1

28

32

36

40

MicroD

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

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.

Identifying Minimal Sets of Inconsistent Constraints in Linear Programs : Deletion, Squeeze and Sensitivity Filtering

by

Erik W. Dravnieks, B.Sc., M.L.I.S.

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
September 8, 1989
©copyright
1989, Erik W. Dravnieks

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

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.

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 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.

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 0-315-54426-0

Canada

Acceptance Sheet

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

Identifying Minimal Sets of Inconsistent Constraints in
Linear Programs : Deletion, Squeeze and Sensitivity Filtering

submitted by Erik W. Dravnieks, B.Sc., M.L.I.S.

in partial fulfilment of the requirements for
the degree of Master of Science
Information and Systems Science

Erik W. Dravnieks.

Thesis Supervisor

John J. ...

Chair, Department of Systems and Computer Engineering

Carleton University

September 8, 1989

Abstract

A linear program (LP) is 'primal infeasible' if it contains a set of inconsistent constraints. At present few methods exist that can efficiently identify minimal inconsistent sets of constraints within a (perhaps very large) set. An efficient method is needed for applied model building, as infeasible LPs occur frequently in the model development process.

Deletion Filtering is introduced as a method of identifying a minimal inconsistent set in small - medium sized LPs. Squeeze Filtering is incorporated into Deletion Filtering to extend the method to medium - large LPs. A further improvement is made by the addition of Sensitivity Filtering into the process. The time complexity of these algorithms is shown to be polynomial in the number of LPs to be solved.

In addition, a characterization of these minimally inconsistent constraint sets is presented.

Acknowledgments

I wish to acknowledge the guidance and support given to me by Dr. John W. Chinneck. I also wish to thank my fellow students for their friendship and comradery.

Table of Contents

Acceptance Sheet	ii
Abstract	iii
Acknowledgments	iv
List of Figures	x
List of Tables	xi
Chapter I Introduction	1
Chapter II Background and Notation	2
II.1 Linear Programming	2
II.1.1 The Dual Problem	4
II.1.2 Sensitivity to Changes in the Right Hand Side	6
II.2 Irreducibly Inconsistent Systems (IISs)	8
Chapter III State of the Art	9
III.1 Roodman	9
III.2 Greenberg	10
III.2.1 Method of Successive Reductions	10
III.2.2 Method of Path/Cycle Generation	11
III.2.3 Improving the Tracebacks	12
III.3 van Loon	13
III.4 Gleeson and Ryan	14

Chapter IV	Problem Statement	16
IV.1	Goal of the Research	16
IV.2	Deficiencies of the State of the Art	16
IV.2.1	Roodman	17
IV.2.2	Greenberg	17
IV.2.3	van Loon	18
IV.2.4	Gleeson and Ryan	18
IV.2.5	Conclusion	19
Chapter V	Characterization of the Problem	21
V.1	Introduction	21
V.2	Irreducibly Inconsistent Systems (IIS)	21
V.3	Elastic Variables and Constraints	23
V.4	Theorems and Conjectures	27
Chapter VI	Deletion Filtering for Small - Medium LPs	30
VI.1	Introduction	30
VI.2	The Deletion Filtering algorithm	31
VI.3	Validity of the Deletion Filtering Algorithm	32
VI.4	Time Complexity of the Deletion Filtering Algorithm	33
VI.5	Example Execution of the Deletion Filtering Algorithm	33
VI.6	Discussion	36

Chapter VII	Squeeze Filtering	38
VII.1	Introduction	38
VII.2	The Squeeze Filtering Algorithm	40
VII.3	Validity of the Squeeze Filtering Algorithm	41
VII.4	Time Complexity of the Squeeze Filtering Algorithm	44
VII.5	Example Execution of the Squeeze Filtering Algorithm	44
VII.6	Discussion	48
Chapter VIII	Deletion/Squeeze Filtering for Medium - Large LPs	49
VIII.1	Introduction	49
VIII.2	The Deletion/Squeeze Filtering Algorithm	49
VIII.3	Time Complexity of the Deletion/Squeeze Algorithm	50
VIII.4	Discussion	50
Chapter IX	Sensitivity Filtering	51
IX.1	Introduction	51
IX.2	The Sensitivity Filtering Algorithm	55
IX.3	Validity of the Sensitivity Filtering Algorithm	60
IX.4	Time Complexity of the Sensitivity Filtering Algorithm	61
IX.5	Example Execution of the Sensitivity Filtering Algorithm	62
IX.6	Discussion	66

Chapter X	Deletion/Squeeze/Sensitivity Filtering for Medium - Large LPs	68
X.1	Introduction	68
X.2	The Squeeze/Sensitivity Filtering Algorithm	69
X.3	Time Complexity of the Squeeze/Sensitivity Filtering Algorithm	71
X.4	The Deletion/Squeeze/Sensitivity Filtering Algorithm	72
X.5	Time Complexity of the Deletion/Squeeze/Sensitivity Filtering Algorithm	72
X.6	Discussion	73
Chapter XI	Efficiency Considerations	74
XI.1	Sparse Matrix Representation	74
XI.2	Scaling the Initial Constraint Set	75
XI.3	Removing Remnant Constraints within Deletion Filtering	75
XI.4	Advanced Starts	76
Chapter XII	Examples	78
XII.1	Example No. 1 - a single IIS	78
XII.1.1	Deletion Filtering	79
XII.1.2	Deletion/Squeeze Filtering	79
XII.1.3	Deletion/Squeeze/Sensitivity Filtering	79

XII.2	Example No. 2 : a single cluster of IISs	80
XII.2.1	Deletion Filtering	81
XII.2.2	Deletion/Squeeze Filtering	81
XII.2.3	Deletion/Squeeze/Sensitivity Filterng	81
XII.3	Example No. 3 : multiple clusters of IISs	82
XII.3.1	Deletion Filtering	83
XII.3.2	Deletion/Squeeze Filterng	84
XII.3.3	Deletion/Squeeze/Sensitivity Filterng	84
XII.4	Summary of Results	85
XII.5	Discussion	86
Chapter XIII	Comparison to State of the Art	90
Chapter XIV	Conclusions	97
XIV.1	Summary of Contributions	98
Chapter XV	Further Research	100
Appendix A	The Directed Graph of a Matrix	103
Appendix B	The 'Bump and Spike' Structure	104
Appendix C	Two Plant - Two Period Model	106
References		110

List of Figures

Figure 1	Netform Dichotomies	13
Figure 2	(a) Isolated IIS (b) Overlapped IISs	22
Figure 3	Relaxation of an elastic constraint	26
Figure 4	Example cluster of 4 IISs with non - IIS constraints	34
Figure 5	Digraph for matrix G	103
Figure 6	Bumps and spikes	104
Figure 7	Rearranged bumps and spikes	105

List of Tables

Table 1	Summary of Results	85
Table 2	Phase I LP sizes in number of constraints	86
Table 3	Computational Time Measures for the 3 Examples	88
Table 4	Table Entries are Computation Time Measures	93

Chapter III State of the Art

Few methods exist which can be used to analyze an inconsistent LP. Wiest [5] gives a brief summary of the methods that are employed in current mathematical programming systems up to early 1989. In addition to the methods covered in her survey one must include the very recent development by Gleeson and Ryan, as well as the ground breaking work by Roodman.

III.1. Roodman

An early formalized attempt at dealing with the problem of infeasible LPs is due to Roodman [6]. Roodman's solution was to examine the state of the LP when the infeasibility was detected.

If a Phase I LP was being used to determine an initial feasible point, then upon termination due to infeasibility, some of the artificial variables that were added to the LP could not be set to zero. For those constraints which correspond to such artificial variables, sensitivity analysis is applied to the right hand side in order to determine the lower bound of the magnitude of the change in the right hand side needed to achieve feasibility. By finding these lower bounds for all of the constraints with nonzero artificial variables, adjustments to the corresponding right hand side values can be made to generate a feasible solution.

If a Dual Simplex method is utilized to solve the LP, then a similar analysis can be carried out to determine the lower bounds for changes to the parameters of the LP (c'_j, b'_i, a'_{ij}) needed to generate a feasible solution.

Chapter II Background and Notation

II.1. Linear Programming

An LP has a linear objective function which is optimized (maximized or minimized) with respect to a set of linear constraints. The LP is either feasible or infeasible. A feasible LP is one in which a solution exists which satisfies all of the constraints, while an infeasible LP does not have any such solution. A feasible solution is optimal if it is the maximum (or minimum) feasible solution. If the LP is feasible but does not contain an optimal solution, then it is unbounded.

The following notation is used within this thesis to identify the components of the LP formulations:

$$\text{maximize (or minimize) } Z = cx$$

subject to :

$$Ax \rho b$$

$$x \geq 0, b \geq 0,$$

$$\rho \in \{\leq, \geq, =\}$$

Where

Z is the linear objective function

c is a cost vector $(c_1, c_2 \dots c_n), c \in R^n$.

x is a variable vector $(x_1, x_2 \dots x_n)^T, x \in R_+^n$.

b is the right hand side vector $(b_1, b_2 \dots b_m)^T, b \in R_+^m$.

A is a m by n coefficient matrix, $a_{ij} \in R$.

R_+^n denotes the n dimensional space of nonnegative Reals.

Alternatively, the set of relationships that are embodied by $Ax \leq b$ can be thought of as a set of m constraints each of the form

$$\sum_j a_{ij} x_j \leq b_i$$

$$\rho_i \in \{\leq, \geq, =\}$$

The LP formulation can be put into the canonical form of

$$Z = [c_{nbv}, c_{bv}] \begin{bmatrix} x_{nbv} \\ x_{bv} \end{bmatrix}$$

subject to :

$$[A_{nbv} | I] \begin{bmatrix} x_{nbv} \\ x_{bv} \end{bmatrix} = b$$

$$x_{nbv}, x_{bv}, b \geq 0$$

Where :

Z = objective function value

$[A_{nbv} | I]$ is the coefficient matrix, with I being an identity matrix

x_{bv} are labelled basic variables and are in general nonzero valued

x_{nbv} are labelled nonbasic variables and are zero valued

c_{nbv} and c_{bv} are respectively the columns of c that are indexed by the nonbasic and basic variables

In order to solve an LP an initial feasible solution is necessary. However, an obvious initial feasible solution may not exist. In this case one can use a Phase I LP formulation to generate an initial feasible solution. Add to each $\geq$ or $=$ constraint a unique artificial variable. For $\leq$ constraints, unique slack variables are available. This ensures that each constraint has at least one variable which a) has a coefficient of one and b) only appears in that constraint. This ensures that an initial feasible solution exists for the Phase I LP. The Phase I objective function is set equal to the sum of the artificial variables. Solving the Phase I LP by minimizing the objective function will lead to a solution that has either a) all artificial variables equal to zero, or b) some artificial variables being nonzero. If all of the artificial variables are zero, then the resultant solution is a feasible solution to the original LP. If there exists one or more nonzero artificial variables then the original LP is infeasible.

II.1.1 The Dual Problem

Every LP formulation has a dual LP formulation [2] in addition to the original primal LP.

For example the dual to the LP

$$\max Z = cx$$

$$\text{subject to } Ax \leq b$$

$$x \geq 0$$

is

$$\begin{aligned}
 \min Y &= yb \\
 \text{subject to :} \\
 yA &\geq c \\
 y &\geq 0
 \end{aligned}$$

where y is a dual variable vector $(y_1, y_2 \dots y_m), y \in R_+^m$.

By the duality theorem there are four possible combinations of feasible, unbounded or infeasible for the primal/dual LP system.

1. Both the dual and the primal are feasible, this results in the same value of the objective function for both the primal and the dual LP.
2. The primal is unbounded and the dual is infeasible.
3. The dual is unbounded and the primal is infeasible.
4. Both the primal and the dual are infeasible.

The primal LP will fail to give an optimal feasible solution if either there is no feasible solution or the primal LP is unbounded. If the primal LP is unbounded then the dual LP is infeasible. It is noted that if the dual is infeasible this does not necessarily ensure that the primal is unbounded; it may be that the primal is also infeasible.

As our emphasis is upon the primal LP, the following notation has been adopted :

primal infeasibility will be used to signify either an infeasible primal / unbounded dual LP or an infeasible primal / infeasible dual LP.

dual infeasibility will be used to signify an unbounded primal / infeasible dual LP.

II.1.2 Sensitivity to Changes in the Right Hand Side

Changing a component of the right hand side, b_i , may affect the value of the optimal solution of an LP. The general technique used to determine the amount of change (if any) is called sensitivity analysis [3]. This method requires that the optimal solution has been obtained.

Let the initial LP solution be stated in the following canonical form :

$$\begin{aligned} Z^* &= [c_{nbv}^*, c_{bv}^*] \begin{bmatrix} x_{nbv}^* \\ x_{bv}^* \end{bmatrix} \\ &\text{subject to :} \\ [A_{nbv}^* | I] \begin{bmatrix} x_{nbv}^* \\ x_{bv}^* \end{bmatrix} &= b^* \\ x_{nbv}^*, x_{bv}^*, b^* &\geq 0 \end{aligned}$$

Where * denotes the initial LP solution. Thus c_{bv}^* denotes the elements of the cost vector indexed by the initial solution basic variables.

Let the final LP solution be stated in the following canonical form :

$$\begin{aligned} Z' &= [c_{nbv}', c_{bv}'] \begin{bmatrix} x_{nbv}' \\ x_{bv}' \end{bmatrix} \\ &\text{subject to :} \\ [A_{nbv}' | I] \begin{bmatrix} x_{nbv}' \\ x_{bv}' \end{bmatrix} &= b' \\ x_{nbv}', x_{bv}', b' &\geq 0 \end{aligned}$$

Where x^* denotes the final LP solution.

The columns of $[A'_{nbv} \mid I]$ that are indexed by x'_{bv} make up the square basis matrix B of the final solution, while the columns of $[A'_{nbv} \mid I]$ that are indexed by x^*_{bv} make up the inverse of the Basis matrix B , B^{-1} .

Thus knowing the basic variables of the final solution allows one to read from the initial LP tableau the vector, c_{bv} , and from the final LP tableau the matrix, B^{-1} .

It can be shown that the optimal solution to the final LP is $Z = c_{bv} \times B^{-1} \times b^*$. Thus if one modifies the i th component of b^* , b^*_i , by an amount ϵ to get the modified vector, b^{**} , one can calculate the new value of the objective function, $Z^{**} = c_{bv} \times B^{-1} \times b^{**}$. Thus the effect of a change of a right hand side element on the optimal objective function can be found by $Z' - Z^{**}$.

This updated objective function value is valid if the corresponding updated solution is still feasible and optimal. As long as the the right hand side value of each constraint remains nonnegative in the updated solution, the updated solution is feasible and optimal. As $B^{-1} \times b^{**} = \text{updated right hand side}$, a simple calculation will indicate if the modification of the i th constraint will result in a feasible and optimal solution. If the updated solution is no longer feasible and optimal, then it is infeasible. In this case, the actual updated optimal solution can be found by applying the Dual Simplex method using the updated solution as the initial dual feasible (primal infeasible) starting point (see XI.4 for a related discussion). The Dual Simplex method uses information from the dual formulation of the LP to solve the LP.

If the right hand side of the initial optimal solution is > 0 , then a small enough ϵ can always be found such that the updated right hand side of the updated solution will also be > 0 . Therefore, for infinitesimal ϵ 's, if the right hand side of the initial optimal solution is strictly positive, then the updated optimal solution will be feasible and optimal. Thus the effect (increase or decrease) on the objective function value by adjusting the i th component of the right hand side, b_i^* , can be found by examining the signs of the ϵ and the corresponding i th component of the vector that results from the multiplication of $c_{bv}^* \times B^{-1}$.

The case in which the initial optimal right hand side has components $b_i = 0$ corresponds to a degenerate system. If the initial optimal right hand side is strictly > 0 , then no degeneracy occurs.

II.2. Irreducibly Inconsistent Systems (IISs)

A system of constraints is primal infeasible if it contains a set of inconsistent constraints.

Definition 2.2.1 : An Irreducibly Inconsistent System (IIS) is a minimal set of inconsistent constraints. (see also [1],[4])

An IIS is an inconsistent set, so an LP is primal infeasible if it contains an Irreducibly Inconsistent System. One focuses on IISs as the removal of any constraint from an IIS removes the inconsistency. Locating an IIS in a large set of constraints localizes the problem so that appropriate corrections can be made.

Chapter III State of the Art

Few methods exist which can be used to analyze an inconsistent LP. Wiest [5] gives a brief summary of the methods that are employed in current mathematical programming systems up to early 1989. In addition to the methods covered in her survey one must include the very recent development by Gleeson and Ryan, as well as the ground breaking work by Roodman.

III.1. Roodman

An early formalized attempt at dealing with the problem of infeasible LPs is due to Roodman [6]. Roodman's solution was to examine the state of the LP when the infeasibility was detected.

If a Phase I LP was being used to determine an initial feasible point, then upon termination due to infeasibility, some of the artificial variables that were added to the LP could not be set to zero. For those constraints which correspond to such artificial variables, sensitivity analysis is applied to the right hand side in order to determine the lower bound of the magnitude of the change in the right hand side needed to achieve feasibility. By finding these lower bounds for all of the constraints with nonzero artificial variables, adjustments to the corresponding right hand side values can be made to generate a feasible solution.

If a Dual Simplex method is utilized to solve the LP, then a similar analysis can be carried out to determine the lower bounds for changes to the parameters of the LP (c'_j, b'_i, a'_{ij}) needed to generate a feasible solution.

However, there is no real control over the generation of this new feasible solution and as such this solution may have little significance for the modeller.

III.2. Greenberg

The topic of the treatment of infeasibility in LPs as a facility for a mathematical programming system has been studied by Greenberg [7 – 11]. The methods that Greenberg employs are discussed below.

III.2.1 Method of Successive Reductions

The basic concept that underlies this approach is that of a redundant constraint. A redundant constraint can be removed from the LP as it does not affect the feasible region of the LP. The topic of redundant constraints and methods that can be used to detect them has been the subject of many papers and books (for example [12 – 18]).

Ideally one would like to be able to simply find and subsequently remove all of the redundant constraints from the LP and thus be left with only those constraints which border the feasible region or the IISs. However, the methods that have been created to find all of the redundant constraints either assume the existence of a feasible solution and/or are inefficient as they require the solution of a large number of LPs.

The approach taken is to change the canonical form of the LP so that the primal and dual variables have lower and upper bounds. A battery of tests [7] have been developed that will attempt to strengthen the bounds on these variables as well as test the constraints for redundancy. This battery of tests is applied repeatedly to the LP until no further change occurs, or until an infeasibility is detected. At

each iteration, any reduction on the bounds of a variable are recorded for future 'tracebacks'. This simplification process will tend to restrict constraints to single variable constraints as well as identify redundant constraints. Both of these types of constraints can be used in an efficient manner to speed up the solution of the LP.

This method is used primarily to simplify or reduce a large LP into a simpler form which can then be solved more efficiently. In the process of simplification the method detects infeasibility in the LP.

Once the method detects an infeasibility, an attempt to identify the IIS can be made by 'tracing back' through the reductions that have occurred. As constraints which are not members of an IIS are also reduced the detection of the IIS can be quite difficult, especially if there are many interrelated IISs.

In conclusion, this method will detect primal infeasibility and primal unboundedness and can in simple cases localize the cause.

III.2.2 Method of Path/Cycle Generation

This is an heuristic method which attempts to find paths or cycles in the directed graph created from the coefficient matrix, A , of the LP (see Appendix A). The method used to find these cycles is a specialized version of that developed by Johnson [19]. If the path or cycle is negative and unbounded then the primal LP is unbounded. The corresponding constraints of these paths/cycles can then be easily found.

This method is used primarily to detect primal unboundedness and the constraint structures that cause it to be so. It has been modified to detect primal infeasibility and thus IISs. In either format, the method does not guarantee the identification of an IIS if one exists, and as such its use is limited.

III.2.3 Improving the Tracebacks

In addition Greenberg has suggested the use of two untested methods that may improve the performance of the traceback process. They are netforming and coefficient screening.

Netforming is based upon the identification of network substructures within the LP. These substructures can be found by various means; one simple approach is to search the coefficient matrix A for columns (or rows) which have either a single nonzero or two nonzeros of opposite sign. This information can then be used to determine if the matrix A can be adjusted by column (or row) switching to produce a matrix of one of the forms in figure 1. If a netform is present in the LP then these structures can be examined for infeasibilities. The special characteristics of a netform make it easier to search for inconsistent sets of constraints. This is partially due to the availability of specialized methods for solving network LPs. As the localization, if any, is to a smaller subset of constraints but not necessarily to an IIS, the method must be used in conjunction with other methods. Its use is further limited by the fact that some IISs may cross the boundary of the netform. As well not all LPs will contain netforms. Thus this method is in essence an heuristic which can be used to localize the search for the IIS.

Coefficient screening works upon a similar concept, that of using the 'bump and spike' structure of the basis of the LP (see Appendix B). Again this method tries to localize the problem to a subset of constraints.

These methods may be useful in localizing the search for the IIS to a smaller subset of constraints, but they must be used in conjunction with other more robust methods to guarantee that only an IIS is output.

$N :=$ Netform Matrix

$R :=$ Remainder Matrix

Figure 1: Netform Dichotomies

III.3. van Loon

A method to identify an IIS in a system of linear inequalities has been described by van Loon [4]. Basically, the set of original constraints is put into a standard form of $Ax \leq b : x \geq 0$ and then a vector of slack variables, y is added to produce the system $Ax + y = b : x, y \geq 0$.

A necessary and sufficient 'row condition' for the existence of an IIS is shown to be :

1. a slack variable y_i as the basis variable for the row.
2. the right hand side of the row is strictly negative.
3. all of the x_i columns have zero coefficients for that row.
4. all of the v_i columns have positive coefficients for that row.

If such a row condition exists then the constraints corresponding to the y_i columns with strictly positive coefficients in that row plus the constraint corresponding to the y_j variable index an IIS.

A simplex related method is applied to the new system to attempt to bring all of the x_i 's into the basis. If at any point the system is feasible, then there are no IISs, else at some point either all of the x_i 's have been placed into the basis or a point where no more can be entered into the basis has been reached. The next phase of the algorithm is to find a row that satisfies the 'row conditions'.

To achieve such a row condition one manipulates the tableau by bringing different y_i 's into the basis (taking out y 's from the basis thus leaving the existing x_i 's in the basis).

This method will identify an IIS and by searching for all rows that satisfy the 'row condition', all of the IISs can be identified. As the search for all IISs is an undirected search, the only real method is to exhaustively search all of the possible combinations of y_i 's into the basis. Thus the search for all IISs is inefficient.

III.4. Gleeson and Ryan

Late in the writing of this thesis, a polynomial time algorithm for the identification of all of the IISs in an infeasible LP was developed by Gleeson and Ryan in an as yet unpublished paper [20].

The basic idea is an extension of van Loon's result. A polyhedron is created such that the 'supports' of the vertices of the polyhedron are the indices of an IIS. The 'support' of a vertex denotes the nonzero components of the vector that describes that vertex. The polyhedron, P , that is used is defined by

$P = \{y \in R_+^m \mid y^T A = 0, y^T b \leq -1\}$ where P is a modified equivalent version of one of the alternatives of Farkas Theorem of the Alternate [21] (the other alternate is of course $\{x \in R^n \mid Ax \leq b\}$). This 'simple' polyhedron, P , is then searched for all of its vertices. The time complexity for locating the vertices is polynomial as shown by Dyer [22].

The time complexity of the algorithm is shown to be $O([n + 1]^2 mr)$ where n is the number of constraints, m is the number of variables and r is the number of IISs.

Chapter IV Problem Statement

IV.1. Goal of the Research

The aim of the research in this thesis is to develop a diagnostic tool to aid the modeller in the analysis of an infeasible LP. The most useful such tool would be a way to identify IISs. This is because they are a) minimal structures that do not carry with them any irrelevant information and b) necessary and sufficient structures for primal infeasibility. Subsequent correction of the constraints of the IISs to remove the inconsistency associated with the IISs will render the model feasible.

The ideal algorithm for identifying IISs is one which would efficiently identify all of the IISs that are present in the LP. This would make available to the modeller sufficient information to properly diagnose the current LP model.

If such an algorithm does not exist, then the next best algorithm would efficiently identify a single IIS. By repeatedly applying the algorithm and removing the IISs as they are identified, the modeller has a methodology for modifying the LP until it becomes feasible.

The goal of this thesis is to develop an efficient and robust method for the identification of a single IIS in a primal infeasible LP.

IV.2. Deficiencies of the State of the Art

There are several difficulties with the current methods of Roodman, Greenberg, van Loon and Gleeson and Ryan that limit the usefulness of these methods.

IV.2.1 Roodman

Roodman does not identify the IIS, but skirts the issue by finding the lower bounds for modifications needed to the right hand side to obtain a feasible solution. No attempt is made to locate the IIS that has caused the infeasibility.

IV.2.2 Greenberg

Successive reduction does not necessarily easily identify the IIS and as the traceback process requires the interaction of the modeller with the algorithm, the use of this method for detecting IISs is cumbersome.

Thus at best this method can be used to reduce the problem size of the LP. However, it does so by manipulating the constraints. Thus one has to maintain a record of changes so as to be able to properly identify which original constraints are members of an IIS.

Path/cycle generation is not robust; it can fail to identify an IIS as it relies on an heuristically determined starting point. By applying the method to all possible starting configurations, the method could possibly find the IIS, but this will reduce its efficiency.

Netforming and coefficient screening are only heuristics that may be useful in pruning down the search space of constraints for the IIS and as such do not identify any IISs. As well, they are untested at present and thus their worth is not known.

Conclusion : as these methods do not guarantee the identification of an IIS they do not solve our research problem.

IV.2.3 van Loon

As the right hand side is not constrained to be non-negative, non-negativity constraints are not implicitly included in the 'simplex' method. In order to match the LP formulation one must include non-negativity constraints, thus extra constraints must be explicitly added to the tableau. In addition, any equality constraints must be represented as two inequalities. The net effect is to increase the number of constraints in the LP by an amount equal to the number of sign restricted variables plus the number of equality constraints.

The other major problem is that there is no efficient method for deciding which of the y_i 's are to be added into the basis and which are to be removed from the basis. The search for a 'row condition' is an unguided search, which will in general only be solved by an exhaustive search of all combinations. Thus the problem of identifying a single IIS is combinatorial in nature. As well, the only way to ensure that all of the IISs have been identified is to check all possible combinations of y_i 's as basis variables. Thus to use this method to identify all of the IISs is combinatorially complex.

In conclusion, this method will identify an IIS, but it is in general inefficient in time and space and thus has limited use for large LPs.

IV.2.4 Gleeson and Ryan

The method presented by Gleeson and Ryan is an elegant solution that appears to be able to find all of the IISs of an infeasible LP.

However, as the polyhedron used is defined for $x \in R^n$, this method does not implicitly imply nonnegativity of the variables. Thus an addition of rows

equal to the number of sign restricted variables is needed in order to properly mimic the original LP formulation. In addition, any equality constraints in the original LP have to be represented as two inequality constraints. Thus an LP with a large number of equality constraints, such as for a network, would require a large increase in the number of rows. As the time complexity of the method is polynomial in the square of the number of constraints, the addition of a large number of extra constraints may cause this method to become inefficient under these conditions.

As the method is extremely recent and untested there may be other problems of implementation that have not been identified as of yet.

Thus this method shows promise of being an efficient method for finding all of the IISs of an infeasible LP. However, it may be inefficient for large network LPs, or other problems which have a large number of equality constraints as well as a large number of nonnegative variables.

IV.2.5 Conclusion

The methods of Greenberg and Roodman do not robustly solve the problem of IIS identification.

The algorithm of van Loon is inefficient in time complexity. For LPs with a large number of equalities and/or sign restricted variables this algorithm will also be inefficient in space complexity. On the other hand, the algorithm of Gleeson and Ryan will efficiently identify all of the IISs, except perhaps for LPs with large numbers of sign restricted variables and/or equality constraints.

The use of sparse matrix techniques may improve the space requirements for both van Loon's and Gleeson and Ryan's methods; especially for the additional

constraints required to maintain nonnegativity of the variables.

Thus Gleeson and Ryan's algorithm is an efficient algorithm for the identification of all IISs in an infeasible LP. However, this method is untested and as it may be inefficient for large network-like LPs the study of alternative methods is still worthwhile. In particular this research may prove worthwhile in the case of LPs with many equality constraints and sign restricted variables. In addition, if alternate methods are found this may a) help to develop the theory of inconsistent sets of constraints b) create a list of methods, each suited for a particular type and size of LP. The net effect is that more tools become available to aid in the diagnosis of infeasible LPs.

Thus it is worthwhile to investigate alternative methods to Gleeson and Ryan, in the hopes of finding a method which will perhaps outperform this algorithm under special conditions.

Chapter V Characterization of the Problem

V.1. Introduction

Insight into the structure of IISs will help to develop and to understand the methods for identifying them. Thus it is necessary to develop the concept of the IIS and to characterize the types of groupings that IISs may take.

As methods exist for analyzing feasible LPs, perhaps creating a controllable environment in which an infeasible LP becomes feasible will lead to a method for identifying IISs. With this in mind, elastic variables and constraints are introduced as a method for transforming an infeasible LP into a feasible LP, while still keeping enough information available for the identification of IISs.

V.2. Irreducibly Inconsistent Systems (IIS)

Definition 5.2.1 : An IIS is isolated if it does not share a constraint with any other IIS.

Definition 5.2.2 : An IIS is overlapped if it shares at least one constraint with another IIS.

Definition 5.2.3 : A cluster of IISs is a maximal collection of IISs such that each IIS overlaps at least one other IIS of the cluster.

Therefore two clusters do not share any constraints and an isolated IIS is a special case of a cluster of IISs.

Figure 2: (a) Isolated IIS (b) Overlapped IISs

In figure 2a the constraints $\{a, b, c, d\}$ form an inconsistent set of constraints, of which $\{a, b, c\}$ form an IIS. The IIS $\{a, b, c\}$ forms a cluster of IISs.

In figure 2b there are two IISs, namely that of $\{a, b, c\}$ and that of $\{a, c, e\}$, which are overlapped IISs as they share the constraints a and c . The IISs $\{a, b, c\}$ and $\{a, c, e\}$ form a cluster of IISs.

In general an LP may have more than one IIS. Given a set of constraints one can isolate the individual IISs within that set.

Definition 5.2.4 : The process of isolating individual IISs from a set of constraints is called filtering.

Definition 5.2.5 : A non-IIS constraint is a constraint which is not a member of any IIS.

Definition 5.2.6 : An IIS constraint is a constraint which is a member of an IIS.

Definition 5.2.7 : A remnant constraint is a constraint which is a member of an IIS of the original LP, but it is not a member of any complete IIS in the current subset of constraints.

During the process of filtering a set of constraints, only some of the constraints of an IIS of the original set may be included in the output set of the filtering process. Thus remnant constraints may occur in the output set of a filtering process.

V.3. Elastic Variables and Constraints

For dealing with infeasible LPs, Roodman [6] has suggested the addition of variables to the constraint set that guarantee a feasible solution by allowing each constraint to be 'relaxed'. In this way a feasible solution may be generated from the infeasible LP. This concept has been labelled 'elastic programming' by Brown and Graves [23].

Let us now formalize the concept of elastic programming by introducing elastic constraints and variables. A constraint is elastic if the constraint can be

adjusted to increase the feasible region, that is the constraint can be 'relaxed'. A constraint can be converted to an elastic constraint by the addition of one or two 'elastic' variable(s).

Consider a constraint of the form $\sum x_i \geq b$, $b \geq 0$. If an 'elastic' variable v , $v \geq 0$, is added to this constraint, the result is the 'elastic' constraint : $\sum x_i + v \geq b$. The elastic variable v has made the constraint elastic. When $v = 0$ the constraint is in its original state and when v is nonzero the value of v determines by how much the constraint has been relaxed.

For a constraint of the form $\sum x_i \leq b$, an elastic variable v is subtracted from the constraint to get the 'elastic' constraint $\sum x_i - v \leq b$. For equality constraints both the addition of an elastic variable, v' and the subtraction of an elastic variable, v'' , is necessary so as to allow the constraint to be elastic in both directions. That is $\sum x_i = b$ becomes $\sum x_i + v' - v'' = b$.

Thus the 'rules of conversion' for changing non-elastic constraints to elastic constraints and vice-versa are:

	non-elastic		elastic
$\geq$ inequalities :	$\sum x_i \geq b$	$\Leftrightarrow$	$\sum x_i + v \geq b$
$\leq$ inequalities :	$\sum x_i \leq b$	$\Leftrightarrow$	$\sum x_i - v \leq b$
$=$ equalities :	$\sum x_i = b$	$\Leftrightarrow$	$\sum x_i + v' - v'' = b$
	$b, v, v', v'' \geq 0$		

The resistance to relaxation implied by the term "elastic" is supplied by adding a term to the objective function which has a worsening effect on the objective

function value as the magnitude of the elastic variable increases.

Definition 5.3.1 : A non-elastic constraint is a constraint from the original LP formulation before the addition of slack, surplus, or artificial variables.

Definition 5.3.2 : An elastic variable is a variable added to a non-elastic constraint, according to the rules of conversion.

Definition 5.3.3 : An elastic constraint is a non-elastic constraint to which elastic variables have been added.

Definition 5.3.4 : A relaxed constraint is an elastic constraint in which one of the elastic variables has a nonzero value.

By sufficiently relaxing a constraint that is a member of an IIS, the inconsistency associated with the IIS is removed. For example constraint b in figure 3 is relaxed by an amount k; constraint b is in effect replaced by b' or constraint b has been relaxed to the new position b', and the IIS has been 'removed'.

Consider a function, 'elastic', which takes a non-elastic constraint and inserts the proper elastic variables, as per the rules of conversion, to create an elastic constraint. As there is a one to one correspondence between the elastic constraint and the non-elastic constraint, the inverse function 'elastic⁻¹' is well defined. Our notation is as follows :

$$\epsilon_i = \text{elastic}(q_i)$$

$$q_i = \text{elastic}^{-1}(\epsilon_i)$$

Figure 3: Relaxation of an elastic constraint

where q_i is the original constraint, and e_i is the corresponding elastic constraint. Extending this notation to include sets as arguments results in

$$E = \text{elastic}(Q)$$

$$Q = \text{elastic}^{-1}(E)$$

where the set E is a set of elastic constraints that result from applying the elastic function to each non-elastic constraint of the set Q .

Consider a function, 'vin' (for "variable in"). This function takes a set of constraints as its argument and returns the set of elastic variables in that set. This allows one to access all of the elastic variables in a constraint set. Our notation is as follows :

$$V = \text{vin}(E)$$

Where E is a set of elastic constraints, and V is the set of all elastic variables in the constraints in the set E .

V.4. Theorems and Conjectures

Theorem 5.4.1 : If there are n variables in the original LP, then the maximum cardinality of any IIS is $n + 1$.

Proof:

See Chvátal [24, p. 146].

The material presented below is used primarily to help explain Sensitivity Filtering as presented in Chapter IX. At present proofs have not been found for the following conjectures. An initial examination indicates that the needed proofs are of a level of complexity that is beyond the scope of this thesis. However, experience has shown that these conjectures seem to be valid. If so, then one can use Sensitivity Filtering in place of Squeeze Filtering (see Chapters VII and X). Otherwise, Sensitivity Filtering can only be used as a heuristic method within Squeeze Filtering.

Conjecture 5.4.2 : For an IIS in R^n of size $m > 2$ constraints, any subset of size $m - 1$ constraints intersect at a common junction. A common junction is either a point, line, plane, or hyper-plane

Definition 5.4.1 : The interior region of an IIS is a set of points for which every point satisfies none of the constraints of the IIS.

Intuitively this region must always be nonempty, but to date a proof has not been found.

Let us examine a non-IIS constraint and its relationship to the interior region of an IIS. Consider an IIS, I_1 , with an interior region, d . Consider a constraint, q , that partitions the interior region, d into two nonempty sets. The constraint q plus some proper nonempty subset of the constraints of I_1 forms an IIS. This is due to the fact that one of the partitioned regions of the region d will be part of the interior region of the new IIS. Thus, a non-IIS constraint will not partition or cross the interior region of any IIS.

The concept of a redundant constraint is extended for infeasible LPs.

Definition 5.4.2 : A redundant constraint is a constraint which when removed from the LP does not change either the solution space of the LP or the interior region of the LP.

Definition 5.4.3 : An extending IIS is an IIS in a cluster of IISs whose interior region is not contained in the union of the interior regions of the other IISs of the cluster.

A cluster of IISs will always contain at least one extending IIS, as otherwise one of the IISs is the union of more than one IIS. This contradicts the minimality of IISs. A cluster of IISs may contain more than one extending IISs.

Conjecture 5.4.3 : If an IIS constraint is relaxed by an amount ϵ in the direction into the interior region, then the cost to relax any other constraint of that IIS so as to eliminate the inconsistency of the IIS has been reduced.

Conjecture 5.4.3 can be shown to be true if Conjecture 5.4.2 is true. This is because the act of relaxing a constraint into the interior region of the IIS will decrease the distance through which the other constraint must be relaxed. This will result in a decreased cost.

Chapter VI Deletion Filtering for Small - Medium LPs

VI.1. Introduction

The goal of Deletion Filtering is to identify a single IIS out of a set of constraints which contain at least one IIS.

Consider any constraint from an infeasible LP. If a constraint is removed and the LP remains infeasible, then at least one IIS is still present in the LP. If a constraint is removed and the LP becomes feasible, then that constraint is a member of all of the IISs in the LP. By systematically checking all of the constraints of the LP in this manner, and removing those whose removal maintains at least one IIS, the set of constraints is pruned down to a single IIS.

A simple Phase I LP can be used to determine the infeasibility/feasibility of the constraint set. It is assumed that the original LP is infeasible.

Nonnegativity of the variables is implicit, rather than explicit, in the LP model. Thus if an IIS exists which includes a nonnegativity constraint, the set output by this algorithm will not include that nonnegativity constraint. For example, $x_1 \leq -1$, becomes an IIS, with an implicit second constraint of $x_1 \geq 0$. In addition, the algorithm will not explicitly indicate that this has occurred. As the addition of explicit nonnegativity constraints will not alter the implied nonnegativity, the algorithm can not be changed to detect nonnegativity.

“Deletion Filtering” is the name given to the process of filtering an inconsistent set of constraints by the successive deletion of constraints with the goal of identifying a single IIS.

The test for feasibility in the algorithm below is an ordinary Phase I simplex.

VI.2. The Deletion Filtering algorithm

Definitions:

- a. Let Q be a set of constraints, q_i .
- b. Let Q_{Input} be the initial set of constraints.
- c. Let Q_{Output} be the final set of constraints.

1. Initialize the set of constraints Q : Set $Q = Q_{\text{Input}}$.
2. Test constraints for deletion :

For each $q_i \in Q$ Do :

- i. Determine the infeasibility/feasibility of the constraint set $Q \setminus \{q_i\}$:
- ii. If $Q \setminus \{q_i\}$ is feasible

THEN continue

ELSE ($Q \setminus \{q_i\}$ is infeasible)

- Set $Q = Q \setminus \{q_i\}$
- continue

3. $Q_{\text{Output}} = Q$.

The set Q_{Output} is an IIS.

VI.3. Validity of the Deletion Filtering Algorithm

Theorem 6.3.1 The Deletion Filtering algorithm will identify a single IIS from an input set of constraints containing at least one IIS.

Proof:

As there exists at least one IIS, the input set Q is infeasible. As a constraint q_i is only removed from the set Q when the set $Q \setminus \{q_i\}$ is infeasible, the set Q is always infeasible, and therefore the set Q always contains at least one IIS.

Any remnant constraints or non-IIS constraints will be deleted from the set Q , as they do not affect the infeasibility of the set Q . Therefore, only the case of clusters of IISs need be examined.

As there are no constraints in common between two clusters of IISs, if there is more than one cluster of IISs in the current set Q , then the cluster constraint being tested will be removed. Therefore, multiple clusters of IISs will reduce down to a single cluster. Any remnant constraints that arise due to these deletions will be removed. Let us now examine what happens when only one cluster is left.

If this cluster is an isolated IIS, then any remnant or non-IIS constraints will be removed to leave just the single IIS.

Otherwise the cluster consists of overlapped IISs. If the constraint being checked does not belong to all of the remaining IISs it will be removed. Therefore, the collection of IISs will also reduce down to a single IIS.

Therefore, as the collection of IISs is reduced down to one IIS, and as all remnant and non-IIS constraints are removed, the Deletion Filtering algorithm will identify a single IIS. Q.E.D.

VI.4. Time Complexity of the Deletion Filtering Algorithm

Let m be the number of constraints in the set Q_{Input} for the Deletion Filtering Algorithm.

Theorem 6.4.1 : The time complexity of the Deletion Filtering algorithm is $[m * \text{Time Complexity of a Phase I LP}]$.

Proof:

As the algorithm checks each constraint in the set Q_{Input} exactly once, m steps are needed. As each check involves a Phase I solution, the Time Complexity is $[m * \text{Time Complexity of a Phase I LP}]$. Q.E.D.

Each step of the algorithm will either remove a constraint, or continue. Therefore, the LP size will never increase, but will most likely decrease.

VI.5. Example Execution of the Deletion Filtering Algorithm

Consider the constraint set of figure 4 :

1. $q_1 : x_1 \leq 2$
2. $q_2 : x_2 \leq 2$
3. $q_3 : x_1 + x_2 \leq 7$
4. $q_4 : x_1 + x_2 \geq 9$
5. $q_5 : 3x_1 + x_2 \geq 15$
6. $q_6 : x_1 + 2x_2 \leq 22$
7. $q_7 : x_1 \leq 7$

Figure 4: Example cluster of 4 IISs with non - IIS constraints

Which by inspection contains the following IISs :

$$\{q_3, q_4\}, \{q_1, q_2, q_4\}, \{q_1, q_2, q_5\}, \{q_1, q_3, q_5\}$$

Step 1 : Initialize the set of constraints Q :

$$Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$$

Step 2 : Test constraints for deletion :

Iteration 1 :

a. Choose q_1 from $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$

b. $Q \setminus \{q_1\}$ is infeasible

$$Q = \{q_2, q_3, q_4, q_5, q_6, q_7\}$$

c. next q_1

Iteration 2 :

- a. Choose q_2 from $Q = \{q_2, q_3, q_4, q_5, q_6, q_7\}$
- b. $Q \setminus \{q_2\}$ is infeasible
- c. $Q = Q \setminus \{q_2\} = \{q_3, q_4, q_5, q_6, q_7\}$
- d. next q_i

Iteration 3 :

- a. Choose q_3 from $Q = \{q_3, q_4, q_5, q_6, q_7\}$
- b. $Q \setminus \{q_3\}$ is feasible
- c. next q_i

Iteration 4 :

- a. Choose q_4 from $Q = \{q_3, q_4, q_5, q_6, q_7\}$
- b. $Q \setminus \{q_4\}$ is feasible
- c. next q_i

Iteration 5 :

- a. Choose q_5 from $Q = \{q_3, q_4, q_5, q_6, q_7\}$
- b. $Q \setminus \{q_5\}$ is infeasible
- c. $Q = Q \setminus \{q_5\} = \{q_3, q_4, q_6, q_7\}$
- d. next q_i

Iteration 6 :

- a. Choose q_6 from $Q = \{q_3, q_4, q_6, q_7\}$
- b. $Q \setminus \{q_6\}$ is infeasible
- c. $Q = Q \setminus \{q_6\} = \{q_3, q_4, q_7\}$

d. next q_i

Iteration 7 :

a. Choose q_7 from $Q = \{q_3, q_4, q_7\}$

b. $Q \setminus \{q_7\}$ is infeasible

c. $Q = Q \setminus \{q_7\} = \{q_3, q_4\}$

d. next q_i

All $q_i \in Q$ have been tested.

Step 3 : $Q_{\text{Output}} = Q = \{q_3, q_4\}$

Q_{Output} is an IIS.

VI.6. Discussion

The algorithm takes a predetermined number of steps to identify a single IIS. As this number is equal to the number of constraints, this method is only suitable for small to medium sized LPs.

When the number of non-IIS constraints is large then the algorithm will spend a large amount of time on these constraints. An algorithm that quickly removes all of the non-IIS constraints from the constraint set would be useful in pruning the constraint set to a size where the Deletion Filtering algorithm could be applied in an efficient manner. In this way, the algorithm could be extended to large LPs as well. This pruning algorithm could be either applied to the set of constraints before the application of Deletion Filtering or during the Deletion Filtering algorithm.

The Deletion Filtering algorithm does not add additional constraints or variables to the LP. If there is enough space on the computer to solve the LP, there is enough space to carry out the algorithm.

On the minus side, Deletion Filtering fails to leave any indication of whether other IISs are present in the LP.

Chapter VII Squeeze Filtering

VII.1. Introduction

The Squeeze Filtering algorithm is designed to remove all of the non-IIS constraints from an infeasible LP constraint set, such that at least one IIS remains.

The algorithm uses the fact that relaxing any constraint in an IIS removes the inconsistency of the IIS. By converting all of the non-elastic constraints to elastic constraints, a system of constraints is produced which will always have a feasible solution. This technique is parallel to the Phase I method. A positive cost is assigned to the relaxation of an elastic constraint by placing the corresponding elastic variable into the objective function. By minimizing the sum of the elastic variables only elastic constraints which correspond to non-elastic constraints in an IIS are relaxed. By selectively "squeezing" on a subset of the elastic constraints (by removing their elastic variables), it is ensured that different elastic constraints are relaxed in each step of the algorithm.

The LP defined by the set of 'squeezed' and elastic constraints is labelled the 'Filtering LP'. The objective function for the Filtering LP is the sum of the elastic variables present in the Filtering LP. The initial Filtering LP consists of all of the elastic constraints. The addition of slack and surplus variables to the $\leq$ and $\geq$ constraints ensures that an initial feasible solution exists. This is because each constraint will have at least one variable with a coefficient of 1 due to either an elastic variable or due to a slack or surplus variable.

The Filtering LP is then solved; the set of relaxed constraints are changed to non-elastic constraints. These non-elastic constraints are all members of an IIS. The constraint set for the Filtering LP now contains elastic constraints and non-elastic constraints (the squeezed constraints). In order to solve this Filtering LP, one adds slack variables to the elastic constraints and artificial variables to the non-elastic constraints (if necessary). An initial Phase I LP (using the artificial variables, etc.) is used to determine if the Filtering LP is feasible or infeasible. If it is feasible, then the solution of the Phase I LP is used as the initial feasible solution to determine the optimal solution for the Filtering LP. After the Filtering LP has been solved, all of the relaxed constraints of this solution are changed to non-elastic constraints. By repeating this process, more and more IIS constraints are identified.

Once all of the constraints of an IIS have been identified, and thus converted to non-elastic constraints, the Filtering LP becomes infeasible. This gives us a simple stopping condition. The set of constraints which are non-elastic form our output set, which contains at least one IIS. As the number of constraints in an IIS is finite, the algorithm terminates in a finite number of steps.

As at each step at least one constraint per IIS is relaxed, the number of steps of the algorithm will be bounded by the number of constraints in the smallest IIS.

The Squeeze Filtering algorithm will take a set of inconsistent constraints, and will remove from this set all of the non-IIS constraints. As it stops when all of the constraints of a particular IIS are identified, some IIS constraints which are members of other IISs may not be identified, and therefore the output set may contain some remnant constraints.

As nonnegativity constraints are not relaxed, all of the IISs that the algorithm identifies may contain some nonnegativity constraints as well.

Thus, Squeeze Filtering will identify a set of constraints which contain at least one complete IIS. In addition, each of the constraints identified will be a member of some IIS, which is perhaps incomplete in the output set.

VII.2. The Squeeze Filtering Algorithm

Definitions:

- i. Let Q_{Input} be the initial set of non-elastic constraints, q_i .
 - ii. Let Q_{Output} be a set of non-elastic constraints, all of which belong to at least one IIS in the set Q_{Input} , and which contains at least one IIS.
 - iii. Let E be the set of elastic constraints, e_i , that are in the constraint set of the Filtering LP.
 - iv. Let $V = \text{vin}(E)$, be the set of elastic variables with non-zero coefficients in the elastic constraint set E .
 - v. Let R be the set of relaxed elastic constraints associated with the solution of a Filtering LP.
1. Initialize the sets of constraints and variables:
 - a. Set $Q_{\text{Output}} = R = \emptyset$.
 - b. Set $E = \text{elastic}(Q_{\text{Input}})$.
 - c. Set $V = \text{vin}(E)$.

2. Solve the initial Filtering LP to detect if the original LP is feasible :

Solve

$$\min Z = \sum_{v_i \in V} v_i$$

such that : $E, Nonnegativity$

If $Z = 0$ THEN STOP, the original LP is feasible

ELSE This results in the set R , set $Q_{Output} = Q_{Output} \cup \text{elastic}^{-1}(R)$, continue

3. Convert the relaxed constraints to non-elastic constraints :

- a. Set $E = E \setminus R$.
- b. Set $R = \emptyset$.
- c. Set $V = \text{vin}(E)$.

4. Solve the Filtering LP.

$$\min Z = \sum_{v_i \in V} v_i$$

such that : $Q_{Output}, E, Nonnegativity$

If Filtering LP is infeasible, THEN STOP

ELSE This results in the set R , $Q_{Output} = Q_{Output} \cup \text{elastic}^{-1}(R)$, go to step 3.

VII.3. Validity of the Squeeze Filtering Algorithm

Theorem 7.3.1 : The Squeeze Filtering algorithm will only relax those elastic constraints which correspond to non-elastic constraints which are elements of an IIS.

Proof:

Consider a non-elastic constraint which is not a member of any IIS. This means that this constraint does not intersect the interior region of any IIS. Therefore, relaxing the corresponding elastic constraint does not affect the interior region of any IIS. Thus the corresponding elastic constraint does not need to be relaxed in the Filtering LP to obtain a feasible solution. Therefore, as the cost of relaxing any constraint is strictly positive, and as the minimum cost solution is always chosen, an elastic constraint which corresponds to a non-elastic constraint which is not a member of an IIS will never be relaxed. Q.E.D.

Lemma 7.3.2 : For the Squeeze Filtering algorithm, if at least one IIS exists in the set Q_{Input} and the current Q_{Output} set does not contain an IIS, then the Filtering LP will only relax previously non-relaxed elastic constraints and it will relax at least one such constraint from each IIS.

Proof:

As at least one IIS exists, the original LP is primal infeasible, and therefore in order to achieve a feasible solution, the Filtering LP must relax at least one of the elastic constraints that correspond to the non-elastic constraints of each IIS in order to eliminate the inconsistency of each IIS.

All of the previously relaxed constraints have been converted to non-elastic constraints, (by step 3) and therefore these constraints can no longer be relaxed. Therefore, only previously non-relaxed constraints can be relaxed.

If the Q_{Output} set does not yet contain an IIS, then at least one constraint per IIS is still elastic in the Filtering LP, and thus a feasible solution to the Filtering

LP exists by relaxing these constraints.

Therefore, the Filtering LP will only relax previously non-relaxed elastic constraints and it will relax at least one such constraint from each IIS. Q.E.D.

Theorem 7.3.3 : The Squeeze Filtering algorithm will find at least one IIS if and only if an IIS exists in the original LP.

Proof:

$\Rightarrow$

If no IIS exists in the original LP, then no constraints are needed to be relaxed to get a feasible solution in the Filtering LP. Any such feasible solution will have a cost of zero in the Filtering LP. As the cost of relaxing any constraint is strictly positive, and as the minimum cost solution is found, the optimal solution will not relax any constraints. By step 2 of the algorithm, if the objective function is equal to zero, the algorithm stops and indicates that no IIS exists. Thus the algorithm will not find an IIS. Q.E.D.

$\Leftarrow$

If an IIS exists in the original LP, then the Filtering LP must relax at least one constraint from each IIS in order to eliminate the inconsistency of each IIS. As the cost of relaxing a constraint is strictly positive, the objective function will be greater than zero. Thus step 2 of the algorithm indicates that an IIS exists. Therefore, the algorithm does not prematurely stop at step 2.

By lemma 7.3.2, as long as the Q_{Output} set does not yet contain an IIS, the Filtering LP will always relax at least one previously non-relaxed elastic constraint which corresponds to a constraint which is a member of an IIS; the corresponding

non-elastic constraint will then be added to the Q_{Output} set. Thus at each iteration of the algorithm the Q_{Output} set is increased by at least one constraint.

Therefore, as each IIS is composed of a finite number of constraints the Squeeze Filtering algorithm must eventually contain at least one IIS in the Q_{Output} set, which will then terminate the algorithm as no feasible solution will exist for the Filtering LP. Q.E.D.

VII.4. Time Complexity of the Squeeze Filtering Algorithm

Let ζ be the minimum of the cardinalities of all of the IISs in the original LP. The cardinality of an IIS is given by the number of constraints in the IIS.

Theorem 7.4.1 : The time complexity of the Squeeze Filtering algorithm is at worst $(\zeta + 1) * \text{Time Complexity of the Filtering LP}$.

Proof:

As the minimum cardinality is ζ , and as at each iteration at least one constraint per IIS is identified, at worst ζ iterations of the algorithm take place which add a constraint to the set Q_{Output} . There remains only one more solution of the Filtering LP to determine that at least one IIS is present in the set Q_{Output} . Thus at worst a total of $(\zeta + 1)$ Filtering LP solutions are needed. As each iteration requires the solution of an LP, the algorithm will have a time complexity of $(\zeta + 1) * \text{Time Complexity of the Filtering LP}$. Q.E.D.

VII.5. Example Execution of the Squeeze Filtering Algorithm

Consider the following original constraints (as shown in figure 4) :

$$q_1 : x_1 \leq 2$$

$$q_2 : x_2 \leq 2$$

$$q_3 : x_1 + x_2 \leq 7$$

$$q_4 : x_1 + x_2 \geq 9$$

$$q_5 : 3x_1 + x_2 \geq 15$$

$$q_6 : x_1 + 2x_2 \leq 22$$

$$q_7 : x_1 \leq 7$$

Converting all original constraints to elastic constraints :

$$e_1 : x_1 - v_1 \leq 2$$

$$e_2 : x_2 - v_2 \leq 2$$

$$e_3 : x_1 + x_2 - v_3 \leq 7$$

$$e_4 : x_1 + x_2 + v_4 \geq 9$$

$$e_5 : 3x_1 + x_2 + v_5 \geq 15$$

$$e_6 : x_1 + 2x_2 - v_7 \leq 22$$

$$e_7 : x_1 - v_6 \leq 7$$

Step 1 : Initialize the sets of constraints and variables

$$Q_{\text{Output}} = Q = R = \emptyset$$

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$$

$$\Rightarrow V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$

Step 2 : Solve the initial Filtering LP :

Chapter IX Sensitivity Filtering

IX.1. Introduction

Let us examine Roodman's method which applies sensitivity analysis to determine the modification of the right hand side of the LP to produce a feasible solution for the infeasible LP

An appropriate adjustment to the right hand side of a constraint is equivalent to relaxing the elastic version of that constraint. Thus a least cost modification will not change the right hand side of a non-IIS constraint, as the relaxation of a non-IIS constraint will not affect the infeasibility of the LP. This means that the method can perhaps be modified to identify IIS constraints. However, the constraints identified will in general not include at least one IIS, as to reduce the inconsistency of an IIS does not require the relaxation of all of the constraints of the IIS.

An extension of this method will lead to a way of identifying a set of constraints which seems to contain at least one IIS from each cluster of IISs.

Consider a single IIS in an LP. Consider the solution of the initial Filtering LP in the Squeeze Filtering algorithm. An optimal solution is generated in which only IIS constraints are relaxed. The Squeeze Filtering algorithm continues by solving further versions of the Filtering LP to identify the remaining IIS constraints. But perhaps there is a better, less expensive way to identify the remaining IIS constraints that does not involve the relatively expensive task of solving LPs.

$$\min Z = v_2 + v_3 + v_5 + v_6 + v_7$$

s.t.

$$x_1 \leq 2$$

$$x_2 - v_2 \leq 2$$

$$x_1 + x_2 - v_3 \leq 7$$

$$x_1 + x_2 \geq 9$$

$$3x_1 + x_2 + v_5 \geq 15$$

$$x_1 + 2x_2 - v_6 \leq 22$$

$$x_1 - v_7 \leq 7$$

$$\forall_i \forall_j \quad x_i, v_j \geq 0$$

This results in $Z = 9$, $v_2 = 5$, $v_3 = 2$, $v_5 = 2$, $\Rightarrow R = \{e_2, e_3, e_5\}$,

$$Q_{\text{Output}} = \{q_1, q_4\} \cup \{q_2, q_3, q_5\} = \{q_1, q_2, q_3, q_4, q_5\}.$$

Step 3' : Convert the relaxed constraints to non-elastic constraints :

$$E = \{e_2, e_3, e_5, e_6, e_7\} \setminus \{e_2, e_3, e_4\} = \{e_6, e_7\}$$

$$Q = \{q_1, q_2, q_3, q_4, q_5\}$$

$$\Rightarrow V = \{v_6, v_7\}$$

Step 4' : Solve the Filtering LP :

$$\min Z = v_6 + v_7$$

$$\begin{aligned}
& s.t. \\
& x_1 \leq 2 \\
& x_2 \leq 2 \\
& x_1 + x_2 \leq 7 \\
& x_1 + x_2 \geq 9 \\
& 3x_1 + x_2 \geq 15 \\
& x_1 + 2x_2 - v_6 \leq 22 \\
& x_1 - v_7 \leq 7 \\
& \forall_i \forall_j \quad x_i, v_j \geq 0
\end{aligned}$$

This results in an infeasible Filtering LP. STOP, $Q_{\text{Output}} = \{q_1, q_2, q_3, q_4, q_5\}$. The set Q_{Output} contain at least one IIS, and all elements of the Q_{Output} set are members of at least one IIS.

VII.6. Discussion

The Squeeze Filtering algorithm removes all of the non-IIS constraints, however, it also may remove some constraints which are elements of an IIS. This output set may contain remnant constraints as well as other IISs.

It is further limited by the fact that it requires extra space for the elastic variables; although the added space is minimal if one uses a sparse matrix format.

Chapter VIII Deletion/Squeeze Filtering for Medium - Large LPs

VIII.1. Introduction

The Deletion Filtering algorithm is not suited for large LPs due to the number of constraints that must be tested. A better algorithm incorporates both Deletion Filtering and Squeeze Filtering. As the Squeeze Filtering algorithm will always keep at least one IIS in its output set, it can be applied to the set of constraints before the application of the Deletion Filtering algorithm and still keep the overall algorithm valid.

As the Squeeze Filtering algorithm removes all of the non-IIS constraints, it makes sense to first Squeeze Filter to remove all of the non-IIS constraints. This will reduce the number of constraints that Deletion Filtering has to test.

In this manner, the two algorithms can be combined to increase the range of LP sizes which can be filtered.

VIII.2. The Deletion/Squeeze Filtering Algorithm

Definitions:

- a. Let Q_{Input} be the initial set of non-elastic constraints.
- b. Let Q_{Output} be the final set of constraints.
1. Squeeze Filter the set Q_{Input} to produce a (perhaps smaller) set Q' .
2. Deletion Filter the set Q' to produce the set Q_{Output}

Q_{Output} is an IIS.

VIII.3. Time Complexity of the Deletion/Squeeze Algorithm

Let μ be equal to the number of constraints which survive the first application of the Squeeze Filtering algorithm. Therefore, $\mu \leq |m - (\text{number of non-IIS constraints})|$.

Theorem 8.3.1 : The time complexity of the Deletion/Squeeze Filtering algorithm is at worst $(\zeta + 1) * \text{Time Complexity of the Filtering LP} + (\mu * \text{Time Complexity of a Phase I LP})$.

Proof:

As the two steps are carried out sequentially, the result trivially follows. Q.E.D.

VIII.4. Discussion

The Deletion/Squeeze Filtering algorithm identifies a single IIS. The initial Squeeze Filtering will remove all of the non-IIS constraints. As in general the number of non-IIS constraints will be substantial, this algorithm can be used for medium - large LPs.

As before, the initial Squeeze Filtering algorithm has extra space requirements, but as most large LP solvers use sparse matrix techniques, the additional space required for the extra elastic variables is minimal.

Chapter IX Sensitivity Filtering

IX.1. Introduction

Let us examine Roodman's method which applies sensitivity analysis to determine the modification of the right hand side of the LP to produce a feasible solution for the infeasible LP

An appropriate adjustment to the right hand side of a constraint is equivalent to relaxing the elastic version of that constraint. Thus a least cost modification will not change the right hand side of a non-IIS constraint, as the relaxation of a non-IIS constraint will not affect the infeasibility of the LP. This means that the method can perhaps be modified to identify IIS constraints. However, the constraints identified will in general not include at least one IIS, as to reduce the inconsistency of an IIS does not require the relaxation of all of the constraints of the IIS.

An extension of this method will lead to a way of identifying a set of constraints which seems to contain at least one IIS from each cluster of IISs.

Consider a single IIS in an LP. Consider the solution of the initial Filtering LP in the Squeeze Filtering algorithm. An optimal solution is generated in which only IIS constraints are relaxed. The Squeeze Filtering algorithm continues by solving further versions of the Filtering LP to identify the remaining IIS constraints. But perhaps there is a better, less expensive way to identify the remaining IIS constraints that does not involve the relatively expensive task of solving LPs.

The cost of eliminating the interior region is just the value of the optimal objective function value of the Filtering LP. Adjusting the right hand side of a non-IIS constraint so as to relax this constraint will not affect the optimal objective function value. This is due to the fact that a non-IIS constraint is redundant to the interior region of the IIS. Conjecture 5.4.3 states that relaxing an IIS constraint in the direction of the interior region of the IIS will decrease the net cost of removing the inconsistency associated with the IIS. That is, relaxing an IIS constraint in the direction of the interior region of the IIS will reduce the value of the optimal objective function.

To relax a $\leq$ constraint one adds an amount ϵ to the right hand side. To relax a $\geq$ constraint one subtracts an amount ϵ from the right hand side. To relax an equality constraint one has to carry out two separate adjustments, as the equality constraint can be relaxed in two directions. Thus one adds an amount ϵ to the right hand side and then, in a separate operation, one subtracts an amount ϵ from the right hand side.

So, what is needed is an efficient method for determining the effect on the objective function of changing the right hand side of a single constraint.

The first idea that comes to mind is to use shadow prices to tell us how the optimal objective function value changes if the right hand side of a constraint is changed (shadow prices are the dual variable values in the final tableau indexed by the initial basic variables). These shadow prices give information on how the objective function is changed if the right hand side of the constraint is increased. However, these shadow prices will have nonpositive values (if one minimizes the objective function) and thus will only indicate that the objective function value

has decreased if one has added an amount ϵ to the right hand side. It will never indicate that the objective function value has decreased due to the subtraction of an amount ϵ from the right hand side. Thus shadow prices can not be used to detect $\geq$ constraints (as well as some equality constraints).

However, the effect on the objective function of adjusting the right hand side can be obtained by the use of sensitivity analysis (see section II.1.2). Sensitivity analysis involves the multiplication of a vector, c_{bv} , by a matrix, B^{-1} and then subsequent multiplication by the adjusted right hand side vector, b^{**} . The vector-matrix multiplication is required only once as it is a constant quantity for a given optimal solution. The additional multiplication by the adjusted right hand side vector must be repeated for each constraint to be tested.

A simple check can be used to determine the validity of this updating effect on the optimal solution. If the right hand side of the optimal solution is strictly greater than zero, then the updated solution is feasible and optimal. In fact, by checking the sign of the i th component of the vector $p = c_{bv} \times B^{-1}$ and the sign of ϵ , one can determine the effect (increase or decrease) on the objective function value (see section II.1.2). As equality constraints can be right hand side adjusted in both directions, a decrease in the objective function value will occur if the corresponding component of p is nonzero.

If the updating effect is not valid, then the new solution will be infeasible. In this case, the new optimal solution can be found by applying the Dual Simplex method, beginning from the primal infeasible solution. The number of iterations of the Dual Simplex should be small, as the new solution should be close to the previous solution, for a small change in the right hand side. If the extra

Dual Simplex solution is deemed too expensive, the option remains of drawing no further conclusions about a constraint which results in an infeasible updated solution. In the general case, the adjustment of the right hand side will result in an updated solution which is feasible and optimal. Thus Sensitivity Filtering will efficiently identify IIS constraints.

However, can this be extended for an LP with a cluster of IISs? Each cluster of IISs contains at least one extending IIS. The interior region of this extending IIS is such that it is not contained within any other IISs. Thus part of the total cost of eliminating all of the interior regions of all of the IISs of the cluster is made up of the cost of removing the interior region of the extending IIS. Conjecture 5.4.3 states that by adjusting the right hand side of any constraint of an IIS, the net cost of removing the IIS is reduced. Thus, adjusting the right hand side of the constraints of the extending IIS into the interior region of the IIS will reduce the cost of removing the corresponding interior region. Thus the net cost of relaxing all of the interior regions of the cluster of IISs is also reduced. Therefore all of the constraints of the extending IIS will be identified by the combination of a Filtering LP solution (to generate the optimal solution) and the subsequent use of sensitivity analysis to determine the effect of changing the right hand sides of the constraints on the optimal objective function value.

For the case of multiple clusters of IISs, the independent nature of the clusters ensures that the set generated will include at least one IIS from each of the clusters.

This method will not identify nonnegativity constraints which are elements of an IIS due to the use of solution methods for the LP which imply nonnegativity.

As the validity of the Sensitivity Filtering algorithm relies on the validity of

Conjecture 5.4.3, this algorithm can stand alone if Conjecture 5.4.3 is shown to be valid. Thus caution must be exercised in its use. If the conjecture is unproven or proven false, then one must develop some way of verifying that at least one IIS has been identified. One such approach is the Squeeze/Sensitivity Filtering algorithm developed in Chapter X.

The net effect of using Sensitivity Filtering instead of Squeeze Filtering is that only one Filtering LP needs to be solved. The only other additional cost is due to the application of sensitivity analysis. In most cases this will involve vector by vector multiplications or the occasional solution of a Dual Simplex LP. As the cost of solving a Filtering LP is greater than that of the sensitivity analysis, the Sensitivity Filtering algorithm should be more efficient.

The final output set of Sensitivity Filtering will contain at least one IIS. In addition, all of the constraints in the output set will be IIS constraints.

IX.2. The Sensitivity Filtering Algorithm

Definitions :

- i. Let c_{bv} be the columns of the initial cost vector c which are indexed by the basic variables of the optimal solution of the filtering LP.
- ii. Let B^{-1} be the inverse of the basis matrix for the optimal solution of the Filtering LP.
- iii. Let b^* be the initial right hand side vector.
- iv. Let b' be the final right hand side vector.
- v. Let Q_{Input} be the initial set of non-elastic constraints.

- vi. Let Q_{Output} be the final set of constraints.
 - vii. Let R be the set of relaxed elastic constraints associated with the solution of a Filtering LP.
 - viii. Let E be the set of elastic constraints, e_i , that are in the constraint set of the Filtering LP.
 - ix. Let State be a 'flag' variable indicating the status of the update solution which can be in one of the two following states: optimal or unknown.
 - x. Let $V = \text{vin}(E)$, be the set of elastic variables with non-zero coefficients in the elastic constraint set E .
1. Initialize the sets of constraints and variables:
 - a. Set $Q_{\text{Output}} = R = \emptyset$.
 - b. Set $E = \text{elastic}(Q_{\text{Input}})$.
 - c. Set $V = \text{vin}(E)$.
 2. Solve the initial Filtering LP to detect if the original LP is feasible :
Solve

$$\min Z = \sum_{v_i \in V} v_i$$

such that : E, Nonnegativity

If $Z = 0$ THEN STOP, the original LP is feasible

ELSE This results in the set R , set $Q_{\text{Output}} = Q_{\text{Output}} \cup \text{elastic}^{-1}(R)$, continue

3. Calculate $c_{bv} \cdot \times B^{-1} = p$, a vector, from information from the Filtering LP formulation and optimal solution.

4. Read from the optimal solution of the Filtering LP the value of the right hand side, b' .

If $b' > 0$

THEN State = optimal

ELSE State = unknown

For each $e_i \in E \setminus R$ do :

a.

CASE 1: e_i is a $\leq$ constraint

IF State = optimal

THEN

If $p_i < 0$

Then ($Z^{**} < Z$) go to step 4.b

Else go to step 4.c

ELSE (State = unknown)

- add ϵ to b_i^* to get b^{**}

- If $B^{-1} \times b^{**} \geq 0$

Then {update is optimal} $Z^{**} = p \times b^{**}$

Else {update is infeasible} apply the Dual Simplex method to determine the new value of Z^{**}

- If $Z^{**} < Z$

Then go to step 4.b

Else go to step 4.c

CASE 2: e_i is a $\geq$ constraint

IF State = optimal

THEN

 If $p_i > 0$

 Then ($Z^{**} < Z$) go to step 4.b

 Else go to step 4.c

ELSE (State = unknown)

- subtract ϵ from b_i^* to get b^{**}

- If $B^{-1} \times b^{**} \geq 0$

 Then {update is optimal} $Z^{**} = p \times b^{**}$

 Else {update is infeasible} apply the Dual Simplex method to determine the new value of Z^{**}

- If $Z^{**} < Z$

 Then go to step 4.b

 Else go to step 4.c

CASE 3: e_i is an $=$ constraint

IF State = optimal

THEN

 If $p_i \neq 0$

 Then ($Z^{**} < Z$) go to step 4.b

 Else go to step 4.c

ELSE (State = unknown)

- add ϵ to b_1^* to get b^{**}

- If $B^{-1} \times b^{**} \geq 0$

THEN {update is optimal} $Z^{**} = p \times b^{**}$

ELSE {update is infeasible} apply the Dual Simplex method to determine the new value of Z^{**}

- If $Z^{**} < Z$

Then go to step 4.b

Else

— subtract ϵ from b_1^* to get b^{**}

— If $B^{-1} \times b^{**} \geq 0$

Then {update is optimal} $Z^{**} = p \times b^{**}$

Else {update is infeasible} apply the Dual Simplex method to determine the new value of Z^{**}

— If $Z^{**} < Z$

Then go to step 4.b

Else go to step 4.c

- b. Add the constraint to the set of identified IIS constraints

$$Q_{\text{Output}} = Q_{\text{Output}} \cup \{\text{elastic}^{-1}(e_i)\}$$

- c. Continue with the next e_i

STOP

The set Q_{Output} contains at least one IIS.

IX.3. Validity of the Sensitivity Filtering Algorithm

Conjecture 9.3.1 : The Sensitivity Filtering algorithm will identify a set of constraints that contains at least one IIS if and only if at least one IIS exists.

Note : This conjecture relies only on Conjecture 5.4.3 being true.

Proof:

$\Rightarrow$

If no IIS exists then step 2 of the algorithm will result in a Z value of zero. This will then terminate the algorithm and indicate that no IIS has been found.

$\Leftarrow$

If an IIS exists then at least one IIS constraint per IIS must be relaxed in step 2 of the algorithm.

Let us examine the case of a single IIS in the LP. Let R be the set of IIS constraints relaxed in step 2 of the algorithm. By Conjecture 5.4.3, the net cost of relaxing the set R is decreased if an IIS constraint is right hand side adjusted in the direction of the interior region of the IIS.

The net cost of relaxing the set R stays the same if the right hand side of a non-IIS constraint is adjusted in the direction of the interior region of the IIS. The non-IIS constraint is redundant and thus does not affect the interior regions of the IISs. Thus the algorithm will identify each of the IIS constraints which are not elements of R in step 4 of the algorithm. Thus all of the IIS constraints of the IIS are identified.

Let us now examine the case of a single multiple IIS cluster of IISs. Let R be the set of IIS constraints relaxed in step 2. Each of the constraints of set R is relaxed the minimum amount necessary to eliminate all of the interior regions of the IIS. When overlapped IISs are present, the relaxation of a single IIS constraint may eliminate more the interior regions of more than one IIS. For example, relaxing the constraint a in figure 2 (b) can eliminate both of the interior regions of the two IISs. This occurs because one IIS's interior region is contained within another IIS. Thus right hand side adjusting an IIS constraint of the enclosed IIS will in general not decrease the cost of relaxing the set R . However, right hand side adjusting an IIS constraint from an extending IIS will always reduce the net cost of relaxing the set R as the interior region of an extending IIS is not contained within any other IIS. Thus for a single cluster of IISs, the algorithm will generate a set of constraints which includes at least one IIS.

If there is more than one cluster of IISs, then as each cluster is independent of the other clusters, the algorithm will identify at least one IIS per cluster.

Therefore, the Sensitivity Filtering algorithm will identify a set of constraints that contains at least one IIS if and only if at least one IIS exists, under the condition that Conjecture 5.4.3 is valid.

IX.4. Time Complexity of the Sensitivity Filtering Algorithm

Theorem 9.4.1 : The time complexity of the Sensitivity Filtering algorithm is at worst the Time Complexity of the Filtering LP + $[O(\text{number of constraints}) * \text{Time Complexity of the Dual Simplex Method}] + [\text{Cost of a vector by matrix multiplication}]$.

Proof:

The Filtering LP produces a set R of relaxed constraints. One vector by matrix multiplication is carried out in step 3. For step 4, at worst all but one constraint must be checked by sensitivity analysis. Each constraint need only be checked once with the exception of equality constraints which may have to be checked twice. Thus $O[\text{number of constraints}]$ sensitivity analysis checks are made. Each sensitivity analysis check involves either the multiplication of two vectors or the solution of a Dual Simplex Method. As the Time Complexity of the Dual Simplex Method $\gg$ Cost of a vector by vector multiplication, this results in a total cost for the sensitivity analysis of $O[\text{number of constraints}] * \text{Time Complexity of the Dual Simplex Method}$. As the sensitivity analysis occurs after the Filtering LP solution, the result follows trivially. Q.E.D.

IX.5. Example Execution of the Sensitivity Filtering Algorithm

Consider the following constraints (as shown in figure 4) :

$$q_1 : x_1 \leq 2$$

$$q_2 : x_2 \leq 2$$

$$q_3 : x_1 + x_2 \leq 7$$

$$q_4 : x_1 + x_2 \geq 9$$

$$q_5 : 3x_1 + x_2 \geq 15$$

$$q_6 : x_1 + 2x_2 \leq 22$$

$$q_7 : x_1 \leq 7$$

Converting these to elastic constraints :

$$e_1 : x_1 - v_1 \leq 2$$

$$e_2 : x_2 - v_2 \leq 2$$

$$e_3 : x_1 + x_2 - v_3 \leq 7$$

$$e_4 : x_1 + x_2 + v_4 \geq 9$$

$$e_5 : 3x_1 + x_2 + v_5 \geq 15$$

$$e_6 : x_1 + 2x_2 - v_7 \leq 22$$

$$e_7 : x_1 - v_6 \leq 7$$

Step 1 : Initialize the sets of constraints and variables

$$Q_{\text{Output}} = R = \emptyset$$

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$$

$$\Rightarrow V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$

Step 2 : Solve the initial Filtering LP :

$$\min Z = v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7$$

s.t.

$$x_1 - v_1 \leq 2$$

$$x_2 - v_2 \leq 2$$

$$x_1 + x_2 - v_3 \leq 7$$

$$x_1 + x_2 + v_4 \geq 9$$

$$3x_1 + x_2 + v_5 \geq 15$$

$$x_1 + 2x_2 - v_6 \leq 22$$

$$x_1 - v_7 \leq 7$$

$$\forall_i \forall_j \quad x_i, v_j \geq 0$$

This results in $Z = 5$, $v_1 = 7/3$, $v_4 = 8/3 \Rightarrow R = \{e_1, e_4\}$, $Q_{\text{Output}} = \{q_1, q_4\}$.

From the Filtering LP solution the value of c_{bv^*} is $(0,0,0,1,1,0,0)$.

The value of B^{-1} is :

0	-1/3	0	0	1/3	0	0
0	1	0	0	0	0	0
0	-2/3	1	0	-1/3	0	0
0	-2/3	0	1	-1/3	0	0
-1	-1/3	0	0	1/3	0	0
0	-5/3	0	0	-1/3	1	0
0	1/3	0	0	-1/3	0	1

$$\text{thus } p = c_{bv^*} \times B^{-1} = (-1, -1, 0, 1, 0, 0, 0).$$

$$\text{Step 4 : } b' = B^{-1} \times b^* = (13/3, 2, 2/3, 8/3, 7/3, 41/3, 8/3), \text{ Thus } b' > 0,$$

State = optimal

For each $e_i \in \{e_2, e_3, e_5, e_6, e_7\}$ Do :

for e_2 :

$$\square \quad x_2 - v_2 \leq 2$$

$$\square \quad e_2 \text{ has } \leq$$

- ☐ $p_2 < 0$
- ☐ Thus $Z^{**} < Z$
- ☐ $Q_{\text{Output}} = \{q_1, q_4\} \cup \{\text{elastic}^{-1}(e_2) = \{q_1, q_2, q_4\}$
- ☐ get next e_i

for e_3 :

- ☐ $x_1 + x_2 - v_3 \leq 7$
- ☐ e_3 has $\leq$
- ☐ $p_3 = 0$
- ☐ get next e_i

for e_5 :

- ☐ $3x_1 + x_2 + v_5 \geq 15$
- ☐ e_5 has $\geq$
- ☐ $p_5 = 0$
- ☐ get next e_i

for e_6 :

- ☐ $x_1 + 2x_2 - v_7 \leq 22$
- ☐ e_6 has $\leq$
- ☐ $p_6 = 0$
- ☐ get next e_i

for e_7 :

- ☐ $x_1 - v_6 \leq 7 + \epsilon$

- ☐ e_7 has $\leq$
- ☐ $p_7 = 0$
- ☐ get next e_i

Thus $Q_{\text{Output}} = \{q_1, q_2, q_4\}$, which contains the IIS $\{q_1, q_2, q_4\}$

IX.6. Discussion

At present the validity of the Sensitivity Filtering algorithm depends on the validity of Conjecture 5.4.3.

If this conjecture is invalid, then the algorithm can still be used as an heuristic inside the Squeeze Filtering LP. One would alternate a sensitivity analysis with a Filtering LP solution to detect other IIS constraints at virtually no cost, until the Filtering LP indicates that an IIS has been identified. As the sensitivity analysis makes direct use of the information from the Filtering LP solution, the additional cost of carrying out sensitivity analysis between Filtering LPs is small. In Chapter XII three example LPs are tested using this type of approach. In all three examples (as well as others used in developing this method), the validity of the Sensitivity Filtering algorithm was supported by the fact that only two Filtering LPs were needed, the second always indicating that at least one IIS had been identified.

If Conjecture 5.4.3 is valid, then this method will be more efficient than that of Squeeze Filtering as the cost of 1 Filtering LP will be in general much more than the cost of the sensitivity analysis.

The additional space requirements of this algorithm are mainly due to the elastic variables for the Filtering LP solution. The additional space for sensitivity analysis is minimal.

In either case this algorithm is useful due to its low cost of execution. For a more complete discussion on its use within the Deletion/Squeeze method see Chapter X.

Chapter X Deletion/Squeeze/Sensitivity Filtering for Medium - Large LPs

X.1. Introduction

Upon examining the Deletion/Squeeze algorithm it is noted that the major cost is due to the number of Filtering LPs that are used. An attempt to decrease this cost is made by the addition of Sensitivity Filtering into the Squeeze Filtering portion of the Deletion/Squeeze Filtering algorithm. As the Sensitivity Filtering algorithm requires a Filtering LP solution and as the Squeeze Filtering algorithm provides one at each step, it makes sense to apply Sensitivity Filtering to each of these Filtering LPs. The additional cost is minimal and the benefits are that more IIS constraints per Filtering LP solution will be identified.

If Sensitivity Filtering is shown to be valid, then due to its lower cost one replaces the Squeeze Filtering algorithm in step 2 of the Deletion/Squeeze Filtering algorithm by a Sensitivity Filtering algorithm.

If Sensitivity Filtering is not valid, then it can still be used in conjunction with a method which ensures that at least one IIS has been identified. To test for an IIS is quite easy: simply formulate a second Filtering LP from the Q_{output} set of the Sensitivity Filtering. If the Filtering LP solution indicates that an IIS has not been identified, apply sensitivity analysis to the new Filtering LP solution. The process repeats until a Filtering LP indicates that at least one IIS has been identified. This process of alternating Filtering LP solutions with Sensitivity Filtering is labelled Squeeze/Sensitivity Filtering. Thus Squeeze/Sensitivity Filtering can be

used to replace the Squeeze Filtering algorithm in step 2 of the Deletion/Squeeze Filtering algorithm.

The validity of these algorithms is assured as either the Sensitivity Filtering algorithm or the Squeeze/Sensitivity algorithm will always identify a set of constraints which includes at least one IIS. In addition, all of the constraints which are identified are IIS constraints. Thus at least one IIS will always be present for the Deletion Filtering algorithm to identify.

X.2. The Squeeze/Sensitivity Filtering Algorithm

To be used if Sensitivity Filtering is invalid or if the validity of Sensitivity Filtering is unsure.

Definitions:

- a. Let Q_{Input} be the initial set of non-elastic constraints.
- b. Let Q_{Output} be the final set of constraints.
- c. Let E be the set of elastic constraints, e_i , that are in the constraint set of the Filtering LP.
- d. Let $V = \text{vin}(E)$, be the set of elastic variables with non-zero coefficients in the elastic constraint set E .
- e. Let R be the set of relaxed elastic constraints associated with the solution of a Filtering LP.
1. Initialize the sets of constraints and variables:
 - a. Set $Q_{\text{Output}} = R = \emptyset$.

- b. Set $E = \text{elastic}(Q_{\text{Input}})$.
 - c. Set $V = \text{vin}(E)$.
2. Solve the initial Filtering LP to detect if the original LP is feasible :
- Solve

$$\min Z = \sum_{v_i \in V} v_i$$

such that : E, Nonnegativity

If $Z = 0$ THEN STOP, the original LP is feasible

ELSE This results in the set R, set $Q_{\text{Output}} = Q_{\text{Output}} \cup \text{elastic}^{-1}(R)$,

$E = E \setminus R$, continue

3. Calculate $c_{bv} \times B^{-1} = p$, a vector, from information from the Filtering LP formulation and optimal solution.
4. For each $e_i \in E \setminus R$ use sensitivity analysis to determine if the value of the optimal objective function can be reduced (This is step 4 of the Sensitivity Filtering algorithm). If so set $Q_{\text{Output}} = Q_{\text{Output}} \cup \{\text{elastic}^{-1}(e_i)\}$, $E = E \setminus \{e_i\}$.
5. Reset the Filtering LP sets :
- a. Set $R = \emptyset$.
 - b. Set $V = \text{vin}(E)$.

Solve the Filtering LP.

$$\min Z = \sum_{v_i \in V} v_i$$

such that : Q_{Output} , E, Nonnegativity

If Filtering LP is infeasible, THEN STOP

ELSE This results in the set R , $Q_{\text{Output}} = Q_{\text{Output}} \cup \text{elastic}^{-1}(R)$,

$E = E \setminus R$, go to step 3.

X.3. Time Complexity of the Squeeze/Sensitivity Filtering Algorithm

Let ζ be the cardinality of the smallest IIS in the set Q_{Input}

Theorem 10.3.1 : The time complexity of the Squeeze/Sensitivity Filtering algorithm is at worst $[(\zeta + 1) * (\text{Time Complexity of the Filtering LP})] + [\zeta * O[\text{number of constraints}] * \text{Time Complexity of the Dual Simplex Method}] + [\zeta * (\text{Cost of a vector by matrix multiplication})]$

Proof:

Without the Sensitivity Filtering section of steps 3 and 4 the algorithm has the complexity of the Squeeze Filtering algorithm, $[(\zeta + 1) * (\text{Time Complexity of the Filtering LP})]$. As the Sensitivity Filtering section can only add constraints to the set Q_{Output} , at worst none are added and $(\zeta + 1)$ Filtering LPs are solved.

For $(\zeta + 1)$ Filtering LP solutions there are ζ iterations of steps 3 and 4. Each iteration of step 4 will sensitivity analysis check at most all of the constraints twice. Thus $O[\text{number of constraints}]$ checks per iteration of step 4 are required. Each sensitivity analysis check of a constraint involves either the multiplication of a vector by a vector or the solution of a Dual Simplex Method. Thus $\zeta * O[\text{number of constraints}] * [\text{Time Complexity of the Dual Simplex Method}]$ is the complexity due to step 4.

Each iteration of step 3 involves a vector by matrix multiplication. Thus $\zeta * (\text{Cost of a vector by matrix multiplication})$ is the total cost of step 4.

Thus the total time complexity is as stated. Q.E.D.

X.4. The Deletion/Squeeze/Sensitivity Filtering Algorithm

Definitions:

- a. Let Q_{Input} be the initial set of non-elastic constraints.
- b. Let Q_{Output} be the final set of constraints.
1. Squeeze/Sensitivity Filter the set Q_{Input} to produce a (perhaps smaller) set Q' .
2. Deletion Filter the set Q' to produce the set Q_{Output} .

The set Q_{Output} is an IIS.

X.5. Time Complexity of the Deletion/Squeeze/Sensitivity Filtering Algorithm

Let m be the number of constraints in the set Q' . $Q' \leq Q_{\text{Input}}$.

Let ζ be the minimum of the cardinalities of all of the IISs in the original LP.

Theorem 10.5.1 : The time complexity of the Deletion/Squeeze/Sensitivity Filtering algorithm is at worst $[(\zeta + 1) * (\text{Time Complexity of the Filtering LP})] + [\zeta * O(\text{number of constraints}) * \text{Time Complexity of the Dual Simplex Method}] + [\zeta * (\text{Cost of a vector by matrix multiplication})] + [O(m) * (\text{Time Complexity of a Phase I LP})]$.

Proof:

At most all of the constraints survive the Squeeze/Sensitivity Filtering algorithm. Thus $O[m]$ Phase I LPs are solved by the Deletion Filtering algorithm. As the two parts, Squeeze/Sensitivity Filtering and subsequent Deletion Filtering are carried out in a sequential order, the result follows trivially. Q.E.D.

X.6. Discussion

In general, the addition of Sensitivity Filtering into the Deletion/Squeeze Filtering algorithm will improve the computation time. This is due to the fact that Sensitivity Filtering is inexpensive when compared to Squeeze Filtering. The savings will be even greater if Sensitivity Filtering is shown to be valid, because only one Filtering LP is needed.

In comparison to Deletion Filtering, Deletion/Squeeze/Sensitivity Filtering will in general have lower computation times if the IISs are small relative to the size of the LP. This condition allows the Squeeze/Sensitivity Filtering stage to remove a large number of constraints. The trade off is in space requirements as the Deletion/Squeeze/Sensitivity Filtering algorithm requires additional space for the elastic variables.

Thus the algorithm will identify a single IIS. In addition, if one compares the set of constraints output from the Squeeze/Sensitivity Filtering stage to that of the final IIS, any difference indicates that additional IISs are present in the LP. Those constraints from the Squeeze Sensitivity Filtering stage which have been eliminated by the Deletion Filtering stage are of course members of these additional IISs. This information can be passed on to the modeller. It is noted that if no difference is found, this does not exclude the presence of additional IISs.

Chapter XI Efficiency Considerations

There are several techniques that could possibly be applied to the Deletion, Squeeze, or Sensitivity Filtering algorithms to improve run time efficiency. Some of these are discussed below.

XI.1. Sparse Matrix Representation

As the size of the LP grows, problems arise due to the size of the coefficient matrix. However, in most cases this matrix will be relatively sparse and as such representing the matrix in a sparse matrix format will allow one to solve large LPs that could not be solved in full matrix representation. The trade off is in time efficiency. The introduction of sparse matrix representation will add some overhead to the matrix operations that are carried out in the solution of the LP.

If a sparse matrix representation is being used, then the additional space requirements for the addition of elastic variables is of the order of the addition of one to two full columns. This is due to the fact that each constraint requires either one or two elastic variables. This is less than the full matrix representation which would require the addition of columns equal to the number of elastic variables. The same type of savings can occur for van Loon's and Gleeson and Ryan's algorithms when the addition of nonnegativity constraints and compensating constraints for the equality constraints are added to the LP.

However, the savings are only in space complexity and not in time complexity as the sparse matrix representation does not decrease the number of rows in the matrix, only the number of elements that must be stored.

XI.2. Scaling the Initial Constraint Set

In the original set of constraints, each constraint may have been created somewhat independently from the other constraints. Thus the relative magnitudes of the coefficients between some constraints may differ by several orders of magnitudes. Such large differences may cause numerical errors. A simple method to deal with this problem is to scale the constraints (or rows) to some convenient common level.

This procedure will serve two purposes: a) to reduce the effect of these types of errors and b) to identify the pathological cases that can not be eliminated by a simple scaling procedure.

XI.3. Removing Remnant Constraints within Deletion Filtering

As stated in section VI.6, an algorithm that prunes away non-IIS constraints could be added either before the Deletion Filtering algorithm or within the Deletion Filtering algorithm. In Chapters VIII and X the option of the addition of this pruning algorithm before the Deletion Filtering algorithm was considered. Let us now consider the other option.

Consider an iteration of the Deletion Filtering algorithm which removes a constraint from the working set, Q . In this operation an IIS may be removed. This may cause some of the remaining constraints which were members of the IIS to become remnant constraints. It may be advantageous to Squeeze/Sensitivity Filter (or Sensitivity Filter) the set of constraints before carrying out the next iteration of the Deletion Filtering algorithm.

As the Squeeze/Sensitivity Filtering algorithm always guarantees that the output set will contain at least one IIS, this will not affect the validity of the Deletion Filtering algorithm.

If the LP contains many large isolated IISs, then this additional Filtering could perhaps be cost effective, as the solution of a few extra LPs could remove a large number of constraints. The problem is that one does not know *a priori* the nature of the IISs. In general, if a large number of constraints are present in the working set of the Deletion Filtering algorithm, then this option may be cost effective, if not then it is likely to be ineffective. Thus the use of Squeeze/Sensitivity Filtering within the Deletion Filtering algorithm is only to be encouraged when there is still a large number of constraints to be tested.

If Sensitivity Filtering is valid, then due to the lower cost of Sensitivity Filtering the use of this option can be extended somewhat.

XI.4. Advanced Starts

For the Filtering LPs in Squeeze Filtering, after each optimal solution, a non-zero elastic variable is removed from the Filtering LP. This causes the optimal solution to be infeasible for the next Filtering LP. This points to the use of the Dual Simplex method to provide an advanced start for the next Filtering LP. Instead of removing the non-zero elastic variable, v_i , let us add in a constraint of the form $v_i + slack = 0$. As desired, the addition of this constraint will cause the previous optimal solution to be infeasible.

If one has an optimal solution, and the addition of a constraint makes this solution infeasible, then an initial dual solution can be constructed from the

previous optimal solution by adding a new column to the previous basis (see Nemhauser and Wolsey [21]). As the starting solution will likely be near primal feasible, the new optimal solution should be achieved in a small number of iterations.

Thus the addition of the constraint $v_i + slack = 0$ will allow us to create an initial dual solution to the new LP. The solution of the new LP will force the elastic variable, v_i , to zero as both v_i and $slack$ are nonnegative variables. After the new optimal solution is found, v_i is equal to zero. Therefore, the constraint $v_i + slack = 0$ as well as the elastic variable can be deleted from the LP. This gives a net result of changing an elastic constraint into a non-elastic constraint.

Some of the time saved using advanced starts will be lost due to the increased overhead that is required to keep track of the constraints and variables that are to be removed. For small LPs the increased amount of overhead will offset any real benefit, however savings may be very substantial for large LPs.

At present this technique works for the addition of a single constraint. The use of this method will partially rely on whether this method can be extended for the addition of more than one constraint. It will also depend on the validity of the Sensitivity Filtering algorithm. If Sensitivity Filtering is valid, then only one Filtering LP solution is needed in any case.

Chapter XII Examples

All of the examples in this chapter use as a base the constraints of the two plant - two period LP found in Appendix C.

Each example is tested by applying the following algorithms to the constraint set :

1. Deletion Filtering
2. Deletion/Squeeze Filtering
3. Deletion/Squeeze/Sensitivity Filtering

A summary of the results is given in Table No. 1 at the end of this section.

XII.1. Example No. 1 : a single IIS

Consider the addition of the following constraints to the set of constraints of Appendix C :

- Constraint₂ : $x_1 + x_{AP1} \geq 250,000$
- Constraint₃ : $x_1 - 2x_2 \leq 0$
- Constraint₄ : $x_2 - 5x_3 \leq 0$
- Constraint₅ : $x_3 - 5x_4 \leq 0$
- Constraint₆ : $x_4 - 10x_5 \leq 0$
- Constraint₇ : $x_5 \leq 50$

This results in the formation of a single IIS of {P_{SALEA1}, Constraint₂, Constraint₃, Constraint₄, Constraint₅, Constraint₆, Constraint₇}.

XII.1.1 Deletion Filtering

Application of the Deletion Filtering algorithm to the set of constraints resulted in the set $\{P_{SALEA1}, \text{Constraint}_2, \text{Constraint}_3, \text{Constraint}_4, \text{Constraint}_5, \text{Constraint}_6, \text{Constraint}_7\}$. This is the IIS.

The algorithm required 42 Phase I LP solutions, one for each constraint.

XII.1.2 Deletion/Squeeze Filtering

Application of the initial Squeeze Filtering algorithm resulted in the set $\{P_{SALEA1}, \text{Constraint}_2, \text{Constraint}_3, \text{Constraint}_4, \text{Constraint}_5, \text{Constraint}_6, \text{Constraint}_7\}$. The subsequent application of the Deletion Filtering algorithm to this set resulted in the set $\{P_{SALEA1}, \text{Constraint}_2, \text{Constraint}_3, \text{Constraint}_4, \text{Constraint}_5, \text{Constraint}_6, \text{Constraint}_7\}$. This is the IIS.

The algorithm required 8 Filtering LP solutions, plus 7 Phase I LP solutions. A total of 15 LP solutions.

XII.1.3 Deletion/Squeeze/Sensitivity Filtering

The initial Squeeze Filtering LP solution resulted in the set $\{\text{Constraint}_7\}$. The subsequent application of Sensitivity Filtering to the solution of the initial Squeeze LP resulted in the set $\{P_{SALEA1}, \text{Constraint}_2, \text{Constraint}_3, \text{Constraint}_4, \text{Constraint}_5, \text{Constraint}_6, \text{Constraint}_7\}$. Upon testing by a second application of the Squeeze Filtering LP this terminated the Squeeze/Sensitivity Phase of the algorithm. The subsequent application of the Deletion Filtering algorithm to this set resulted in the set $\{P_{SALEA1}, \text{Constraint}_2, \text{Constraint}_3, \text{Constraint}_4, \text{Constraint}_5, \text{Constraint}_6, \text{Constraint}_7\}$. This is the IIS.

The algorithm required 1 Filtering LP solutions, plus 41 sensitivity checks, plus 7 Phase I LP solutions, as well as one extra Filtering LP solution to verify that at least one IIS had been identified. A total of 9 LP solutions plus a minimal amount of computation due to sensitivity checks.

XII.2. Example No. 2 : a single cluster of IISs

Consider the addition of the following constraints to the set of constraints of Appendix C :

- Constraint₁ : $z_{WP1} \leq 90,000$
- Constraint₈ : $re_1 \geq 8,000$
- Constraint₉ : $v_1 - re_1 \geq 50,000$
- Constraint₁₀ : $v_2 - re_1 \geq 87,000$

This results in the formation of a single cluster of IISs of constraints {DEBT_{LIM}₁, INTREST₁, DEBT_{LIM}₂, Constraint₈, Constraint₉, Constraint₁₀} with 5 IISs :

1. {DEBT_{LIM}₁, INTREST₁, Constraint₈}
2. {DEBT_{LIM}₁, INTREST₁, Constraint₉}
3. {DEBT_{LIM}₁, Constraint₈, Constraint₉}
4. {INTREST₁, DEBT_{LIM}₂, Constraint₈, Constraint₁₀}
5. {DEBT_{LIM}₂, Constraint₉, Constraint₁₀}

XII.2.1 Deletion Filtering

Application of the Deletion Filtering algorithm to the set of constraints resulted in the set {DEBTLM₂, Constraint₉, Constraint₁₀}. This is one of the IISs.

The algorithm required 40 Phase I LP solutions.

XII.2.2 Deletion/Squeeze Filtering

Application of the initial Squeeze Filtering algorithm resulted in the set {DEBTLM₁, INTREST₁, DEBTLM₂, Constraint₈, Constraint₁₀}. The subsequent application of the Deletion Filtering algorithm to this set resulted in the set {INTREST₁, DEBTLM₂, Constraint₈, Constraint₁₀}. This is one of the IISs.

The algorithm required 3 Filtering LP solutions, plus 5 Phase I LP solutions. A total of 8 LP solutions.

XII.2.3 Deletion/Squeeze/Sensitivity Filtering

The initial Squeeze Filtering LP solution resulted in the set {INTREST₁, Constraint₈, Constraint₁₀}. The subsequent application of Sensitivity Filtering to the solution of the initial Squeeze LP resulted in the set {DEBTLM₁, INTREST₁, DEBTLM₂, Constraint₈, Constraint₉, Constraint₁₀}. Upon testing by a second application of the Squeeze Filtering LP this terminated the Squeeze/Sensitivity Phase of the algorithm. The subsequent application of the Deletion Filtering algorithm to this set resulted in the set {DEBTLM₂, Constraint₉, Constraint₁₀}. This is one of the IISs.

The algorithm required 1 Filtering LP solution, plus 37 sensitivity checks, plus 6 Phase I LP solutions, as well as 1 extra Filtering LP solution to verify that

at least one IIS had been identified. A total of 8 (7) LP solutions plus minimal sensitivity checks.

XII.3. Example No. 3 : multiple clusters of IISs

Consider the addition of the following constraints to the set of constraints of Appendix C :

- Constraint₁ : $z_{WP1} \leq 90,000$
- Constraint₂ : $x_1 + x_{AP1} \geq 250,000$
- Constraint₃ : $x_1 - 2x_2 \leq 0$
- Constraint₄ : $x_2 - 5x_3 \leq 0$
- Constraint₅ : $x_3 - 5x_4 \leq 0$
- Constraint₆ : $x_4 - 10x_5 \leq 0$
- Constraint₇ : $x_5 \leq 50$
- Constraint₈ : $re_1 \geq 8,000$
- Constraint₉ : $r_1 - re_1 \geq 50,000$
- Constraint₁₀ : $r_2 - re_1 \geq 87,000$
- Constraint₁₁ : $2y_{BS1} - z_{CS1} \geq 3,500$

This results in the formation of three clusters of IISs :

1. {DEBT_{LIM}₁, INTREST₁, DEBT_{LIM}₂, Constraint₈, Constraint₉, Constraint₁₀}

2. {PSALEA₁, PSALEB₁, PINVENA₁, PRAWMAW₁, SSALEB₁, SINVENA₁, SRAWMAW₁, PROFIT₁, Constraint₁, Constraint₂, Constraint₃, Constraint₄, Constraint₅, Constraint₆, Constraint₇}
3. {SRAWMAU₁, Constraint₁₁}

The first cluster is composed of 5 IISs :

1. {DEBTLIM₁, INTREST₁, Constraint₈}
2. {DEBTLIM₁, INTREST₁, Constraint₉}
3. {DEBTLIM₁, Constraint₂, Constraint₉}
4. {INTREST₁, DEBTLIM₂, Constraint₈, Constraint₁₀}
5. {DEBTLIM₂, Constraint₉, Constraint₁₀}

The second cluster is composed of 2 IISs :

1. {PSALEA₁, Constraint₂, Constraint₃, Constraint₄, Constraint₅, Constraint₆, Constraint₇}
2. {PSALEB₁, PINVENA₁, PRAWMAW₁, SSALEB₁, SINVENA₁, SRAWMAW₁, PROFIT₁, Constraint₁, Constraint₂, Constraint₃, Constraint₄, Constraint₅, Constraint₆, Constraint₇}

The third cluster is composed of an isolated IIS.

XII.3.1 Deletion Filtering

Application of the Deletion Filtering algorithm to the set of constraints resulted in the set {DEBTLIM₂, Constraint₉, Constraint₁₀}. This is one of the IISs.

The algorithm required 47 Phase I LP solutions.

XII.3.2 Deletion/Squeeze Filtering

Application of the initial Squeeze Filtering algorithm resulted in the set $\{S_{RAWMAU1}, DEBTIM_1, INTREST_1, DEBTIM_2, Constraint_6, Constraint_7, Constraint_8, Constraint_{10}, Constraint_{11}\}$. The subsequent application of the Deletion Filtering algorithm to this set resulted in the set $\{INTREST_1, DEBTIM_2, Constraint_8, Constraint_{10}\}$. This is one of the IISs.

The algorithm required 3 Filtering LP solutions, plus 9 Phase I LP solutions. A total of 12 LP solutions.

XII.3.3 Deletion/Squeeze/Sensitivity Filtering

The initial Squeeze Filtering LP solution resulted in the set $\{S_{RAWMAU1}, INTREST_1, Constraint_7, Constraint_8, Constraint_{10}\}$. The subsequent application of Sensitivity Filtering to the solution of the initial Squeeze LP resulted in the set $\{P_{SALEA1}, S_{RAWMAU1}, DEBTIM_1, INTREST_1, Constraint_2, Constraint_3, Constraint_4, Constraint_5, Constraint_6, Constraint_7, Constraint_8, Constraint_9, Constraint_{10}, Constraint_{11}\}$. Upon testing by a second application of the Squeeze Filtering LP this terminated the Squeeze/Sensitivity Phase of the algorithm. The subsequent application of the Deletion Filtering algorithm to this set resulted in the set $\{DEBTIM_1, Constraint_8, Constraint_9\}$. This is one of the IISs.

The algorithm required 1 Filtering LP solutions, plus 42 sensitivity checks, plus 14 Phase I LP solutions, as well as one extra Filtering LP solution to verify that at least one IIS had been identified. A total of 16 LP solutions plus a small amount of computation for the sensitivity checks.

Chapter XIII Comparison to State of the Art

As the methods of Roodman and Greenberg do not guarantee the identification of an IIS, they are not considered in this analysis. In addition, it must be noted that the computation time measures assume that the asymptotic equations hold. These are very rough estimates and are useful only for order-of-magnitude comparisons.

To solve these examples via van Loon's algorithm in essence requires visiting each of the vertices of the polyhedron defined by the constraint set for van Loon's algorithm in an undirected search until the row conditions are met. This is in general more time consuming than a solution of an LP of equivalent size, which is a directed search. This is due to the large number of the vertices of the polyhedron of the LP that may be searched before a proper solution is found. An estimate based on the solution of an equivalent sized LP will underestimate the computation time for van Loon's algorithm. As before, the asymptotic formulae assumption is made. That is lower order terms are ignored and the coefficient of the highest order is disregarded.

Therefore, a rough estimate of the computation time for this algorithm can be obtained by calculating the cube of the number of constraints needed by van Loon's algorithm. As van Loon's algorithm does not imply nonnegativity of the variables and as all equality constraints must be represented as two oppositely directed inequalities, the number of constraints for van Loon's algorithm is equal to the number of constraints in the example plus the number of equality constraints plus the number of nonnegative variables.

XII.4. Summary of Results

The results of the example tests are summarized in Tables 1 and 2. Table 1 records the number of Phase I LPs, the number of Filtering LPs and the number of sensitivity checks. Table 2 records the sizes of the Phase I LPs for the three examples in terms of the number of constraints in the LP. The size of the Filtering LP is constant relative to the example and is equal to the number of constraints of the LP.

		Example No. 1	Example No. 2	Example No.3
Deletion Filtering	Number of Phase I LPs	42	40	47
Deletion Squeeze	Number of Phase I LPs	7	5	9
Filtering	Number of Filtering LPs	8	3	3
Deletion Squeeze	Number of Phase I LPs	7	6	14
Sensitivity Filtering	Number of Filtering LPs	1 + 1	1 + 1	1 + 1
	Number of Sensitivity Checks	41	37	42

Table 1: Summary of Results

	Example No.1	Example No. 2	Example No. 3
Deletion Filtering	1 of sizes 7 - 40 + 2 of size 41 + 6 of size 6	1 of sizes 2 - 34 + 1 of size 6 + 1 of size 2	1 of sizes 3 - 46 + 2 of size 3
Deletion Squeeze Filtering	7 of size 6	1 of size 4 + 4 of size 3	1 of sizes 3 - 8 + 2 of size 6 + 1 of size 4
Deletion Squeeze Sensitivity Filtering	7 of size 6	1 of sizes 4 - 5 + 2 of size 3 + 2 of size 2	1 of sizes 3 - 13 + 1 of size 11 + 1 of size 4 + 1 of size 3

Table 2: Phase I LP sizes in number of constraints

XII.5. Discussion

The results of the example tests of the algorithms are summarized in Tables 1 and 2. However, due to the varying nature and size of the LPs, it is difficult to compare the relative performance of each algorithm. Therefore, a common scale must be developed that will estimate the amount of computation needed by each method.

One such scale results from using the rule of thumb [25, p. 192] that the computation time for the simplex method tends to be roughly proportional to the cube of the number of constraints of the LP. It is assumed that the asymptotic assumption holds, that is : a) all lower order terms are ignored and b) the coefficient of the highest order term is assumed to be 1, i.e. its effect is disregarded. As the computational cost for the sensitivity checks did not involve

the use of any application of the Dual Simplex Method, and as only a very rough estimate is being made, these costs are ignored.

Thus a very rough measure of the total estimated computation time that each algorithm requires to identify an IIS is given by the sum of the cube of the number of constraints in each of the LPs that were solved (be they Filtering LPs or Phase I LPs). For example the computation time measure for the Deletion/Squeeze method on Example No. 2 is :

- 3 Filtering LPs of size 40 equals $3 \times 40^3 = 192,000$
- 5 Phase I LPs of size 4, 3, 3, 3 and 3 equals $4^3 + 4 \times 3^3 = 172$
- Total 192,172

The resulting computation time cost measure for each algorithm on each example is summarized in Table 3. The entries that are enclosed in parentheses indicate the cost of the Deletion/Squeeze/Sensitivity Filtering algorithm if the conjecture that Sensitivity Filtering will identify at least one IIS is valid.

As is evident from Table No. 3, the smallest computation time measure was achieved by the Deletion/Squeeze/Sensitivity Filtering algorithm. The second best performance was by the Deletion/Squeeze Filtering algorithm which used roughly 1.5 times as much computation time as the Deletion/Squeeze/Sensitivity Filtering algorithm. In last place was the Deletion Filtering algorithm which used roughly 6 times as much computation time as the Deletion/Squeeze/Sensitivity Filtering algorithm. This trend in performance is due to the efficiency with which the algorithm reduces the initial constraint set down to a small subset.

	Example No. 1	Example No. 2	Example No.3
Deletion	811,097	608,623	1,170,803
Filtering			
Deletion	594,216	192,381	313,252
Squeeze			
Filtering			
Deletion	149,688	128,334	217,321
Squeeze	(75,600)	(64,334)	(113,498)
Sensitivity			
Filtering			

[Computational Time Measure if Sensitivity

Filtering is valid is shown in brackets]

Table 3. Computational Time Measures for the 3 Examples

If Conjecture 5.4.3 is valid then the time cost measure of Deletion/Squeeze/Sensitivity Filtering is decreased by roughly a factor of two. It is noted that in all three examples, this conjecture was verified.

Both Deletion/Squeeze/Sensitivity Filtering and Deletion/Squeeze Filtering require the addition of 1 (or 2) elastic variables per constraint to the LP. Deletion Filtering does not require this extra cost in space. Thus there is a trade off between time and space complexity for these two classes of algorithms.

For large LPs the additional space requirement of Deletion/Squeeze/Sensitivity Filtering and Deletion/Squeeze Filtering may conflict with that of the maximum capacity of the LP solver, thus inhibiting the use of these two algorithms. The use of sparse matrix representations (as discussed in section XI.1) would extend the

maximum LP size upon which one could use the Deletion/Squeeze Filtering and Deletion/Squeeze/Sensitivity Filtering algorithms. Overall, these two algorithms would seem to perform better than that of the Deletion Filtering algorithm.

As the Deletion/Squeeze/Sensitivity Filtering algorithm requires the same amount of space as the Deletion/Squeeze Filtering algorithm, the best algorithm of the three tested is the Deletion/Squeeze/Sensitivity Filtering algorithm.

Chapter XIII Comparison to State of the Art

As the methods of Roodman and Greenberg do not guarantee the identification of an IIS, they are not considered in this analysis. In addition, it must be noted that the computation time measures assume that the asymptotic equations hold. These are very rough estimates and are useful only for order-of-magnitude comparisons.

To solve these examples via van Loon's algorithm in essence requires visiting each of the vertices of the polyhedron defined by the constraint set for van Loon's algorithm in an undirected search until the row conditions are met. This is in general more time consuming than a solution of an LP of equivalent size, which is a directed search. This is due to the large number of the vertices of the polyhedron of the LP that may be searched before a proper solution is found. An estimate based on the solution of an equivalent sized LP will underestimate the computation time for van Loon's algorithm. As before, the asymptotic formulae assumption is made. That is lower order terms are ignored and the coefficient of the highest order is disregarded.

Therefore, a rough estimate of the computation time for this algorithm can be obtained by calculating the cube of the number of constraints needed by van Loon's algorithm. As van Loon's algorithm does not imply nonnegativity of the variables and as all equality constraints must be represented as two oppositely directed inequalities, the number of constraints for van Loon's algorithm is equal to the number of constraints in the example plus the number of equality constraints plus the number of nonnegative variables.

Consider Example No. 1 : there are 42 constraints, 6 equality constraints and 44 nonnegative variables for a total of 92 constraints. The cube of 92 is 778,688. The computation time measure of van Loon's algorithm relative to the other methods is summarized in table No. 4.

The performance of van Loon's algorithm is on par with Deletion Filtering (see table No. 3). In addition, van Loon's algorithm has a large additional space requirement due to the additional constraints and artificial variables that are required. This limits the practicality of this method for large LPs. In any case, the Deletion/Squeeze/Filtering algorithm seems to be significantly better than van Loon's algorithm for the three examples.

The time complexity of Gleeson and Ryan's algorithm is $O([n + 1]^2mv)$, where n is the number of constraints, m the number of variables and v the number of IISs. The time to find one IIS is $O([n + 1]^2m)$. As before, the asymptotic formulae assumption is made. That is, lower order terms are ignored and the coefficient of the highest order is disregarded. As well, the Gleeson and Ryan algorithm is very recent and undeveloped. Thus it is difficult to determine how its actual performance will compare to its theoretical time complexity. These are very rough estimates used only to give some order-of-magnitude comparisons of the algorithms.

As Gleeson and Ryan's algorithm does not imply nonnegativity of the variables and as all equality constraints must be represented as two oppositely directed inequalities, the number of constraints required is equal to the number of constraints in the example plus the number of equality constraints plus the number of nonnegative variables. The number of variables remains the same.

Consider Example No. 2 : there are 40 constraints, 6 equality constraints and 44 nonnegative variables for a total number of 90 constraints. $[90 + 1]^2 \times 44 = 364,364$. The computation time measure of Gleeson and Ryan's algorithm relative to the other methods is summarized in table No. 4.

The performance of Gleeson and Ryan's algorithm is comparable with that of the Deletion/Squeeze Filtering algorithm. However, Gleeson and Ryan's algorithm has a large additional space requirement due to the additional constraints that are required. The Computation Time Measures for the van Loon's algorithm is about twice that of Gleeson and Ryan's algorithm. Thus one concludes that Gleeson and Ryan's algorithm is a better algorithm than van Loon's not only due to a better computation time, but also as it is a directed search while van Loon's algorithm is not.

Table 4 shows that Deletion/Squeeze/Sensitivity Filtering outperforms both van Loon's and Gleeson and Ryan's algorithms. Both Gleeson and Ryan's algorithm and Deletion/Squeeze/Sensitivity Filtering are improvements over van Loon's algorithm. One must keep in mind that these are only rough estimates of the computation times and that only a few example tests have been used. However, from these results it would seem that at worst the Deletion/Squeeze/Sensitivity Filtering algorithm is on par with that of Gleeson and Ryan.

This is due to the necessary addition of a) explicit nonnegativity constraints and b) paired inequality constraints for each original equality constraint into the coefficient matrix for Gleeson and Ryan's algorithm.

Let us now do a rough analysis on the relative performance between Gleeson and Ryan's algorithm and the Deletion/Squeeze/Sensitivity Filtering algorithm for

	Example No. 1	Example No. 2	Example No. 3
Deletion	149,688	128,334	217,321
Squeeze	(75,600)	(64,334)	(113,498)
Sensitivity			
Filtering			
van Loon (to find a single IIS)	778,688	729,000	912,673
Gleeson and Ryan (to find a single IIS)	380,556	364,364	422,576

[Computational Time Measure if Sensitivity

Filtering is valid is shown in brackets]

Table 4: Table Entries are Computation Time Measures

the case of identifying a single IIS in the LP.

- Let a represent the number of equality constraints
- Let b represent the number of nonequality constraints
- Let g represent the number of nonnegative variables
- Let h represent the number of unrestricted variables
- Let e represent the cardinality of the IIS
- Thus $a + b = n$, the number of constraints
- Thus $g + h = m$, the number of variables

The computation time measure for Gleeson and Ryan's algorithm is $O[(b + 2a + g + 1)^2 \times (g + h)]$. This is due to the fact that two inequalities are required for each equality and one constraint is required for each nonnegative variable.

The computation time measure for the Deletion/Squeeze/Sensitivity Filtering algorithm is $(e + 1)(a + b)^3 + e \times (e - 1)^3$ [it would be $(a + b)^3 + e \times (e - 1)^3$ if Sensitivity Filtering is valid]. This is due to the fact that at most $e + 1$ Filtering LPs of size $a + b$ are needed as well as e Phase I LPs of size $e - 1$. Each unrestricted variable is converted to two nonnegative variables. For example, if x_i is unrestricted, it is replaced by $x_i^+ - x_i^-$, where both x_i^+ and x_i^- are nonnegative variables.

If e is small relative to n , then the computation time measure for the Deletion/Squeeze/Sensitivity Filtering algorithm is of the order $(a + b)^3$.

Let us consider the case where $a \gg b$ and $g \gg h$. Thus $a + b \simeq a$ and $g + h \simeq g$.

Thus the two computation time measures become a^3 for the Deletion/Squeeze/Sensitivity Filtering algorithm and $(2a + g)^2 \times g = 4a^2g + 4ag^2 + g^3$ for Gleeson and Ryan's algorithm. If $a = 4g$ then the computation time measure for Gleeson and Ryan's algorithm becomes $a^3 \times (1 + 1/4 + 1/4^3)$. Thus as long as a is not more than about 4 times as large as g , the computation time measure for Gleeson and Ryan's algorithm will be larger than that of the Deletion/Squeeze/Sensitivity Filtering algorithm.

This supports the conclusion that if the LP has a large number of equality constraints and nonnegativity constraints, then the Deletion/Squeeze/Sensitivity Filtering algorithm will outperform Gleeson and Ryan's algorithm in identifying

a small IIS. As most network LP models will have a) predominantly equality constraints b) nonnegativity of variables and c) about the same number of variables as constraints, this suggests that one would want to use Deletion/Squeeze/Sensitivity Filtering for identifying the IISs in these types of LPs.

Let us now consider the case of a cluster of IISs in the LP. A cluster of IISs is a collection of interrelated IISs. The deletion of one IIS constraint from a cluster may remove several IISs. The number of IISs in a cluster can be large compared to the number of constraints. Gleeson and Ryan's algorithm is designed to find all of the IISs of the LP. The computation time measure for finding all of the IISs is estimated by multiplying the computation time measure for identifying one IIS by the number of IISs. The question that arises is whether one always want to identify all of the IISs of this type of structure? It may be more relevant to the problem to obtain the constraints of the cluster as well as the extending IISs of the cluster. If this is the case, then the Deletion/Squeeze/Sensitivity Filtering algorithm may be more appropriate, especially if a large number of IISs are present in the cluster.

For the general case of multiple clusters of IISs with only a few IISs per cluster, Gleeson and Ryan's algorithm performs very well. A possible condition in which the Deletion/Squeeze/Sensitivity Filtering algorithm would be better is if the LP contained only isolated IISs of relatively small sizes (in addition to a large number of equality constraints and/or sign restricted variables). In this case, repeated applications of the Deletion/Squeeze/Sensitivity Filtering algorithm would be able to identify each of the isolated IISs in less computation time than Gleeson and Ryan's algorithm.

Thus the Deletion/Squeeze/Sensitivity Filtering algorithm outperforms Glee-

son and Ryan's algorithm in identifying isolated IISs under the conditions of an LP with a large number of nonnegativity constraints and/or a large number of equality constraints. If the number of overlapped IISs per cluster is reasonable, then the use of Gleeson and Ryan's algorithm would seem to be preferred.

As stated before, the Deletion/Squeeze/Sensitivity Filtering algorithm does not indicate if a nonnegativity constraint is a member of the IIS identified. From the modellers point of view, this information may or may not be useful. In some cases the modeller has no control over nonnegativity constraints, that is, the type of model assumes nonnegativity of all of its variables. In such a case there is no real need for the identification of nonnegativity IIS constraints. In other cases where the modeller has control over these variables, there is a definite need for knowledge about the complete IIS, including nonnegativity IIS constraints

Information on the presence of nonnegativity constraints in an IIS can be obtained by either applying Gleeson and Ryan's algorithm, complete with nonnegativity constraints, to the IIS constraints identified by the Deletion/Squeeze/Sensitivity Filtering algorithm or perhaps by the application of some equivalent method from linear algebra (see section XV.4). As the cardinality of the IIS will in most cases be smaller than the original set of constraints, the additional cost of applying Gleeson and Ryan's algorithm will be minimal. Thus for a small extra cost, the nonnegativity constraints of an IIS can be identified. If this cost is less than the difference in the cost of the two algorithms, then it is still advantageous to use the Deletion/Squeeze/Sensitivity Filtering algorithm over that of Gleeson and Ryan's algorithm.

Chapter XIV Conclusions

- The Deletion, Deletion/Squeeze and Deletion/Squeeze/Sensitivity Filtering algorithms correctly identify a single IIS from an inconsistent set of constraints. They do so in a time complexity that is polynomial in the number of LPs to be solved. These algorithms are relatively simple to implement, and have in general small requirements for additional space. From an operational point of view they are practical algorithms. Thus the main research problem has been solved: efficient and robust algorithms for the identification of a single IIS have been developed.
- If Sensitivity Filtering is shown to be valid, then the computation time for the Deletion/Squeeze/Sensitivity Filtering algorithm will be cut by about 50% for the identification of small IISs. This would increase the practicality of the algorithm.
- Using a rough computation time measure estimate, it was found that the Deletion/Squeeze/Sensitivity Filtering algorithm outperforms all of the current State of the Art algorithms with the possible exception of that of Gleeson and Ryan. The Deletion/Squeeze/Sensitivity Filtering algorithm may be better than Gleeson and Ryan's algorithm for LPs with a large number of equality constraints and/or sign restricted variables and if one of the following is true :
 - ☐ a single IIS exists in the LP
 - ☐ only isolated IISs exists in the LP
 - ☐ each cluster consists of relatively large number of IISs

As some classes of infeasible LPs may be more efficiently analyzed by the Deletion/Squeeze/Sensitivity Filtering algorithm, it would be worthwhile to continue with the investigation of this algorithm.

- The algorithms are limited in that only one IIS is identified per application of the algorithm (in some cases, the existence of other IISs is detected by the identification of 'extra' IIS constraints). If the IIS identified is corrected then additional IISs can be identified by repeated applications of the algorithm. As the removal of an IIS in a cluster may remove other IISs, this technique will not identify all of the IISs. However, all of the IISs of the LP are removed by this technique.

The algorithms are also limited in that they do not identify nonnegativity IIS constraints. However, inexpensive methods are available for the identification of any nonnegativity IIS constraints of an IIS. In addition, for some applications the detection of nonnegative IIS constraints is irrelevant as the modeller might be restricted to the use of nonnegative variables.

XIV.1. Summary of Contributions

- The major contribution of this thesis is the Deletion/Squeeze/Sensitivity algorithm for identifying IISs. It is a major improvement over all but the very recent and untested method of Gleeson and Ryan. In some applications it is likely to be better than Gleeson and Ryan's algorithm.
- The theoretical characterization of IISs is an important contribution. The formalization of the terminology for the types and groupings of the IISs as well as for the components of the IISs provides a foundation for further research

into the structure of IISs. Among these is the concept of the infeasibility region, which should lead to the development of the theory of the structure of IISs.

- The development of the framework for the Sensitivity Filtering algorithm, which if it is shown to be valid will drastically improve the efficiency of the Deletion/Squeeze/Sensitivity algorithm in most applications.
- The formalization of the concept of elastic programming by the introduction of elastic variables and constraints.

Chapter XV Further Research

Theory of IISs : Structure of the Interior Region of an IIS

Additional research is needed to determine the validity of the conjectures presented in this thesis, especially those that pertain to the validity of the Sensitivity Filtering algorithm. This research would focus on the structure of the interior region of an IIS, initially verifying that the interior region of an IIS is nonempty and then focusing on the possible structures of IISs. One promising approach is to determine the different possible structures of an IIS of a given cardinality, s , in R^n , $2 < s < n + 1$. From this information, one can then possibly develop theorems on the cardinality of the constraints that intersect at a common junction.

Extension of Gleeson and Ryan to Include Implicit Nonnegativity

If one uses a different version of Farkas's Theorem of the Alternate, say that of $\{x \in R_+^n \mid Ax \leq b\}$ and $\{y \in R_+^m \mid y^T A \geq 0, y^T b < 0\}$ [21] then one may be able to set up a parallel theorem to that of Gleeson and Ryan. This would solve the problem of having to add in extra constraints to mimic the nonnegativity of the variables.

Extension to Identify the Causes of Primal Unbounded and Dual Infeasible LP Solutions

An obvious extension is to detect primal unboundedness, or dual infeasibility. Dual infeasibility is easily solved by simply applying the methods to the dual LP. By carrying out the analysis on both the primal and dual, one can determine the type of non-feasibility, i.e. infeasible or unbounded, and for which LP. It will remain to determine if the information from identifying the dual infeasibility will be sufficient to aid the modeller in correcting the primal unboundedness.

The Detection of Nonnegativity Constraints of an IIS

The Deletion/Squeeze/Sensitivity system of algorithms do not identify the nonnegativity constraints of an IIS. That is, the IIS has hidden nonnegativity constraints. It is possible that an efficient method could be found or developed that would a) determine if any hidden nonnegativity constraints exist and b) identify the hidden constraints. This entails exploring such methods as Gaussian elimination as well as methods of determining consistency and inconsistency conditions for sets of linear constraints.

Efficient Implementation of the Algorithms

A future goal is to implement the Deletion/Squeeze/Sensitivity Filtering algorithms using fast LP solution techniques as well as some of the ideas presented in Chapter XI. In this way, testing of the performance of these algorithms could be undertaken and accurate comparisons to the algorithms of Gleeson and Ryan and van Loon can be made.

Extension to Identify all of the IISs

The ideal algorithm would quickly identify all of the IISs in an LP. As the current algorithm only identifies a single IIS, one task would be to extend the algorithm so that it finds all of the IISs.

One possible approach is to develop a backtracking methodology for the Deletion Filtering algorithm. This would allow the algorithm to choose different orderings of the constraints being tested. In this way, all of the IISs could be found. However the time complexity is exponential and as such this will be a very inefficient algorithm in comparison to Gleeson and Ryan.

Appendix A The Directed Graph of a Matrix

The basic principle involved in creating a directed graph [26], digraph, from a rectangular matrix is to construct two sets of nodes one corresponding to the rows of the matrix and the other to the columns of the matrix. If the a_{ij} component of A is nonzero then an arc is created between the i th row node and j th column node. The direction of the arc is dependent upon the sign of the a_{ij} component.

Two variations on this are row and column digraphs. A row digraph is a set of nodes corresponding to the rows of the matrix with arcs from the i th to j th arc if there exists a k such that a_{ik} and a_{kj} are both elements of A . A similar construction produces the column digraph.

For example : consider the matrix, G

$$G = \begin{bmatrix} 3 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 1 & -2 \end{bmatrix}$$

This matrix has the digraph shown in figure 5.

Figure 5: Digraph for matrix G

Appendix B The 'Bump and Spike' Structure

The 'bump and spike' structure of a square matrix (see figure 6) [7], [24] is obtained by attempting to sparsely triangularize the matrix. Each column which has nonzero elements above the diagonal is a spike, while the bumps are collections of spikes that form a 'square' on the diagonal.

After rearrangement of the matrix, one can obtain a matrix of the form in figure 7. The location of the infeasible activity within one of the three labelled sections can be used to prune the search space during the traceback process.

Figure 6: Bumps and spikes

- [9] H. J. Greenberg, "Diagnosing Infeasibility for Min-Cost Network Flow Models, Part II: Primal Infeasibility," *IMA Journal of Mathematics Applied in Business and Industry*, vol. 2, pp. 1-12, 1988.
- [10] H. J. Greenberg, "A Functional Description of ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models," *ACM Transactions on Mathematical Software*, vol. 9, no. 1, pp. 18-56, March 1983.
- [11] H. J. Greenberg, "ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models," *Operations research Letters*, vol. 6, no. 5, pp. 249-255, November 1987.
- [12] A. Brearly, G. Mitra, and H. Williams, "Analysis of Mathematical Programming Problems Prior to Applying the Simplex Algorithm," *Mathematical Programming*, vol. 8, pp. 54-83, 1975.
- [13] T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics*. New York: McGraw-Hill, 1979.
- [14] J. Telgen, *Redundancy and Linear Programs*. Mathematical Centre Tracts No. 137, Amsterdam: Mathematisch Centrum, 1981.
- [15] J. Telgen, "Identifying Redundant Constraints and Implicit Equalities in Systems of Linear Constraints," *Management Science*, vol. 29, no. 10, pp. 1209-1222, October 1983.
- [16] M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, *Redundancy in Mathematical Programming : A State of the Art Survey*. New York: Springer-Verlag, 1983.
- [17] H. Williams, *Model Building in Mathematical Programming*. Chichester: John Wiley and Sons, 2nd ed., 1985.

Appendix C Two Plant - Two Period Model

This LP is a budget model for two plants, P and S, which produce two products, A and B, over two time periods, 1 and 2.

The LP contains 36 constraints of which 6 are equality, 20 are less than or equal to and 10 are greater than or equal to constraints. There are 44 nonnegative variables.

It is a corrected version of the LP found in Jaaskelainen [27].

The LP consists of the following constraints :

- $P_{SALEA1} : x_{AP1} \leq 220,000$
- $P_{SALEB1} : x_{BP1} \leq 300,000$
- $P_{INVENA1} : x_{AP1} - y_{AP1} - t_{APS1} + t_{ASP1} + i_{AP1} \leq 5,000$
- $P_{INVENB1} : -x_{BP1} + y_{BP1} + t_{BPS1} - t_{BSP1} - i_{BP1} \geq 10,000$
- $P_{PRODCA1} : 0.5y_{AP1} + 0.6y_{BP1} \leq 300,000$
- $P_{RAWMAW1} : -3y_{AP1} + z_{WP1} \geq 10,000$
- $P_{RAWMAU1} : 2y_{BP1} - z_{UP1} \leq 10,000$
- $S_{SALEA1} : x_{AS1} \leq 180,000$
- $S_{SALEB1} : x_{BS1} \leq 150,000$
- $S_{INVENA1} : x_{AS1} - y_{AS1} + t_{APS1} - t_{ASP1} + i_{AS1} \leq 2,000$
- $S_{INVENB1} : x_{BS1} - y_{BS1} + t_{BPS1} - t_{BSP1} + i_{BS1} \leq 1,500$
- $S_{PRODCA1} : 0.6y_{AS1} + 0.7y_{BS1} \leq 220,000$

- $S_{\text{RAWMAW1}} : 3y_{AS1} - z_{WS1} \leq 2,000$
- $S_{\text{RAWMAU1}} : 2y_{BS1} - z_{US1} \leq 3,000$
- $P_{\text{SALEA2}} : x_{AP2} \leq 250,000$
- $P_{\text{SALEB2}} : x_{BP2} \leq 300,000$
- $P_{\text{INVENA2}} : -x_{AP2} + y_{AP2} + t_{APS2} - t_{ASP2} + v_{AP1} \geq 0$
- $P_{\text{INVENB2}} : -x_{BP2} + y_{BP2} + t_{BPS2} - t_{BSP2} + v_{BP1} \geq 0$
- $P_{\text{PRODCA2}} : 0.5y_{AP2} + 0.6y_{BP2} \leq 280,000$
- $P_{\text{RAWMAW2}} : -3y_{AP2} + z_{WP2} \geq 0$
- $P_{\text{RAWMAU2}} : -2y_{BP2} + z_{UP2} \geq 0$
- $S_{\text{SALEA2}} : x_{AS2} \leq 150,000$
- $S_{\text{SALEB2}} : x_{BS2} \leq 200,000$
- $S_{\text{INVENA2}} : -x_{AS2} + y_{AS2} - t_{APS2} + t_{ASP2} + v_{AS1} \geq 0$
- $S_{\text{INVENB2}} : -x_{BS2} + y_{BS2} - t_{BPS2} + t_{BSP2} + v_{BS1} \geq 0$
- $S_{\text{PRODCA2}} : 0.6y_{AS2} + 0.7y_{BS2} \leq 230,000$
- $S_{\text{RAWMAW2}} : -3y_{AS2} + z_{WS2} \geq 0$
- $S_{\text{RAWMAU2}} : -2y_{BS2} + z_{US2} \geq 0$
- $\text{CASH}_1 :$

$$\begin{aligned}
& 3.15x_{AP1} + 4.28x_{BP1} - 1.4y_{AP1} - 1.68y_{BP1} - 0.14z_{WP1} - 0.22z_{UP1} \\
& - 0.3t_{APS1} - 0.3t_{ASP1} - 0.4t_{BPS1} - 0.4t_{BSP1} + 3.42x_{AS1} + 4.57x_{BS1} \\
& - 1.68y_{AS1} - 1.96y_{BS1} - 0.14z_{WS1} - 0.22z_{US1} + v_1 - m_1 = 1,451,000
\end{aligned}$$

- **DEBTLIM₁** : $v_1 \leq 50,000$
- **INTREST₁** : $-0.05v_1 + re_1 = 5,000$
- **PROFIT₁** :

$$1.58x_{AP1} + 2.2x_{BP1} - 0.58t_{APS1} - 0.02t_{ASP1} - 0.68t_{BPS1} - 0.12t_{BSP1} \\ + 1.6x_{AS1} + 2.02x_{BS1} - re_1 + 0.03m_1 - 2noe_1 = 1,465,000$$

- **CASH₂** :

$$0.28x_{AP1} + 0.13x_{BP1} - 0.01z_{WP1} - 0.015z_{UP1} + 0.31x_{AS1} + 0.14x_{BS1} \\ - 0.01z_{WS1} - 0.015z_{US1} + 3.15x_{AP2} + 4.28x_{BP2} - 1.4y_{AP2} - 1.68y_{BP2} \\ - 0.14z_{WP2} - 0.22z_{UP2} - 0.3t_{APS2} - 0.3t_{ASP2} - 0.4t_{BPS2} - 0.4t_{BSP2} \\ + 4.57x_{BS2} - 1.68y_{AS2} - 1.96y_{BS2} - 0.14z_{WS2} - 0.22z_{US2} - 0.25v_1 \\ + 3.42x_{AS2} + 1.03m_1 - noe_1 + v_2 - m_2 = 1,229,500$$

- **DEBTLIM₂** : $0.8v_1 + v_2 \leq 100,000$
- **INTREST₂** : $-0.04v_1 - 0.05v_2 + re_2 = 4,500$
- **PROFIT₂** :

$$1.58x_{AP2} + 2.2x_{BP2} - 0.58t_{APS2} - 0.02t_{ASP2} - 0.68t_{BPS2} - 0.12t_{BSP2} \\ + 1.6x_{AS2} + 2.02x_{BS2} - re_2 + 0.03m_2 - 2noe_2 = 1,465,000$$

plus the objective function :

$$\begin{aligned}
 \max z = & 1.58x_{AP1} + 2.20x_{BP1} - 0.58t_{APS1} - 0.02t_{ASP1} - 0.68t_{BPS1} \\
 & -0.12t_{BSP1} + 1.60x_{AS1} + 2.02x_{BS1} - 0.11t_{AP1} - 0.13t_{BP1} \\
 & -0.11t_{AS1} - 0.13t_{BS1} + 1.58x_{AP2} + 2.20x_{BP2} - 0.58t_{APS2} \\
 & -0.02t_{ASP2} - 0.68t_{BPS2} - 0.12t_{BSP2} + 1.60x_{AS2} + 2.02x_{BS2} \\
 & -rc_1 + 0.03m_1 - rc_2 + 0.03m_2
 \end{aligned}$$

This LP has an optimal solution value of $Z = 3,253,000$.

References

- [1] K. Fan, "On Systems of Linear Inequalities," in *Linear Inequalities and Related Systems* (A. Kuhn, H.W. Tucker, ed.), Annals of Mathematical Studies No. 38, pp. 99–156, Princeton: Princeton University Press, 1956.
- [2] G. B. Dantzig, *Linear Programming and Extensions*. Princeton: Princeton University Press, 1963.
- [3] W. L. Winston, *Operations Research : Applications and Algorithms*. Boston: PWS Publishers, 1987.
- [4] J. van Loon, "Irreducibly Inconsistent Systems of Linear Inequalities," *European Journal of Operational Research*, vol. 8, pp. 283–288, 1981.
- [5] S. A. Wiest, "Detecting Infeasibility with an Intelligent Mathematical Programming System," in *Spring Math Clinic Report : Elements of an Intelligent Mathematical Programming System* (H. Greenberg, ed.), pp. 2.1–2.22, Denver: University of Colorado at Denver, 1989.
- [6] G. M. Roodman, "Post-Infeasibility Analysis in Linear Programming," *Management Science*, vol. 25, no. 9, pp. 916–922, Sept. 1979.
- [7] H. J. Greenberg, "Computer-Assisted Analysis for Diagnosing Infeasible or Unbounded Linear Programs," *Mathematical Programming Studies*, vol. 31, pp. 79–97, 1987.
- [8] H. J. Greenberg, "Diagnosing Infeasibility for Min-Cost Network Flow Models, Part I: Dual Infeasibility," *IMA Journal of Mathematics in Management*, vol. 1, pp. 99–109, 1987.

- [9] H. J. Greenberg, "Diagnosing Infeasibility for Min-Cost Network Flow Models, Part II: Primal Infeasibility," *IMA Journal of Mathematics Applied in Business and Industry*, vol. 2, pp. 1–12, 1988.
- [10] H. J. Greenberg, "A Functional Description of ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models," *ACM Transactions on Mathematical Software*, vol. 9, no. 1, pp. 18–56, March 1983.
- [11] H. J. Greenberg, "ANALYZE: A Computer-Assisted Analysis System for Linear Programming Models," *Operations research Letters*, vol. 6, no. 5, pp. 249–255, November 1987.
- [12] A. Brearly, G. Mitra, and H. Williams, "Analysis of Mathematical Programming Problems Prior to Applying the Simplex Algorithm," *Mathematical Programming*, vol. 8, pp. 54–83, 1975.
- [13] T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics*. New York: McGraw-Hill, 1979.
- [14] J. Telgen, *Redundancy and Linear Programs*. Mathematical Centre Tracts No. 137, Amsterdam: Mathematisch Centrum, 1981.
- [15] J. Telgen, "Identifying Redundant Constraints and Implicit Equalities in Systems of Linear Constraints," *Management Science*, vol. 29, no. 10, pp. 1209–1222, October 1983.
- [16] M. H. Karwan, V. Lotfi, J. Telgen, and S. Zionts, *Redundancy in Mathematical Programming : A State of the Art Survey*. New York: Springer-Verlag, 1983.
- [17] H. Williams, *Model Building in Mathematical Programming*. Chichester: John Wiley and Sons, 2nd ed., 1985.

- [18] J. Gleeson, "Diagnosing Redundancy in Large-scale Linear Programs," in *Spring Math Clinic Report : Elements of an Intelligent Mathematical Programming System* (H. Greenberg, ed.), pp. 1.1–1.28, Denver: University of Colorado at Denver, 1989.
- [19] D. B. Johnson, "Finding all the Elementary Circuits of a Directed Graph," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 77–84, March 1975.
- [20] J. Gleeson and J. Ryan, "Identifying Minimally Infeasible Subsystems of Inequalities," tech. rep., University of Colorado at Denver, Mathematics Department, Denver, 1989.
- [21] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York: John Wiley and Sons, 1988.
- [22] M. Dyer, "The Complexity of Vertex Enumeration Methods," *Mathematics of Operations Research*, vol. 8, no. 3, pp. 381–402, August 1983.
- [23] G. Brown and G. Graves, "Use of Elastic Programming," *presented at ORSA/TIMS conference*, November 1977.
- [24] V. Chvatal, *Linear Programming*. New York: W.H. Freeman and Company, 1983.
- [25] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*. Oakland: Holden-Day, Inc., 4th ed., 1986.
- [26] H. J. Greenberg, J. Lundgren, and J. Maybee, "Graph Theoretic Methods for the Qualitative Analysis of Rectangular Matrices," *SIAM Journal of Algebraic and Discrete Methods*, vol. 2, no. 3, pp. 227–239, September 1981.
- [27] V. Jaaskelainen, *Linear Programming and Budgeting*. New York: Petrocelli/Charter, 1975.

END

1809.90

FIN