when P_0 is an IIS: in each iteration, inconsistency is attained when the last constraint of C is added to T . Thus, the first iteration requires n infeasibility proofs for T and 1 for I , the second an overall number of n proofs, the third $(n - 1)$ and so forth until the last iteration which requires 2 proofs. For example, the incremental filtering of ir IRs requires therefore $\frac{1}{2}(ir^2 + 3ir)$. In the same context, a deletion filter requires at most only $ir + lc + bd$ MILP solutions. On the other hand, the incremental filter is likely to outperform the deletion filter on models having relatively small IISs.

Remark: Consider IRs, LCs and BDs just as constraints. Since $ir^2 + lc^2 + bd^2 \leq (ir + lc + bd)^2$, where $\frac{1}{2}(ir + lc + bd)^2$ is the order of the complexity of a plain incremental filter, then similarly, the worst case behaviour of the incremental filter can be significantly reduced by splitting the original set of constraints into several classes of constraints. The average computational cost is on the other hand, likely to increase as the number of steps and controls is increased.

Similarly to the deletion filter, the incremental filter requires little memory apart from the space used by the MILP solver.

8.5 Opportunist Incremental Filter Algorithm

The initial ordering of the constraints determines which IIS is isolated, as for deletion filtering. However, the incremental filter does not have to check all the constraints since it discards constraints past the current one when infeasibility is

attained. When the test set of constraints T happens to be feasible, all constraints past the current one (i.e. constraints that have not yet been incorporated into T) which are satisfied by the current values of the solution can also be added to T . This is likely to happen when adding IRs or BDs or redundant LCs. As a matter of fact, network-like relationships can force continuous variables to take integral values provided that a sufficient number of variables are already integer. This corresponds to a dynamic reordering of the constraints and thus does not modify the validity of the outcome. See Algorithm 7.

Although there is a cost associated to checking whether some ir_j , lc_j or bd_j are satisfied by the current solution, the overhead can be neglected when compared to the cost of solving another MILP. Thus, both algorithms have the same worst case behaviour in $O(ir^2 + lc^2 + bd^2)$, but the “opportunistic” algorithm should outperform the straight incremental filter on average. Note that since this algorithm reorders constraints in the filtering process, it may identify another distinct IIS.

8.6 Example

The following lists the steps necessary to complete the execution of both incremental filters on the inconsistent MILP described in Section 7.5.1:

8.6.1 Algorithm 6: Straight Incremental Filtering.

Step 0: a. $C = \{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5, \underline{x}_6, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 0: b. $I = \emptyset$

Step 1: a. $T = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *FEASIBLE*
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *INFEASIBLE*
 $I = \{\underline{x}_2\}$
 $C = \{\underline{x}_1, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup LC_0 \cup BD_0$ *FEASIBLE*

Step 1: a. $T = \{\underline{x}_2, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_2\}$ *INFEASIBLE*

C : Ordered Set of Constraints in Original Model

I : Current Set of Constraints Contributing to Infeasibility

T : Test Set of Constraints

Input : An *infeasible* MILP or ILP P_0

Step 0 : a. SET $C := IR_0 \cup LC_0 \cup BD_0$

b. SET $I := \emptyset$

Step 1 : a. SET $T := I \cup LC_0 \cup BD_0$

b. FOR EACH $ir_k \in C$ DO

$T := T \cup \{ir_k\}$

IF T *inconsistent* THEN

$I := I \cup \{ir_k\}$

$C := C \setminus \{ir_k, ir_{k+1}, \dots, ir_{IR}\}$

IF $I \cup LC_0 \cup BD_0$ *inconsistent* GOTO Step 2

GOTO Step 1

ELSE

$T := T \cup \{ir_j \mid k < j \text{ and } x_j \in \mathbb{Z}\}$

Step 2 : a. IF $I \cup BD_0$ *inconsistent* GOTO Step 3

b. SET $T := I \cup BD_0$

c. FOR EACH $lc_k \in C$ DO

$T := T \cup \{lc_k\}$

IF T *inconsistent* THEN

$I := I \cup \{lc_k\}$

$C := C \setminus \{lc_k, lc_{k+1}, \dots, lc_{LC}\}$

IF $I \cup BD_0$ *inconsistent* GOTO Step 3

GOTO Step 2b.

ELSE

$T := T \cup \{lc_j \mid k < j \text{ and } lc_j \text{ satisfied}\}$

ALGORITHM 7. (IR-LC-BD) Opportunist Incremental Filter (part 1)

$$I = \{\underline{x}_1, \underline{x}_2\}$$

$$C = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$$

$$I \cup LC_0 \cup BD_0 \text{ INFEASIBLE}$$

Step 2 : a. $\{\underline{x}_1, \underline{x}_2, x_1 \geq 0, \dots, x_6 \geq 0\}$ is *FEASIBLE*

Step 2 : b. $T = \{\underline{x}_1, \underline{x}_2, x_1 \geq 0, \dots, x_6 \geq 0\}$

Step : c. $T := T \cup \{C_1\}$ *FEASIBLE*

Step 2 : c. $T := T \cup \{C_2\}$ *FEASIBLE*

Step 3 :

- a. IF I inconsistent GOTO Step 4
- b. SET $BD_1 := BD_0 \setminus \{ \text{BDs on variables not in } I \in T \}$
- c. SET $C := (C \setminus BD_0) \cup BD_1$
- d. SET $T := I$
- e. FOR EACH $bd_k \in C$ DO
 - $T := T \cup \{bd_k\}$
 - IF T inconsistent THEN
 - $I := I \cup \{bd_k\}$
 - $C := C \setminus \{bd_k, bd_{k+1}, \dots, bd_{BD}\}$
 - IF I inconsistent GOTO Step 4
 - GOTO Step 3d.
 - ELSE
 - $T := T \cup \{bd_j \mid k < j \text{ and } x_j \text{ satisfies } bd_j\}$

Step 4 : **STOP:** I is an IIS

ALGORITHM 7. (IR-LC-BD) Opportunist Incremental Filter (part 2)

Step 2: c. $T := T \cup \{C_3\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_4\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_5\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_6\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_7\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_8\}$ **INFEASIBLE**

$I = \{\underline{x}_1, \underline{x}_2, C_8\}$

$C = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, x_1 \geq 0, \dots, x_6 \geq 0\}$

$I \cup BD_0$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_1\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_2\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_3\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_4\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_5\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_6\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_7\}$ **INFEASIBLE**

$I = \{\underline{x}_1, \underline{x}_2, C_7, C_8\}$

$C = \{C_1, C_2, C_3, C_4, C_5, C_6, x_1 \geq 0, \dots, x_6 \geq 0\}$

$I \cup BD_0$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_1\}$ **FEASIBLE**

Step 2: c. $T := T \cup \{C_2\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_3\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_4\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_5\}$ *INFEASIBLE*
 $I = \{x_1, x_2, C_5, C_7, C_8\}$
 $C = \{C_1, C_2, C_3, C_4, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_1\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_2\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_3\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_4\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_2, C_4, C_5, C_7, C_8\}$
 $C = \{C_1, C_2, C_3, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_1\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_2\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_2, C_2, C_4, C_5, C_7, C_8\}$
 $C = \{C_1, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_1\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_2, C_1, C_2, C_4, C_5, C_7, C_8\}$
 $C = \{x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *INFEASIBLE*
Step 3: a. $\{\underline{x}_1, \underline{x}_2, C_1, C_2, C_4, C_5, C_7, C_8\}$ is *FEASIBLE*
Step 3: b. $T = \{\underline{x}_1, \underline{x}_2, C_1, C_2, C_4, C_5, C_7, C_8\}$
Step 3: c. $T := T \cup \{x_1 \geq 0\}$ *FEASIBLE*
Step 3: c. $T := T \cup \{x_2 \geq 0\}$ *FEASIBLE*
Step 3: c. $T := T \cup \{x_3 \geq 0\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_2, C_1, C_2, C_4, C_5, C_7, C_8, x_3 \geq 0\}$
 $C = \{x_1 \geq 0, x_2 \geq 0\}$
 I *INFEASIBLE*
Step 4: STOP : $I = \{\underline{x}_1, \underline{x}_2, C_1, C_2, C_4, C_5, C_7, C_8, x_3 \geq 0\}$ is an IIS

An overall total of 44 MILP solutions, 12 of which lead to the conclusion of inconsistency of the test set, are required to identify the IIS which contains 2 IRs of 6, 6 LCs of 8 and 1 BD of 6.

8.6.2 Algorithm 7: Opportunist Incremental Filtering.

- Step 0:** a. $C = \{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5, \underline{x}_6, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 0: b. $I = \emptyset$
- Step 1:** a. $T = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *FEASIBLE*
Step 1: b. (ELSE) $T := T \cup \{\underline{x}_4\}$
Step 1: b. $T := \{\underline{x}_1, \underline{x}_4, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\} \cup \{\underline{x}_3\}$
 is *INFEASIBLE*
 $I = \{\underline{x}_3\}$
 $C = \{\underline{x}_4, \underline{x}_1, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup LC_0 \cup BD_0$ *FEASIBLE*
- Step 1:** a. $T = \{\underline{x}_3, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3\}$
 $C = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup LC_0 \cup BD_0$ *INFEASIBLE*
- Step 2:** a. $T = \{\underline{x}_1, \underline{x}_3, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 2: b. T is *FEASIBLE*
Step 2: c. $T := T \cup \{C_1\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_2\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_3\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_4\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_5\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_6\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_7\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_8\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3, C_8\}$
 $C = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
- Step 2:** c. $T := T \cup \{C_1\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_2\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_3\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_4\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_5\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_6\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_7\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3, C_7, C_8\}$
 $C = \{C_1, C_2, C_3, C_4, C_5, C_6, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
- Step 2:** c. $T := T \cup \{C_1\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_2\}$ *FEASIBLE*

- Step 2:** c. $T := T \cup \{C_3\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_4\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_5\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_6\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3, C_6, C_7, C_8\}$
 $C = \{C_1, C_2, C_3, C_4, C_5, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_1\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_2\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_3\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_4\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3, C_4, C_6, C_7, C_8\}$
 $C = \{C_1, C_2, C_3, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_1\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_2\}$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_3\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3, C_3, C_4, C_6, C_7, C_8\}$
 $C = \{C_1, C_2, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *FEASIBLE*
Step 2: c. $T := T \cup \{C_1\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3, C_1, C_3, C_4, C_6, C_7, C_8\}$
 $C = \{x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup BD_0$ *INFEASIBLE*
Step 3: a. I *FEASIBLE*
Step 3: b. $BD_1 = \{x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 3: c. $C = \{x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 3: d. $T = \{\underline{x}_1, \underline{x}_3, C_1, C_2, C_4, C_5, C_7, C_8\}$
Step 3: e. $T := T \cup \{x_1 \geq 0\}$ *FEASIBLE*
Step 3: e. (ELSE) $T := T \cup \{x_3 \geq 0\}$
Step 3: e. (ELSE) $T := T \cup \{x_4 \geq 0\}$
Step 3: e. (ELSE) $T := T \cup \{x_6 \geq 0\}$
Step 3: e. $T := T \cup \{x_2 \geq 0\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3, C_1, C_3, C_4, C_6, C_7, C_8, x_2 \geq 0\}$
 $C = \{x_6 \geq 0, x_4 \geq 0, x_3 \geq 0, x_1 \geq 0\}$
 I *INFEASIBLE*
Step 4 STOP : $I = \{\underline{x}_1, \underline{x}_3, C_1, C_3, C_4, C_6, C_7, C_8, x_2 \geq 0\}$ is an IIS

The identification of the 2 IRs of 6, 6 LCs of 8 and 1 BD of 6 which constitute the IIS requires the proof of feasibility of 36 MILPs and inconsistency of 12.

8.7 Computational Results

Tables 7 and 8 list the performances of Algorithms 6 and 7 on the test set of infeasible MILPs. Objective functions of the models have been set equal to the sum of all variables restricted to take integral values (general and binary IRs).

TABLE 7. Results for (IR-LC-BD) Incremental Filter (Algorithm 6)

- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	53	61689	601275	24	307	19:27
* bell3b	1	65	55	5084	471787	10000	32704	14:46
* bell4	13	65	43	228604	703262	1330	3171	39:26
* bell5	1	49	48	3115	249427	10000	33303	7:32
dell	5	184	825	93906	9004289	2	563	14:50:57
* Flugpl	8	13	10	1988436	1239675	3501	4472	1:00:47
* g503	1	33	15	327	20247	0	45	45
* misc01	12	28	165	5587	74450	60	816	2:41
* misc02	14	30	115	38268	312357	38	437	12:18
* misc03	3	31	319	1806	59880	18	571	5:30
misc04	kil(a)							
* misc05	7	62	71	24683	1275287	80	4866	1:13:37
misc06	kil(b)							
* mod008	37	7	638	191653	917646	526	3443	45:14
* mod013	15	50	51	41148	388142	104	1294	10:33
* p0201	1	2	6	407	29604	2	101	1:50
stein9	6	13	0	1042	2688	38	78	7
stein15	6	36	0	5114	25212	88	329	51
stein27	14	109	0	244824	1323113	1334	6930	41:03
stein45	kil(c)							

* IS(not necessarily IIS) isolated.

kil: killed after (a)200, (b)65 and (c)30 hours.

TABLE 8. Results for (IR-LC-BD) Opportunist Incremental Filter (Algo. 7)

- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	76	51	18350	275009	24	307	9:54
* bell3b	1	65	58	2062	129240	10000	32704	5:35
* bell4	18	65	43	313958	691259	1330	3171	38:35
* bell5	1	49	48	1062	65040	10000	33303	2:34
dell	1	33	825	952	137833	2	563	55:43
* flugpl	8	13	9	1491902	931704	3501	4472	46:18
* g503	1	18	17	320	7139	0	45	23
* misc01	20	27	165	8512	98611	60	816	2:36
* misc02	1	31	117	6809	65374	38	437	1:38
* misc03	10	30	319	5561	124957	18	571	7:00
misc04	kil(a)							
* misc05	16	68	118	46986	2365221	80	4866	1:47:25
misc06	kil(b)							
* inod008	2	7	638	4034	19558	526	3443	1:05
* mod013	16	50	54	25245	341871	104	1294	8:31
* p0201	1	2	6	334	10658	2	101	57
stein9	6	13	0	1069	2435	38	78	7
stein15	6	36	0	5015	22464	88	329	52
stein27	14	106	0	278979	1446623	1334	6930	45:27
stein45	kil(c)							

* IS(not necessarily IIS) isolated.

kil: killed after (a)200, (b)65 and (c)30 hours.

8.8 Discussion

As in Tables 3 and 5 ((IR-LC-BD) and (LC-IR-BD) deletion filters), few IISs but rather ISs are isolated by the two incremental filters (Algorithm 6 and 7). Unlike the deletion filtering algorithm, an incremental filter does not directly identify dubious constraints in the isolation process. An additional deletion filtering of the entire output is required to label and count the questionable constraints. Note also that the MILP solver's ability to detect infeasibility varies greatly with the objective function of the inconsistent MILP.

When the original inconsistent MILP contains several IISs, the deletion and incremental filters behave in different ways. When adding IRs for instance, the incremental filter discards all untested IRs once infeasibility of the test set has been encountered. On the other hand, all IRs have to be tested before the deletion filter identifies the integer part of the IIS. Thus, if the IIS IRs are homogeneously scattered among the set of IRs, the incremental filter is likely to isolate an IIS with a smaller number of IRs than the deletion filter. Algorithms 6 and 7 lead to IISs with fewer IRs and LCs than found by the deletion filter in 4 instances (“bell3b”, “bell4”, “bell5” and “dell”).

Similarly to the deletion filter case, when doubt arises about the feasibility of an intermediate test set, minimality is sacrificed to inconsistency and the troublesome constraint is added to T , i.e. the outcome may therefore not be minimal but is guaranteed to be infeasible. However, unlike the deletion filter, dubious constraints are not positively identified by the incremental algorithm and are therefore not listed in Tables 7 and 8. Inability of the MILP solver to detect inconsistency is likely to happen quite often when adding BDs. One starts with all variables free and progressively adds bounds on them. Thus, early iterations of the incremental filter, when most variables are unbounded, are more likely to be unbounded, leading to more extraneous BD constraints in the IS than deletion filtering the same set of BDs. Not surprisingly, the incremental filter behaves poorly in some cases on the BD part of the isolations; this is particularly true on the “misc*” datasets where nearly twice as many BDs as necessary are identified in the IS.

On average, the incremental algorithm generates more nodes and lengthier execution times compared to the deletion filter when isolating identical IISs with relatively large cardinalities. If the time required to perform the identification also depends on the efficiency of the MILP solver, the sole comparison of the number of nodes leads to the same conclusion. On the other hand, nearly “best-case behaviour” is obtained

when the IISs isolated contain few constraints relative to the original MILP and when these constraints are principally located among the first constraints of each class of constraints (IRs, LCs and BDs): discarding constraints helps focus and speed up the search; this property is illustrated in the filtering of problems such as “bell*” where the number of nodes required can be divided by a factor of a hundred.

The opportunist incremental filter (Algorithm 7) requires in most cases less time than the straight incremental filter (Algorithm 6). In roughly half of the isolations the overall execution time has been divided by a factor of 2 or more and some drastic improvements have been achieved. Isolation time went down from approximately 14 hours on “dell” with Algorithm 6 to less than 1 hour with Algorithm 7. However, the extra cost associated with the opportunist approach may actually increase the isolation time when there is no satisfied constraint to add.

Isolation times for the incremental filters still depend on the initial ordering of the constraints in the original MILP. Like the deletion filtering technique, this approach is more suited to relatively small MILPs. However, if the initial ordering of the constraints, as well as the cardinality of the IISs contained in the infeasible model allow it, the incremental filtering algorithms can also isolate IISs in medium sized inconsistent MILPs, i.e. whose cardinality in IRs, BDs and LCs but also solution complexity is reasonable.

CHAPTER 9
INCREMENTAL/DELETION FILTERING
ALGORITHM FOR SMALL-MEDIUM MILPS

9.1 Overview

The Incremental/Deletion Filter presented in this Chapter combines both previously described approaches. In the first phase, IRs are added to the LP-relaxation of the original MILP, then LCs and BDs are deletion filtered out. Thus, this new algorithm is likely to isolate IISs with fewer IRs than the incremental filter would have identified, without incurring the overhead caused by the worst-case behaviour of the latter, adopting the linear complexity of the deletion filter on the LCs and BDs part. Therefore a more stable behaviour is expected from this technique. Moreover, as remarked in Section 8.7, the outcome is also likely to contain fewer spurious constraints than in the incremental filtering case since BDs are filtered and not added to the initially empty test set.

9.2 Algorithm

Algorithm 8 which implements a straight incremental/deletion filter borrows parts from the incremental filter, for identifying the IR part of the IIS, and from the deletion filter for the continuous (LCs and BDs) part.

9.3 Validity of the Isolation

Theorem 5: The Incremental/Deletion Filtering Algorithm leads to the identification of a single IIS in an infeasible MILP.

Proof: The *integer part* of the output is guaranteed to be minimal, since the Incremental filter isolates an IIS. Thus Step 1 leads to the identification of a minimal set

C : Ordered Set of Constraints in Original Model

I : Current Set of Constraints Contributing to Infeasibility

T : Test Set of Constraints

Input : An infeasible MILP or $M \supset P_0$ containing IR IRs, LC LCs and BD BDs.

Step 0 :
 a. SET $C := IR_0 \cup LC_0 \cup BD_0$
 b. SET $I := \emptyset$

Step 1 :
 a. SET $T := I \cup LC_0 \cup BD_0$
 b. FOR EACH $ir_k \in C$ DO
 $T := T \cup \{ir_k\}$
 IF T inconsistent THEN
 $I := I \cup \{ir_k\}$
 $C := C \setminus \{ir_k, ir_{k+1}, \dots, ir_{IR}\}$
 IF $I \cup LC_0 \cup BD_0$ inconsistent GOTO Step 2
 GOTO Step 1

Step 2 :
 a. SET $T := I \cup LC_0 \cup BD_0$
 b. FOR EACH $lc_k \in LC_0$ DO
 IF $T \setminus \{lc_k\}$ inconsistent SET $T := T \setminus \{lc_k\}$

Step 3 :
 a. SET $BD_1 := BD_0 \setminus \{BDs \text{ on variables not in } lc \in T\}$
 b. SET $T := (T \setminus BD_0) \cup BD_1$
 c. FOR EACH $bd_k \in BD_1$ DO
 IF $T \setminus \{bd_k\}$ inconsistent SET $T := T \setminus \{bd_k\}$

Step 4 : **STOP:** T is an **IIS**

ALGORITHM 8. (IR) Incremental/ (LC-BD) Deletion Filter Algorithm

of IRs that renders the relaxed LP infeasible. Deletion filtering discards LCs and BDs that do not cause infeasibility. The output of Step 2 and 3 is also minimal. The set of constraints (IRs, LCs and BDs) identified by the Incremental/Deletion Filtering Algorithm is therefore an IIS. $\square$

9.4 Time and Space Complexities

The Incremental/Deletion Filtering algorithm inherits some characteristics from

both the incremental and deletion filters in terms of complexity. The worst-case behaviour of this algorithm is in $O(IR^2 + LC + BD)$ MILP solutions (where IR , LC and BD are respectively the numbers of IRs, LCs and BDs in the original model), a combination of the complexities of the incremental filter on the *integer part* of the IIS and of the deletion filter on the *linear part*. On the other hand, adding IRs through the use of the incremental filter may dramatically improve the best-case behaviour, to $O(ir + LC + BD)$ where ir is the number of IRs contained in the isolated IIS.

Memory requirements of the incremental/deletion filter are identical to those of both previously described algorithms.

9.5 Opportunist Incremental/Deletion Filter Algorithm

As in Section 8.5, IRs can be dynamically reordered in the incremental filtering phase. This may lead to the identification of another distinct IIS. See Algorithm 9.

9.6 Example

The following lists the steps necessary to complete the execution of both incremental/deletion filters on the inconsistent MILP described in Section 7.5.1:

9.6.1 Algorithm 8: Straight Incremental/Deletion Filtering.

Step 0: a. $C = \{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5, \underline{x}_6, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 0: b. $I = \emptyset$

Step 1: a. $T = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *FEASIBLE*
Step 1: b. $T := T \cup \{\underline{x}_2\}$ *INFEASIBLE*
 $I = \{\underline{x}_2\}$
 $C = \{\underline{x}_1, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup LC_0 \cup BD_0$ *FEASIBLE*

Step 1: a. $T = \{\underline{x}_2, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_2\}$
 $C = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$

C : Ordered Set of Constraints in Original Model

I : Current Set of Constraints Contributing to Infeasibility

T : Test Set of Constraints

Input : An *infeasible* MILP or ILP P_0

Step 0 : a. SET $C = IR_0 \cup LC_0 \cup BD_0$

b. SET $I = \emptyset$

Step 1 : a. SET $T = I \cup LC_0 \cup BD_0$

b. FOR EACH $ir_k \in IR_0$ DO

$T := T \cup \{ir_k\}$

IF T *inconsistent* THEN

$I := I \cup \{ir_k\}$

$C := C \setminus \{ir_k, ir_{k+1}, \dots, ir_{IR}\}$

IF $I \cup LC_0 \cup BD_0$ *inconsistent* GOTO **Step 2**

GOTO **Step 1**

ELSE

$T := T \cup \{ir_j \mid k < j \text{ and } x_j \in \mathbb{Z}\}$

Step 2 : a. SET $I := I \cup LC_0 \cup BD_0$

b. FOR EACH $lc_k \in LC_0$ DO

IF $T \setminus \{lc_k\}$ *inconsistent* SET $T := T \setminus \{lc_k\}$

Step 3 : a. SET $BD_1 = BD_0 \setminus \{BDs \text{ on variables not in } lc \in T\}$

b. SET $T := (T \setminus BD_0) \cup BD_1$

c. FOR EACH $bd_k \in BD_1$ DO

IF $T \setminus \{bd_k\}$ *inconsistent* SET $T := T \setminus \{bd_k\}$

Step 4 : **STOP**: T is an **IIS**

ALGORITHM 9. (**IR**) Opportunist Incremental/ (**LC-BD**) Deletion Filter

$I \cup LC_0 \cup BD_0$ **INFEASIBLE**

Step 2: a. $T = \{\underline{x}_1, \underline{x}_2, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$

Step 2: b. $T \setminus \{C_1\}$ **FEASIBLE**

Step 2: b. $T \setminus \{C_2\}$ **FEASIBLE**

Step 2: b. $T \setminus \{C_3\}$ **INFEASIBLE**; $T := T \setminus \{C_3\}$

Step 2: b. $T \setminus \{C_4\}$ **FEASIBLE**

Step 2: b. $T \setminus \{C_5\}$ **FEASIBLE**

Step 2: b. $T \setminus \{C_6\}$ **INFEASIBLE**; $T := T \setminus \{C_6\}$

Step 2: b. $T \setminus \{C_7\}$ *FEASIBLE*
Step 2: b. $T \setminus \{C_8\}$ *FEASIBLE*
Step 3: a. $BD_1 = BD_0$
Step 3: b. T not changed
Step 3: c. $T \setminus \{x_1 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_1 \geq 0\}$
Step 3: c. $T \setminus \{x_2 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_2 \geq 0\}$
Step 3: c. $T \setminus \{x_3 \geq 0\}$ *FEASIBLE*
Step 3: c. $T \setminus \{x_4 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_4 \geq 0\}$
Step 3: c. $T \setminus \{x_5 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_5 \geq 0\}$
Step 3: c. $T \setminus \{x_6 \geq 0\}$ *INFEASIBLE*; $T := T \setminus \{x_6 \geq 0\}$
Step 4: STOP : $T = \{\underline{x}_1, \underline{x}_2, C_1, C_2, C_4, C_5, C_7, C_8, x_3 \geq 0\}$ is an IIS

2 IRs of 6, 6 LCs of 8 and 1 BD of 6 make up the isolated IIS; 21 MILP solutions are required to perform the isolation, 11 of which reach feasibility.

9.6.2 Algorithm 9: Opportunist Incremental/Deletion Filtering.

Step 0: a. $C = \{\underline{x}_1, \underline{x}_2, \underline{x}_3, \underline{x}_4, \underline{x}_5, \underline{x}_6, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 0: b. $I = \emptyset$
Step 1: a. $T = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *FEASIBLE*
Step 1: b. (ELSE) $T := T \cup \{\underline{x}_2\}$
Step 1: b. $T := T \cup \{\underline{x}_3\}$ *INFEASIBLE*
 $I = \{\underline{x}_3\}$
 $C = \{\underline{x}_1, \underline{x}_2, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup LC_0 \cup BD_0$ *FEASIBLE*
Step 1: a. $T = \{\underline{x}_3, C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 1: b. $T := T \cup \{\underline{x}_1\}$ *INFEASIBLE*
 $I = \{\underline{x}_1, \underline{x}_3\}$
 $C = \{C_1, \dots, C_8, x_1 \geq 0, \dots, x_6 \geq 0\}$
 $I \cup LC_0 \cup BD_0$ *INFEASIBLE*
Step 2: a. $T = \{\underline{x}_1, \underline{x}_3, C_1, \dots, C_2, x_1 \geq 0, \dots, x_6 \geq 0\}$
Step 2: b. $T \setminus \{C_1\}$ *FEASIBLE*
Step 2: b. $T \setminus \{C_2\}$ *INFEASIBLE*; $T := T \setminus \{C_2\}$
Step 2: b. $T \setminus \{C_3\}$ *FEASIBLE*
Step 2: b. $T \setminus \{C_4\}$ *FEASIBLE*
Step 2: b. $T \setminus \{C_5\}$ *INFEASIBLE*; $T := T \setminus \{C_5\}$
Step 2: b. $T \setminus \{C_6\}$ *FEASIBLE*

Step	2:	b.	$T \setminus \{C_7\}$	<i>FEASIBLE</i>
Step	2:	b.	$T \setminus \{C_8\}$	<i>FEASIBLE</i>
Step	3:	b.	$T \setminus \{x_1 \geq 0\}$	<i>INFEASIBLE</i> ; $T := T \setminus \{x_1 \geq 0\}$
Step	3:	b.	$T \setminus \{x_2 \geq 0\}$	<i>FEASIBLE</i>
Step	3:	b.	$T \setminus \{x_3 \geq 0\}$	<i>INFEASIBLE</i> ; $T := T \setminus \{x_3 \geq 0\}$
Step	3:	b.	$T \setminus \{x_4 \geq 0\}$	<i>INFEASIBLE</i> ; $T := T \setminus \{x_4 \geq 0\}$
Step	3:	b.	$T \setminus \{x_5 \geq 0\}$	<i>INFEASIBLE</i> ; $T := T \setminus \{x_5 \geq 0\}$
Step	3:	b.	$T \setminus \{x_6 \geq 0\}$	<i>INFEASIBLE</i> ; $T := T \setminus \{x_6 \geq 0\}$
Step	4:		STOP : $T = \{\underline{x}_1, \underline{x}_3, C_1, C_3, C_4, C_6, C_7, C_8, x_2 \geq 0\}$ is an IIS	

Although different, the two identified IISs share the same characteristics, i.e. 2 IRs, 6 LCs and 1 BD. An identical number of MILP solutions is necessary to complete the isolation.

9.7 Computational Results

Table 9 lists the performances of the incremental/deletion filtering approach (Algorithm 8) on the test set of infeasible MILPs and Table 10 the performances of the opportunist incremental/deletion filtering technique (Algorithm 9). Objective functions of the infeasible models have been modified according to the scheme described in Section 8.7.

9.8 Discussion

In most cases the incremental/deletion filters identify ISs rather than IISs. A better handling of the dropped BDs can isolate IISs more successfully as shown in Table 4. Unlike the incremental filters, since LCs and BDs are deletion filtered, dubious continuous constraints can be singled out and counted. In all but one case (“bell5”), no IR can be dubious since the MILP solver is able to detect infeasibility of the original MILP in a finite number of nodes.

The incremental/deletion filter isolates IISs whose *integer part* is identical to one of the IISs produced by the incremental filter. Thus, IISs with fewer IRs than in

TABLE 9. Results for (IR) Incremental/ (LC-BD) Deletion Filter (Algo. 8)

- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	52(1)	7459	76643	24	307	1:54
* bell3b	1	65	56(7)	562	27994	10000	32704	26
* bell4	13	65	43(1)	173969	278692	1330	3171	18:17
* bell5	1	49	49(2)	444	15882	10000	33303	14
* dell	5	320	621(1)	10119	1369927	2	13	2:01:13
* flugpl	8	13(4)	7(5)	432436	306378	3501	4472	14:04
* g503	1	33	16(1)	180	5262	0	45	5
* misc01	12	28	71(2)	8012	91611	60	816	2:38
* misc02	14	30	57(7)	11499	116684	38	437	2:25
* misc03	3	31	113(5)	2428	47035	18	571	1:58
* misc04	3	1186	3732(2)	23272	12387963	6	3335	33:49:30
* misc05	7	62	60(11)	6673	379509	80	4866	16:49
* misc06	4	605	1046(8)	16125	3369270	54	5289	6:39:07
* mod008	37	7	636(11)	98437	623213	526	3443	23:37
* mod013	15	50	98(66)	24946	236770	104	1294	5:08
* p0201	1	2	7(1)	961	26107	2	101	2:03
stein9	6	13	0	1251	2819	38	78	4
stein15	6	36	0	4416	17950	88	329	23
* stein27	14	109	2(2)	132198	741976	1334	6930	21:05
* stein45	16	227	35(31)	912355	9358798	1090	13532	7:12:55

* IS(not necessarily IIS) isolated.

the deletion filter case were isolated as well. It also outperforms the incremental filter when IISs have a large proportion of LCs and BDs; the IIS identifications performed on the “bell*” series clearly exhibit this property. Moreover, since BDs are deleted and not added, as in the pure incremental filtering case, fewer extraneous BDs are left in the ISs, which improves the quality of the output. Also, worst-case behaviours observed with the straight incremental filter have disappeared.

The incremental/deletion filter offers an interesting compromise between the deletion and incremental filters. And, indeed, isolations on medium-sized MILPs which may require several hours when using a deletion or an incremental filter can be

**TABLE 10. Results for Opportunist (IR) Incremental/ (LC-BD) Deletion Filter
(Algorithm 9)**

- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	8	76	51(3)	27094	113103	24	307	3:59
* bell3b	1	65	56(7)	562	27994	10000	32704	26
* bell4	15	65	43(1)	785144	1795039	1330	3171	1:24:10
* bell5	1	49	49(2)	444	15882	10000	33303	14
* dell	1	33	114(1)	5718	4248	2	563	3:26
* flugpl	8	13(5)	6(4)	429312	298729	3501	4472	13:28
* g503	1	18	18(1)	180	2806	0	45	3
* misc01	23	25	77(7)	12708	115698	60	816	2:20
* misc02	11	30	55(4)	4472	46460	38	437	46
* misc03	8	29	112(20)	16021	92117	18	571	3:13
* misc04	3	1186	3732(3)	22187	11112753	6	3335	30:37:05
* misc05	12	73	100(88)	49412	1216934	80	4866	48:35
* misc06	4	605	1046(6)	16178	3331787	54	5289	6:42:38
mod008	2	7	638	5062	38946	526	3443	1:39
* mod013	15	50	56(6)	18169	144123	104	1294	2:47
* p0201	1	2	7(1)	961	8953	2	101	29
stein9	6	13	0	1238	2721	38	78	4
stein15	7	36	0	4464	17154	88	329	21
* stein27	14	109	2(2)	132148	742034	1334	6930	21:04
* stein45	16	225	9(5)	958773	10632330	1090	13532	8:00:03

* IS(not necessarily IIS) isolated.

achieved in a matter of a few minutes or even seconds (e.g. “bell*” series or “p0201”) with this filter.

However, lengthy, unacceptable isolation times may be observed for large MILPs, which restricts the use of this algorithm to small to medium-sized MILPs.

CHAPTER 10

SPEED CONSIDERATIONS

Any of the algorithms described in Chapter 7, 8 or 9 requires numerous, likely in the hundreds or even thousands, tests of feasibility of a test MILP which is modified as the algorithm proceeds. The efficiency of these algorithms can be improved by two different means; firstly, by accelerating each one of these tests of feasibility and secondly, by reducing the overall number of feasibility tests required to complete the identification.

10.1 On MILP Solutions

10.1.1 Settings of the MILP Solver.

The proof of feasibility of a MILP, based on a B & B tree, requires the development of the binary tree until a feasible solution is reached. In case of an infeasible MILP, the proof is not completed until the whole tree has been searched and the existence of a feasible solution has been shown to be impossible.

Two conclusions can be drawn from this remark:

- For two MILPs with similar characteristics, one feasible and the other not, the proof of consistency of the former is likely to be faster than establishing that the latter is inconsistent.
- The strategy used to develop the tree, i.e. the node selection and branching variable selection schemes, has a considerable impact on the duration of the proof.

The deletion, incremental and deletion/incremental filters require the solution of a sequence of test MILPs. The consistency of each one of these temporary for-

TABLE 11. Comparison of Different B & B Strategies (Algorithm 3)

• Depth-First Search, Branching on Maximum-Infeasibility Variable.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	45881	112078	558	1159	6:04
* bell3b	1	66	113(24)	645660	1730108	10000	21337	1:25:51
* bell4	13	65	43(14)	2207851	2426466	10000	10395	3:17:25
* bell5	18	48	63(43)	1136269	2004419	10000	29106	1:40:18
dell	5	382	707	9128	1151618	8	744	1:39:40
* flugpl	6	6	6(6)	416307	244243	10000	8593	11:55
* g503	1	33	15(4)	207	6123	0	45	6
* misc01	12	28(1)	65(25)	5286	41542	16	213	1:02
* misc02	14	30(1)	43(7)	7308	49825	32	345	1:03
* misc03	3	31(1)	110(40)	3644	58174	14	282	2:16
* misc04	3	1186(294)	3731(2477)	19147	8139143	4	2466	19:21:40
* misc05	7	62(4)	55(32)	9667	190400	42	1084	7:54
* misc06	11	613(26)	1050(673)	45906	3219977	60	3811	6:29:17
* mod008	73	7	313(292)	335940	1141094	752	3121	1:07:40
* mod013	15	50	50(21)	17266	73032	128	608	1:43
* p0201	1	2	6(5)	1081813	6954630	10000	82322	5:24:39
stein9	6	13	0	1109	2217	44	86	3
stein15	6	36	0	3937	16185	92	340	19
stein27	14	109	0	139602	625873	1218	5517	19:29
stein45	16	227	4	1035530	10126733	1112	11298	7:40:14

* IS(not necessarily IIS) isolated.

mulations mainly depends on the initial ordering of the constraints in the original inconsistent MILP and cannot therefore be controlled.

A best-bound node selection and branching variable selection scheme is traditionally adopted for optimization purposes. The best-bound and depth-first exploration strategies of the binary tree are not likely to differ much in case of an infeasible MILP, but a depth-first node selection scheme is likely to outperform the best-bound approach when the MILP is consistent. In fact, a depth-first scheme often generates a poor but feasible solution quickly whereas the best-bound scheme focuses its search on finding “good” solutions only. Moreover, in a depth-first policy, the dual-simplex

TABLE 12. Comparison of Different B & B Strategies (Algorithm 3)

• Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	7284	44847	36	190	1:55
* bell3b	8	75	77(48)	1325018	2307950	2	127	2:18:33
* bell4	15	65	43(14)	1349744	1455399	2	110	2:01:55
* bell5	6	53	78(25)	1010942	2105135	2	108	1:39:32
dell	5	381	764	7512	1133866	6	767	1:35:03
* flugpi	6	6	6(6)	414413	241286	5712	4248	11:45
* g503	1	33	15(4)	213	6180	0	45	6
* misc01	12	28(1)	65(25)	4929	41813	28	242	59
* misc02	14	30(1)	43(7)	6610	45758	34	326	55
* misc03	3	31(1)	110(40)	2982	53854	6	214	2:01
* misc04	3	1186(294)	3731(2477)	20385	8199496	2	2392	19:41:38
* misc05	7	62(4)	55(32)	9486	188309	52	1007	7:57
* misc06	11	613(26)	1017(684)	44277	3201026	58	3774	6:28:52
* mod008	73	7	313(292)	583635	2056161	926	3832	1:56:59
* mod013	15	50	50(21)	16889	74834	130	681	1:44
* p0201	1	2	6(5)	911440	6338045	224	2463	5:04:14
stein9	6	13	0	1109	2217	44	86	3
stein15	6	36	0	3937	16185	92	340	19
stein27	14	109	0	139602	625873	1218	5517	19:43
stein45	16	227	4	1035530	10126733	1112	11298	7:40:27

* IS(not necessarily IIS) isolated.

algorithm, based on the final simplex tableau derived for the solution of the parent node, can considerably speed up the solution of the MILP associated to the current node.

Similarly, different branching variable selection procedures have been developed. Usually, estimates of the value of the objective function, depending on the variable branched on, are used to select the most promising variable. The maximum infeasibility procedure designed by Burdet et al.[74] selects variables whose branching value violate the set of constraints the most.

Tables 11-14 list the performances of four different combinations of node and

TABLE 13. Comparison of Different B & B Strategies (Algorithm 3)

- Best-Bound Search, Branching on Max. Infeasibility Variable.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	42723	106534	558	1159	5:46
* bell3b	1	66	113(36)	645765	1387616	10000	15272	1:23:10
* bell4	15	65	43(14)	2183489	2440861	10000	11565	3:32:17
* bell5	26	53	30(40)	1232163	2086530	10000	16297	1:57:40
dell	5	382	707	9129	1151683	8	744	1:39:13
* flugpl	6	7(1)	12(10)	362263	228639	10000	8835	13:09
* g503	1	33	15(4)	207	6123	0	45	6
* misc01	12	28(1)	65(25)	5286	41600	16	213	1:08
* misc02	14	30(1)	43(7)	6970	49634	30	343	1:01
* misc03	3	31(1)	110(40)	3644	58174	14	282	2:16
* misc04	3	1186(294)	3731(2477)	19148	8139402	4	2466	19:18:57
* misc05	7	62(4)	55(32)	9598	190123	42	1084	7:59
* misc06	11	613(26)	1050(684)	45885	3217840	60	3811	6:27:48
* mod008	73	7	313(292)	334938	1131424	690	2837	1:09:33
* mod013	15	50	50(21)	17639	73554	128	608	1:45
* p0201	1	2	6(5)	1085992	7356356	332	2818	6:41:09
stein9	6	13	0	1087	2181	44	86	3
stein15	6	36	0	3776	15708	92	340	19
stein27	14	109	0	123042	567554	1218	5517	18:33
stein45	16	227	4	998512	10169768	1112	11298	7:55:33

* IS(not necessarily IIS) isolated.

branching variable selection strategies: depth-first versus best-bound node selection and maximum-infeasibility versus best-bound branching variable selection. Table 15 lists the relative rankings of the different policies.

The criteria used to compare the different strategies are the quality of the isolation, i.e. how large are the ISs relative to the smallest one, the overall number of nodes to proceed to the isolation and the overall time required to carry out the filtering algorithm.

The inspection of the performances of the different combinations on the original proof of infeasibility of the initial inconsistent MILP stresses the importance of an

TABLE 14. Comparison of Different B & B Strategies (Algorithm 3)

• Best-Bound Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
	IRs	LCs	BDs	IIS Isolation		Inf. Detection		Time h:m:s
				Nodes	LP It.	Nodes	LP It.	
* bell3a	7	68	51(26)	7298	44850	36	190	1:15
* bell3b	21	92	43(24)	1182918	1958119	2	127	2:27:44
* bell4	15	65	43(14)	1214974	1326922	2	110	1:59:27
* bell5	17	76	79(30)	748679	1499837	2	108	1:25:17
dell	5	381	764	7515	1134112	6	767	1:34:34
* flugpl	6	8(1)	12(11)	349267	208073	5012	3623	12:32
* g503	1	33	15(4)	213	6180	0	45	6
* misc01	12	28(1)	65(25)	4956	41985	28	242	59
* misc02	14	30(1)	43(7)	6653	46669	34	326	56
* misc03	3	31(1)	110(40)	2982	53854	6	214	2:00
* misc04	3	1186(294)	3731(2477)	20397	8199965	2	2392	19:35:18
* misc05	7	62(4)	55(32)	9498	188242	50	991	7:53
* misc06	11	613(26)	1017(673)	42994	3156894	40	3072	6:21:09
* mod008	73	7	313(292)	604813	2102641	956	3949	2:01:18
* mod013	15	50	50(21)	16602	73785	124	604	1:43
* p0201	1	2	6(5)	914276	6799952	5946	56448	6:21:18
stein9	6	13	0	1087	2181	44	86	3
stein15	6	36	0	3776	15708	92	340	19
stein27	14	109	0	123042	567554	1218	5517	18:17
stein45	16	227	4	998512	10169768	1112	11298	7:52:18

* IS(not necessarily IIS) isolated.

efficient branching variable strategy. The "Maximum Infeasibility" branching scheme is not able to recognize the infeasibility of several of the initial inconsistent MILPs within the 10,000 upper boundary on the number of nodes. However, since the default branching strategy recognizes the inconsistency of these problems in a finite number of nodes, the case of Figure 4 does not characterize these problems. As constraints are filtered out in the deletion process, the two strategies based on this variable selection procedure are even less likely to be able to establish inconsistency of the test MILPs. These methods exhibit therefore a poor quality of isolation. Moreover, the mediocre result of the best-bound node selection combined with the default branching scheme

TABLE 15. Rankings of Different B & B Strategies (Algorithm 3)

Method 1: Depth-First Search, Branching on Max. Infeasibility Variable

Method 2: Depth-First Search, Default Branching Scheme

Method 3: Best-Bound Search, Branching on Max. Infeasibility Variable

Method 4: Best-Bound Search, Default Branching Scheme

Infeasible MILP	Method 1			Method 2			Method 3			Method 4		
	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time
bell3a	1	4	3	1	1	1	1	3	2	1	2	1
bell3b	1	1	2	2	4	3	1	2	1	3	3	4
bell4	1	4	3	2	2	2	2	3	4	2	1	1
bell5	3	3	2	1	2	3	4	4	4	2	1	1
dell	1	3	3	1	1	2	1	4	4	1	2	1
flugpl	1	4	2	1	3	1	2	2	4	3	1	3
g503	1	1	1	1	2	1	1	1	1	1	2	1
misc01	1	3	2	1	1	1	1	3	4	1	2	1
misc02	1	4	4	1	1	1	1	3	3	1	2	2
misc03	1	2	2	1	1	2	1	2	2	1	1	1
misc04	1	1	2	1	3	4	1	2	1	1	4	3
misc05	1	4	2	1	1	1	1	2	4	1	2	1
misc06	2	4	4	1	2	3	2	3	2	1	1	1
mod008	1	2	1	1	3	2	1	1	3	1	4	4
mod013	1	3	1	1	2	3	1	4	2	1	1	1
p0201	1	3	2	1	1	1	1	4	4	1	2	3
stein9	1	2	1	1	2	1	1	1	1	1	1	1
stein15	1	2	1	1	2	1	1	1	1	1	1	1
stein27	1	2	3	1	2	4	1	1	2	1	1	1
stein45	1	2	1	1	2	2	1	1	4	1	1	3
Avg	1.15	2.70	2.10	1.10	1.90	1.95	1.30	2.35	2.65	1.35	1.75	1.75

demonstrates that the best bound node selection is not suitable either.

A depth-first node selection scheme, combined with a best-bound variable selection procedure leads to the most satisfactory results in terms of isolation, although it ranked poorly in the isolation times column. However, this is likely to have a rather restricted impact on the rest of the experiments since the overall time gap for the whole set of inconsistent formulations is fairly negligible.

10.1.2 Using Past Isolation Results.

The subset identified by any of the algorithms previously described depends on

TABLE 16. Using Priority Lists based on Previous IIS Isolations

• Depth-First Search, Branching on Maximum-Infeasibility Variable.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	6288	43007	20	158	1:08
* bell3b	8	75	77(48)	1485220	2498866	10000	18622	2:33:38
* bell4	15	65	43(14)	1626128	1744661	7314	7936	2:26:35
* bell5	5	48	88(28)	1191338	2413187	10000	25386	1:54:29
dell	5	381	764	7193	1133329	10	874	1:35:01
* flugpl	6	6	6(6)	424553	248696	10000	8261	12:24
* g503	1	33	15(4)	213	6180	0	45	6
* misc01	12	28(1)	65(25)	4877	41269	24	215	59
* misc02	14	30(1)	43(7)	6632	46312	46	382	56
* misc03	3	31(1)	110(40)	2872	49974	6	185	1:50
* misc04	3	1186(294)	3731(2477)	20409	8200622	6	244	19:39:34
* misc05	7	62(4)	55(32)	6464	150969	14	436	5:33
* misc06	11	613(26)	1017(684)	40870	3088518	28	2843	6:07:20
* mod008	73	7	313(292)	570532	2004465	506	2366	1:54:57
* mod013	15	50	50(21)	17168	75948	162	782	1:47
* pu201	1	2	6(5)	1076	59452	2	234	1:38
stein9	6	13	0	1110	2222	48	89	4
stein15	6	36	0	3788	15758	80	312	19
stein27	14	109	0	139650	628360	1238	5588	19:38
stein45	16	227	4	1034043	10127078	1054	11418	7:40:25

* IS(not necessarily IIS) isolated.

the ordering of the constraints when several IISs are present in the original infeasible MILP. Thus, different IISs are likely to be isolated if the algorithm is run on a reordered set of constraints. However, information based on the previous isolation may be used and may help to focus the search of the B & B trees. Indeed one may choose to give priority to branching on variables that have already been identified in previous IISs. Priority files, based on the integer part of the IIS, can be input into the MILP solver.

Table 16 lists the performance of a straight deletion filter with such priorities.

This approach may lead to disappointing results for two main reasons. First,

2 of/de 2

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

```

Input      :   An infeasible MILP or ILP  $P_0$ , a blocking size  $BlockSize$ 
Step   0 :   IF  $LP_0$  inconsistent GOTO Step 2
Step   1 :   a. SET  $k := 1$ ,  $Blocking := ON$ 
               b. WHILE  $k \leq |IR|$  DO
                   IF  $Blocking$  THEN
                       IF  $T \setminus \{ir_k, \dots, ir_{k+BlockSize-1}\}$  inconsistent THEN
                           SET  $T := T \setminus \{ir_k, \dots, ir_{k+BlockSize-1}\}$ 
                           SET  $k := k + BlockSize$ 
                       ELSE
                           SET  $Blocking := OFF$ 
                   ELSE
                       IF  $T \setminus \{ir_k\}$  inconsistent THEN
                           SET  $T := T \setminus \{ir_k\}$ 
                           SET  $Blocking := ON$ 
                           SET  $k := k + 1$ 
Step   2 :   a. SET  $k := 1$ ,  $Blocking := ON$ 
               b. WHILE  $k \leq |LC|$  DO
                   IF  $Blocking$  THEN
                       IF  $T \setminus \{lc_k, \dots, lc_{k+BlockSize-1}\}$  inconsistent THEN
                           SET  $T := T \setminus \{lc_k, \dots, lc_{k+BlockSize-1}\}$ 
                           SET  $k := k + BlockSize$ 
                       ELSE
                           SET  $Blocking := OFF$ 
                   ELSE
                       IF  $T \setminus \{lc_k\}$  inconsistent THEN
                           SET  $T := T \setminus \{lc_k\}$ 
                           SET  $Blocking := ON$ 
                           SET  $k := k + 1$ 

```

ALGORITHM 10. (IR-LC-BD) Optimistic Blocking Deletion Filter (part 1)

the integer part of the IIS isolated based on the new ordering of the constraints may not have anything in common with the one encoded in the priority file, thus these priorities may dramatically mislead the B & B tree searches, as illustrated by the “bell*” series. An increase or decrease in the number of nodes required to prove the original MILP infeasible is likely to reveal a similar tendency for the whole filtering

```

Step 3 :   a. SET  $BD_1 = BD_0 \setminus \{ \text{BDs on variables not in } lc \in LC \}$ 
             b. SET  $T := (T \setminus BD_0) \cup BD_1$ 
             c. SET  $k := 1$ .  $Blocking := ON$ 
             d. WHILE  $k \leq |BD_1|$  DO
                 IF  $Blocking$  THEN
                     IF  $T \setminus \{bd_k, \dots, bd_{k+BlockSize-1}\}$  inconsistent THEN
                         SET  $T := T \setminus \{bd_k, \dots, bd_{k+BlockSize-1}\}$ 
                         SET  $k := k + BlockSize$ 
                     ELSE
                         SET  $Blocking := OFF$ 
                 ELSE
                     IF  $T \setminus \{bd_k\}$  inconsistent THEN
                         SET  $T := T \setminus \{bd_k\}$ 
                         SET  $Blocking := ON$ 
                     SET  $k := k + 1$ 

Step 4 :   STOP: T is an IIS

```

ALGORITHM 10. (IR-LC-BD) Optimistic Blocking Deletion Filter (part 2)

process, e.g. proving the inconsistency of “p0201” with priorities costs only **2** nodes, and **224** without any indication, accordingly, the overall number of nodes is slashed down roughly by a factor of 200: the filtering takes less than **2** minutes compared to the original **5** hours. Secondly, among the IRs belonging to the priority file, IRs appearing first are branched on first: this arbitrary ordering may be far from optimal.

10.2 Grouping Constraints

As an attempt to reduce the overall number of MILP solutions to perform an isolation, constraints can be grouped in blocks of fixed or varying size. In the incremental filter for instance, constraints can be added by blocks to the temporary set T until inconsistency is reached. Then this last block is removed and each constraint of the block is added separately: the first constraint that renders T infeasible is moved to I . This grouping procedure can be adapted to the other algorithms as well. A

blocking deletion filtering algorithm is described below.

10.2.1 Fixed-Size Blocks.

TABLE 17. Fixed-Size Block Deletion Filter (Algorithm 10)

- Fixed Size Blocks of 4 Constraints.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
	IRs	LCs	BDs	IIS Isolation		Inf. Detection		Time h:m:s
				Nodes	LP It.	Nodes	LP It.	
* bell3a	7	68	51(26)	4419	34287	36	190	53
* bell3b	8	67	70(48)	638997	1120247	2	127	1:03:43
* bell4	15	65	43(14)	617194	678039	2	110	54:11
* bell5	6	53	47(28)	401381	864177	2	108	39:58
dell	5	278	652	9636	1485209	6	767	1:51:01
* flugpl	6	6	6(6)	284722	166838	5718	4248	8:12
* g503	1	33	15(4)	181	5682	0	45	6
* misc01	12	28(1)	59(20)	3970	34031	28	242	47
* misc02	14	30(1)	54(12)	5680	39222	34	326	46
* misc03	3	31(1)	110(40)	1895	29008	6	214	58
* misc04	3	1186(294)	3731(2477)	18465	9910283	2	2392	27:54:59
* misc05	7	62(4)	54(32)	3549	91106	52	1007	3:17
* misc06	11	613(26)	1047(680)	24302	3033869	58	3774	5:07:27
* mod008	73	7	313(293)	358084	1225822	926	3832	1:11:43
* mod013	15	50	51(21)	13255	64403	130	681	1:28
* p0201	1	2	6(5)	283867	1933913	224	2463	1:36:26
stein9	6	13	0	607	1390	44	86	2
stein15	6	36	0	2559	11353	92	340	14
stein27	14	109	0	81138	372467	1218	5517	11:53
stein45	16	227	4	761809	7436935	1112	11298	5:40:20

* IS(not necessarily IIS) isolated.

In this section we consider blocks of constraints with a fixed size, pre-determined by the modeller. Algorithm 10 describes a blocking algorithm based on the (**IR-LC-BD**) deletion filter. The same modifications could be applied to the incremental (Algorithms 3 and 4) and incremental/deletion techniques (Algorithms 6 and 7).

Table 17 lists the performances of a deletion filter using blocks of 4 constraints.

TABLE 18. Fixed-Size Block Incremental Filter

- Fixed Size Blocks of 4 Constraints.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:mm:ss
* bell3a	7	68	188	8387	95822	24	397	3:09
* bell3b	1	65	188	756	49090	10000	32704	1:50
* bell4	1	65	164	136080	248065	1330	3171	15:33
* bell5	1	49	148	440	26591	10000	33303	54
dell	5	184	825	27941	2732549	2	563	4:28:48
* flugpl	8	13	11	1057659	708535	3501	4472	32:22
* g503	1	33	22	198	10304	0	45	18
* misc01	12	28	165	3791	48057	60	816	1:22
* misc02	14	30	115	27140	244418	38	437	0:03
* misc03	3	31	319	885	29311	18	571	1:58
misc04	kil(a)							
* misc05	7	62	199	4306	414494	80	4866	18:54
misc06	kil(b)							
* mod008	73	7	638	316757	1370087	526	3443	1:09:12
* mod013	15	50	58	25273	263259	104	1294	5:32
* p0201	1	2	6	324	10084	2	101	43
stein9	6	13	0	1189	2705	38	78	6
stein15	6	36	0	5707	23858	88	329	36
stein27	14	109	0	264429	1365215	1334	6930	39:52
stein45	kil(c)							

* IS(not necessarily IIS) isolated.

kil: killed after (a)200, (b)65 and (c)30 hours.

Using blocks of identical size, Table 18 and 19 list the performance of blocking incremental and incremental/deletion filters.

When deletion filtering, if the test set $T \setminus \{ir_k, \dots, ir_{k+BlockSize-1}\}$ happens to be inconsistent, then all the $\{ir_k, \dots, ir_{k+BlockSize-1}\}$ constraints can be discarded from T . However, when, on the other hand, it is feasible, *Blocking* is turned *OFF* and $T \setminus \{ir_k\}$ has to be tested before allowing to block constraints again (SET *Blocking* = *ON*). Thus, computational overhead, due to unsuccessful blocking attempts, may

TABLE 19. Fixed-Size Block Incremental/Deletion Filter

- Fixed Size Blocks of 4 Constraints.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	52(1)	6007	66649	24	307	1:43
* bell3b	1	65	56(8)	512	25249	10000	32704	25
* bell4	13	65	44(1)	133358	230260	1330	3171	14:54
* bell5	1	49	49(2)	418	15646	10000	33303	15
* dell	5	334	580(1)	10894	1363997	2	563	1:51:18
* flugpl	8	13(4)	7(5)	407947	279441	3501	4472	13:16
* g503	1	33	16(1)	164	5642	0	45	5
* misc01	12	28	69(1)	8457	93938	60	816	2:42
* misc02	14	30	54(3)	11981	120882	38	437	2:34
* misc03	3	31	114(5)	2199	41674	16	571	1:50
* misc04	3	1186	3731(2)	22674	16760623	6	3335	36:37:31
* misc05	7	62	59(11)	3371	202536	80	4866	8:59
* misc06	4	605	1042(4)	9844	3368624	54	5289	4:45:01
* mod008	37	7	640	105213	732435	526	3443	37:49
* mod013	15	50	100(65)	22108	218149	104	1294	4:52
* p0201	1	2	7(1)	477	23014	2	101	1:55
stein9	6	13	0	743	1834	38	78	3
stein15	6	36	0	2704	11414	88	329	16
* stein27	14	109	18(18)	76224	402574	1334	6930	11:55
* stein45	16	227	34(30)	756430	7755206	1090	13532	6:03:22

* IS(not necessarily IIS) isolated.

arise.

A conservative algorithm (Algorithm 11) where the *Blocking* flag is set back *ON* only once *BlockSize* constraints have been individually filtered has been implemented and tested as well. Results are listed in Table 20.

With both algorithms, if the IIS isolated is relatively small compared to the original infeasible MILP, successful blockings are likely to make up for the unsuccessful ones: the overall process is sped up ("bell*", "p0201"). But, if the IIS is fairly large,

Input : An infeasible MILP or ILP P_0 , a blocking size $BlockSize$

Step 0 : IF LP_0 inconsistent GOTO **Step 2**

Step 1 : a. SET $k := 1$, $NoBlockSize := 0$
b. WHILE $k \leq |IR|$ DO
IF $NoBlockSize = 0$ THEN
IF $T \setminus \{ir_k, \dots, ir_{k+BlockSize-1}\}$ inconsistent THEN
SET $T := T \setminus \{ir_k, \dots, ir_{k+BlockSize-1}\}$
SET $k := k + BlockSize$
ELSE
SET $NoBlockSize := BlockSize$
ELSE
SET $NoBlockSize := NoBlockSize - 1$
IF $T \setminus \{ir_k\}$ inconsistent THEN SET $T := T \setminus \{ir_k\}$
SET $k := k + 1$

Step 2 : a. SET $k := 1$, $NoBlockSize := 0$
b. WHILE $k \leq |LC|$ DO
IF $NoBlockSize = 0$ THEN
IF $T \setminus \{lc_k, \dots, lc_{k+BlockSize-1}\}$ inconsistent THEN
SET $T := T \setminus \{lc_k, \dots, lc_{k+BlockSize-1}\}$
SET $k := k + BlockSize$
ELSE
SET $NoBlockSize := BlockSize$
ELSE
SET $NoBlockSize := NoBlockSize - 1$
IF $T \setminus \{lc_k\}$ inconsistent THEN
SET $T := T \setminus \{lc_k\}$
SET $k := k + 1$

Step 3 : a. SET $BD_1 := BD_0 \setminus \{BDs \text{ on variables not in } lc \in LC\}$
b. SET $T := (T \setminus BD_0) \cup BD_1$
c. SET $k := 1$, $NoBlockSize := 0$
d. WHILE $k \leq |BD_1|$ DO
IF $NoBlockSize = 0$ THEN
IF $T \setminus \{bd_k, \dots, bd_{k+BlockSize-1}\}$ inconsistent THEN
SET $T := T \setminus \{bd_k, \dots, bd_{k+BlockSize-1}\}$
SET $k := k + BlockSize$
ELSE
SET $NoBlockSize := BlockSize$
ELSE
IF $T \setminus \{bd_k\}$ inconsistent THEN SET $T := T \setminus \{bd_k\}$
SET $k := k + 1$

Step 4 : STOP: **T** is an IIS

TABLE 20. Fixed-Size Block Deletion Filter (Algorithm 11)

- Fixed Size Blocks of 4 Constraints.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	4223	29575	36	190	48
* bell3b	8	73	65(49)	818253	1401317	2	127	1:22:00
* bell4	15	65	43(14)	665881	724382	2	110	58:24
* bell5	6	53	48(32)	460922	956234	2	108	45:35
dell	5	377	596	8154	1205454	6	767	1:40:05
* flugpl	6	6	6(6)	334498	192472	5718	4248	9:33
* g503	1	33	15(4)	159	4726	0	45	5
* misc01	12	28(1)	65(25)	3763	31519	28	242	44
* misc02	14	30(1)	54(12)	5416	36774	34	326	44
* misc03	3	31(1)	110(40)	1836	27447	6	214	56
* misc04	3	1186(294)	3731(2477)	15274	7263993	2	2392	19:16:31
* misc05	7	62(4)	55(32)	3867	88612	52	1007	3:18
* misc06	11	613(26)	1019(686)	25901	2534418	58	3774	4:37:11
* mod008	73	7	313(293)	377935	1305983	926	3832	1:15:39
* mod013	15	50	50(21)	13639	63361	130	681	1:28
* p0201	1	2	6(5)	288547	1986777	224	2463	1:38:22
stein9	6	13	0	605	1315	44	86	2
stein15	6	36	0	2436	10401	92	340	13
stein27	14	109	0	76689	47969	1218	5517	10:59
stein45	16	227	4	683225	6708833	1112	11298	5:07:51

* IS(not necessarily IIS) isolated.

then the cost attached to these failures is likely to overcome the gains and even to deteriorate the overall cost of the isolation ("dell", "misc06"). However, Algorithm 11 outperforms Algorithm 10 on large, dense IISs since the number of attempts to group constraints is reduced: performances no worse and in most cases far better than the straight deletion filter are obtained.

10.2.2 Adaptive-Size Blocks.

The preceding deletion filtering algorithms which attempt to delete a fixed num-

ber of constraints at each iteration can be modified to use blocks of varying size.

Four different adaptive-blocking strategies, listed below, have been tested on the set of inconsistent MILPs. Two adaptive blocking strategies modify the size of the chunks of constraints by increments of 2 (“Additive Adaptive Blocking Strategy” (1) and (2)), the other two multiply and divide by a factor of 2 the size of the blocks (“Multiplicative Adaptive Blocking Strategy” (1) and (2)). See Algorithm 12.

Additive Adaptive Blocking Strategy (1)

```

SET BlockSize := 2
IF Test-MILP inconsistent THEN BlockSize := BlockSize + 2
                               ELSE BlockSize := MAX(BlockSize - 2, 0).

```

Additive Adaptive Blocking Strategy (2)

```

SET BlockSize := 2
IF Test-MILP inconsistent THEN BlockSize := BlockSize + 2
                               ELSE BlockSize := 2.

```

Multiplicative Adaptive Blocking Strategy (1)

```

SET BlockSize := 1
IF Test-MILP inconsistent THEN BlockSize := BlockSize * 2
                               ELSE BlockSize := MAX(BlockSize/2, 1).

```

Multiplicative Adaptive Blocking Strategy (2)

```

SET BlockSize := 1
IF Test-MILP inconsistent THEN BlockSize := BlockSize * 2
                               ELSE BlockSize := 1.

```

ALGORITHM 12. Examples of Different Adaptive Blocking Strategies

Remark: A blocking size equal to 0 or 1 actually means that no blocking occurs.

Results of these experiments are listed in Tables 21–24 for the “optimistic” approach to blocking constraints, i.e. Algorithm 10, and in tables 25–28 for the more

conservative one, i.e. Algorithm 11. Rankings of the different techniques are provided in Tables 29 and 30.

TABLE 21. Additive Blocking Size Deletion Filter (Algorithm 10)

- Additive Adaptive Size Blocks (1): 2-Constraint Increments.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	4219	33790	36	190	53
* bell3b	8	67	67(52)	816334	1553868	2	127	1:24:49
* bell4	15	65	43(14)	602075	659901	2	110	56:55
* bell5	6	53	50(38)	520966	1036333	2	108	50:38
dell	5	385	728	10459	1764168	6	767	2:22:50
* flugpl	6	6	6(6)	335506	195474	5718	4248	10:06
* g503	1	33	15(4)	196	5703	0	45	6
* misc01	12	28(1)	60(20)	3942	30706	28	242	43
* misc02	14	30(1)	54(8)	6231	42822	34	326	53
* misc03	3	31(1)	112(41)	1486	24495	6	214	50
* misc04	3	1186(294)	3729(2477)	18582	9888191	2	2392	27:10:10
* misc05	7	62(4)	53(32)	4060	92224	52	1007	3:15
* misc06	11	613(26)	1048(680)	24997	3155833	58	3774	5:12:13
* mod008	73	7	313(291)	468931	1630219	926	3832	1:35:12
* mod013	15	50	48(20)	13058	64649	130	681	1:30
* p0201	1	2	6(5)	105754	738017	224	2463	38:06
stein9	6	13	0	650	1487	44	86	3
stein15	6	36	0	2487	11280	92	340	14
stein27	14	109	0	71909	330352	1218	5517	10:32
stein45	16	227	4	680578	6739158	1112	11298	5:12:47

* IS(not necessarily IIS) isolated.

With both types, i.e. additive and multiplicative, of adaptive blocking strategies, strategies which reset the blocking size upon reaching an inconsistent test set provide lower quality outputs but require a shorter execution time. Moreover, additive blocking strategies exhibit a better accuracy and take less time to complete the filtering. As in the fixed-size case, the “pessimist” approach guarantees overall execution times

TABLE 22. Additive Blocking Size Deletion Filter (Algorithm 10)

- Additive Adaptive Size Blocks (2): 2-Constraint Increments.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	4480	34932	36	190	55
* bell3b	8	69	65(47)	677623	1177529	2	127	1:08:37
* bell4	15	65	43(14)	544327	601626	2	110	49:48
* bell5	6	53	47(37)	381052	814784	2	108	38:11
dell	5	355	594	8642	1516530	6	767	2:00:19
* flugpl	6	6	6(6)	265507	159132	5718	4248	8:09
* g503	1	33	15(4)	189	5723	0	45	6
* misc01	12	28(1)	60(20)	3920	32723	28	242	45
* misc02	14	30(1)	43(8)	6327	43122	34	326	52
* misc03	3	31(1)	110(40)	1978	27708	6	214	54
* misc04	3	1186(294)	3731(2477)	19148	10062463	2	2392	28:07:31
* misc05	7	62(4)	54(32)	3463	89813	52	1007	3:10
* misc06	11	613(26)	1026(692)	22392	2943773	58	3774	4:55:24
* mod008	73	7	313(291)	371209	1271658	926	3832	1:15:05
* mod013	15	50	50(21)	12675	62802	130	681	1:26
* p0201	1	2	6(5)	117542	821086	224	2463	42:04
stein9	6	13	0	650	1487	44	86	3
stein15	6	36	0	2181	10071	92	340	13
stein27	14	109	0	79374	363352	1218	5517	11:31
stein45	16	227	4	706436	6936678	1112	11298	5:21:27

* IS(not necessarily IIS) isolated.

lower than the straight deletion filter but may not take full advantage of relatively sparse IISs and exhibit lower performance than the “optimist” one.

TABLE 23. Multiplicative Blocking Size Deletion Filter (Algorithm 10)

- Multiplicative Adaptive Size Blocks (1).
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h.m.s
* bell3a	7	68	51(26)	4592	35048	36	190	59
* bell3b	8	63	96(30)	1316434	2003565	2	127	2:04:25
* bell4	15	65	43(14)	559441	615823	2	110	50:42
* bell5	6	53	79(29)	571047	1131841	2	108	55:34
dell	5	199	354	12835	1295031	6	767	1:36:02
* flugpl	6	6	6(6)	281664	166460	5718	4248	9:08
* g503	1	33	15(4)	227	6608	0	45	7
* misc01	12	28(1)	60(20)	4246	37124	28	242	53
* misc02	14	30(1)	43(8)	5643	38878	34	326	49
* misc03	3	31(1)	119(41)	1668	27121	6	214	56
* misc04	3	1186(294)	3729(2477)	19966	10268902	2	2392	28:01:04
* misc05	7	62(4)	54(32)	6300	124940	52	1007	4:59
* misc06	11	613(26)	1213(662)	34631	3585538	58	3774	6:21:30
* mod008	73	7	313(292)	521178	1821633	926	3832	1:45:36
* mod013	15	50	48(20)	14987	71276	130	681	1:39
* p0201	1	2	6(5)	80928	591479	224	2463	31:42
stein9	6	13	0	619	1444	44	86	3
stein15	6	36	0	2201	9991	92	340	13
stein27	14	109	0	77848	358220	1218	5517	11:21
stein45	16	227	4	726107	7158141	1112	11298	5:31:42

* IS(not necessarily IIS) isolated.

TABLE 24. Multiplicative Blocking Size Deletion Filter (Algorithm 10)

- Multiplicative Adaptive Size Blocks (2).
- Depth-First Search. Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	4252	39319	36	190	1:03
* bell3b	8	71	68(50)	818206	1338531	2	127	1:21:16
* bell4	15	65	43(14)	529138	588114	2	110	53:54
* bell5	6	53	47(34)	541127	1138178	2	108	55:17
dell	5	367	722	10606	1661456	6	767	2:10:47
* flugpl	6	6	6(6)	271666	160556	5718	4248	9:04
* g503	1	33	15(4)	214	6472	0	45	7
* misc01	12	28(1)	65(25)	4475	38225	28	242	56
* misc02	14	30(1)	43(8)	5920	40195	34	326	50
* misc03	3	31(1)	110(40)	1875	26770	6	214	55
* misc04	3	1186(294)	3731(2477)	21427	10648845	2	2392	29:02:23
* misc05	7	62(4)	55(32)	4892	109685	52	1007	4:14
* misc06	11	613(26)	1001(687)	25198	3064040	58	3774	5:08:02
* mod003	73	7	313(292)	386287	1326724	926	3832	1:17:50
* mod013	15	50	50(21)	13546	66160	130	681	1:39
* p0201	1	2	6(5)	90258	656335	224	2463	34:47
stein9	6	13	0	634	1472	44	86	3
stein15	6	36	0	2049	9401	92	340	12
stein27	14	109	0	79374	364209	1218	5517	11:26
stein45	16	227	4	894888	8814284	1112	11298	6:43:46

* IS(not necessarily IIS) isolated.

TABLE 25. Additive Blocking Size Deletion Filter (Algorithm 11)

- Additive Adaptive Size Blocks (1): 2-Constraint Increments.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	mins
* bell3a	7	68	51(26)	4231	30274	36	190	48
* bell3b	8	69	129(30)	1217124	1878375	2	127	1:59:15
* bell4	15	65	43(14)	659812	717575	2	110	1:03:20
* bell5	6	53	52(37)	620879	1221350	2	108	1:00:50
dell	5	335	631	7355	1162409	6	767	1:34:13
* flugpl	6	6	6(6)	335725	195636	5718	4248	10:24
* g503	1	33	15(4)	214	6139	0	45	7
* misc01	12	28(1)	65(25)	3809	32098	28	242	45
* misc02	14	30(1)	54(12)	6201	42725	34	326	51
* misc03	3	31(1)	110(40)	1851	26690	6	214	56
* misc04	3	1186(294)	3731(2477)	17619	7657061	2	2392	18:48:40
* misc05	7	62(4)	55(32)	5032	105271	52	1007	3:45
* misc06	11	612(26)	1017(684)	32477	2928067	58	3774	5:23:24
* mod008	73	7	313(292)	473300	1656949	926	3832	1:36:18
* mod013	15	50	50(21)	15271	72268	130	681	1:40
* p0201	1	2	6(5)	105754	738017	224	2463	38:09
stein9	6	13	0	680	1570	44	86	3
stein15	6	36	0	2456	11239	92	340	14
stein27	14	109	0	70848	322874	1218	5517	10:19
stein45	16	227	4	638136	6296508	1112	11298	4:51:54

* IS(not necessarily IIS) isolated.

TABLE 26. Additive Blocking Size Deletion Filter (Algorithm 11)

- Additive Adaptive Size Blocks (2): 2-Constraint Increments.
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	4005	29361	36	190	48
* bell3b	8	72	98(34)	1357040	1937393	2	127	2:09:45
* bell4	15	65	43(14)	616303	673202	2	110	1:00:44
* bell5	6	53	77(26)	481090	1027630	2	108	48:40
dell	5	302	585	8339	1144732	6	767	1:28:23
* flugpl	6	6	6(6)	315371	184868	5718	4248	9:38
* g503	1	33	15(4)	183	5166	0	45	6
* misc01	12	28(1)	60(20)	3600	30117	28	242	43
* misc02	14	30(1)	43(7)	6059	41172	34	326	51
* misc03	3	31(1)	110(40)	1883	26396	6	214	54
* misc04	3	1186(294)	3730(2477)	16576	7803579	2	2392	20:39:05
* misc05	7	62(4)	54(32)	4065	91622	52	1007	3:20
* misc06	11	613(26)	1041(679)	23763	2527265	58	3774	4:25:57
* mod008	73	7	313(292)	386050	1334119	926	3832	1:17:48
* mod013	15	50	50(21)	13557	63002	130	681	1:30
* p0201	1	2	6(5)	132284	936226	224	2463	48:14
stein9	6	13	0	641	1408	44	86	3
stein15	6	36	0	2351	10287	92	340	13
stein27	14	109	0	71871	327921	1218	5517	10:28
stein45	16	227	4	638558	6296999	1112	11298	4:53:01

* IS(not necessarily IIS) isolated.

TABLE 27. Multiplicative Blocking Size Deletion Filter (Algorithm 11)

- Multiplicative Adaptive Size Blocks (1).
- Depth-First Search. Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	4793	31522	36	190	54
* bell3b	8	72	101(29)	1416367	2114756	2	127	2:23:12
* bell4	15	65	43(14)	839709	906995	2	110	1:18:25
* bell5	6	53	65(35)	570824	1254106	2	108	57:20
dell	5	295	610	8345	1140819	6	767	1:31:49
* flugpl	6	6	6(6)	335637	193750	5718	4248	10:24
* g503	1	33	15(4)	215	6004	0	45	7
* misc01	12	28(1)	60(20)	4309	36479	28	242	54
* misc02	14	30(1)	43(8)	5475	36848	34	326	45
* misc03	3	31(1)	110(40)	1977	27351	6	214	55
* misc04	3	1186(294)	3731(2477)	19399	8037958	2	2392	19:28:05
* misc05	7	62(4)	55(32)	6975	129206	52	1007	5:00
* misc06	11	613(26)	1017(684)	38410	3006137	58	3774	5:49:53
* mod008	73	7	313(292)	503666	1766492	926	3832	1:41:34
* mod013	15	50	50(21)	15229	69848	130	681	1:39
* p0201	1	2	6(5)	80928	591230	224	2463	31:45
stein9	6	13	0	599	1338	44	86	3
stein15	6	36	0	2020	8951	92	340	11
stein27	14	109	0	75092	341303	1218	5517	10:54
stein45	16	227	4	723316	7115696	1112	11298	5:29:41

* IS(not necessarily IIS) isolated.

TABLE 28. Multiplicative Blocking Size Deletion Filter (Algorithm 11)

- Multiplicative Adaptive Size Blocks (2).
- Depth-First Search, Default Branching Scheme.

Infeasible MILP	IIS Characteristics			Efficiency				
				IIS Isolation		Inf. Detection		Time
	IRs	LCs	BDs	Nodes	LP It.	Nodes	LP It.	h:m:s
* bell3a	7	68	51(26)	5144	35859	36	190	58
* bell3b	8	71	72(50)	1028845	1799289	2	127	1:46:29
* bell4	15	65	43(14)	685144	746675	2	110	1:04:17
* bell5	6	53	49(34)	580966	1231827	2	108	59:24
dell	5	307	493	11525	1418075	6	767	1:55:03
* flugpl	6	6	6(6)	321537	187133	5718	4248	10:08
* g503	1	33	15(4)	196	5776	0	45	7
* misc01	12	28(1)	60(20)	4237	35758	28	242	54
* misc02	14	30(1)	43(8)	5448	36969	34	326	46
* misc03	3	31(1)	110(40)	1927	27907	6	214	56
* misc04	3	1186(294)	3731(2477)	19261	8808667	2	2392	23:11:06
* misc05	7	62(4)	55(32)	5798	120252	52	1007	4:40
* misc06	11	613(26)	993(683)	29421	2884728	58	3774	5:10:23
* mod008	73	7	313(292)	441434	1536843	926	3832	1:29:02
* mod013	15	50	50(21)	13512	63284	130	681	1:30
* p0201	1	2	6(5)	90258	656086	224	2463	34:43
stein9	6	13	0	599	1338	44	86	3
stein15	6	36	0	1906	8535	92	340	11
stein27	14	109	0	76227	347421	1218	5517	11:04
stein45	16	227	4	805467	7937648	1112	11298	6:06:24

* IS(not necessarily IIS) isolated.

TABLE 29. Rankings of Adaptive Blocking Strategies (Algorithm 10)**Method 1:** Additive Blocking Strategy (1)**Method 2:** Additive Blocking Strategy (2)**Method 3:** Multiplicative Blocking Strategy (1)**Method 4:** Multiplicative Blocking Strategy (2)

Infeasible MILP	Method 1			Method 2			Method 3			Method 4		
	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time
bell3a	1	1	1	1	2	2	1	3	3	1	4	4
bell3b	2	2	3	3	1	1	1	4	4	4	3	2
bell4	1	3	4	1	4	1	1	2	2	1	1	3
bell5	2	2	2	1	1	1	3	4	4	1	3	3
dell	- *	-	-	-	-	-	-	-	-	-	-	-
flugpl	1	4	4	1	1	1	1	3	3	1	2	2
g503	1	2	1	1	1	1	1	4	2	1	3	2
misc01	2	2	1	2	1	2	2	3	3	1	4	4
misc02	1	3	3	1	4	4	1	1	1	1	2	2
misc03	2	1	1	1	4	2	3	2	4	1	3	3
misc04	1	1	1	2	2	2	1	3	3	2	4	4
misc05	1	2	2	2	1	1	2	4	4	3	3	3
misc06	2	2	3	1	1	1	4	4	4	3	3	2
mod008	1	3	3	1	1	1	1	4	4	1	2	2
mod013	1	2	2	2	1	1	1	4	3	2	3	3
p0201	1	3	3	1	4	4	1	1	1	1	2	2
stein9	1	3	1	1	3	1	1	1	1	1	2	1
stein15	1	4	3	1	2	2	1	3	2	1	1	1
stein27	1	1	1	1	3	4	1	2	2	1	3	3
stein45	1	1	1	1	2	2	1	3	3	1	4	4
Avg	1.26	2.21	2.10	1.31	2.05	1.79	1.47	2.89	2.79	1.47	2.73	2.63

* Different IISs were isolated.

TABLE 30. Rankings of Adaptive Blocking Strategies (Algorithm 11)**Method 1: Additive Blocking Strategy (1)****Method 2: Additive Blocking Strategy (2)****Method 3: Multiplicative Blocking Strategy (1)****Method 4: Multiplicative Blocking Strategy (2)**

Infeasible MILP	Method 1			Method 2			Method 3			Method 4		
	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time
bell3a	1	2	1	1	1	1	1	3	3	1	4	2
bell3b	1	2	2	3	3	3	4	4	4	2	1	1
bell4	1	2	2	1	1	1	1	3	4	1	4	3
bell5	2	4	4	4	1	1	3	2	2	1	3	3
dell	- *	-	-	-	-	-	-	-	-	-	-	-
flugpl	1	4	3	1	1	1	1	3	3	1	2	2
g503	1	3	2	1	1	1	1	4	2	1	2	2
misc01	2	2	2	1	1	1	1	4	3	1	3	3
misc02	2	4	3	1	3	3	1	2	1	1	1	2
misc03	1	1	3	1	2	1	1	4	2	1	3	3
misc04	2	2	1	1	1	3	2	4	2	2	3	4
misc05	2	2	2	1	1	1	2	4	4	2	3	3
misc06	1	3	3	4	1	1	3	4	4	2	2	2
mod008	1	3	3	1	1	1	1	4	4	1	2	2
mod013	1	4	3	1	2	1	1	3	2	1	1	1
p0201	1	3	3	1	4	4	1	1	1	1	2	2
stein9	1	3	1	1	2	1	1	1	1	1	1	1
stein15	1	4	3	1	3	2	1	2	1	1	1	1
stein27	1	1	1	1	2	2	1	3	2	1	4	3
stein45	1	1	1	1	2	2	1	3	3	1	4	4
Avg	1.26	2.63	2.26	1.42	1.74	1.63	1.47	3.05	2.52	1.21	2.42	2.32

* Different IISs were isolated.

CHAPTER 11
B & B TREE DELETION FILTERING
ALGORITHMS FOR MEDIUM-LARGE MILPS

11.1 Overview

Algorithms described in the previous chapters solve numerous, possibly in the thousands, of MILP test sets. However, no information apart from the (in)feasibility of the test set is retained from these solutions. More specifically in the case of B & B tree solutions, Section 5.4 detailed two types of information that can be gathered at marginal additional cost as the proofs proceed. Algorithms described in the present Chapter use the “Node and Leaf” as well as the “Branch” information contained in the original proof of infeasibility of the initial MILP or ILP to speed up the isolation process.

The set of variables that take an integer value at any intermediate node of the original B & B tree can certainly not constitute the integral part of any IIS; the same remark also holds for any subset of any such combination. Hence, a pool of all the combinations reached in the proof of infeasibility is maintained. Sensitivity analyses can also be carried out at the leaves (or final nodes) of the B & B tree. They allow the identification of the LCs and BDs that are not involved in the IIS as demonstrated in Section 5.4.2.

The set of IRs that have been branched on along any path of the B & B tree may point out the IRs contained in the IIS. Some of these combinations of IRs do not have to be considered if they also appear in the set of the integral combinations attained. Moreover, the set of all of the IRs introduced in the B & B search contains the integral part of the IIS (Section 5.4.3). This last remark has strong practical implications. If

we consider Table 2 which lists the number of nodes involved in the original proofs of infeasibility of the test set of problems we can observe that more than half of the proofs required less than 31 nodes. (The number of nodes in a binary tree is given by $(2^{depth+1} - 1)$ whereas the number of possible IRs introduced is $(2^{depth} - 1)$.) Thus, in over half of the cases, at most 15 IRs have been branched on, which reduces by at least a factor 4 the number of IRs to consider, in roughly half of the problems. As for the LCs and BDs, sensitivity filtering of these paths may discard IRs that have been “needlessly” branched on.

11.2 Algorithms

All three algorithms produce a similar kind of output with two different components. A linear component identifies the LCs and BDs that are involved with the IIS associated with the B & B tree developed to prove the original MILP inconsistent. If the sensitivity filter has not been used, all LCs and BDs contained in the original formulation are identified. The integral part is made of the collection of the paths of the B & B tree, where each set of IRs indicates a superset of the potential integral part of the IIS. These combinations of IRs are sequentially tested, from the smallest to the largest one, together with the identified LCs and BDs until infeasibility is proved. If none of them satisfies this criterion, the set of all the IRs introduced in the binary search renders the formulation inconsistent.

Once the minimal path of IRs leading to inconsistency is found, an IS has been isolated. This subset can therefore be used as an advanced start in the IIS isolation.

Two main IIS identification techniques have been developed so far. Unlike deletion filtering which exhibits performances rather independent of the size of the IIS isolated, the incremental filter demonstrates a worst-case behaviour when the IIS is relatively large in the original formulation. Hence, we combine the B & B tree filtering with a deletion filter to complete the IIS isolation.

11.2.1 Notation.

The B & B Tree is represented by a stack (**S**) of nodes or subproblems (**N**). The top element of the stack can be accessed via a $POP(S) = N$ operation and a node can be added on top of the stack via a $PUSH(S, N)$. Each node **N** of the stack is represented by $N = (V, LP)$ where:

- **V** is a set of couples (variable,value) or (X_i, V_i) .
- **LP** is the LP problem associated with this node, and these values are a feasible solution of **LP**.

We define the function $Value(F, V)$ which computes the value of the objective function F with each variable's value defined in V .

We assume that two additional functions are defined by the B & B exploration strategy:

- a. $BranchVar(N)$ returns the variable that has to be branched on when developing subproblem N .
- b. $SOLVE(LP)$ returns the set **V** of (X_i, V_i) , (variable , value) optimal solution of **LP**. **LP** is the couple (objective function, all linear constraints and bounds) (F, C) .

We are also assuming that, without loss of generality, all objective functions are to be minimized.

Updated at any node of the binary search, $Pool_{IR}$ contains the set of all the integral combinations reached in the B & B tree so far. This set of combination is used to discard the paths of the tree which can not contain the integer part of the identified IIS. Moreover, $Path(V, LP)$ is the set of IRs for the branch label variables in a path derived from the root of the B & B tree to the subproblem (V, LP) .

The following three algorithms share an identical *initialization* phase listed below:

Initialization:


```

SET  $S := \emptyset$ 
SET  $V_0 := SOLVE(LP_0)$ ;   Update  $Pool_{IR}$ 
SET  $Path(V_0, LP_0) := \emptyset$ 
PUSH( $S, (V_0, LP_0)$ )
SET  $Pool_{IR} := \emptyset$ 

```

11.2.2 B & B Tree Filtering Algorithm.

Algorithm 13 described below is a B & B algorithm which keeps and updates the $Pool_{IR}$ variable as well as the set of paths as branches and nodes are developed during the original binary search.

Input : An infeasible MILP or ILP $P_0 = (IR_0, (F_0, C_0))$

Initialization

WHILE ($S \neq \emptyset$) DO

- a. SET $(V, (F, C)) := POP(S)$
- b. SET $X_i := BranchVar(V, (F, C))$, i index of node N
- c. SET $LEFTLP := (F_0, C_0 \cup \{X_i \leq \lfloor V_i \rfloor\})$
- d. SET $RIGHTLP := (F_0, C_0 \cup \{X_i \geq \lceil V_i \rceil\})$
- e. SET $LEFTV := SOLVE(LEFTLP)$; Update $Pool_{IR}$ Paths
- f. SET $RIGHTV := SOLVE(RIGHTLP)$; Update $Pool_{IR}$ Paths
- g. FOR $(V', LP') \in \{(LEFTV, LEFTLP), (RIGHTV, RIGHTLP)\}$ DO
 SET $Path(V', LP') := Path(V, LP) \cup \{X_i\}$
 IF $(V' \neq \emptyset)$ THEN
 PUSH($S, (V', LP')$)
 ELSE IF
 $Path(V', LP') \in Pool_{IR}$ THEN DISCARD $Path(V', LP')$

OUTPUT: A set of $Path(V, LP)$ (Combinations of IRs).

ALGORITHM 13. B & B Tree Filtering Algorithm

11.2.3 Sensitivity B & B Tree Filtering Algorithm.

In addition to the bookkeeping operations concerning the paths of the binary tree, Algorithm 14 performs the sensitivity filtering of LCs and BDs at the leaves of the tree. Non-marked LCs and BDs, once the tree is completed, are discarded before applying the deletion filter. Similarly, the paths of the tree are also sensitivity filtered, i.e. only the IRs whose branching introduced an inequality that is not sensitivity filtered in the processing of the corresponding leaf, are kept in the path.

Input : An infeasible MILP or ILP $P_0 = (IR_0, (F_0, C_0))$

Initialization

WHILE ($S \neq \emptyset$) DO

- a. SET $(V, (F, C)) := POP(S)$
- b. SET $X_i := BranchVar(V, (F, C))$, i index of node N
- c. SET $LEFTLP := (F_0, C_0 \cup \{X_i \leq \lfloor V_i \rfloor\})$
- d. SET $RIGHTLP := (F_0, C_0 \cup \{X_i \geq \lceil V_i \rceil\})$
- e. SET $LEFTV := SOLVE(LEFTLP)$; Update $Pool_{IR}$ Paths
- f. SET $RIGHTV := SOLVE(RIGHTLP)$; Update $Pool_{IR}$ Paths
- g. FOR $(V', LP') \in \{(LEFTV, LEFTLP), (RIGHTV, RIGHTLP)\}$ DO
 SET $Path(V', LP') := Path(V, LP) \cup \{X_i\}$
 IF $(V' = \emptyset)$ THEN
 MARK SENSITIVE LCs & BDs
 SENSITIVITY FILTERING OF $Path(V', LP')$
 IF $Path(V', LP') \in Pool_{IR}$ THEN DISCARD $Path(V', LP')$
 ELSE
 PUSH($S, (V', LP')$)

OUTPUT: A set of $Path(V, LP)$ (Combinations of IRs)
 and a set of Marked LCs and BDs.

ALGORITHM 14. Sensitivity B & B Tree Filtering Algorithm

11.2.4 Heuristic/Sensitivity B & B Tree Filtering Algorithm.

In addition to the tasks performed by the previous algorithm and in order to get as few paths as possible in the output, we try to obtain as many integer values in the intermediate nodes of the B & B tree as possible (stored in $Pool_{IR}$) which increases the chances of discarding a path in a leaf. To do so, instead of introducing constraints of the form $x \leq \lfloor \alpha \rfloor$ for instance, we introduce equality constraints such as $x + slack = \lfloor \alpha \rfloor$ and set the objective function to minimize the sum of the *slack* variables. See Algorithm 15.

11.3 Validity of the Isolations

The validity of the three algorithms described in this Chapter relies on properties in Section 5.4.2 and 5.4.3 which ensure that at least one combination or path of IRs, or eventually the set of IRs introduced in the binary search, plus the set of constraints (LCs and BDs) marked by any of these B & B tree filtering techniques is an IS. Since the deletion filter outputs an IIS when given an inconsistent set of constraints, this guarantees that the final outcome of these two combined procedures is an IIS.

11.4 Time and Space Complexities

The B & B tree filter is basically a classical B & B algorithm combined with book-keeping operations which keep track of the integer combinations obtained during the search as well as the paths developed in the tree. Hence, these three algorithms behave like a B & B algorithm and thus their efficiency mainly depends on the subproblem selection strategy and branching variable selection scheme.

Some additional memory is required to store all the lists of IRs generated by the algorithms. Each node may generate a possibly large list of IRs. However, a proper implementation of these algorithms would not store smaller lists of IRs which are contained in some other list. Such an implementation would therefore minimize the extra memory space required on top of the B & B tree, at the expense of a longer

Input : An infeasible MILP or ILP $P_0 = (IR_0, (F_0, C_0))$

Initialization

WHILE ($S \neq \emptyset$) DO

- a. SET $(V, (F, C)) := POP(S)$
- b. SET $X_i := BranchVar(V, (F, C))$, i index of node N
- c. SET $AC := C \setminus C_0$, AC stands for Added Constraints
- d. IF ($AC = \emptyset$) THEN
 - SET $LEFTLP := (slack_{2i}, C_0 \cup \{X_i + slack_{2i} = \lfloor V_i \rfloor\})$
 - SET $RIGHTLP := (slack_{2i+1}, C_0 \cup \{X_i - slack_{2i+1} = \lceil V_i \rceil\})$
 ELSE
 - SET $LEFTLP := (F_0 + slack_{2i}, C_0 \cup AC \cup \{X_i + slack_{2i} = \lfloor V_i \rfloor\})$
 - SET $RIGHTLP := (F_0 + slack_{2i+1}, C_0 \cup AC \cup \{X_i - slack_{2i+1} = \lceil V_i \rceil\})$
- e. SET $LEFTV := SOLVE(LEFTLP)$; Update $Pool_{IR}$ Paths
- f. SET $RIGHTV := SOLVE(RIGHTLP)$; Update $Pool_{IR}$ Paths
- g. FOR $(V', LP') \in \{(LEFTV, LEFTLP), (RIGHTV, RIGHTLP)\}$ DO
 - SET $Path(V', LP') := Path(V, LP) \cup \{X_i\}$
 - IF ($V' = \emptyset$) THEN
 - MARK SENSITIVE LCs & BDs
 - SENSITIVITY FILTERING OF $Path(V', LP')$
 - IF $Path(V', LP') \in Pool_{IR}$ THEN DISCARD $Path(V', LP')$
 - ELSE
 - PUSH($S, (V', LP')$)

OUTPUT: A set of $Path(V, LP)$ (Combinations of IRs)
and a set of Marked LCs and BDs.

ALGORITHM 15. Heuristic/Sensitivity B & B Tree Filtering Algorithm

processing time.

11.5 Example

We illustrate these three algorithms on the inconsistent ILP described in Section 7.5.1. Figure 8 describes the binary tree obtained by proving the inconsistency of the formulation as well as some relevant information associated with this problem and collected by the algorithms.

11.5.1 Algorithm 12: B & B Tree Deletion Filtering.

We develop the B & B tree in a depth-first search fashion. Variables with large fractional parts are branched on first.

Input : $P_0 = (\underline{x}_1, \dots, \underline{x}_6, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))$

Initialize $S = \text{Path}(LP_0) = \emptyset$

$V_0 = \{(x_1, 0.25), (x_2, 0), \dots\}; x_2 \text{ and } x_5 \text{ integers}$

$S = \{((x_1, 0.25), (x_2, 0), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))\}$

$\text{Pool}_{IR} = \emptyset$

a. $(V, (F, C)) = ((x_1, 0.25), (x_2, 0), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))$

b. $X_1 = x_1$

c. $\text{LEFTLP} = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\} \cup \{x_1 \leq 0\})$

d. $\text{RIGHTLP} = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\} \cup \{x_1 \geq 1\})$

e. Solve LEFTLP : inconsistent

f. Solve RIGHTLP : x_1 and x_4 integers, $\text{DISCARD}\{\underline{x}_1\}$

g. $\text{Path}(\text{LEFTV}, \text{LEFTLP}) = \emptyset \cup \{\underline{x}_1\}$

$\text{STORE}\{\underline{x}_1\}$

$\text{Path}(\text{RIGHTV}, \text{RIGHTLP}) = \emptyset \cup \{\underline{x}_1\}$

$\text{PUSH}(S, (\text{RIGHTV}, \text{RIGHTLP}))$

a. $(V, (F, C)) = ((x_1, 1), (x_2, 0.25), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 \geq 1))$

b. $X_3 = x_2$

c. $\text{LEFTLP} = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 \geq 1\} \cup \{x_2 \leq 0\})$

d. $\text{RIGHTLP} = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 \geq 1\} \cup \{x_2 \geq 1\})$

e. Solve LEFTLP : inconsistent

f. Solve RIGHTLP : inconsistent

g. $\text{Path}(\text{LEFTV}, \text{LEFTLP}) = \{\underline{x}_1\} \cup \{\underline{x}_2\}$

$\text{STORE}\{\underline{x}_1, \underline{x}_2\}$

$\text{Path}(\text{RIGHTV}, \text{RIGHTLP}) = \{\underline{x}_1\} \cup \{\underline{x}_2\}$

$\text{STORE}\{\underline{x}_1, \underline{x}_2\}$

$S = \emptyset$: STOP

OUTPUT: 1 Path $\{\underline{x}_1, \underline{x}_2\}$.

Deletion Filtering applied to the MILP made of the IRs $\{\underline{x}_1, \underline{x}_2\}$ combined with the LCs and BDs in the LP relaxation, leads to the following IIS:

$$\{\underline{x}_1, \underline{x}_2, c_1, c_2, c_4, c_5, c_7, c_8, x_3 \geq 0\}$$

11.5.2 Algorithm 14: Sensitivity B & B Tree Deletion Filtering.

Input : $P_0 = (\underline{x}_1, \dots, \underline{x}_6, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))$

Initialize $S = Path(LP_0) = \emptyset$

$V_0 = \{(x_1, 0.25), (x_2, 0), \dots\}; x_2 \text{ and } x_5 \text{ integers}$

$S = \{((x_1, 0.25), (x_2, 0), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))\}$

$Pool_{IR} = \emptyset$

a. $(V, (F, C)) = ((x_1, 0.25), (x_2, 0), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))$

b. $X_1 = x_1$

c. $LEFTLP = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\} \cup \{x_1 \leq 0\})$

d. $RIGHTLP = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\} \cup \{x_1 \geq 1\})$

e. Solve $LEFTLP$: inconsistent

f. Solve $RIGHTLP$: x_1 and x_4 integers

g. $Path(LEFTV, LEFTLP) = \emptyset \cup \{\underline{x}_1\}$

$MARK\{c_1, c_2, c_3, c_5, c_6, c_7, c_8, x_1 \geq 0, x_5 \geq 0\}$

NO Path to STORE

$Path(RIGHTV, RIGHTLP) = \emptyset \cup \{\underline{x}_1\}$

$PUSH(S, (RIGHTV, RIGHTLP))$

a. $(V, (F, C)) = ((x_1, 1), (x_2, 0.25), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 \geq 1))$

b. $X_3 = x_2$

c. $LEFTLP = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 \geq 1\} \cup \{x_2 \leq 0\})$

d. $RIGHTLP = (F_0, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 \geq 1\} \cup \{x_2 \geq 1\})$

e. Solve $LEFTLP$: inconsistent

f. Solve $RIGHTLP$: inconsistent

g. $Path(LEFTV, LEFTLP) = \{\underline{x}_1\} \cup \{\underline{x}_2\}$

$MARK\{c_1, c_2, c_3, c_4, c_5, c_6, c_8, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0\}$

$DISCARD\{\underline{x}_1\}$

$Path(RIGHTV, RIGHTLP) = \{\underline{x}_1\} \cup \{\underline{x}_2\}$

MARK $\{c_1, \dots, c_8, x_3 \geq 0, x_5 \geq 0\}$
STORE $\{\underline{x}_1, \underline{x}_2\}$

$S = \emptyset$: STOP

OUTPUT: 1 Path $\{\underline{x}_1, \underline{x}_2\}$ and $\{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\}$

Deletion Filtering applied to the MILP made of the IRs $\{\underline{x}_1, \underline{x}_2\}$ combined with the LCs and BDs marked in the search, i.e. $\{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\}$ leads to the following IIS: $\{\underline{x}_1, \underline{x}_2, c_1, c_2, c_4, c_5, c_7, c_8, x_3 \geq 0\}$.

11.5.3 Algorithm 15: Heuristic/Sensitivity B & B Tree Deletion Filtering.

Input : $P_0 = (\underline{x}_1, \dots, \underline{x}_6, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))$

Initialize $S = Path(LP_0) = \emptyset$

$V_0 = \{(x_1, 0.25), (x_2, 0), \dots\}$; x_2 and x_5 integers

$S = \{((x_1, 0.25), (x_2, 0), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))\}$

$Pool_{IR} = \emptyset$

a. $(V, (F, C)) = ((x_1, 0.25), (x_2, 0), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0))$

b. $X_1 = x_1$

c. $AC = \emptyset$

d. $LEFTLP = (slack_2, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\} \cup \{x_1 + slack_2 = 0\})$
 $RIGHTLP = (slack_3, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0\} \cup \{x_1 - slack_3 = 1\})$

e. Solve $LEFTLP$: inconsistent

f. Solve $RIGHTLP$: x_1 and x_4 integers

h. $Path(LEFTV, LEFTLP) = \emptyset \cup \{\underline{x}_1\}$

$MARK\{c_2, c_3, c_4, c_5, c_6, c_7, c_8, x_1 \geq 0\}$

NO Path to STORE

$Path(RIGHTV, RIGHTLP) = \emptyset \cup \{\underline{x}_1\}$

$PUSH(S, (RIGHTV, RIGHTLP))$

a. $(V, (F, C)) =$

$((x_1, 1), (x_2, 0.25), \dots, (\sum_1^6 x_i, c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 - slack_3 = 1))$

b. $X_3 = x_2$

- c. $AC = \{x_1 - slack_3 = 1\}$
 - d. $LEFTLP =$
 $(slack_2 + slack_6, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 - slack_3 = 1\} \cup \{x_2 + slack_6 = 0\})$
 $RIGHTLP =$
 $(slack_2 + slack_6, \{c_1, \dots, c_8, x_1 \geq 0, \dots, x_6 \geq 0, x_1 - slack_3 = 1\} \cup \{x_2 - slack_7 = 1\})$
 - e. Solve $LEFTLP$: inconsistent
 - f. Solve $RIGHTLP$: inconsistent
 - g. $Path(LEFTV, LEFTLP) = \{\underline{x_1}\} \cup \{\underline{x_2}\}$
 $MARK\{c_2, c_3, c_4, c_6, c_8, x_2 \geq 0, x_4 \geq 0\}$
 $NO\ Path\ to\ STORE$
 $Path(RIGHTV, RIGHTLP) = \{\underline{x_1}\} \cup \{\underline{x_2}\}$
 $MARK\{c_1, \dots, c_8, x_4 \geq 0\}$
 $DISCARD\{\underline{x_2}\}$
 $S = \emptyset$: STOP
- OUTPUT:** NO Path BUT Introduced IRs $\{\underline{x_1}, \underline{x_2}\}$ and marked LCs and BDs are $\{c_1, \dots, c_8, x_1 \geq 0, x_2 \geq 0, x_4 \geq 0\}$

Deletion Filtering applied to the MILP made of the IRs $\{\underline{x_1}, \underline{x_2}\}$ combined with the LCs and BDs marked in the search, i.e. $\{c_1, \dots, c_8, x_1 \geq 0, x_2 \geq 0, x_4 \geq 0\}$ leads to the following IIS: $\{\underline{x_1}, \underline{x_2}, c_1, \dots, c_8\}$.

11.6 Computational Results

Tables 31–33 list the respective performances of Algorithms 13, 14 and 15 on the set of test MILPs described in Chapter 5, the “dell” problem excepted since MINTO 2.1 does not allow any re-scaling of the formulation.

The strategies adopted in the B & B searches are a best-bound search for the node selection and a non quasi-integer variable selection scheme, i.e. the variable farthest from any integer value is branched on.

Each row of any table contains the following information. The first three columns list characteristics of the subset of constraints isolated by the B & B tree filtering algorithm: number of IRs introduced in the search (as well as the number of non-discarded paths) and the number of marked LCs and BDs. The overall number of

nodes required to complete this first round is listed in the “B & B Filter Nodes” column, including the few infeasibility proofs necessary to identify the first inconsistent path. The subset is then deletion filtered. The cardinality of the IIS, in IRs, LCs and BDs, is indicated in the next three columns. The number of nodes developed by the deletion filter is given in the next to last column. The overall time to complete both phases is listed in the last column.

11.7 Discussion

The inability of all three algorithms to detect the inconsistency of several formulations in a reasonable number of nodes, less than 3,000, reflects the inefficiency of the branching schemes rather than a lack of power on behalf of the B & B tree filters. In fact, the B & B algorithm which serves as a basis for these filters fails in most cases where a depth-first or best-bound search together with the maximum infeasibility branching variable selection scheme would fail (Tables 11 and 13). However, these problems require only a few nodes when an appropriate B & B strategy is used (Tables 12 and 14). Therefore, B & B tree filters, combined with an efficient strategy, are likely to succeed and to collect the few paths necessary to start the deletion filtering phase.

When the B & B algorithm implemented is able to detect infeasibility, Algorithms 14 and 15 lead to net reductions in the overall time required to complete the isolation. For instance, “misc04” whose filtering takes about 20 hours with a straight deletion filter requires less than 8 hours. Increases in the isolation time are mainly due to the fact that, with more free variables, the B & B algorithm is less able to detect inconsistency and therefore iterates until the threshold of 10,000 nodes is attained which slows down the deletion process.

Also, Algorithm 15 distinguishes itself from Algorithm 14 by the fact that, in all cases, the number of paths identified by the B & B tree filter is smaller. However, this

TABLE 31. B & B Tree Deletion Filtering Approach (Algorithm 13)

Infeasible MILP	B & B Tree Filtering			IIS Constraints After Del. Filter			Efficiency		
	IRs(Paths)	LCs	BDs	IRs	LCs	BDs	B & B Filter Nodes	Del. Filter Nodes	Time h:m:s
*bell3a	19(4)	124	204	7	70	49(4)	505	6774	1:20
bell3b	-	-	-	-	-	-	-	-	-
bell4	-	-	-	-	-	-	-	-	-
bell5	-	-	-	-	-	-	-	-	-
flugpl	-	-	-	-	-	-	-	-	-
*g503	1(1)	41	72	1	18	17(4)	5	159	5
*misc01	9(1)	55	164	5	28(1)	64(34)	35	563	24
*misc02	15(1)	40	116	9	32(1)	33(22)	110	1454	26
*misc03	3(1)	97	398	3	22(1)	105(63)	13	1610734	2:00:01
*misc04	2(1)	1726	4691.	1	1173(301)	3706(2534)	7	6974	17:24:58
*misc05	14(2)	301	204	10	74(4)	59(43)	99	25389	10:32
*misc06	13(3)	821	1894	3	602(26)	1017(708)	75	9509	2:45:58
mod008	-	-	-	-	-	-	-	-	-
*mod013	20(0)	63	144	14	49	51(20)	241	11238	1:18
p0201	-	-	-	-	-	-	-	-	-
stein9	8(1)	14	18	6	13	0	84	1073	4
stein15	13(4)	37	30	7	36	0	276	4019	23
stein27	-	-	-	-	-	-	-	-	-
stein45	-	-	-	-	-	-	-	-	-

- Maximum Number of Nodes Reached.

* IS (not necessarily IIS) isolated.

TABLE 32. B & B Tree Deletion Filtering Approach (Algorithm 14)

Infeasible MLP	B & B Tree Filtering			IIS Constraints After Del. Filter			Efficiency		
	IRs(Paths)	LCs	BDs	IRs	LCs	BDs	B & B Filter Nodes	Del. Filter Nodes	Time h:m:s
*bell3a	19(6)	103	113	7	70(38)	41(46)	505	3838	4:07
bell3b	-	-	-	-	-	-	-	-	-
bell4	-	-	-	-	-	-	-	-	-
bell5	-	-	-	-	-	-	-	-	-
flugpl	-	-	-	-	-	-	-	-	-
*g503	1(1)	41	43	1	30(9)	18(11)	3	100	5
*misc01	9(1)	55	112	5	28(1)	56(48)	19	530	23
*misc02	15(1)	40	97	9	32(1)	33(22)	41	1428	29
*misc03	3(1)	96	171	3	22(1)	105(88)	7	570597	42:44
*misc04	2(1)	1486	4009	1	1169(1104)	3699(3599)	5	1569	7:25:54
*misc05	14(3)	287	153	10	74(5)	58(40)	97	20920	9:22
*misc06	13(2)	774	1561	3	602(516)	1141(872)	49	3017	1:22:35
mod008	-	-	-	-	-	-	-	-	-
*mod013	20(0)	63	120	14	49(2)	49(26)	117	10232	1:25
p0201	-	-	-	-	-	-	-	-	-
stein9	13(4)	14	18	6	13	0	35	1073	6
stein15	-	-	-	-	-	-	-	-	-
stein27	-	-	-	-	-	-	-	-	-
stein45	-	-	-	-	-	-	-	-	-

- Maximum Number of Nodes Reached.

* IS (not necessarily IIS) isolated.

Table 33. B & B Tree Deletion Filtering Approach (Algorithm 15)

Infeasible MILP	B & B Tree Filtering			IIS Constraints After Del. Filter			Efficiency		
	IRs(Paths)	LCs	BDs	IRs	LCs	BDs	B & B Filter Nodes	Del. Filter Nodes	Time h:m:s
*bell3a	12(6)	113	127	7	70(10)	46(35)	135	4688	1:10
*bell3b	3(0)	100	105	1	69(53)	33(32)	13	221	19
bell4	-	-	-	-	-	-	-	-	-
bell5	-	-	-	-	-	-	-	-	-
flugpl	-	-	-	-	-	-	-	-	-
*g503	1(0)	40	44	1	30(9)	19(10)	3	103	5
*misc01	9(1)	55	127	6	26(1)	81(39)	35	877	25
*misc02	18(0)	40	104	9	31(1)	34(22)	83	1187	28
*misc03	8(1)	97	209	3	22(1)	105(82)	33	760716	56:48
*misc04	2(0)	1493	4028	1	1164(1094)	3707(3586)	7	1654	7:39:20
*misc05	20(2)	291	172	9	67(5)	67(41)	140	14043	9:44
*misc06	13(2)	775	1566	3	602(500)	1093(874)	75	5434	1:47:22
mod008	-	-	-	-	-	-	-	-	-
*mod013	28(1)	63	129	14	48(2)	51(26)	313	11912	1:56
p0201	-	-	-	-	-	-	-	-	-
stein9	7(0)	14	18	6	13	0	93	985	7
stein15	11(0)	37	30	8	36	0	183	3792	24
stein27	-	-	-	-	-	-	-	-	-
stein45	44(0)	332	90	16	220	2	1979	1150298	8:27:44

- Maximum Number of Nodes Reached.

* IS (not necessarily IIS) isolated.

does not seem lead to improved performance , at least on the sample of inconsistent MILPs, since paths tend to be larger which increases the number of MILP solutions in the filtering of the IRs.

CHAPTER 12

A FRAMEWORK FOR ANALYZING INFEASIBLE MILPS

12.1 A Few Remarks

- The number of nodes in the B & B tree developed to prove the original MILP infeasible shows a great sensitivity to at least three factors: the subproblem selection and branching variable strategies (Tables 11, 12, 13 and 14), the objective function (Table 7) and the use of priority lists which can either be based on a certain knowledge of the problem or on previous isolations (Table 16). All three factors can be tuned to improve the performance of the MILP solver.
- Finite BDs on variables combined with a modified deletion filter (Table 4), where dropped BDs are replaced by looser, but finite, bounds allow the identification of IISs in most cases, by enabling a B & B based MILP solver to detect in a finite number of nodes the infeasibility of the test set.
- Table 34 lists the rankings of the deletion filter (Algorithm 3), incremental filter (Algorithm 6) and incremental/deletion filter (Algorithm 8). As noted in Section 6.3, the ranking based on the overall number of nodes provides a safe way of comparing these three different algorithms, since they are implemented differently. Algorithm 8 appears to be the best choice on average. Moreover, performance of the incremental/deletion filter can be improved by adding IRs in an opportunist way (Algorithm 8). Also, as in the deletion filter case, dubious constraints can be identified.
- The conservative approach in blocking (Algorithm 11) is on average almost as efficient as the optimist one (Algorithm 10), see Table 35, and outperforms the latter for large IISs (as in “misc04” and “misc06”, see Tables 3, 17 and 20).

Moreover, in the case of large IISs, the overall isolation time required by Algorithm 10 may exceed the isolation time of a straight deletion filter (Algorithm 3), as in “dell” and “misc04”. Algorithm 10 shows shorter isolation times in these cases and even isolates an IS in “misc04” before Algorithm 3.

TABLE 34. Rankings of Isolation Algorithms

Method 1: Deletion Filter (Algorithm 3)
 Method 2: Incremental Filter (Algorithm 6)
 Method 3: Incremental/Deletion Filter (Algorithm 8)

Infeasible MILP	Method 1			Method 2			Method 3		
	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time
bell3a	1	1	1	3	3	3	2	2	2
bell3b	- *	-	-	-	-	-	-	-	-
bell4	1	3	3	2	2	2	2	1	1
bell5	- *	-	-	-	-	-	-	-	-
dell	- *	-	-	-	-	-	-	-	-
flugpl	- *	-	-	-	-	-	-	-	-
g503	1	2	2	3	3	3	1	1	1
misc01	1	2	1	3	1	3	2	3	2
misc02	1	1	1	3	3	3	2	2	2
misc03	1	3	2	3	1	3	2	2	2
misc04	- *	-	-	-	-	-	-	-	-
misc05	1	2	1	3	3	3	2	1	2
misc06	- *	-	-	-	-	-	-	-	-
mod008	- *	-	-	-	-	-	-	-	-
mod013	1	1	1	2	3	3	1	2	2
p0201	1	3	3	2	1	1	1	2	2
stein9	1	2	1	1	1	3	1	3	2
stein15	1	1	1	1	3	3	1	2	2
stein27	1	3	1	1	2	3	3	1	2
stein45	- *	-	-	-	-	-	-	-	-
Avg	1.00	2.00	1.50	2.25	2.16	2.75	1.67	1.83	1.83

* Different IISs were isolated.

12.2 A Framework for Isolating IISs in Inconsistent MILPs

Algorithm 16 gives a sequence of actions, based on the previous remarks, for identifying IISs in inconsistent MILPs.

TABLE 35. Rankings of Blocking Strategies

Method 1: Fixed-Size Optimist Blocking Strategy (Algo. 10)
 Method 2: Fixed-Size Conservative Blocking Strategy (Algo. 11)
 Method 3: Additive Adaptive Blocking Strategy 2 (Algo. 10)
 Method 4: Additive Adaptive Blocking Strategy 2 (Algo. 11)

Infeasible MILP	Method 1			Method 2			Method 3			Method 4		
	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time	Isol.	Nodes	Time
bell3a	1	3	2	1	2	1	1	4	3	1	1	1
bell3b	1	1	1	4	3	3	2	2	2	3	4	4
bell4	1	3	2	1	4	3	1	1	1	1	2	4
bell5	1	2	2	2	3	3	1	1	1	3	4	4
dell	- *	-	-	-	-	-	-	-	-	-	-	-
flugpl	1	2	2	1	4	3	1	1	1	1	3	4
g503	1	2	2	1	1	1	1	4	2	1	3	2
misc01	1	3	4	3	4	2	2	2	3	2	1	1
misc02	2	1	2	2	2	1	1	4	4	1	3	3
misc03	1	2	3	1	1	2	1	4	1	1	3	1
misc04	2	3	3	2	1	1	2	4	4	1	2	2
misc05	1	2	2	2	3	3	1	1	1	1	4	4
misc06	4	3	4	1	4	2	2	1	3	3	2	1
mod008	1	1	1	1	3	3	1	2	2	1	4	4
mod013	2	2	2	1	4	2	1	1	1	1	3	3
p0201	1	3	3	1	4	4	1	1	1	1	2	2
stein9	1	2	1	1	1	1	1	4	2	1	3	2
stein15	1	4	2	1	3	1	1	1	1	1	2	1
stein27	1	4	4	1	2	2	1	3	3	1	1	1
stein45	1	4	4	1	2	2	1	3	3	1	1	1
Avg	1.32	2.47	2.42	1.37	2.68	2.10	1.21	2.32	2.05	1.37	2.53	2.37

* Different IISs were isolated.

Remark: A “large” MILP is either a MILP with thousands of constraints, or a MILP whose initial proof of infeasibility requires thousands of nodes.

Input : An *infeasible* MILP or ILP P_0

Step 0 : Try to decrease the number of nodes required by the initial proof of infeasibility, add BDs if necessary.

Step 1 : IF P_0 is "large" THEN
Preprocess the MILP by a Sensitivity B & B Tree Filter.

Step 2 : Use Opportunist Incremental/Deletion Filter with Conservative Blocking Strategy on LCs and BDs and use "safety BDs".

OUTPUT : An IS or IIS.

ALGORITHM 16. A Framework for Isolating IISs in Inconsistent MILPs

CHAPTER 13

CONCLUSIONS

Several empirical conclusions concerning the formulation of inconsistent MILPs, the general settings of the MILP solver and the IIS-isolation algorithms can be drawn from the set of varied experiments carried out on the test set of inconsistent MILPs.

13.1 Settings of B & B MILP Solver and Inconsistent MILP Formulations

- A depth-first search of the binary tree, combined with a best-bound variable selection strategy gave satisfactory results in most cases.
- Finite bounds on variables, even if these bounds cause some redundancy within the relaxed LP, can speed up the solution of the MILP.
- Modified objective functions can be used in the post-infeasibility analysis of infeasible MILPs. MILP solutions, in terms of number of developed nodes and solution time, are sensitive to the choice of the objective function. Discrepancies between two different objective functions can be considerable.

13.2 IIS-Isolation Algorithms

- The deletion filter, incremental filter and incremental/deletion filter are valid approaches for identifying IISs in an infeasible MILP. However, the inability of the B & B MILP solver to decide the consistency of the test set in a finite number of nodes, mainly when too many infinite bounds on variables appear, leads in almost all cases to the isolation of ISs rather than IISs. Unlike incremental techniques, deletion filters can label and count dubious constraints, i.e. constraints whose removal destroys the ability of the MILP solver to detect inconsistency.

Incremental/deletion filters can do so on LCs and BDs only, but this is not too restrictive since no IR can be labelled as dubious if the B & B solver succeeds in detecting the infeasibility of the initial MILP in a finite number of nodes. These questionable constraints, mainly BDs on variables, can then be separately examined. If lower (respectively upper) BDs, dropped in the deletion filtering process, are replaced by smaller (resp. larger) BDs rather than infinite values then the deletion filter, based on a plain B & B algorithm, isolates IISs in most cases. Also, incremental filters tend to output more extraneous BDs than deletion filters. If IISs are necessary, filtering algorithms, based on B & Cut searches can be used.

- The deletion filter requires a computation time to complete the identification which is rather independent of the characteristics of the isolated IIS. On the other hand, the incremental filter can isolate small IISs in a time polynomial on the order of the size of the IIS. However, the order of complexity of this algorithm is not linear since worst-case behaviours are observed when the IIS contains a large proportion of the original constraints. The incremental/deletion filter offers a “middle-way” in between these two algorithms, with a worst case behaviour linear in the number of LCs and BDs (and polynomial in IRs) and an ability to, as with the incremental technique, quickly identify smaller integer parts of the IISs.
- Blocking strategies combined with one of the previous algorithms can dramatically accelerate the isolation. However, a too optimistic blocking approach, i.e. too many unsuccessful attempts to block, may turn out to effectively increase the overall identification time if the IIS is relatively large compared with the original MILP. In such cases, a more cautious approach will yield isolation times roughly of the order of the ones required by the straight algorithm, i.e. without any blocking.

- Preprocessing of the infeasible formulation by B & B tree filtering techniques proved to be a valid approach for discarding relatively large numbers of constraints which do not belong to the IIS. A straight deletion filter of the output of the B & B tree filter may isolate an IIS. The combination of both techniques is a good alternative to blocking algorithms for isolating large (dense) IISs.

13.3 Contributions

- First-ever MILP infeasibility analyzer.
- Numerous new algorithms for isolating ISs and IISs in inconsistent MILPs have been described, implemented and benchmarked.
- Experiments on a test set of inconsistent MILPs allow the first empirical comparison of the incremental and deletion filtering approaches: two rather distinct behaviours have been observed.
- A few characteristics of IISs in infeasible MILPs have been exposed: possibly large numbers of LCs and BDs in an IIS and exponential number of IISs in an inconsistent MILP.

13.4 Future Research

- A future goal is an implementation of a B & Cut algorithm which would allow the algorithms presented in this thesis to identify IISs in all cases. Also, efficient implementations of the algorithms based on the principle of a B & B tree filter, integrated with deletion filtering techniques would provide an efficient framework for isolating IISs in MILPs.
- In the case of an all integer formulation (ILP), infeasibility of a system of equations can be determined by the integrality of a vector obtained by the product of the inverse of the Hermite transform of the matrix of the coefficients by the vector of the RHS. The overall transformation is therefore polynomial both in time and space. An inconsistent ILP is identified when this vector contains at least one

non-integer value. A study of the constraints which prevent this coefficient from taking an integral value may indicate equations and integer restrictions involved in the IIS. However, this approach suffers from two flaws being that, in the first place, this method is fairly restricted to infeasible systems of Diophantine equations only and secondly that, as mentioned in [Nemhauser et al. 88] computation of the Hermite transform leads fairly rapidly to numerical difficulties as the dimensions of the matrix of the coefficients grow.

APPENDIX A: DERIVATION OF A CUT

First, we recall the formulation of P_α , MILP illustrated in Figure 4:

$$\begin{aligned}
 P_\alpha \quad & \textbf{Minimize} \quad F_\alpha(x, y) = 2x + y \\
 & \textbf{Subject to} \quad (1 + \alpha)x - y \geq 1.2 \\
 & \quad \quad \quad x - y \leq 1.8 \\
 & \quad \quad \quad x, y \text{ non-negative integers}
 \end{aligned}$$

For $\alpha = 0.01$ the optimal solution to the relaxed LP is given by:

$$\begin{cases} x = 1.1881 \\ y = 0.0 \end{cases}$$

and the integral solution by:

$$\begin{cases} x = 20 \\ y = 19 \end{cases}$$

We follow here [Winston 91 pp 513–514] in the derivation of the cutting plane expression. The introduction of two slack variables, s_1 and s_2 results in the equivalent standard form of the LP-relaxation of P_α :

$$\begin{aligned}
 \textbf{Minimize} \quad & F_\alpha(x, y) = 2x + y \\
 \textbf{Subject to} \quad & (1 + \alpha)x - y - s_1 = 1.2 \\
 & x - y + s_2 = 1.8 \\
 & x, y, s_1, s_2 \geq 0
 \end{aligned}$$

The final tableau of the simplex algorithm leads to the following expression:

$$\begin{cases} s_1 = -1.8 + \frac{1}{1 + \alpha} 1.2 - \frac{\alpha}{1 + \alpha} y - \frac{1}{1 + \alpha} s_2 \\ x = \frac{1}{1 + \alpha} 1.2 + \frac{1}{1 + \alpha} y - \frac{1}{1 + \alpha} s_2 \end{cases} \quad (1, 2)$$

where s_1 and x are the basic variables.

For $\alpha \ll 1$ the following approximation holds:

$$\frac{1}{(1 + \alpha)} \simeq 1 - \alpha$$

Thus (1) and (2) become:

$$\begin{cases} s_1 \simeq -1.8 + (1 - \alpha)1.2 - \alpha y - (1 - \alpha)s_2 \\ x \simeq (1 - \alpha)1.2 + (1 - \alpha)y - (1 - \alpha)s_2 \end{cases} \quad (2,3)$$

If we rewrite the second equation as

$$x - y + s_2 - 1 = 0.2 - \alpha(1.2) - \alpha y + \alpha s_2 \quad (4)$$

then the following relationship is a *valid inequality*:

$$0.2 - \alpha(1.2) - \alpha y + \alpha s_2 \leq 0 \quad (5)$$

And, finally, from the original standard formulation:

$$x \geq \frac{0.2}{\alpha} + 0.6 \quad (6)$$

which leads to the new equivalent formulation P'_α

$$\begin{aligned} P'_\alpha \quad & \text{Minimize} \quad F_\alpha(x, y) = 2x + y \\ & \text{Subject to} \quad (1 + \alpha)x - y \leq 1.2 \\ & \quad \quad \quad x - y \geq 1.8 \\ & \quad \quad \quad x \geq \frac{0.2}{\alpha} + 0.6 \\ & \quad \quad \quad x, y \text{ non-negative integers} \end{aligned}$$

Remark: For $\alpha = 0$, an infeasible MILP, this derivation leads to adding the following infeasible cut $x \geq \infty$. Thus, unlike the B & B method, we can conclude that the original MILP is inconsistent.

APPENDIX B: HERMITE NORMAL FORM TRANSFORMATION

Consider the following linear equation integer feasibility problem [Nemhauser and Wolsey 88, pp 189–191]:

$$2x_1 + 6x_2 + 1x_3 = 7$$

$$4x_1 + 7x_2 + 7x_3 = 4.$$

Where

x_1, x_2, x_3 are integer

Using the notation introduced in Section 2.2.8, this problem has a solution if and only if $\mathbf{b} \in L(A)$ where:

- $L(A)$ is the lattice generated by the columns of A

$$A = \begin{pmatrix} 2 & 6 & 1 \\ 4 & 7 & 7 \end{pmatrix}$$

- $\mathbf{b}$ is the vector of the RHS

$$\mathbf{b} = \begin{pmatrix} 7 \\ 4 \end{pmatrix}$$

A sequence of *elementary column operations* transforms the matrix A into a Hermite matrix H . These operations are:

- a. Interchange two columns
- b. Multiply a column by -1
- c. Add to a column an integer multiple of another column

Starting from matrix A' , operating from left to right on each row:

$$A' = \begin{pmatrix} 2 & 6 & 1 \\ 4 & 7 & 7 \\ 0 & 0 & 1 \end{pmatrix},$$

the following sequence of column operations leads to the Hermite normal form H' of A' :

Step 1 : Subtract 3 times Col. 1 from Col. 2

$$A' = \begin{pmatrix} 2 & 0 & 1 \\ 4 & -5 & 7 \\ 0 & 0 & 1 \end{pmatrix},$$

Step 2 : Interchange Col. 1 and Col. 3

$$A' = \begin{pmatrix} 1 & 0 & 2 \\ 7 & -5 & 4 \\ 1 & 0 & 0 \end{pmatrix},$$

Step 2 : Subtract 2 times Col. 1 from Col. 2

$$A' = \begin{pmatrix} 1 & 0 & 0 \\ 7 & -5 & 10 \\ 1 & 0 & 2 \end{pmatrix},$$

...

...

Until

$$A' = \begin{pmatrix} 1 & 0 & 0 \\ -3 & 5 & 0 \\ -1 & 0 & 2 \end{pmatrix}.$$

Which gives the Hermite transform H of A

$$H = \begin{pmatrix} 1 & 0 \\ -3 & 5 \end{pmatrix}$$

Since

$$\begin{pmatrix} 2 & 6 & 1 \\ 4 & 7 & 7 \end{pmatrix} \times \begin{pmatrix} 1 & 3 & -7 \\ 0 & -1 & 2 \\ -1 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -3 & 5 & 0 \end{pmatrix}$$

Then

$$C = \begin{pmatrix} 1 & 3 & -7 \\ 0 & -1 & 2 \\ -1 & 0 & 2 \end{pmatrix}$$

is a unimodular matrix ($|\det C| = 1$).

All solutions to the problem are given by:

$$x = \begin{pmatrix} 1 & 3 \\ 0 & -1 \\ -1 & 0 \end{pmatrix} \cdot H^{-1}\mathbf{b} + \begin{pmatrix} -7 \\ 2 \\ 2 \end{pmatrix} z \quad z \in \mathbb{Z}$$

Thus, integer solutions exist if and only if

$$H^{-1}\mathbf{b} = \begin{pmatrix} 1 & 0 \\ -3 & 5 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 7 \\ 4 \end{pmatrix} = \begin{pmatrix} 7 \\ 5 \end{pmatrix} \in \mathbb{Z}^2$$

Solutions are then given by:

$$x = \begin{pmatrix} 22 \\ -5 \\ -7 \end{pmatrix} + \begin{pmatrix} -7 \\ 2 \\ 2 \end{pmatrix} z \quad z \in \mathbb{Z}$$

For another set of RHS values, say $b = \begin{pmatrix} 8 \\ 4 \end{pmatrix}$, the problem has no integer solution since:

$$\begin{pmatrix} 1 & 0 \\ -3 & 5 \end{pmatrix}^{-1} \cdot \begin{pmatrix} 8 \\ 4 \end{pmatrix} = \begin{pmatrix} 8 \\ \frac{36}{5} \end{pmatrix} \notin \mathbb{Z}^2$$

The solution of the linear equation integer feasibility problem requires in general $O(n^2)$ (where n is the larger dimension of the matrix) steps to compute the Hermite

transform of the coefficient matrix and one matrix inversion. It is therefore a polynomially bounded problem in the number of operations. However, unusually large numbers may appear while computing the Hermite transform matrix, specifically if the determinant of A happens to be large [Bradley 71]. Thus, algorithms that perform modulo arithmetic operations with respect to the value of that determinant are required to avoid any computational problems.

For instance the following matrix ($| \det A | = 25390$):

$$A = \begin{pmatrix} 13 & 21 & 0 & -37 \\ 10 & 20 & -15 & 0 \\ 1 & 3 & 2 & 1 \\ 7 & -1 & -1 & 0 \end{pmatrix}$$

has the following Hermite transform:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ -1 & 0 & 2 & 0 \\ -1570 & -1651 & -2415 & 2539 \end{pmatrix}.$$

References

- N. Agin, *Optimum Seeking with Branch and Bound*, Management Science **13** (1966), no. 4, B 176–185.
- R.W. Ashford and R.C. Daniel, *Some Lessons in Solving Practical Integer Programs*, Journal of the Operational Research Society **43** (1992), no. 5, 425–433.
- C. Barnhart, E.L. Johnson, G.L. Nemhauser, G. Sigismondi and P.Vance, *Formulating a Mixed Integer Programming Problem to Improve Solvability*, Operations Research **41** (1993), no. 6, 1013–1019.
- E.M.L. Beale, *Integer Programming*, The State of the Art of Numerical Analysis (D.A.H. Jacob, ed.), Academic Press, London, 1977, pp. 409–448.
- M. Benichou, J.M. Gauthier, P. Girodet, G. Hentges, G. Ribiere and D. Vincent, *Experiments in Mixed Integer Linear Programming*, Mathematical Programming **1** (1971), 76–94.
- A.L. Brearley, G. Mitra and H.P. Williams, *Analysis of Mathematical Programming Problems Prior to Applying the Simplex Algorithm*, Mathematical Programming **8** (1975), 54–83.
- G.H. Bradley, *Algorithms for Hermite and Smith Normal Matrices and Linear Diophantine Equations*, Mathematics of Computation **25** (1971), 897–907.
- R. Breu and C.A. Burdet, *Branch and Bound Experiments in Zero-One Programming*, Mathematical Programming Study, vol. 2, 1974, pp. 1–50.
- N. Chakravarti, *Some Results Concerning Post-Infeasibility Analysis*, European Journal of Operational Research **73** (1994), 139–143.
- J.W. Chinneck, *Finding Minimal Sets of Constraints in Infeasible Mathematical Pro-*

- grams, technical report SCE-93-01. Systems and Computer Engineering, Carleton University, Canada, 1993.
- , *Finding the Most Useful Subset of Constraints for Analysis in an Infeasible Linear Program*, technical report SCE-93-07. Systems and Computer Engineering, Carleton University, Canada, 1993.
- , *MINOS(IIS) - Infeasibility Analysis Using Minos*, Computers and Operations Research **21** (1994), no. 1, 1–9.
- , *An Effective Polynomial-Time Heuristic for the Minimum-Cardinality IIS Set-Covering Problem*, to appear in Annals of Mathematics and Artificial Intelligence (1995).
- J.W. Chinneck and E.W. Dravnieks, *Locating Minimal Infeasible Constraint Sets in Linear Programs*, ORSA Journal on Computing **3** (1991), no. 2, 157–168.
- V. Chvátal, *Linear Programming*, W.H. Freeman and Company, New York, 1983.
- G.B. Dantzig, *On the Significance of Solving Linear Programming Problems with some Integer Variables*, Econometrica **28** (1960), no. 1, 30–44.
- R.E. Davis, D.A. Kendrick and M. Weitzman, *A Branch and Bound Algorithm for zero-one Mixed Integer Programming Problems*, Operations Research **19** (1971), 1036–1044.
- P.D. Domich, R. Kannan and L.E. Trotter, *Hermite Normal Form Computation Using Modulo Determinant Arithmetic*, Mathematics of Operations Research **12** (1987), no. 1, 50–59.
- E.W. Dravnieks, *Identifying Minimal Sets of Inconsistent Constraints in Linear Programs: Deletion, Squeeze and Sensitivity Filtering*, M.Sc. thesis, Systems and Computer Engineering, Carleton University, Ottawa, Canada, 1989.
- M.E. Dyer, *The Complexity of Vertex Enumeration Methods*, Mathematics of Operations Research **8** (1983), 381–402.
- J.J.H. Forrest, J.P.H. Hirst and J.A. Tomlin, *Practical Solution of Large Mixed Integer*

- Programming Problems with (U)mpire*, Management Science **20** (1974), no. 5, 736-773.
- A.M. Geoffrion and R. Nauss, *Parametric and Postoptimality Analysis in Integer Linear Programming*, Management Science **23** (1977), no. 5, 453-466.
- F. Gleeson and J. Ryan, *Identifying Minimally Infeasible Subsystems of Equations*, ORSA Journal of Computing **2** (1990), no. 1, 61-63.
- R.E. Gomory, *An Algorithm for Integer Solutions to Linear Programs*, Recent Advances in Mathematical Programming, Mc Graw Hill Book Co., New York, 1963.
- H.J. Greenberg, *ANALYZE: a Computer-Assisted Analysis System for Linear Programming Models*, Operations Research Letters **6** (1987), no. 5, 249-255.
- , *How to Analyze the Results of Linear Programs-Part 1: Preliminaries*, Interfaces **23** (1993), no. 4, 56-67.
- , *How to Analyze the Results of Linear Programs-Part 2: Price Interpretation*, Interfaces **23** (1993), no. 5, 97-114.
- , *How to Analyze the Results of Linear Programs-Part 3: Infeasibility Diagnosis*, Interfaces **23** (1993), no. 6, 120-139.
- , *How to Analyze the Results of Linear Programs-Part 4: Forcing Substructures*, Interfaces **24** (1994), no. 1, 121-130.
- A.J. Hoffman and M. Padberg, *LP-Based Combinatorial Problem Solving*, Annals of Operations Research **4** (1985), 145-194.
- S. Holn and D. Klein, *Discrete Right Hand Side Parametrization for Linear Integer Programs*, European Journal of Operational Research **2** (1978), 50-53.
- , *Three Methods for Postoptimal Analysis in Integer Linear Programming*, Mathematical Programming Study, vol. 21, 1984, pp. 97-109.
- R.E. Jensen, *Sensitivity Analysis and Integer Linear Programming*, The Accounting Review XLIII (1968), 425-446.
- R.G. Jeroslaw, *Minimal Inequalities*, Mathematical Programming **17** (1979), 1-15.

- , *Cutting Plane Theory: Algebraic Methods*, Discrete Mathematics **23** (1978), 121–150.
- R. Kannan and A. Bachem, *Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix*, SIAM Journal of Computing **9** (1979), 499–507.
- E.L. Lawler, J.K. Lenstra, A.H.G. Rinnoy Kan and D.B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, New York, 1985.
- R.E. Marsten and T.L. Morin, *Parametric Integer Programming: the Right-Hand-Side Case*, Annals of Discrete Mathematics **1** (1977), 375–390.
- R.R. Meyer, *On the Existence of Optimal Solutions to Integer and Mixed-Integer Programming Problems*, Mathematical Programming **7** (1974), 223–235.
- G.L. Nemhauser, *The Age of Optimization: Solving Large-Scale Real-World Problems*, Operations Research **42** (1994), no. 1, 5–13.
- G.L. Nemhauser, M.W.P. Savelsbergh and G.C. Sigismondi, *MINTO, a Mixed INTe-ger Optimizer*, Operations Research Letters **15** (1994), 47–58.
- G.L. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.
- M. Parker and J. Ryan, *Finding the Maximum Weight Feasible Subsystem of an Infeasible System of Linear Inequalities*, technical report, the University of Colorado at Denver, preprint, 1994.
- R.G. Parker and R.L. Rardin, *Heuristic Aspects of Branch and Bound*, Discrete Optimization, Academic Press, Boston, 1988, pp. 194–204.
- C.J. Piper and A.A. Zoltners, *Some Easy Postoptimality Analysis for Zero-One Programming*, Management Science **22** (1976), no. 7, 759–765.
- G.M. Roodman, *Post-Optimality Analysis in Integer Programming by Implicit Enumeration: the Mixed Integer Case*, Naval Research Logistics Quarterly **20** (1973),

595–607.

G.M. Roodman, *Post-Infeasibility Analysis in Linear Programming*, Management Science **25** (1979), no. 9, 916–922.

J.K. Sankaran, *A Note on Resolving Infeasibility in Linear Programs by Constraint Relaxation*, Operation Research Letters **13** (1993), 19–20.

M.W.P. Savelsbergh, *Preprocessing and Probing Techniques for Mixed Integer Programming Problems*, ORSA Journal on Computing **6** (1994), no. 4, 445–454.

M. Tamiz, S.J. Mardle and D.F. Jones, *Detecting IIS in Infeasible Linear Programmes using Techniques from Goal Programming*, to appear in Computers and Operations Research (1995).

———, *Resolving Inconsistency in Infeasible Linear Programmes*, technical report, School of Mathematical Studies, University of Portsmouth, 1995.

J. Telgen, *Identifying Redundant Constraints and Implicit Equalities in Systems of Linear Constraints*, Management Science **29** (1983), 1209–1222.

J.N.M. van Loon, *Irreducibly Inconsistent Systems of Linear Inequalities*, European Journal of Operational Research **8** (1981), 283–288.

W.L. Winston, *Operations Research: Applications and Algorithms*, PWS-KENT Publishing Company, California, 1991.

L.A. Wolsey, *Integer Programming Duality: Price Functions and Sensitivity Analysis*, Mathematical Programming **20** (1981), 173–195.

———, *Strong Formulations for Mixed Integer Programming: A Survey*, Mathematical Programming **45** (1989), 173–191.

END

0|5-06-9|6

FIN