Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

MATHEMATICAL CENTRE TRACTS 137

REDUNDANCY AND LINEAR PROGRAMS

J. TELGEN

MATHEMATISCH CENTRUM AMSTERDAM 1981

1980 Mathematics subject classification: 52-02, 52A25, 52A40, 68-01, 68C25, 90-02, 90C05

ISBN 90 6196 215 3

... and the masses sang in glory:
"Linear programming what a treasure!"
(Wheelwright and McFarlane [1973])

This work was partly supported by a personal grant from the Ministry of Education and Sciences (Min. van O & W), a NATO Science Fellowship for the Netherlands Organization for the Advancement of Pure Research (ZWO) and a CORE Research Fellowship.

CONTENTS

Contents i
Preface iii

Introduction 1

REDUNDANCY 5

1. Introduction and survey 6
2. Practical considerations 9
3. Theory 12
   3.1. Inequalities 13
   3.2. Equalities 20
   3.3. Implicit equalities 22
   3.4. Minimal representation 27
   3.5. Existing theory 37
4. Methods 41
   4.1. Implicit equalities 41
   4.2. Redundant constraints 45
   4.3. Minimal representation 52
   4.4. Existing methods 54
5. Applications 59
   5.1. Results from literature 59
   5.2. Experimental results 61
6. Related Topics 66
   6.1. Nonbinding constraints 67
   6.2. Primal-dual relations 73
7. Conclusion 75

LINEAR PROGRAMS 77

8. Introduction 78
9. The simplex method 80
10. The complexity of linear programming 90
11. LP-equivalent problems 93
12. The ellipsoidal method 103
13. Conclusion 108

References 113
Subject index 122
Author index 124

This book contains a revised version of a monograph presented at Erasmus University Rotterdam in October 1979.

The revision consists of two parts: first, the occurrence of ever present minor errors is reduced. Second and most important, the work is brought up to date. This required a major revision of the material in the second part of this work. While the computational complexity of linear programming was still open at the time the thesis was written, it is determined by now. An algorithm developed by L.G. Khachian solves linear programming problems in polynomially bounded time. We are glad to be able to include this new result in this book.

There are many people that contributed greatly although indirectly to the realization of this work. First of all I would like to thank prof. dr. ir. H.W. van den Meerendonk for acting as my thesis advisor; prof. dr.dr. T. Gal (Gernuniversität Hagen, B.R.D.) gave valuable advice in the early stages of the thesis and later acted as a referee.

Although not officially involved, Alexander Rinnooy Kan probably had the strongest influence on both my scientific development and my work. I feel deeply grateful for his efforts, enthusiasm, encouragement and guidance, and I am glad to mention that he acted as co-author for a preliminary version of part II of this work.

Other individuals whose interest and help stimulated me include Ton Vorst (theoretical development in sections 3.3-3.4), prof. dr. J.F. Benders (sections 3.1-3.4), prof. Stanley Zionts (part I), Bob Koudenburg (programming assistance), Bert Meyerman (testing assistance), Jaap Spronk, Wim van Dam, Gerrit Timmer, Leen Stougie and Guus Boender (various services, including coffee, soccer, table tennis and proof reading).

I thank the Mathematical Centre for the opportunity to publish this monograph in their series Mathematical Centre Tracts and all those at the Mathematical Centre who have contributed to its technical realization.

INTRODUCTION

Operations Research is not so much a coherent structure of theory and methods, but more like a collection of techniques applicable to problems arising in many different areas.

Mathematical programming, embodying such diverse features as linear, integer, nonlinear, geometric, stochastic and dynamic programming, has become a major tool of operations research.

Generally mathematical programming is applied as part for the following process:

- the practical system to be considered
- gives rise to an abstract view of it, which may be expressed by a mathematical formulation: the model.
- From the model, a mathematical programming problem is constructed,
- to which a solution is determined by means of mathematical programming techniques.
- The solution is translated back into the results for the model;
- the results are given their (economic) interpretation, which may be used to indicate the measures to be taken in practice.

Considering three levels of abstraction: real world, the model and the problem, the process is summarized in figure A.

Except for the relations between the real world and the model, all steps are deductive or can be made along well established paths. Therefore the quality of the implementation depends heavily on the quality of the model. This implies that the modeler's skill and ability are of great importance.

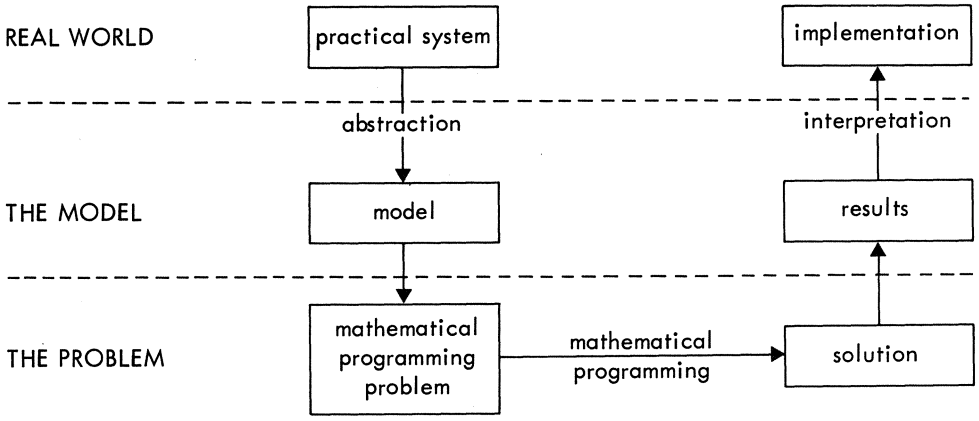

Figure A. Mathematical programming in practice

Once the mathematical model is obtained, the mathematical programming problem is constructed according to the aspects of the system which should be considered and the goals to be achieved. This generally means selecting some constraints and variables and (eventually) one or more objective functions. It should be stressed that the (mathematical) model and the mathematical programming problem generally are not identical.

The solution to a mathematical programming problem usually is obtained by a standard procedure. Some of these procedures are more efficient than others, which may tempt one to model the practical system in such a way that the more efficient procedure may be applied. In doing so one should try to strike a balance between the quality of the results obtained and the effort spent in obtaining these results.

The results for the model, obtained via the solution to the mathematical programming problem, should be given a careful interpretation. This interpretation, in turn, may indicate which measures, if any, are appropriate in the real world situation.

In the scheme described above several loops may occur, especially if the practical system is a complicated one or if interactions between problem solution and modeling are desired.

A major mathematical programming technique is *Linear Programming* (LP), used in the process sketched above, if the mathematical programming problem has linear constraints and a linear objective function. Starting with the development of the *simplex method* by G.B. Dantzig in 1947 a lot of research has been done on methods to obtain the solution to a linear programming problem (*e.g.* Kantorovich [1939], Dantzig [1963]) and the mathematical problems that arise in these methods (*e.g.* Orchard-Hays [1968], Bartels and Golub [1969], Forrest and Tomlin [1972]). The (economic) interpretation of the solution to a linear programming problem is also well established by the work of *e.g.* Koopmans [1951], Dorfman *et al.* [1958].

Far less work has been done on the proper *formulation* of the linear programming problem and the inherent *computational complexity* of linear programming. In this work we concentrate on these topics.

In the first part of this work (chapters 1 through 7) we consider questions such as: "how should the LP problem be formulated?" and "can a given LP problem be replaced by a simpler one?".One of the main characteristics of an LP problem is the number of (linear) constraints, together determining the set of feasible solutions for the problem. Since the size of the system can obviously be reduced if there is *redundancy* in the system of linear constraints we study the latter topic in detail, thus extending our scope beyond linear programming alone.

We establish the concept of a *minimal representation* for any system of linear constraints, and we show that such a minimal representation is obtained if and only if the system contains no *implicit equalities* and redundant constraints. To obtain a minimal representation we develop efficient methods (based on the simplex method) to identify implicit equalities and redundant constraints. Furthermore we show that the theory and methods introduced here provide a generalization of previously known theory and methods, all of which can be obtained as special cases.

In the second part of this work (chapters 8 through 12) we deal with questions such as: "how difficult is LP as compared to other problems?" and "what is the relation between the size of the LP problem to be solved and the number of computations required?". Interest in this aspect of linear and other mathematical programming problems is relatively young and originated within computer science; it is generally referred to as the theory of *computational complexity*.

Results in this area are scattered throughout literature and not very accessible to operational researchers. We have summarized and extended these results giving a "state of the art" survey of the computational complexity of linear programming.

The two parts of this work relate as follows:

- results derived in the first part for systems of linear constraints obviously apply to linear programming as well; in fact, these results simplify some derivations and proofs in the second part;
- in the second part we show that the general linear programming problem is equivalent to the problem of identifying redundant constraints; therefore all results on the complexity of linear programming are equally valid for the theoretical complexity of the redundancy problem.

Throughout this work a basic knowledge of the simplex method is assumed. Good textbooks on this topic are Hadley [1962], Simmonnard [1966] and Luenberger [1973].

No prior knowledge of either computer science or the theory of computational complexity is required.

part 1

REDUNDANCY

1. INTRODUCTION AND SURVEY

Systems of linear *equality* constraints have been studied extensively, but systems of linear *inequality* constraints excited virtually no interest until the advent of game theory in 1944 and linear programming in 1947. However, after the formulation of many practical problems as linear programming problems (Kantorovich [1939]) and the development of the simplex method by G.B. Dantzig [1948] a widespread interest in systems of linear constraints arose.

Only recently the emphasis in these studies has shifted from the system as a whole to the individual constraints. Until the early sixties systems of linear equalities and inequalities were studied from a "*system*"-point of view, in the sense that the system was more important than the individual constraints. A number of interesting results were derived for the solvability and the geometric properties of a system of linear constraints without considering the constraints individually (Farkas [1902], Motzkin [1936], Kuhn and Tucker [1956], Tschernikow [1966]).

From the early sixties on a number of papers were published treating the subject from a "*constraint*"-point of view in the sense that more attention is paid to the individual constraints within the system. As a consequence *redundancy*, which is a phenomenon typically related to individual constraints within a system, was taken into consideration as well. The first paper entirely devoted to redundancy was written by J.C.G. Boot in 1962 (Boot [1962]).

Before proceeding we briefly sketch the general setting of redundancy in systems of linear constraints. A system of linear constraints may contain both equality constraints and inequality constraints. A linear equality constraint corresponds to a hyperplane, namely the set of all points satisfying the equality. A linear inequality constraint corresponds to a halfspace consisting of all points satisfying the inequality. The set of all points satisfying the system of linear constraints is the intersection of all halfspaces and hyperplanes corresponding to the constraints. This set is termed the *feasible region*. The feasible region may be empty, if there is no point which satisfies all individual constraints simultaneously. In that case the system is called infeasible. If the system is feasible, the feasible region being the intersection of a number of halfspaces and hyperplanes, is a (not necessarily bounded) convex polytope.

For the moment (see definitions 3.1.1 and 3.1.2 for an exact mathematical formulation) we define a redundant constraint as a constraint which may be dropped from the system without changing the feasible region. Instead of the term "*redundant*" some authors use other terms: "*trivial*" (Boot [1962]), "*superfluous*" (Thompson *et al.* [1966]), "*irrelevant*" (Mattheis [1973]), "*inessential*" (Zeleny [1974]). From the context of the papers it is clear that all mean the same thing with these different terms.

In figures 1.1.A and 1.1.B some redundant constraints are sketched.

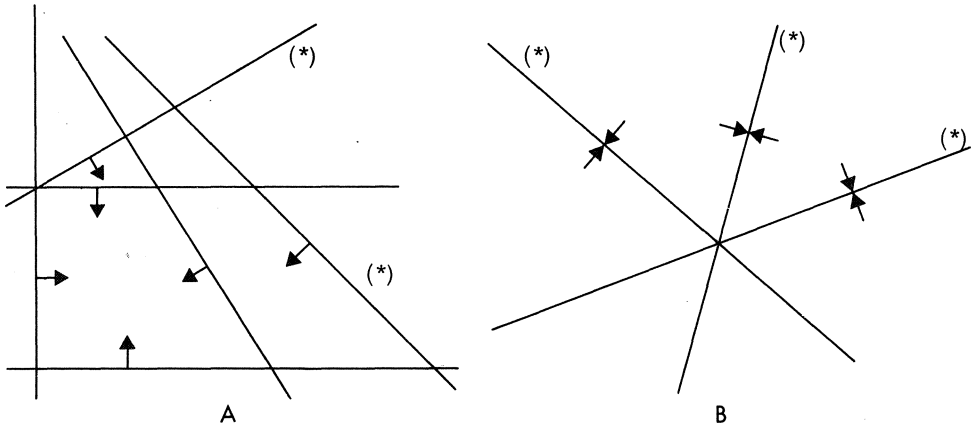

Figure 1.1. Redundant constraints.
A single arrow (⟼) corresponds to an inequality constraint, double arrows (⟼⟻) correspond to an equality constraint; arrows point at the feasible region; constraints marked by an asterix (*) are redundant.

In *chapter 2* we consider the possible origins of redundant constraints and the practical consequences of their presence. These *practical aspects* are treated both from a mathematical point of view and from an information theoretical point of view.

Chapter 3 contains a new and complete *theory* of redundancy published here. We prove that for any system of linear constraints there exists a minimal representation that can be obtained by removing all redundant constraints and implicit equalities. This theory is an extension and a generalization of current theory, since all known results can be derived

from it.

In *chapter 4* we describe *algorithms* to identify all redundant constraints and implicit equalities. All known methods to eliminate redundancy are shown to be special cases of this approach.

Chapter 5 contains the results of some computational *experiments* with the algorithms introduced in chapter 4 and a confrontation with empirical data from literature.

A number of topics closely related to redundancy are considered in *chapter 6*. The most important of these topics is the concept of *nonbinding* constraints; these are nonredundant constraints with the property that removing them does not change the set of optimal solutions to a linear programming problem. Some relations between nonbinding and redundant constraints are established and some ways to identify nonbinding constraints are indicated. A second topic is the presence of redundant and nonbinding constraints in a *dual pair* of linear programming problems. We shall draw some conclusions for the dual problem, if the primal problem exhibits one of these phenomena.

We conclude this first part by some remarks on the questions whether or not redundant constraints should be identified and removed from a general linear programming problem and whether or not the minimal representation of the set of constraints should be obtained.

2. PRACTICAL CONSIDERATIONS

From the nature of redundancy it will be clear that redundant constraints can be omitted from a problem. Since redundant constraints are often present in problems arising in practice (already noted in Hoffman [1955]), we first consider some reasons for the fact that this phenomenon occurs.

These reasons may be distinguished according to the different steps from the real-world system to the model and finally the (programming) problem.

(i) The main reason for redundant constraints to originate in the step from the real world system to the model is insufficient knowledge of the practical system. This may in turn be caused by superficial consideration of the system or because the system itself is too intricate, too big or too difficult to conceive of redundancy as well as other undesirable properties such as inconsistency. Furthermore, in determining the mathematical formulation of the model, modelers may want to "play safe" *i.e.* they want to avoid the possibility that they have to return to this stage of the process because of an omission that is discovered later on *e.g.* by infeasibility or unboundedness. Therefore they may specify more constraints than is strictly necessary. This implies that they obtain a larger model, which they prefer as compared to a smaller model, which may have to be revised later on.

This reason causes more redundant constraints to be present as the systems involved become larger and more complicated. Then modeling is usually done by teams, in which case redundancy may arise since the interactions between the different parts are more difficult to survey.

(ii) In the step from the model to the problem too, the reasons mentioned above (insufficient knowledge, "play safe", teamwork) may cause redundant constraints to be specified. Another reason, which becomes more important as the models involved become larger, is the influence of automation. Often a model is constructed to give rise to several different problems. An example of this phenomenon is provided by a system which has to be regarded under slightly varying conditions; because of changes in the coefficients constraints may change from nonredundant to redundant and vice-versa. For example in production planning models a capacity constraint is specified because it may become binding in future, but it could be redundant at the moment.

(iii) In solving the (programming) problem some techniques require the specification of extra constraints, which may cause redundancy; these techniques include all cutting plane methods for linear- (Dantzig-Wolfe decomposition, dual form; Dantzig and Wolfe [1960]), integer- (Gomory [1958]), mixed integer- (Benders [1962]) and convex nonlinear programming (Kelly [1963]) and all branch and bound methods (*e.g.* Garfinkel and Nemhauser [1972]). In parametric programming (*e.g.* Gal [1979]) redundant constraints may become nonredundant and vice-versa (see Gal [1975 C]).

There are many disadvantageous effects caused by the inclusion of redundant constraints in systems of linear constraints. The most important ones are related to the simplex method of linear programming:

(a) In general there will be more basic solutions if redundant constraints are present in a problem, so the simplex method may require more iterations and even cycling may occur (examples on which the simplex method cycles (Dantzig [1963]) all contain redundant constraints). Furthermore a phenomenon called "near-cycling" (very small changes in the objective function value during a number of iterations) has been noted by Thompson *et al.* [1966] to arise more frequently in the presence of redundant constraints. Finally, redundant constraints may worsen the performance of some non-simplex methods (see *e.g.* Künzi and Tschach [1967]).

(b) Redundant constraints necessitate more calculations per simplex iteration, for example in the determination of the variable to leave the basis. Furthermore they may cause numerical difficulties since redundant constraints often are (nearly) dependent.

(c) Redundant constraints require storage space, which may be critical if the problem can hardly be solved by an in-core code. The extra storage space required by redundant constraints may even cause the problem solver to rely on other procedures *e.g.* decomposition (Thompson *et al.* [1966]).

Note that all disadvantageous effects mentioned above occur in the problem solving stage, although the redundancy may be caused much earlier in the process.

From an *information theoretic* point of view redundant constraints may have both an advantageous and a disadvantageous effect. Sometimes

redundant constraints may express information, that is included in other constraints in a very revealing way. Consider for example the system of linear constraints:

$$\begin{cases} -x_1 - x_2 + x_3 + x_4 \leq 2 \\ x_1 + x_2 + x_3 + x_4 \leq 8 \\ - x_4 \leq -2 \end{cases}$$

The constraint $x_3 \leq 3$ would be redundant in this system (add the first two constraints, divide by 2 and add the last constraint: the result is $x_3 \leq 3$), but conceptually provides useful information about the system in the form of an upper bound on x_3.

Of course redundant constraints objectively do not add any information about the system in the sense that they do not exclude any possibilities (solutions), that would be admitted without these constraints (see also Gal [1975 A]).

On the other hand, just by the sheer fact of their presence in the problem, redundant constraints make the impression of being nonredundant (nobody would specify redundant constraints, isn't it?). This may be a confusing element in the model and obscure the user's view of the system.

Identification (and removal) of redundant constraints results in some simplifications. First, as indicated above, the resulting problem may be solved with minimal computational effort. We shall return to this point in the next chapters. Second, there is an advantageous effect in an information-theoretical sense. Another favorable effect of the identification of redundant constraints may be the recognition of the fact that the slacks of these constraints may be used in an alternative way in any feasible solution (for an example see Zimmerman and Gal [1975] and Telgen [1979 A]).

For all of these points the extent of the resulting simplifications depends on the purpose of specifying the model and the problem. It may make quite a difference if it is formulated just to gain insight into a system or to find an optimal solution. But in all cases some simplifications will result. Even the knowledge of the fact that no redundant constraints have been specified, may be regarded as a simplification.

3. THEORY

We consider the system of linear constraints*

$$(3.1) \qquad \begin{cases} Ax = a \\ Bx \leq b \end{cases}$$

in which $A \in \mathbb{R}^{m_a \times n}$, $B \in \mathbb{R}^{m_b \times n}$, $x \in \mathbb{R}^n$, $a \in \mathbb{R}^{m_a}$ and $b \in \mathbb{R}^{m_b}$
The rows of A and B are denoted by A_i and B_i respectively.

The *feasible region* corresponding to the system (3.1) is defined as:

$$S \equiv \left\{ x \in \mathbb{R}_n \;\middle|\; \begin{matrix} Ax = a \\ Bx \leq b \end{matrix} \right\}$$

Throughout it is assumed that there exists a feasible solution to the system (3.1), *i.e.* $S \neq \emptyset$.

We define $u = (u_1, \ldots, u_{m_b}) \in \mathbb{R}^{m_b}$ as the vector of slack variables of the inequality constraints. Then the system (3.1) can be written as

$$\begin{bmatrix} A & 0 \\ B & I \end{bmatrix} \cdot \begin{bmatrix} x \\ u \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

with I an $(m_b \times m_b)$ identity matrix and 0 a $(m_a \times m_b)$ matrix of zeroes. We replace variables that are not restricted to be nonnegative by the difference of two nonnegative variables and premultiply with the inverse of a basis. Redefinition of the variables (including u) as x_j^N or x_j^B, according to their status (N for nonbasic and B for basic) yields the equivalent system

$$[Y \quad I] \cdot \begin{bmatrix} x^N \\ x^B \end{bmatrix} = [y_o]$$

$$\text{with } x^N, x^B \geq 0$$

in which y_o is the 'updated right hand side'. The matrix Y is usually referred to as the *contracted simplex tableau* (Dantzig [1963]); for simplicity of notation we assume that $Y \in \mathbb{R}^{m \times n}$.

---

* Throughout for two vectors y and z, $y \leq z$ is taken to mean $y_i \leq z_i \quad \forall i$.

3.1. *Inequalities*

Redundant constraints in the system (3.1) are of no interest at all in determining the feasible region S. To define redundant inequality constraints more formally we denote for any fixed $k \in (1, \ldots, m_b)$:

$$S_k \equiv \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} Ax = a & \\ B_i x \leq b_i & \forall i \neq k \end{array} \right\}$$

Definition 3.1.1

The constraint $B_k x \leq b_k$ $(1 \leq k \leq m_b)$ is a *redundant inequality* in the system (3.1) if and only if $S_k = S$.

Define

$$u_k(x) \equiv b_k - B_k x \quad ;$$

then it is easy to see that $B_k x \leq b_k$ $(1 \leq k \leq m_b)$ is a redundant inequality if and only if

$$\hat{u}_k = \min\{u_k(x) \mid x \in S_k\} \geq 0 \tag{3.1.1}$$

To see this note that $S_k = S$ if and only if $B_k x \leq b_k \quad \forall x \in S_k$, which is equivalent to (3.1.1).

If $\hat{u}_k = 0$, then the inequality is termed *weakly redundant*, if $\hat{u}_k > 0$ it is termed *strictly redundant*. Unless explicitly mentioned otherwise the term 'redundant' will be used to indicate constraints that are either strictly or weakly redundant. Both kinds of redundant constraints are sketched in figure 3.1.

From the definition of a redundant constraint it is clear that any inequality in the system (3.1) can be identified as being redundant or not by solving the linear programming problem:

$$\left| \begin{array}{ll} \min & b_k - B_k x \\ \text{s.t.} & Ax = a \\ & B_i x \leq b_i \qquad \forall i \neq k \end{array} \right. \tag{3.1.2}$$

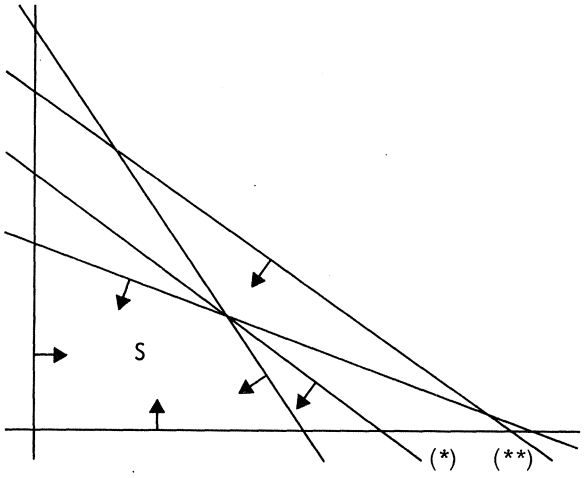

Figure 3.1. A weakly redundant constraint (*) and a strictly redundant constraint (**)

Alternatively, redundant inequality constraints can also be identified by using the *'turn-over'* lemma:

LEMMA 3.1.1 ('turn-over' lemma; Boot [1962])

The constraint $B_k x \leq b_k$ *is redundant in the system (3.1) if and only if the system*

$$(3.1.3) \qquad \begin{cases} Ax = a \\ B_i x \leq b_i & \forall i \neq k \\ B_k x > b_k \end{cases}$$

is infeasible.

Proof:

IF: If (3.1.3) is infeasible and $S_k \neq \emptyset$ (since $S \neq \emptyset$), there is no $x \in S_k$ such that $B_k x > b_k$ and therefore $\hat{u}_k \geq 0$.

ONLY IF: If $B_k x \leq b_k$ is redundant, we have $B_k x \leq b_k \quad \forall x \in S_k$, contradicting $B_k x > b_k$, thus (3.1.3) is infeasible. □

A direct consequence of the 'turn-over' lemma is the following:

COROLLARY 3.1.1

The constraint $B_k x \leq b_k$ is strictly redundant in the system (3.1) if and only if the system

$$\begin{cases} Ax = a \\ B_i x \leq b_i & \forall i \neq k \\ B_k x \geq b_k \end{cases}$$

is infeasible.

Thus a redundant inequality constraint can be identified by checking the system of linear constraints (3.1.3) for feasibility. In part 2 we shall show that this problem too is computationally equivalent to solving a linear programming problem.

However in certain situations it can be seen immediately without solving a linear programming problem whether an inequality is redundant or not by applying one of the following theorems.

THEOREM 3.1.1 (Zionts [1965], Thompson et al. [1966])

The constraint $B_k x \leq b_k$ is redundant in the system (3.1) if there is some basic solution in which $u_k = x_r^B$ with $y_{ro} \geq 0$ and $y_{rj} \leq 0 \quad \forall j$.

Proof:

In a basic solution we have

$$x_r^B = y_{ro} - \sum_{j=1}^{n} y_{rj} x_j^N \tag{3.1.4}$$

since the value of the x_j^N can only increase, the sum will be nonpositive and because $y_{ro} \geq 0$ this yields $x_r^B = u_k \geq 0$. □

As a consequence the constraint $B_k x \leq b_k$ is strictly redundant if $u_k = x_r^B$ with $y_{ro} > 0$ and $y_{rj} \leq 0 \quad \forall j$.

COROLLARY 3.1.2 (Zionts [1965], Thompson et al. [1966])

The constraint $B_k x \leq b_k$ is redundant in the system (3.1) if in some basic solution $u_k = x_p^N$ and there is a row r with

$$\begin{cases} y_{rp} < 0 \\ y_{rj} \geq 0 & \forall j \neq p \\ y_{ro} \leq 0 \end{cases}$$

Proof:

Pivoting on y_{rp} yields the situation described in theorem 3.1.1. □

Again the constraint in the corollary above is strictly redundant if $y_{rp} < 0$, $y_{rj} \geq 0 \quad \forall j \neq p$ and $y_{ro} < 0$.

Note that the row in which a slack variable is basic corresponds to the criterion row of the linear programming problem (3.1.2).

THEOREM 3.1.2 (Gal [1975 B, 1978])

The constraint $B_k x \leq b_k$ is redundant in the system (3.1) if and only if in some basic feasible solution $u_k = x_r^B$ with $y_{rj} \leq 0 \quad \forall j$.

Proof:

IF: In a basic feasible solution $y_{ro} \geq 0$. Because $y_{rj} \leq 0 \quad \forall j$, it follows from (3.1.4) that $\hat{u}_k = y_{ro} \geq 0$.

ONLY IF: Consider the r-th row as the criterion row for the problem (3.1.2), then if $\hat{u}_k \geq 0$, in the optimal solution we should have $y_{rj} \leq 0 \quad \forall j$ and $y_{ro} \geq 0$.

□

Since $\hat{u}_k = y_{ro}$ the constraint is strictly redundant if $y_{ro} > 0$ and weakly redundant if $y_{ro} = 0$ in the above theorem.

THEOREM 3.1.3 (Eckhardt [1971])

The constraint $B_k x \leq b_k$ is redundant in the system (3.1) if and only if in some basic feasible solution $u_k = x_r^B$ with

$$(3.1.5) \qquad \begin{cases} y_{rj} \leq \mu . y_{sj} & \forall j \\ y_{ro} \geq \mu . y_{so} \end{cases}$$

for some index $s \neq r$ and some $\mu \geq 0$.

Proof:

$$\text{IF:} \qquad u_k = x_r^B = y_{ro} - \sum_{j=1}^{n} y_{rj} x_j^N \geq \mu y_{so} - \mu \sum_{j=1}^{n} y_{sj} x_j^N = \mu x_s^B \geq 0.$$

ONLY IF: If the constraint $B_k x \leq b_k$ is redundant, then according to theorem 3.1.2 in some basic feasible solution $u_k = x_r^B$ with $y_{rj} \leq 0 \quad \forall j$. Thus for $\mu = 0$ the conditions (3.1.5) are satisfied. □

In some cases it may be easier to identify constraints that are <u>not</u> redundant. The following theorems relate to that case.

THEOREM 3.1.4 (Mattheis [1973], Gal [1975 B, 1978])

In a nondegenerate basic feasible solution to the system (3.1) all nonbasic variables u_k correspond to nonredundant inequality constraints.

Proof:

For all basic variables we may write

$$(3.1.6) \qquad x_i^B = y_{io} - \sum_{j=1}^{n} y_{ij} \cdot x_j^N$$

Since $y_{io} > 0 \quad \forall i$, a small negative value for some x_j^N will not cause other infeasibilities. Thus $\exists x \in S_k$ such that $u_k(x) < 0$ and therefore $\hat{u}_k < 0$. □

THEOREM 3.1.5

The constraint $B_k x \leq b_k$ is not redundant in the system (3.1) if in some degenerate basic feasible solution $u_k = x_p^N$ and $y_{ip} \geq 0 \quad \forall i$ with $y_{io} = 0$.

Proof:

Consider (3.1.6); a small negative value for x_j^N will not cause infeasibilities in rows for which $y_{io} > 0$; in rows with $y_{io} = 0$ the condition $y_{ip} \geq 0$ prevents new infeasibilities. □

THEOREM 3.1.6 (Telgen [1977 A] modified from Gal [1975 B])

The constraint $B_k x \leq b_k$ *is not redundant in the system (3.1) if there is some basic feasible solution with* $u_k = x_r^B$ *and*

$$(3.1.7) \qquad \frac{y_{ro}}{y_{rs}} = \min_i \left\{ \frac{y_{io}}{y_{is}} \mid y_{is} > 0 \right\}$$

is unique for some s.

Proof:

Pivoting on y_{rs} determined in (3.1.7) yields a new basic feasible solution. If the original solution was nondegenerate, this one is nondegenerate too and we can apply theorem 3.1.4 since u_k is nonbasic. If the new solution is degenerate, the original was degenerate too *i.e.* for some set T we had $y_{to} = 0 \quad \forall t \in T$. From the uniqueness of (3.1.7) it follows that $y_{ts} \leq 0$; thus for the new value, denoted by a prime, we have

$$y'_{ts} = -\frac{y_{ts}}{y_{rs}} \geq 0 \qquad \forall t \in T$$

and therefore the new solution satisfies the conditions for theorem 3.1.5. □

Note that uniqueness of (3.1.7) is not required to prove that $B_k x \leq b_k$ is not strictly redundant.

The following theorems apply to strictly redundant constraints only.

THEOREM 3.1.7

The constraint $B_k x \leq b_k$ *is not strictly redundant in the system (3.1) if in some basic feasible solution* u_k *is a nonbasic variable.*

Proof:

Since $u_k(x) = 0$, $\hat{u}_k \leq 0$. □

THEOREM 3.1.8

The constraint $B_k x \leq b_k$ *is not strictly redundant in the system (3.1) if in some (degenerate) basic feasible solution* $x_r^B = u_k$ *with* $y_{ro} = 0$.

Proof:

Since $u_k(x) = y_{ro} = 0$, $\hat{u}_k \leq 0$. □

Finally, some more relations between redundancy of a constraint and feasibility of a system of linear constraints can be obtained by dualizing (3.1.2). As an example we mention

THEOREM 3.1.9

The constraint $B_k x \leq b_k$ is redundant in the system (3.1) if and only if the system

$$\begin{aligned} &\sum_{i=1}^{m_a} w_i A_i + \sum_{\substack{i=1 \\ i\neq k}}^{m_b} v_i B_i = B_k \\ &\sum_{i=1}^{m_a} w_i a_i + \sum_{\substack{i=1 \\ i\neq k}}^{m_b} v_i b_i \leq b_k \\ &v_i \geq 0 \quad \forall i \neq k \end{aligned} \tag{3.1.8}$$

is feasible.

Proof:

The constraint $B_k x \leq b_k$ is redundant if and only if the optimal value of the linear programming problem

$$\begin{aligned} &\max \ B_k x \\ &\text{s.t. } Ax = a \\ &\quad B_i x \leq b_i \quad \forall i \neq k \end{aligned}$$

is smaller than or equal to b_k. This is true if and only if the corresponding dual problem has an optimal value smaller than or equal to b_k. That, in turn, is true if there is some $w \in \mathbb{R}^{m_a}$ and $v \in \mathbb{R}^{m_b}$ with the properties (3.1.8). □

3.2. *Equalities*

Similar to the case of inequality constraints in the preceding section some equality constraints may be redundant in determining S. We define for any fixed $k \in (1, \ldots, m_a)$:

$$S^k \equiv \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} A_i x = a_i & \forall i \neq k \\ Bx \leq b & \end{array} \right\}$$

Definition 3.2.1

The constraint $A_k x = a_k$ is a *redundant equality* constraint in the system (3.1) if and only if $S^k = S$.

Note that $A_k x = a_k$ may be replaced by

$$\begin{cases} A_k x \leq a_k \\ A_k x \geq a_k \end{cases}$$

This implies that an equality constraint is redundant if and only if both inequality constraints replacing it are redundant. Therefore $S^k = S$ is equivalent to

$$\begin{cases} \min \{a_k - A_k x \mid x \in S^k\} = 0 \\ \max \{a_k - A_k x \mid x \in S^k\} = 0 \end{cases}$$

Again it is not always necessary to solve these linear programming problems to identify redundant equalities. Denote by r(A) the rank of the matrix A.

THEOREM 3.2.1

The constraint $A_k x = a_k$ is redundant in the system (3.1) if r(A) does not change by removing A_k from A.

Proof:

If r(A) does not change by removing A_k from A then $r(A) \leq m_a - 1$, which implies $\exists \lambda \in \mathbb{R}^{m_a}$ such that $\lambda A = 0$ with $\lambda_k \neq 0$ and, since $S \neq \emptyset$, also $\lambda a = 0$. Thus

$$A_k x = -\frac{1}{\lambda_k} \sum_{\substack{i=1 \\ i\neq k}}^{m_a} \lambda_i (A_i x) = -\frac{1}{\lambda_k} \sum_{\substack{i=1 \\ i\neq k}}^{m_a} \lambda_i a_i = a_k$$

and therefore $A_k x = a_k \quad \forall x \in S^k$, which implies $S^k = S$. □

COROLLARY 3.2.1

The system (3.1) contains at least one redundant equality constraint if $r(A) < m_a$.

Proof:

If $r(A) < m_a$, at least one of the equality constraints is linearly dependent on the others, so it may be removed without changing r(A). Then theorem 3.2.1 applies. □

The converse of the corollary is obviously not true since any redundant equality may be replaced by two inequalities such as to make $r(A) = m_a$. Only under some additional assumptions it can be proved that the system (3.1) contains redundant equality constraints if and only if $r(A) < m_a$ (see section 3.4, theorem 3.4.1).

Finally note that it is not necessarily true that every equality constraint is redundant if $r(A) < m_a$.

3.3. *Implicit equalities*

In the preceding section we have seen that one equality constraint was replaced by two inequality constraints. This may be done for all equality constraints. Another way to convert a set of linear equalities into a set of linear inequalities is given by the following theorem.

THEOREM 3.3.1 (Dantzig [1963])

Let ι *denote the vector* $(1, 1, \ldots, 1)^T \in R^m$, *then*

$$\{x \in R^n \mid Ax = a\} = \left\{x \in R^n \;\middle|\; \begin{matrix} Ax \leq a \\ \iota^T Ax \geq \iota^T a \end{matrix}\right\}$$

Proof:

Denote the sets above by V and W respectively. Trivially $\forall x : x \in V \Rightarrow x \in W$. Suppose now $x' \in W$ but $x' \notin V$, then $A_s x' < a_s$ for some s ($A_s x' > a_s$ is impossible because $x' \in W$).
Define $\eta \in \mathbb{R}^m$ with $\eta_i = 1 \quad \forall i \neq s$ and $\eta_s = 0$; then from $Ax \leq a$ we have

$$A_s x' = \iota^T Ax' - \eta^T Ax' \geq \iota^T a - \eta^T a = a_s$$

which contradicts the assumption that $A_s x' < a_s$. □

Conversely to replace inequalities by equalities one usually adds slack variables to the inequalities. However this is not always necessary: some inequalities may be replaced by equalities without enlarging the dimension (number of variables) of the system. Denote

$$V_k \equiv \{x \in \mathbb{R}^n \mid B_k x = b_k\}$$

Definition 3.3.1

The constraint $B_k x \leq b_k$ in the system (3.1) is an *implicit equality* if and only if $S \subset V_k$.

The concept of implicit equalities is introduced here. Some similarity exists with the concept of instable inequalities, used in the theory of the stability of systems of linear constraints (see *e.g.* Robinson [1975]). Using this concept in relation to redundancy seems to be new and

enables us to develop a complete theory of redundancy.

Note that $S \subset V_k$ is equivalent to $S = S_k \cap V_k$. This can be seen from the following argument:
if $S = S_k \cap V_k$ then certainly $S \subset V_k$;
if $S \subset V_k$ then $S \subset (V_k \cap S_k)$ since $S \subset S_k$. It is trivial that $V_k \cap S_k = V_k \cap S$ and thus $(V_k \cap S_k) \subset S$, implying that $S = V_k \cap S_k$.
Denote

$$\tilde{u}_k = \max \{u_k(x) \mid x \in S_k\} \tag{3.3.1}$$

then we can prove

THEOREM 3.3.2

The constraint $B_k x \leq b_k$ is an implicit equality in the system (3.1) if and only if $\tilde{u}_k = 0$.

Proof:

IF: From $\max \{u_k(x) \mid x \in S_k\} = 0$ we see that $B_k x \geq b_k \quad \forall x \in S_k$.
For all $x \in S$ we have $B_k x \leq b_k$; together this yields $B_k x = b_k$ *i.e.* $x \in V_k \quad \forall x \in S$ thus $S \subset V_k$.

ONLY IF: If $B_k x \leq b_k$ is an implicit equality then $u_k(x) = 0 \quad \forall x \in S$ thus

$$\max \left\{ b_k - B_k x \;\middle|\; \begin{array}{l} x \in S_k \\ B_k x \leq b_k \end{array} \right\} = 0$$

implying $\max \{u_k(x) \mid x \in S_k\} = \tilde{u}_k = 0$ □

In theorem 3.3.2 we may replace $\tilde{u}_k = 0$ by the condition

$$\max \{u_k(x) \mid x \in S\} = 0$$

It is trivial that this condition is implied by $\max \{u_k(x) \mid x \in S_k\} = 0$; the converse follows from the first two lines of the 'ONLY IF' part of the proof above.

Note that by theorem 3.3.2 some similarity is shown between implicit equality constraints and redundant inequality constraints. Replacing the max operator in (3.3.1) by a min operator yields (3.1.1).

The following theorems are analogous to theorems 3.1.1 and 3.1.2 and

are therefore stated without proof.

THEOREM 3.3.3 (cf. Zionts [1965])

The constraint $B_k x \leq b_k$ *is an implicit equality in the system (3.1) if in some basic solution* $u_k = x_r^B$ *with* $y_{rj} \geq 0 \quad \forall j$ *and* $y_{ro} = 0$.

COROLLARY 3.3.1 (cf. Zionts [1965])

The constraint $B_k x \leq b_k$ *is an implicit equality in the system (3.1) if in a basic solution* $u_k = x_p^N$ *with* $y_{rp} > 0$ *and* $y_{rj} \geq 0 \quad \forall j \neq p$ *and* $y_{ro} = 0$ *for some row r.*

THEOREM 3.3.4

The constraint $B_k x \leq b_k$ *is an implicit equality in the system (3.1) if and only if in some basic feasible solution* $u_k = x_r^B$ *with* $y_{rj} \geq 0 \; \forall j$ *and* $y_{ro} = 0$.

In some cases a constraint which is <u>not</u> an implicit equality can easily be identified by one of the following theorems.

THEOREM 3.3.5

The constraint $B_k x \leq b_k$ *is not an implicit equality in the system (3.1) if in some basic feasible solution* $u_k = x_r^B$ *and* $y_{ro} > 0$.

Proof:

Trivial. □

THEOREM 3.3.6

The system (3.1) contains no implicit equalities if there exists a nondegenerate basic feasible solution.

Proof:

In a nondegenerate basic feasible solution $y_{io} > 0 \quad \forall i$ and thus theorem 3.3.5 applies to all basic (slack) variables u_k. If u_k is a nonbasic variable it can be introduced into the basis on a positive level by a feasible pivot step and in this new tableau theorem 3.3.5 applies. □

THEOREM 3.3.7

The constraint $B_k x \leq b_k$ is not an implicit equality in the system (3.1) if in some basic feasible solution $u_k = x_j^N$ with $y_{io} > 0$ for all i with $y_{ij} > 0$.

Proof:

From the fact that $y_{io} > 0$ for all i with $y_{ij} > 0$ and

$$x_i^B = y_{io} - \sum_{j=1}^{n} y_{ij} x_j^N$$

we see that x_j^N can attain a small positive value without causing any infeasibilities. □

THEOREM 3.3.8

A strictly redundant constraint is not an implicit equality and conversely.

Proof:

Since $\tilde{u}_k \geq \hat{u}_k$ we cannot have both $\tilde{u}_k = 0$ and $\hat{u}_k > 0$. □

Combining this theorem with theorems on strictly redundant constraints gives the following corollaries.

COROLLARY 3.3.2

The constraint $B_k x \leq b_k$ is not an implicit equality in the system (3.1) if there is some basic solution in which $u_k = x_r^B$ with $y_{rj} \leq 0$ $\forall j$ and $y_{ro} > 0$.

COROLLARY 3.3.3

The constraint $B_k x \leq b_k$ is not an implicit equality in the system (3.1) if there is some basic solution in which $x_p^N = u_k$ with

$$\begin{cases} y_{rp} < 0 \\ y_{rj} \geq 0 & \forall j \neq p \\ y_{ro} < 0 \end{cases}$$

for some row r.

Finally by dualizing the linear programming problem in (3.3.1) we may obtain some theorems that relate the feasibility of a system of linear constraints to implicit equalities. As an example we mention:

THEOREM 3.3.9

The constraint $B_k x \leq b_k$ *is an implicit equality in the system (3.1) if and only if the system*

$$\begin{cases} A^T w + B^T v = - B_k \\ a^T w + b^T v = - b_k \\ \qquad\quad v \geq 0 \end{cases}$$

is feasible.

Proof:

The constraint $B_k x \leq b_k$ is an implicit equality if and only if the optimal value of the linear programming problem

$$\begin{cases} \max \ -B_k x \\ \text{s.t. } Ax = a \\ \qquad Bx \leq b \end{cases}$$

is equal to $-b_k$. This is true if and only if the dual problem

$$\begin{cases} \min \ a^T w + b^T v \\ \text{s.t. } A^T w + B^T v = - B_k \\ \qquad\qquad\quad v \geq 0 \end{cases}$$

has the same optimal value. That in turn is true if the conditions for the theorem are satisfied. □

3.4 *Minimal representation*

We have seen that the set of feasible solutions S may be represented in various ways *i.e.* by different sets of linear constraints. For various purposes it may be desirable to have a representation that is as small as possible.

Definition 3.4.1

A *minimal representation* of the set S is a system of linear constraints (3.1) such that

$$S = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{l} Ax = a \\ Bx \leq b \end{array} \right\}$$

with $m = m_a + m_b$ and every other system describing S has at least m constraints.

In the remainder of this section we establish some relations between redundant constraints and implicit equalities and finally prove the *main redundancy theorem* which says that a minimal representation is obtained by explicitly stating all implicit equalities and by removing all redundant constraints.

First we introduce some more notation:

$V \equiv \{x \in \mathbb{R}^n \mid Ax = a\}$

$V_k \equiv \{x \in \mathbb{R}^n \mid B_k x = b_k\}$ (as before)

$r \equiv n - m_a$

It will be convenient to refer to the following conditions by their number:

(I) $r(A) = m_a$; A is of full row rank

(II) $\hat{u}_i < 0 \quad \forall i$; the system (3.1) contains no redundant inequalities

(III) $\tilde{u}_i > 0 \quad \forall i$; the system (3.1) contains no implicit equalities

The following three lemmas require some relatively involved mathematical derivations.

We denote by lin(S) the smallest linear manifold containing S *i.e.* the subset of $\mathbb{R}^n$ consisting of all linear combinations of vectors from S.

By $\overset{\circ}{S}$ we denote the interior of S in lin(S). Finally we define

dim S = dim(lin(S)).

LEMMA 3.4.1

If the system (3.1) satisfies (III) then

$$\overset{\circ}{S} = \left\{ x \in \mathbb{R}^n \quad \begin{array}{l} Ax = a \\ Bx < b \end{array} \right\}$$

Proof:

First, suppose $x' \in \mathbb{R}^n$ is such that $Ax' = a$ and $Bx' < b$, then around x' there is an ε-ball $o(x', \varepsilon)$ such that $Bz < b \quad \forall z \in o(x', \varepsilon)$. Any $z \in \{o(x', \varepsilon) \cap \text{lin}(S)\}$ is a linear combination of x' and elements from S and so $Az = a$ thus $z \in S$. Therefore $\{o(x', \varepsilon) \cap \text{lin}(S)\} \subset S$ and $x' \in \overset{\circ}{S}$.

For the remainder of the proof it might be helpful to consider the following figure.

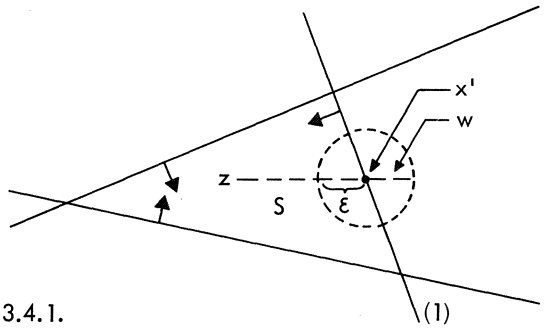

Figure 3.2.
The second part of the proof of lemma 3.4.1.

Now suppose $x' \in \mathbb{R}^n$ is such that $Ax' = a$ and $B_i x' < b_i \quad \forall i = 2,\ldots,m_b$ but $B_1 x' = b_1$. Since the system (3.1) satisfies (III) there is a point $z \in S$ with $B_1 z < b_1$. We have to prove that there is a point w in any $\{o(x', \varepsilon) \cap \text{lin}(S)\}$ with $w \notin S$:

choose $w = \lambda x' + (1-\lambda)z$ with $\lambda = 1 + \frac{1}{2}\varepsilon$.

Then $B_1 w = (1 + \frac{1}{2}\varepsilon)B_1 x' - \frac{1}{2}\varepsilon B_1 z$

$$(1 + \tfrac{1}{2}\varepsilon)b_1 - \tfrac{1}{2}\varepsilon b_1 = b_1$$

thus $w \notin S$. □

LEMMA 3.4.2 (Eckhardt [1975]) (Interior point lemma)

If the system (3.1) satisfies (III) then $\overset{\circ}{S} \neq \emptyset$.

Proof:

From (III) we see that for all i there is a point $y^i \in S_i$ such that $B_i y^i < b_i$. Consider

$$y = \frac{1}{m_b} \sum_{i=1}^{m_b} y^i \quad ;$$

it is easy to see that $Ay = a$ and $By < b$ which means $y \in \overset{\circ}{S}$.

LEMMA 3.4.3

If the system (3.1) satisfies (I) and (III) then dim S = n - m_a = r.

Proof:

From the interior point lemma we know that there is a point $y \in \overset{\circ}{S}$ such that $\{V \cap o(y, \varepsilon)\} \subset S$. Because of (I) $\dim(V) = n - m_a = r$ and therefore for some basis $\{\alpha_1, \ldots, \alpha_r\}$ we can write

$$V = \{y + \lambda_1\alpha_1 + \ldots + \lambda_r\alpha_r \mid (\lambda_1, \ldots, \lambda_r)^T \in \mathbb{R}^r\}$$

Furthermore $\exists\delta > 0$ such that, if $|\lambda_i| \leq \delta$ for all i then

$$\{y + \lambda_1\alpha_1 + \ldots + \lambda_r\alpha_r\} \in o(y, \varepsilon)$$

and $y, y+\delta\alpha_1, \ldots, y+\delta\alpha_r \in \{V \cap o(y, \varepsilon)\}$
Since $\delta\alpha_1, \ldots, \delta\alpha_r$ are linearly independent we have
$\dim\{V \cap o(y, \varepsilon)\} \geq r$ which implies $\dim S \geq r$.
But since $S \subset V$ we also have $\dim S \leq r$ thus $\dim S = r$. □

Using these lemmas we can prove the following theorems, which apply to situations in which implicit equalities are explicitly stated as equalities. Note that under such an operation the set of feasible solutions does not change.

THEOREM 3.4.1

The system (3.1) which satisfies (III) contains redundant equalities if and only if r(A) < m_a.

Proof:

IF: Follows from corollary 3.2.1.

ONLY IF: Suppose $r(A) = m_a$ and (III) holds, then $\dim S = n - m_a$ (lemma 3.4.3). If there are redundant equalities in the system, we can remove them and the new system still satisfies (III) and now satisfies (I)

too. Then according to lemma 3.4.3 dim S = n - m_a' > n - m_a and this contradicts the assumption. □

THEOREM 3.4.2

If the system (3.1) contains exactly one inequality constraint which is an implicit equality, then this inequality constraint is redundant as well.

Proof:

Assume the first inequality is an implicit equality: $B_1x = b_1$ $\forall x \in S$.
According to the interior point lemma there is an $y \in S$ such that
$B_iy < b_i$ $\forall i = 2, \ldots, m_b$.
Suppose the first inequality is not redundant, then there is a point
$z \in S_1$ with $B_1z > b_1$.
Take $w(\lambda) = \lambda z + (1-\lambda)y$. Since $o(y, \varepsilon) \subset S_1$ and $z \in S_1$ we have
$w(\lambda) \in S_1$ for $-\frac{1}{2}\varepsilon \leq \lambda \leq 1$. But

$$B_1w(-\tfrac{1}{2}) = -\tfrac{1}{2}B_1z + (1 + \tfrac{1}{2})B_1y < -\tfrac{1}{2}\varepsilon b_1 + (1 + \tfrac{1}{2}\varepsilon)b_1 = b_1$$

thus the first inequality is not an implicit equality, contradicting the assumption. □

COROLLARY 3.4.1

If in the system (3.1) all implicit equalities are replaced by equality constraints, then the new system contains at least one redundant equality.

Proof:

Replacing an implicit equality $B_ix \leq b_i$ by an equality constraint, is equivalent to adding the constraint $B_ix \geq b_i$ to the system; the latter constraint is obviously redundant.
Furthermore the last implicit equality is redundant by theorem 3.4.2 and if both $B_ix \leq b_i$ and $B_ix \geq b_i$ are redundant the equality is redundant as well. □

Now we turn to a number of lemmas which are helpful in proving the main redundancy theorem.

LEMMA 3.4.4

If the system (3.1) satisfies (I), (II) and (III), then the system

$$(3.4.1) \qquad \left| \begin{array}{ll} Ax = a & \\ B_1x = b_1 & \\ B_ix \leq b_i & \forall i = 2, \ldots, m_b \end{array} \right.$$

contains no redundant equalities and implicit equalities.

Proof:

Assume the system (3.4.1) contains a redundant equality, then there is some $\lambda \in \mathbb{R}^{m_a}$ such that $B_1 = \lambda_1 A_1 + \ldots + \lambda_{m_a} A_{m_a}$. This yields for all $x \in V$

$$\begin{aligned} B_1x &= \lambda_1 A_1 x + \ldots + \lambda_{m_a} A_{m_a} x \\ &= \lambda_1 a_1 + \ldots + \lambda_{m_a} a_{m_a} \\ &= c \quad \text{(constant)} \end{aligned}$$

and this leads to a contradiction, since $b_1 \geq c$ contradicts (II) and $b_1 \leq c$ contradicts (III) for the system (3.1). Therefore $r\begin{pmatrix} A \\ B_1 \end{pmatrix} = m_a+1$ and the system (3.4.1) contains no redundant equalities.

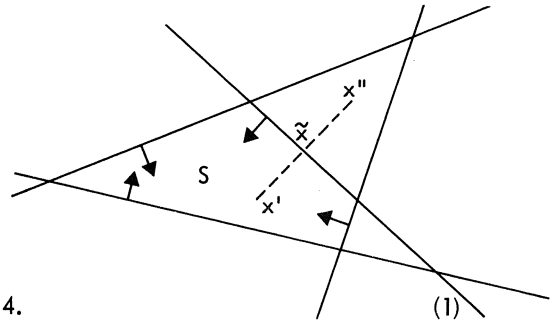

Figure 3.3.
Proving the second part of lemma 3.4.4.

In following the proof that the system contains no implicit equalities, figure 3.3 might be helpful. According to the interior point lemma there is some $x' \in V$ with $B_ix' < b_i \quad \forall i$. Since (III) holds for the system (3.1) there is some $x'' \in V$ with $B_1x'' > b_1$ and $B_ix'' \leq b_i$ $\forall i = 2, \ldots, m_b$.

Choose $0 < \lambda < 1$ such that $B_1(\lambda x' + (1-\lambda)x'') = b_1$.

(this is possible: $\lambda = (b_1 - B_1x')/(B_1x'' - B_1x')$)

For all $2 \leq i \leq m_b$ we have

$$B_i(\lambda x' + (1-\lambda)x") = \lambda B_i x' + (1-\lambda)B_i x"$$
$$< \lambda b_i + (1-\lambda)b_i = b_i$$

So $\tilde{x} = \lambda x' + (1-\lambda)x"$ satisfies

$$\begin{cases} A\tilde{x} = a \\ B_1\tilde{x} = b_1 \\ B_i\tilde{x} < b_i \qquad \forall i = 2, \ldots, m_b \end{cases}$$

and therefore the system (3.4.1) contains no implicit equalities. □

Figure 3.4 might provide some help in studying the following corollary and the next lemma.

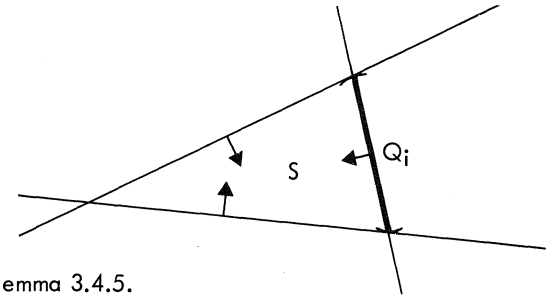

Figure 3.4.
The situation for corollary 3.4.2. and lemma 3.4.5.

COROLLARY 3.4.2

If the system (3.1) satisfies (I), (II) and (III) then for any $i \in (1, \ldots, m_b)$ there is a convex set Q_i such that $Q_i \subset (S_i \cap V_i)$ and dim $Q_i = r - 1$.

Proof:

For all $x \in (S_i \cap V_i)$ we have

$$\begin{cases} Ax = a \\ B_i x = b_i \\ B_j x \leq b_j \qquad \forall j \neq i \end{cases}$$

Take $Q_i = S_i \cap V_i$, then clearly Q_i is convex. According to lemma 3.4.4 the conditions for lemma 3.4.3 are satisfied in this new system and thus dim $Q_i = r-1$. □

LEMMA 3.4.5

If the system (3.1) satisfies (III) then for any convex set Q with $Q \subset (S \setminus \mathring{S})$ and dim Q = r - 1 there is some V_j with $Q \subset V_j$.

Proof:

Suppose no V_j contains Q; then by condition (III) for all j there is an $y^j \in Q$ such that

$$\begin{cases} Ay^j = a \\ B_j y^j < b_j \\ B_i y^j \leq b_i \qquad \forall i \neq j \end{cases}$$

Take $y = \frac{1}{m_b} \sum_{j=1}^{m_b} y^j$ then by convexity of Q we have $y \in Q$ and

$$\begin{cases} Ay = a \\ By < b \end{cases}$$

thus $y \in \mathring{S}$, contradicting the fact that $Q \subset (S \setminus \mathring{S})$. □

Note that it may be proved that $Q \subset V_j$ for exactly one j; this however, is not required for the following.

LEMMA 3.4.6

If the system (3.1) satisfies (III) then $V \neq V_j$ for all j.

Proof:

Suppose there is some j such that $V \subset V_j$, then $B_j x = b_j \quad \forall x \in V$, implying $\max \{b_j - B_j x \mid x \in V\} = 0$. But since $S \subset V \cap G$ this yields $\max \{b_j - B_j x \mid x \in S\} \leq 0$, contradicting (III). □

LEMMA 3.4.7

If the system (3.1) satisfies (I), (II) and (III) then V is the smallest linear manifold which contains both Q_k and Q_j ($k \neq j$; $Q_k \subset (V_k \cap S_k)$; $Q_j \subset (V_j \cap S_j)$; dim Q_k = dim Q_j = r - 1).

Proof:

The linear manifold V contains Q_k and Q_j by definition; dim V = r. Suppose there is another linear manifold W, containing Q_k and Q_j;

since dim Q_k = dim Q_j = r - 1 we must have dim W ≥ r - 1.
If dim W = r and W ≠ V then dim (V ∩ W) = r - 1, since (V ∩ W) = W' contains Q_k and Q_j; thus W' would also be a linear manifold that contains both Q_k and Q_j. Hence we may restrict ourselves to the case: dim W = r - 1.
Since both W and V_k are linear manifolds and $Q_k \subset V_k$ we have $W \subset V_k$; similarly $W \subset V_j$. From the assumption we have $W \subset V$ thus $W \subset (V \cap V_k \cap V_j)$, but since dim$(V \cap V_k)$ = r - 1 (lemma 3.4.3) this implies $(V \cap V_k) \subset V_j$. The latter is impossible because of the following (see also figure 3.5):

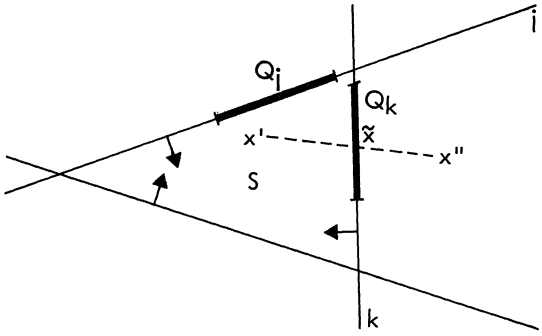

Figure 3.5.
Proving lemma 3.4.7.

Choose some $x' \in \mathring{S}$ (interior point lemma) thus $B_i x < b_i \quad \forall i$.
Choose $x'' \in S_k$ such that $B_k x'' > b_k$ which is possible according to (II).
Then there is some $0 < \lambda < 1$ with $B_k(\lambda x' + (1-\lambda)x'') = b_k$.
Define $\tilde{x} = \lambda x' + (1-\lambda)x''$; since $\tilde{x} \in V$ and $\tilde{x} \in V_k$ also $\tilde{x} \in (V \cap V_k)$.
However $B_j\tilde{x} = \lambda B_j x' + (1-\lambda)B_j x'' < \lambda b_j + (1-\lambda)b_j = b_j$ and therefore $\tilde{x} \notin V_j$, thus $(V_k \cap V) \not\subset V_j$. □

At this point we are fully equipped to prove the main theorem.

THEOREM 3.4.3 (MAIN REDUNDANCY THEOREM)

The system (3.1) is a minimal representation of the set S if and only if it contains no redundant constraints and implicit equalities.

Proof:

IF: Note that the system (3.1) contains no redundant constraints and implicit equalities if and only if the conditions (I), (II) and (III) are satisfied.
Suppose that there is another system (indicated by primes) that also

represents S but with $m' < m$; we assume that (I), (II) and (III) hold for that system as well.

First we prove that the number of equalities in both systems is equal. Then we prove that the sets of points satisfying all equality constraints are the same for both systems.

According to lemma 3.4.3 $\dim S = n - m_a = r$; however also $\dim S = n - m'_a = r'$ and therefore $m_a = m'_a$.

The smallest linear manifold containing S has dimension $n - m_a$, but also $S \subset V$ and $\dim V = n - m_a$; therefore V is the smallest linear manifold containing S. The same is true for V' and thus $V' = V$.

Now we show that every inequality constraint in one system corresponds to at least one inequality in the other system, but no two correspond to the same inequality. This implies that the number of inequality constraints in both systems is equal.

Choose $i \in (1, \ldots, m_b)$; corollary 3.4.2 says that there is some $Q_i \subset (S_i \cap V_i)$ for this i and according to lemma 3.4.5 there is some $V_{i'}$ corresponding to this Q_i. Thus, related to any i there is an i'.

Suppose now for $i \neq j$ we have $i' = j'$ thus $V_{i'} = V_{j'}$ thus $Q_i \subset V_{i'}$ and $Q_j \subset V_{i'}$; but according to lemma 3.4.7 the smallest linear manifold containing both Q_i and Q_j is V.

Therefore $V \subset V_{i'}$ which would be contradicting lemma 3.4.6. Thus for $i \neq j$ we have $i' \neq j'$ and thus $m_b \leq m'_b$.

In the same way we prove $m'_b \leq m_b$ and thus $m_b = m'_b$.

ONLY IF: Suppose the system (3.1) contains redundant constraints; these may be removed to obtain a smaller system. This contradicts the fact that the system is a minimal representation. Suppose the system (3.1) contains implicit equalities; replacing all of them by equality constraints would make at least one of them redundant (corollary 3.4.1), so this one may be removed to obtain a smaller system. Again this is a contradiction with the minimality of the representation. □

The main redundancy theorem has some nice practical aspects. In fact it says that given a system of linear constraints one can obtain a minimal representation for the same set of solutions, by just leaving out some (redundant) constraints and replacing inequality signs by equality signs. Note that the coefficients of the system do not change under these operations.

Finally note that although the number of equality and inequality constraints in the minimal representation is uniquely determined, the minimal representation in itself is not unique.

3.5. *Existing theory*

Existing theory on redundancy has led to some kind of minimal representation in only two instances. These will be treated in chronological order and shown to be special cases of the theory we developed.

The first contribution in this respect was the work of Shefi [1969], later modified and extended in Luenberger [1973].

Definition 3.5.1 (Shefi [1969], Luenberger [1973])

A *minimal similar representation* for the convex polyhedral set T is a system of linear constraints

$$\begin{cases} Dx = d \\ x \geq 0 \end{cases} \tag{3.5.1}$$

with $D \in \mathbb{R}^{m_d \times n_d}$, x and $0 \in \mathbb{R}^{n_d}$ and $d \in \mathbb{R}^{m_d}$ such that:

(i) for $\bar{T} = \left\{ x \in \mathbb{R}^{n_d} \;\middle|\; \begin{matrix} Dx = d \\ x \geq 0 \end{matrix} \right\}$

there is a linear invertible mapping Σ (called a similarity transformation) that maps lin(T) onto lin($\bar{T}$) such that $\Sigma(T) = \bar{T}$ and $\Sigma^{-1}(\bar{T}) = T$;

(ii) m_d is minimal;

(iii) n_d is minimal.

Loosely speaking, this means that a minimal similar representation of a set T is a system of linear equalities in nonnegative variables that determines a set $\bar{T}$ with the same shape and dimensions as T and does this in the smallest number of equalities and the smallest number of variables.

The main difference between a minimal representation and a minimal similar representation is that the latter is embedded in a linear manifold of smallest possible dimension and is given relative to that linear manifold.

Shefi [1969] and Luenberger [1973] indicate a way how to obtain a minimal similar representation. To achieve this they introduce some terminology:

- *null* variables are defined to be zero in every feasible solution; in our terminology this means that the corresponding inequality constraint is an implicit equality. Null variables can be removed by striking out the corresponding column in the matrix D and adding the equality $x_j = 0$.
- variables are *nonextremal* if the corresponding nonnegativity constraint $x_j = 0$ is redundant. Nonextremal variables can be removed from the system, by eliminating them from a constraint in which they have a nonzero coefficient. Then this constraint can be replaced by a definition of the removed variable in terms of the other variables *e.g.* if x_1 is nonextremal and $a_{11} \neq 0$ then the constraint $\sum_{j=1}^{n} a_{1j}x_j = b_1$ may be replaced by

$$x_1 = \frac{1}{a_{11}} (b_1 - \sum_{j=2}^{n} a_{1j}x_j)$$

By means of a fairly complex proof Shefi and Luenberger show:

THEOREM 3.5.1 (Shefi [1969], Luenberger [1973])

Suppose $T \neq \emptyset$ and bounded: then the system (3.5.1) is a minimal similar representation of T if and only if it contains no redundant equations, null variables and nonextremal variables.

We give a new proof of this theorem, using our main redundancy theorem.

Proof:

IF: If the system (3.5.1) contains no redundant equations, null variables or nonextremal variables, it satisfies the conditions (I), (II) and (III), so we can apply the main redundancy theorem and therefore the system (3.5.1) is a minimal representation.
We have proved that the number of faces of T of dimension equal to (dim T-1) is fixed and equal to the number of inequality constraints. Since the number of these faces does not change under a similarity transformation, we cannot transform T to a space with smaller dimension.
Since we also proved that dim T = $n_d - m_d$ and hence is constant we cannot find a system with less equations in the same number of variables.

ONLY IF: Assume that the system (3.5.1) is a minimal similar representation. If it contains a redundant equality, this can be dropped,

contradicting point (ii) of the definition of a minimal similar representation.
If it contains a null variable, point (iii) would be contradicted.
If it contains a nonextremal variable x_k, points (ii) and (iii) are contradicted. □

In fact we proved a somewhat stronger theorem, because we did not use the boundedness assumption of Shefi [1969] and Luenberger [1973]. This implies that theorem 3.5.1 is a corollary of our main redundancy theorem, but not the converse.

In practice the minimal similar representation of a set T may seem to be smaller than the minimal representation but this is an illusion. In fact this constitutes an important point of criticism on the theory of Shefi [1969] and Luenberger [1973].

In the minimal similar representation a minimal dimension (number of variables) is required: variables corresponding to dimensions that are dropped are appended to the system as equalities. Their values can be calculated afterwards. However, this is no reason to consider these variables as not being a part of the system. From a practical point of view there is no such thing as lowest dimension: all variables specified in the set T should be present in a minimal representation as well. Therefore the variables that are appended to the system in a minimal similar representation, form an integral part of the system and should also be incorporated in the minimal representation.

Another way to interprete this point is to consider the minimal similar representation as a one-to-one transformation of the original system. Given this minimal similar representation, nothing can be said about the original system if the transformation is not known. Therefore the minimal similar representation is useless without this one-to-one transformation. This implies that, since the transformation is embedded in the equalities that are appended to the system, these equalities should be part of a minimal representation,as is done in our theory.

In case equalities appended to the system are considered as a part of the minimal similar representation, that representation will never be smaller than our minimal representation, as can be seen from the following table.

Shefi [1969] Luenberger [1973]	Telgen
- redundant equality: remove 1 equality	- redundant equality remove 1 equality
- null variable remove 1 equality $x_p \geq 0$ strike out 1 column in matrix D append 1 equality $x_p = 0$	- implicit equality $x_p \geq 0$: replace inequality $x_p \geq 0$ by equality $x_p = 0$
- nonextremal variable x_q: remove 1 inequality $x_q \geq 0$ use 1 equality to eliminate x_q from all equalities. append 1 equality net result: remove 1 inequality	- redundant inequality $x_q \geq 0$ remove 1 inequality $x_q \geq 0$

Table 3.1. Relations between the concepts used by Shefi [1969] and Luenberger [1973] and the concepts used here.

Another important point can also be seen from the table. Whereas our reductions require no additional calculations, in the theory of Shefi [1969] and Luenberger [1973] the removal of nonextremal variables requires extra calculations for all coefficients in the system.

Finally, if necessary a minimal similar representation can readily be obtained from a minimal representation by forcing all variables that are not sign restricted to enter into the basis.

Eckhardt [1977] considers systems in which only linear inequality constraints are present. By defining *instable* inequalities in the same way as our implicit equalities, Eckhardt [1977] can prove a number of special cases ($m_a = 0$) of our theorems and lemmas.

However the minimal representation in Eckhardt [1977] is defined as a system that contains no redundant constraints. Thus, no attention is paid to the minimality of the number of constraints. By defining a minimal representation in that way Eckhardt [1977] avoids a confrontation with the main redundancy theorem.

4. METHODS

4.1. Implicit equalities

Recall from section 3.3 that the constraint $B_k x \leq b_k$ in the system

$$\begin{cases} Ax = a \\ Bx \leq b \end{cases} \tag{4.1.1}$$

is an implicit equality if

$$\max\{u_k(x) \mid x \in S_k\} = 0$$

This means, that to identify implicit equalities we could solve the linear programming problem

$$\begin{cases} \max \; b_k - B_k x \\ \text{s.t. } Ax = a \\ \qquad B_i x \leq b_i \qquad \forall i \neq k \end{cases}$$

for all $k \in (1, \ldots, m_b)$, thus solving m_b linear programming problems.

However by using theorems 3.3.3 through 3.3.8 a more efficient algorithm can be developed. This algorithm, described in detail below can be sketched as follows: in a basic feasible solution the tableau is scanned for the existence of the conditions required for the application of theorems 3.3.3 through 3.3.8, by which a constraint can be identified as either an implicit equality or not. Then from the remaining constraints we select one (*e.g.* the one with smallest index k) and perform a pivot operation so as to maximize $u_k(x)$. After this constraint has been identified we turn to the next unidentified one, meanwhile checking all intermediate tableaux as we did the first tableau.

Formally the algorithm consists of the following steps:

Algorithm IMPLEQ

Initialize: We assume a basic feasible solution is given and the corresponding contracted simplex tableau is set up.
Let G be the set of all indices of constraints which should be checked for being implicit equalities.

Step 1: If the solution is nondegenerate, then by theorem 3.3.6 there are no implicit equalities: STOP;

Step 2: Check all basic variables $x_i^B = u_k$ with $k \in G$ for the property $y_{io} > 0$;
if this holds then $B_k x \leq b_k$ is not an implicit equality by theorem 3.3.5 and k should be removed from G;

Step 3: Check all basic variables $x_i^B = u_k$ with $k \in G$ for the property $y_{ij} \geq 0 \quad \forall j$;
If this holds the constraint $B_k x \leq b_k$ is an implicit equality by theorem 3.3.3 and k should be removed from G;

Step 4: Check all nonbasic variables $x_p^N = u_k$ with $k \in G$ for the property: $y_{rp} > 0$, $y_{rj} \geq 0 \quad \forall j \neq p$ and $y_{ro} = 0$ for some r.
If this holds then the constraint $B_k x \leq b_k$ is an implicit equality by corollary 3.3.1; remove k from G;

Step 5: Check all nonbasic variables $x_j^N = u_k$ with $k \in G$ for the property: $y_{io} > 0$ for all i with $y_{ij} > 0$.
If this holds then the constraint $B_k x \leq b_k$ is not an implicit equality by theorem 3.3.7; remove k from G;

Step 6: If $G = \emptyset$: STOP;

Step 7: If there is no basic variable $x_i^B = u_k$ with $k \in G$, introduce a nonbasic variable $x_j^N = u_k$ with $k \in G$ into the basis, *e.g.* the one with the smallest index k; continue with step 1;

Step 8: Select a basic variable $x_r^B = u_k$ with $k \in G$ (*e.g.* the one with the smallest index k. Perform a feasible pivot step in column p with $y_{rp} = \min_j y_{rj}$. Continue with step 1.

The IMPLEQ algorithm is finite, since the cardinality of G, denoted $|G|$, is nonincreasing. In iterations, in which $|G|$ does not change, the algorithm maximizes $u_k(x)$ for some fixed $k \in G$ using the simplex method. Because the simplex method is finite (if necessary use an anti-cycling device *e.g.* Bland [1977]), $|G|$ will decrease after a finite number of steps.

The fact that the IMPLEQ algorithm correctly identifies all implicit equalities is implied by the correctness of theorems 3.3.3 through 3.3.7.

The flow chart of the IMPLEQ algorithm is given in figure 4.1.

A disadvantage of the IMPLEQ algorithm is the fact that a basic feasible solution for the system has to be known before the algorithm can start. However, in the process of determining such a basic feasible

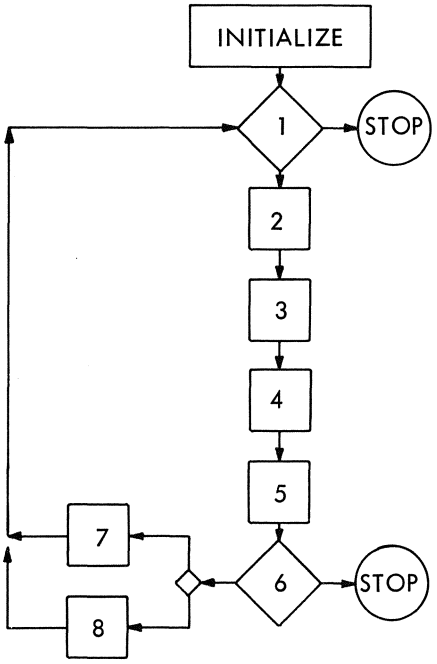

Figure 4.1.

The flow chart of the IMPLEQ algorithm; the numbers indicate the steps

solution (if necessary) one can apply theorem 3.3.3 and corollaries 3.3.1, 3.3.2 and 3.3.3 to any intermediate tableau, since they do not require a feasible solution. Then G may be initialized without the indices of the constraints identified in this preliminary stage.

Furthermore as soon as the constraint $B_k x \leq b_k$ has been identified as an implicit equality, it may be converted into an explicit equality, by dropping the column in which the associated u_k is nonbasic, thus diminishing the size of the tableau.

To illustrate the IMPLEQ algorithm consider the example

$$\begin{cases} x_1 - x_2 \leq 0 \\ x_1 + x_2 \leq 1 \\ -x_1 + 2x_2 \leq 0 \\ x_1, x_2 \geq 0 \end{cases}$$

A basic feasible solution is $(x,u)^T = 0$ and the corresponding contracted simplex tableau is:

	x_1	x_2	RHS
u_1	1	-1	0
u_2	1	1	1
u_3	-1	2	0

Applying the IMPLEQ algorithm yields:

step 2: u_2 does not correspond to an implicit equality;

step 8: select u_1, pivot on $y_{32} = 2$, yielding the tableau:

	x_1	u_3	RHS
u_1	½	½	0
u_2	3/2	-½	1
x_2	-½	½	0

step 3: u_1 corresponds to an implicit equality;

step 4: $x_1 \geq 0$ is an implicit equality;

u_3 corresponds to an implicit equality;

step 8: select x_2, pivot on $y_{11} = 1$, which gives the tableau:

	u_1	u_3	RHS
x_1	1	1	0
u_2	-3	-2	1
x_2	1	1	0

step 3: $x_2 \geq 0$ is an implicit equality;

step 6: STOP.

4.2. *Redundant constraints*

Recall from section 3.1 that the constraint $B_k x \leq b_k$ is redundant in the system (4.1.1) if and only if

$$\min\{u_k(x) \mid x \in S_k\} \geq 0$$

This means that we could identify all inequality constraints to be redundant or not by solving the linear programming problem

$$\begin{cases} \min \; b_k - B_k x \\ \text{s.t. } Ax = a \\ \qquad B_i x \leq b_i \quad \forall i \neq k \end{cases}$$

for all $k \in (1, \ldots, m_b)$, thus solving m_b linear programming problems. A more efficient way to identify (all) redundant inequalities is given by the algorithm below, based upon theorems 3.1.1 through 3.1.6. Basically the algorithm checks the tableau associated with a given basic feasible solution for the conditions for theorems 3.1.1 to 3.1.6. If any of these theorems applies some constraints may be identified as redundant or not. From the remaining unidentified constraints one is selected (*e.g.* the one with smallest index) and its corresponding slack variable is minimized. All intermediate tableaux obtained in the course of this minimization are scanned for the conditions for theorems 3.1.1 through 3.1.6. After this selected constraint has been identified we turn to the next one and so on.

The algorithm is described more formally below:

Algorithm REDINQ

Initialize: We assume a basic feasible solution is given and the corresponding contracted simplex tableau is set up. Let H be the set of all indices of constraints to be identified as either redundant or nonredundant inequalities.

Step 1: If the solution is nondegenerate, all $u_k = x_j^N$ correspond to nonredundant inequalities (theorem 3.1.4); remove these k from H and continue with step 3;

Step 2: Check all nonbasic variables $x_p^N = u_k$ with $k \in H$ for the property $y_{ip} \geq 0 \quad \forall i$ with $y_{io} = 0$

If this holds then the constraint $B_k x \leq b_k$ is not redundant (theorem 3.1.5): remove k from H;

Step 3: Check all basic variables $x_i^B = u_k$ with $k \in H$ for the property $y_{ij} \leq 0 \quad \forall j$.
If this holds then the constraint $B_k x \leq b_k$ is redundant (theorem 3.1.1): remove k from H;

Step 4: Check all basic variables $x_r^B = u_k$ with $k \in H$ for the property

$$\frac{y_{ro}}{y_{rs}} = \min_i \left\{ \frac{y_{io}}{y_{is}} \;\middle|\; y_{is} > 0 \right\} \text{ is unique for some s.}$$

If this holds the constraint $B_k x \leq b_k$ is not redundant (theorem 3.1.6): remove k from H;

Step 5: If $H = \emptyset$, then STOP;

Step 6: If there is no basic variable $x_i^B = u_k$ with $k \in H$, then introduce a nonbasic variable $x_j^N = u_k$ with $k \in H$ (*e.g.* the one with the smallest index k) into the basis and continue with step 1;

Step 7: Select a basic variable $x_i^B = u_k$ with $k \in H$ (*e.g.* the one with the smallest index k) and perform a feasible pivot step in column p with $y_{ip} = \max_j y_{ij}$;
continue with step 1.

The finiteness of the REDINQ algorithm can be proved along the same lines as the finiteness of the IMPLEQ algorithm. The cardinality of H is nonincreasing; in iterations in which |H| does not decrease, the algorithm merely applies the simplex method to minimize $u_k(x)$ for some fixed $k \in H$. Because of the finiteness of the simplex method $\hat{u}_k$ will be determined in a finite number of steps and then |H| decreases.

The correctness of the REDINQ algorithm is implied by the correctness of theorems 3.1.1 through 3.1.6.

As in the case of the IMPLEQ algorithm, the REDINQ algorithm requires a basic feasible solution to start with; this is a major drawback. However, while looking for this feasible solution we can apply theorem 3.1.1 and corollary 3.1.2 to all intermediate tableaux, since they do not require a feasible solution. The indices of all constraints that have been identified as redundant or not in this preliminary stage, can be excluded from H.

Furthermore, rows in which u_k is basic, while $B_k x \leq b_k$ is identified as redundant may be deleted as soon as this identification takes place.

This may reduce the number of computations by diminishing the size of the tableau.

The flow chart of the REDINQ algorithm is given in figure 4.2.

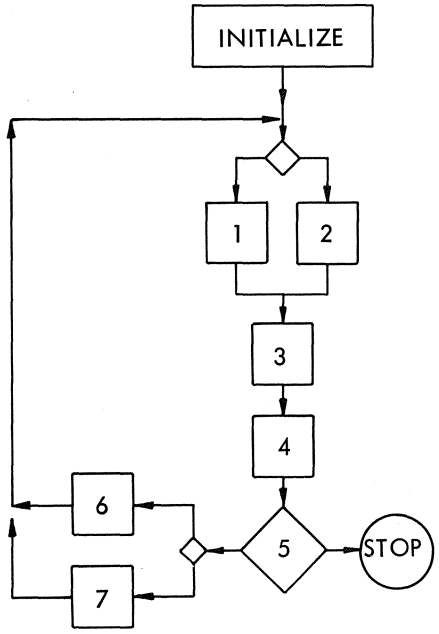

Figure 4.2.

The flow chart of the REDINQ algorithm; the numbers refer to the steps

To illustrate the use of the REDINQ algorithm we give the following example. Consider the system:

$$
\begin{cases}
x_1 - x_2 \leq 2 & (1) \\
2x_1 + x_2 \leq 7 & (2) \\
x_1 \leq 2 & (3) \\
-x_1 + 2x_2 \leq 4 & (4) \\
2x_2 \leq 5 & (5) \\
x_1 + x_2 \leq 4 & (6) \\
x_1 , x_2 \geq 0 & (7,8)
\end{cases}
$$

which is shown graphically in figure 4.3.

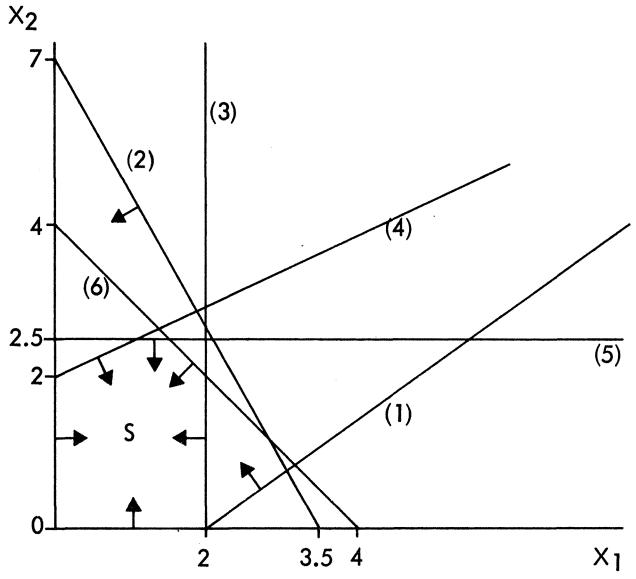

Figure 4.3. Example of the REDINQ algorithm

A basic feasible solution is given by $(x,u)^T = 0$ and the corresponding contracted simplex tableau is:

	x_1	x_2	RHS
u_1	1	-1	2
u_2	2	1	7
u_3	1		2
u_4	-1	2	4
u_5		2	5
u_6	1	1	4

Applying the REDINQ algorithm yields:

Step 1: $x_1 \geq 0$ and $x_2 \geq 0$ nonredundant;

Step 4: u_3 corresponds to a nonredundant constraint;

Step 7: select u_1, pivot on $y_{31} = 1$ to obtain the tableau:

	u_3	x_2	RHS
u_1	-1	-1	0
u_2	-2	1	3
x_1	1		2
u_4	1	2	6
u_5		2	5
u_6	-1	1	2

Step 3: u_1 corresponds to a redundant constraint.
Step 4: u_6 corresponds to a nonredundant constraint.
Step 7: select u_2, pivot on $y_{62} = 1$.

	u_3	u_6	RHS
u_1	-2	1	2
u_2	-1	-1	1
x_1	1		2
u_4	3	-2	2
u_5	2	-2	1
x_2	-1	1	2

Step 1: u_3 corresponds to a nonredundant constraint,
Step 3: u_2 corresponds to a redundant constraint,
Step 4: u_5 corresponds to a nonredundant constraint,
Step 5: STOP.

If only strictly redundant constraints are required to be identified a modified variant of the REDINQ algorithm may be used.

Algorithm STREDINQ

Initialize: We assume a basic feasible solution is given and the corresponding contracted simplex tableau is set up.
Let E be the set of all indices of constraints to be identified as either strictly redundant or not.

Step 1: All nonbasic variables $x_j^N = u_k$ correspond to constraints which are not strictly redundant (theorem 3.1.7): remove these k from E;

Step 2: Check all basic variables $x_i^B = u_k$ with $k \in E$ for the property $y_{io} = 0$;

if it holds the constraint $B_k x \leq b_k$ is not strictly redundant (theorem 3.1.8): remove these k from E;

Step 3: Check all basic variables $x_i^B = u_k$ with $k \in E$ for the property $y_{ij} \leq 0 \quad \forall j$;
if it holds the constraint $B_k x \leq b_k$ is strictly redundant (theorem 3.1.2): remove these k from E;

Step 4: Check all basic variables $x_r^B = u_k$ with $k \in E$ for the property

$$\frac{y_{ro}}{y_{rs}} = \min_i \left\{ \frac{y_{io}}{y_{is}} \;\middle|\; y_{is} > 0 \right\}$$

if this holds then the constraint is not strictly redundant (theorem 3.1.6): remove these k from E;

Step 5: If $E = \emptyset$, then STOP;

Step 6: Select a basic variable $x_i^B = u_k$ (*e.g.* the one with the smallest index k) and perform a feasible pivot step in column p with $y_{ip} = \max y_{ij}$;
continue with step 2.

The flow chart of the STREDINQ algorithm is given in figure 4.4.

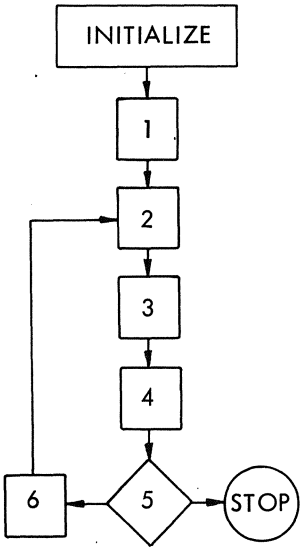

Figure 4.4.
The flow chart of the STREDINQ algorithm; the numbers refer to the steps

The correctness and finiteness proof for the STREDINQ algorithm is similar to the one for the REDINQ algorithm.

The STREDINQ algorithm too requires a basic feasible solution to start with. Again in the process of determining such a solution all intermediate tableaux can be checked for the conditions of theorem 3.1.1 and corollary 3.1.2, since these do not require a feasible solution.
Also constraints may be deleted as soon as they have been identified as strictly redundant, thus diminishing the size of the tableau.

Recall from section 3.2 that the *equality* constraint $A_k x = a_k$ is redundant if and only if

$$\begin{cases} \max \{a_k - A_k x \mid x \in S^k\} = 0 \\ \min \{a_k - A_k x \mid x \in S^k\} = 0 \end{cases}$$

To identify whether an equality constraint is redundant or not, we could solve these two linear programming problems or use the IMPLEQ and REDINQ algorithms. However it is usually a more efficient method to state all implicit equalities explicitly as equalities (after identifying then by the IMPLEQ algorithm) and then use theorem 3.4.1, which says that the system contains redundant equalities if and only if $r(A) < m_a$. We can determine $r(A)$ by checking a basic solution (which is at hand if the IMPLEQ algorithm is used) for an all-zero row. Then from theorem 3.2.1 we know that the constraint corresponding to this row is redundant and may be removed.

4.3. Minimal representation

An algorithm to obtain a minimal representation can be constructed easily from the IMPLEQ and REDINQ algorithms

Algorithm MINREP

Step 1: Determine a basic feasible solution; while doing so check all intermediate tableaux for the conditions of theorems 3.3.3 and 3.1.1 and corollaries 3.1.2, 3.3.1, 3.3.2 and 3.3.3.
If redundant constraints are identified remove them, if implicit equalities are identified replace them by explicit equalities;

Step 2: Apply the IMPLEQ algorithm, where G contains the indices of all nonidentified constraints; replace all implicit equalities by explicit equalities;

Step 3: Apply the REDINQ algorithm, where H contains the indices of all nonidentified constraints; remove redundant inequalities as soon as they have been identified as such;

Step 4: Remove all redundant equalities.

A slightly more efficient variant of this algorithm can be formulated by checking all intermediate tableaux, obtained during the application of the IMPLEQ algorithm for properties by which (non)redundant constraints can be identified, *i.e.* applying steps 1, 2, 3 and 4 of the REDINQ algorithm to the intermediate tableaux.

The finiteness of the MINREP algorithm follows directly from the finiteness of the IMPLEQ algorithm and the REDINQ algorithm.
Regarding the correctness of the MINREP algorithm we know that the resulting system contains no redundant constraints because these were removed in steps 3 and 4. The resulting system could contain implicit equalities only if these were created in steps 3 and 4, since all implicit equalities were removed in step 2. However removing redundant constraints can not cause implicit equalities to originate; if $\max\{u_k(x) \mid x \in S\} > 0$ then certainly $\max\{u_k(x) \mid x \in S_j\} > 0$ and also $\max\{u_k(x) \mid x \in S^j\} > 0$. Thus the resulting system contains no implicit equalities and hence is a minimal representation by the main redundancy theorem.

Note that the order in which the steps 2, 3 and 4 are given in the algorithm is essential. Checking for implicit equalities should precede checking for redundancy. Otherwise there is no guarantee that a minimal

representation is obtained: replacing implicit equalities by explicit equalities after application of the IMPLEQ algorithm causes some constraints to become redundant (cf. theorem 3.4.1).

4.4. *Existing methods*

The methods described in the preceding sections are general for two reasons: firstly all implicit equalities and all redundant constraints can be identified with these methods. Secondly all methods known from the literature can be shown to be special cases of the methods introduced here. In the following we review these methods briefly (see also Telgen [1977 C]).

From the literature no methods are known to identify implicit equalities or to obtain a minimal representation. Wolfe [1955] describes a method to reduce a problem to a 'simplest problem in standard form'. However the meaning of this expression is not clearly defined although it certainly does not correspond to a minimal representation.

Shefi [1969] gives an algorithm to obtain a minimal similar representation. This algorithm amounts to solving the linear programming problems $\max\{u_k(x) \mid x \in S_k\}$ and $\min\{u_k(x) \mid x \in S_k\}$ for all k. Since the feasible region changes every time a nonextremal variable or a null variable is removed, the calculations may have to be repeated a number of times. Therefore the algorithm has to solve at least $2n_d$ and at most n_d^2 linear programming problems of size comparable to the original system.

Redundant equality constraints are usually identified by checking for $r(A) < m_a$. The fact that this is not a necessary condition for an equality constraint to be redundant, seems to be recognized here for the first time (theorem 3.2.1 and theorem 3.4.1). Therefore our method to identify redundant equality constraints by first stating explicitly the implicit equalities is a generalization of the usual method.

In the remainder of this section we concentrate on methods to identify redundant inequality constraints. We distinguish between three classes of methods according to the goals for which they may be used.

(i) *deterministic methods*: to determine whether or not a certain constraint is redundant. As an extension of these methods, we can check which constraints of a given set of constraints are redundant and which are not.

(ii) *probabilistic methods*: to check a system for conditions by which possibly some constraints can be identified as redundant or not. The result of these methods are correct, but it is not known *a priori*

which constraints can be identified.

(iii) *heuristic methods*: to check a system for conditions by which, under some assumptions, some constraints can be identified as redundant or not. The results of these methods are conditional upon the validity of the assumptions.

A number of deterministic methods to identify redundant inequality constraints may be derived from the 'turn-over' lemma.

Boot [1962, 1964] proposed to determine the feasibility of the system

$$(4.4.1) \qquad \begin{cases} Ax = a \\ B_i x \leq b_i & \forall i \neq k \\ B_k x > b_k \end{cases}$$

by considering a related system in which the last constraint is replaced by

$$(4.4.2) \qquad B_k x = b_k + \varepsilon \qquad (\varepsilon > 0;\ \text{sufficiently small})$$

Then this equality can be substituted in all other constraints by eliminating some variable. The feasibility of the remaining system can be determined by standard methods.

Thompson *et al.* [1966] tried to improve the computational performance of this method by considering the constraint (4.4.2) as an inequality with slack variable $-\varepsilon$. Then this slack variable is kept in the basis on the same value and the remaining system is solved. In this way some computational improvements are achieved.

But as well as in Boot's original scheme, for every inequality constraint that is to be identified as being redundant or not, the feasibility of a system of linear constraints has to be tested. In practice this means that a linear programming problem has to be solved for every constraint that should be checked for redundancy. To some extent this is true for our method as well. However in our method a number of other constraints may be identified without extra computations, as we are working towards identifying a given constraint.

Furthermore it should be noted that the original system (4.4.1) and the modified system including (4.4.2) are not equivalent with respect to

the redundancy of the constraint $B_k x \leq b_k$. If the modified system is feasible, the original system is feasible too and hence the k-th constraint is not redundant. However the constraint is redundant if the original system (4.4.1) is infeasible and that may be concluded only if the modified system is infeasible for all $\varepsilon > 0$. Finally the choice of a particular $\varepsilon > 0$, which should be small, may cause numerical difficulties.

Since determining the feasibility of the system (4.4.1) is essentially the same as determining $\hat{u}_k = \min\{u_k(x) \mid x \in S_k\}$, these methods, based on the 'turn-over' lemma, require the same computations as the REDINQ algorithm for one fixed k. However if more constraints have to be identified the REDINQ algorithm is clearly superior.

Some deterministic methods to identify redundant constraints have been developed from methods to find all extreme points of a convex polyhedron (*e.g.* Balinsky [1961], Shefi [1969], Mattheis [1973], Greenberg [1975], Holm and Klein [1975]). In these methods all basic feasible solutions (extreme points of the convex polyhedron) are checked for the conditions for theorem 3.1.2. If these hold for some constraint, then this constraint is identified as redundant (step 3 of the REDINQ algorithm). In this way all extreme points are checked. Only if also all feasible bases are checked, the conclusion may be drawn that the unidentified constraints are nonredundant. Clearly the REDINQ algorithm is more general and computationally superior to these methods.

Following an idea of Lisy [1971], Gal [1975 B] presented a deterministic method to identify redundant inequality constraints. The method of Gal [1975 B] concentrates on strictly redundant constraints although weakly redundant constraints are sometimes identified too. The method can be considered as a variant of the STREDINQ algorithm in which step 2 is omited and step 4 stated slightly less general (see also Gal [1978]). Among the constraints that are identified as nonredundant there may be some weakly redundant ones, because step 1 of the algorithm does not distinguish between nonredundant and weakly redundant constraints. Therefore, although quite similar, the method of Gal [1975 B, 1978] is less general than the method introduced here.

In the literature most methods to identify redundant constraints are probabilistic methods.

One of the earliest proposals for a probabilistic method is due to Llewellyn [1964]. Llewellyn gives some rules to identify a special class of redundant constraints, namely those constraints that are redundant by one other constraint and nonnegativity constraints on all variables. The same rules were reintroduced in a different format in Zeleny [1974]. Eckhardt [1971] showed that theorem 3.1.3 provided a generalization of these rules. However it was not recognized that the rules Llewellyn [1964] gave were incorrect; they only hold if restricted to some special cases, *e.g.* all coefficients greater than or equal to zero. In these special cases the rules of Llewellyn [1964] can be shown to be an immediate consequence of our theorem 3.1.2. A detailed treatment is given in Telgen [1977 B].

Zionts [1965] and Thompson *et al.* [1966] introduced the concept of a *definitional constraint*, which is a constraint that satisfies the conditions of our theorem 3.1.1. The method they propose consists of scanning each tableau for a row satisfying these conditions, and for rows in which these conditions can be satisfied after one iteration (corollary 3.1.2). If the tableau exhibits these properties some constraints can be identified.

Thompson *et al.* [1966] describe a Monte Carlo technique, in which situations as described above are constructed by generating random combinations of all rows of the tableau.

Tischer [1968] and independently Brearly *et al.* [1975] give an extensive list of simple methods to identify redundant inequalities. These methods can be seen as an application of the REDINQ algorithm if only one constraint at a time and all bounds on the variables are taken into consideration.

Another probabilistic method has recently been suggested by Boneh and Golan [1979]. Basically in their method a direction is generated randomly in a given feasible solution; then it is determined which constraint is reached first if one follows this direction from the given point. The constraint that is reached first is not redundant (this follows from the 'turn-over' lemma). This procedure is repeated a great number of times and after that all nonidentified constraints are considered to be redundant. Thus the method is always correct in classifying nonredundant constraints, whereas the probability that constraints are classified

incorrectly as redundant decreases as the number of randomly generated directions increases. Since the method of Boneh and Golan [1979] does not use the linearity of the constraints, it is applicable to systems of non-linear constraints as well.

Heuristic methods are applications of rules that are valid only if certain conditions are satisfied. These conditions are not checked *a priori*, either because it is impossible or because it is too laborious. Therefore the results should be validated *aposteriori* or interpreted very carefully.

Dantzig [1955] proposes to use all kinds of experience, intuition, ideas and information to predict which constraints will not be binding in the optimal solution. Then the slack variables of these constraints can be forced into the basis and marked as being no candidate for leaving the basis. These slack variables, or rather these constraints are placed behind a "curtain" where they do not affect the subsequent solution procedure. Of course a similar thing can be done with constraints that are predicted to be binding in the optimal solution; their slack variables are placed behind a curtain from where they are temporarily not allowed to enter into the basis (for a detailed treatment see *e.g.* Orchard-Hays [1968]).

Apart from the fact that this technique may be profitable for the computing speed and storage needed (everything behind the curtains may be stored in secondary memory) it may be used to obtain a good starting solution (crashing) and to select a pivot. It should be noted that more curtains can be used at the same time, separating variables with different probabilities to enter the basis. This may depend on the available *a priori* information, but also on the solution path being followed.

A solution path is called convex if every two dimensional projection of the path orthogonal to each hyperplane corresponding to a constraint, is convex. Zionts [1965] and Thompson *et al.* [1966] proved that, if the solution path is convex, and a variable enters, leaves and reenters the basis in a series of iterations, the variable will be basic in the optimal solution. If a variable leaves, enters and again leaves the basis, then it will have a zero value in the optimal solution. However there is no simple way known to ensure a convex solution path or to check whether the solution path is convex.

5. APPLICATIONS

5.1 Results from literature

The number of redundant constraints and implicit equalities in a problem depends on various factors such as: the kind of problem, experience of the problem and model formulator, size of the problem etc. Therefore in general it is not possible to predict the rate of reduction in the number of constraints that can be achieved in a particular problem.

Very few experimental results on this question are reported in literature. Zionts [1965] performed some computational tests with the probabilistic and heuristic methods he proposed on a number of practical linear programming problems. Size reductions ranging up to 50% were noted, but the time required to obtain these reductions together with the time required to solve the reduced problem usually was not significantly better (and sometimes significantly worse) than the time required to solve the original problem.

Thompson *et al.* [1966] stated that size reductions of 50% are not unusual for small problems and they hoped that results on larger problems would be even more significant. They considered linear programming problems and applied the probabilistic methods they gave to both the primal and the dual problem. With this procedure results on 5 practical problems ranging in size from 9×20 to 55×104, show size reductions ranging from 30% to 75%.

Tischer [1968] mentions a large number of empirical results on practical linear programming problems (production planning). The size reductions with the very simple rules he uses go up to 99%, but these figures are considered to be somewhat unrealistic since nonnegativity constraints and bounds on the variables are explicitly stated as constraints in the original formulation.

However most of the methods used are also described by Brearly *et al.* [1975] to be part of the REDUCE routines available in advanced commercial LP packages such as MPSX/370 and APEX III, in which they are thought to be useful.

Brown [1977] supports this observation by reporting some computational results using a similar set of simple rules, from which he concludes that 'preprocessing' (scanning for size reductions with simple methods) should be done on any problem. This conclusion is based on both practical, computational and (information) theoretical arguments.

All the percentages mentioned so far are conservative since the methods used do not identify all redundant constraints (only probabilistic and heuristic methods were used). However the size reductions will not be that dramatic in all practical problems, but the fact that reductions are almost always possible is agreed upon by many practitioners (Hofmann [1955], Zionts [1965], Thompson *et al.* [1966], Brearly *et al.* [1975], Brown [1977]).

Even more rare than experimental results with methods to identify redundant constraints are applications of these methods to practical problems reported in literature. The author is aware of only three:

Firstly, the probabilistic and heuristic methods of Zionts [1965] are applied to a blast furnace burdening problem, a customer order combination problem (both with favorable result) and to a raw material allocation problem (unfavorable result), all reported in Zionts [1965].

Secondly, the deterministic method of Boot [1962] is used by Van Amerongen [1978] on a number of small machine generated constraint sets for a nonlinear programming problem, arising in the context of network flow scheduling of electricity. In these small problems (25×6) reductions were huge (80% on the average) at reasonable costs.

Thirdly, Boneh and Golan [1979] reported of a problem instance that was amenable only after application of their method. In this problem of 34 linear and 34 quadratic constraints in 25 variables, a size reduction of 45% was achieved.

5.2 Experimental results

For a comparison of methods to identify redundant constraints at least two factors should be regarded: (i) the number of constraints identified as redundant or not and (ii) the effort spent in trying to identify redundant constraints, measured in computer time or number of computations.

The REDINQ algorithm can only be compared to the deterministic method of Boot [1962] (or the modified version of Thompson *et al.* [1966]) with regard to the second factor; the effectiveness of these methods is the same. It will be clear that the REDINQ algorithm is superior to these other methods especially if more than one constraint has to be identified.

The method of Gal [1975 B , 1978] cannot be compared to the methods mentioned above in the same way because it does not always identify weakly redundant constraints in a correct way. Based on this observation we consider the REDINQ algorithm to be superior.

Comparing the REDINQ algorithm to probabilistic methods is not an easy task, since the REDINQ algorithm will generally perform better regarding (i) but worse regarding (ii).

The same reasoning applies to heuristic methods, but the relative simplicity of the heuristic methods (the advantage it has regarding (ii)) should outweigh the extra computations required to check the conditions or the uncertainty inherent to a heuristic method. Since this last aspect is very much problem dependent we concentrate on a comparison of the REDINQ algorithm and probabilistic methods.*

As representatives of the probabilistic methods we chose the methods introduced in Zionts [1965] and Thompson *et al.* [1966] and the methods incorporated in the REDUCE option of the IBM MPSX/370 package, described in Brearly *et al.* [1975].

We programmed the REDINQ algorithm in PL/I using IBM's MPSX features. The experiments with the REDINQ algorithm and the REDUCE option were performed on the IBM 370/158 computer of Technical University Delft using the PL/I optimizing compiler. The probabilistic method of Thompson *et al.* [1966] was programmed in Algol by B.G. Meyerman. Tests with this method were performed on the Control Data Cyber 74-16 of the University of Groningen using a CDC Algol 60 compiler version 3.1.

All problems used in the experiments are linear programming problems

---

* A large scale comparison of many methods to identify redundant constraints is currently in progress (Telgen and Zionts [1980]).

emerging from practice. Some statistics are given in table 5.1.

problem	size[+]	density[×]	kind of problem	source
A	16×20	15.70 %	queueing theory	Kotiah and Steinberg [1977] page 110, reformulated using theorem 3.3.1
B	22×15	10.57 %	production planning	Meyerman [1979]
C	36×23	7.80 %	production planning	Meyerman [1966]
D	24×5	15.26 %	production planning	Tischer [1968] page 323
E	65×26	10.47 %	production planning	Tischer [1968] pp.335-336
F	9×20	55.42 %	diet problem	Dantzig [1963] pp.553-555 quoted from Stigler
G	28×25	6.96 %	production planning	Nijkamp and Spronk [1978]
H	14×12	14.06 %	disbursement problem	Nijkamp and Spronk [1978]

+ number of inequalities × number of nonnegative variables

× including a slack variable for each constraint

Table 5.1. Testproblems

Concerning these testproblems we make the following remarks. Problem A is known to be a numerically difficult problem; in Kotiah and Steinberg [1977] it is reported to have cycled with MPS. Problems G and H have alternative optimal solutions.

The experiments consisted of solving the testproblems with five different strategies:

(i) straightforward use of IBM's MPSX package with the PRIMAL option;

(ii) first apply IBM's REDUCE option and then solve the problem with the PRIMAL option of the MPSX package;

(iii) first find a feasible solution with the MPSX package, then apply the REDINQ algorithm and remove all redundant constraints and after that solve the problem using the PRIMAL option of the MPSX package;

(iv) apply the probabilistic method of Thompson *et al.* [1966] to all intermediate tableaux and remove redundant constraints, while solving the problem with a standard simplex program (Algol 60);

(v) solve the problem with a standard simplex program (Algol 60).

The results of our experiments are given in table 5.2.

		A	B	C	D	E	F	G	H
(i)	iterations phase I	33	0	0	4	0	2	4	7
	iterations phase II	4	7	21	4	28	7	0	0
	CPU time IBM 370/158	5.33	3.71	4.30	4.28	6.04	4.04	3.94	3.56
(ii)	iterations phase I	11	4	8	0	6	2	0	5
	iterations phase II	8	6	17	4	13	7	0	0
	CPU time IBM 370/158	5.71	4.78	5.47	4.67	7.45	4.97	5.13	4.81
	matrix passes	2	3	2	2	2	2	2	2
	free variables	7	1	4	4	4	0	3	4
	redundant constraints	0	4	0	0	0	0	0	0
(iii)	iterations phase I	33	0	0	4	0	2	4	7
	iterations phase II	4	7	14	4	26	6[1)]	0	0
	CPU time IBM 370/158	6.45	4.60	6.72	4.63	34.19	3.98	5.53	4.23
	iterations REDINQ	41	16	37	30	171	12	28	9
	free variables	11	1	4	4	4	0	8	7
	redundant constraints	0	13	1	13	34	0	0	5
(iv)	iterations phase I	16	0	0	4	0	25	14	5
	iterations phase II	2	7	21	7	40	2	0	0
	CPU time Cyber 74-16	3.236	1.145	7.129	1.647	28.581	3.628	5.637	1.140
	free variables	2	0	4	4	0	0	2[2)]	0
	redundant constraints	0	2	0	0	8	0	15[2)]	
(v)	iterations phase I	*	0	0	4	0	25	4	7
	iterations phase II		7	19	7	32	3	0	0
	CPU time Cyber 74-16		0.741	4.775	1.301	21.902	2.102	1.210	0.493

* terminated due to numerical problems

1) differences caused by different reinversion time

2) larger than in row (iii) since the constraint $c^T x \geq z^o$ (z^o is present solution value) is added to every intermediate tableau

Table 5.2. Experimental results

Regarding the CPU times reported here it should be kept in mind that they were obtained in a time-sharing environment. Therefore they may be influenced by external factors and should not be given absolute values. For a fair comparison of the methods the number of extra iterations (or matrix passes) should be taken into consideration as well.

From table 5.2 we see that the number of redundant constraints in the testproblems usually is considerably larger than the number identified by the REDUCE option of MPSX or the method of Thompson *et al.* [1966]. Identifying all redundant constraints by the REDINQ algorithm sometimes reduces the number of phase II iterations, but generally requires a number of extra iterations and consequently more CPU time than solving the problem directly with MPSX.

Whether or not the identification of all redundant constraints is worth this extra time (effort) is an open question, that has to be decided upon for each individual problem. However, if an LP problem has to be solved more than once, as is the case in many practical situations, identifying all (or part) of the redundant constraints is more attractive.

The method of Thompson *et al.* [1966] does not identify all redundant constraints, but it may be useful since it requires no extra iterations.

A major disadvantage of both the REDINQ algorithm and the method of Thompson *et al.* [1966] is the fact that they require information about all coefficients of the tableau. Since all modern LP packages use some variant of the product form of the inverse (PFI) simplex algorithm, this is a rather costly demand. Since the REDUCE option does not require this information it has a great advantage above the other two methods. However for special purposes or small problems the REDINQ algorithm may still be attractive as illustrated by our testproblems.

The performance of the REDINQ algorithm is sketched in figure 5.1, where a typical pattern for the relation between iterations of the REDINQ algorithm and the number of constraints identified is given.

A practical variant of the REDINQ algorithm could be to stop the algorithm after some prefixed number of iterations, depending on the amount of time one wants to spend on the identification of redundant constraints.

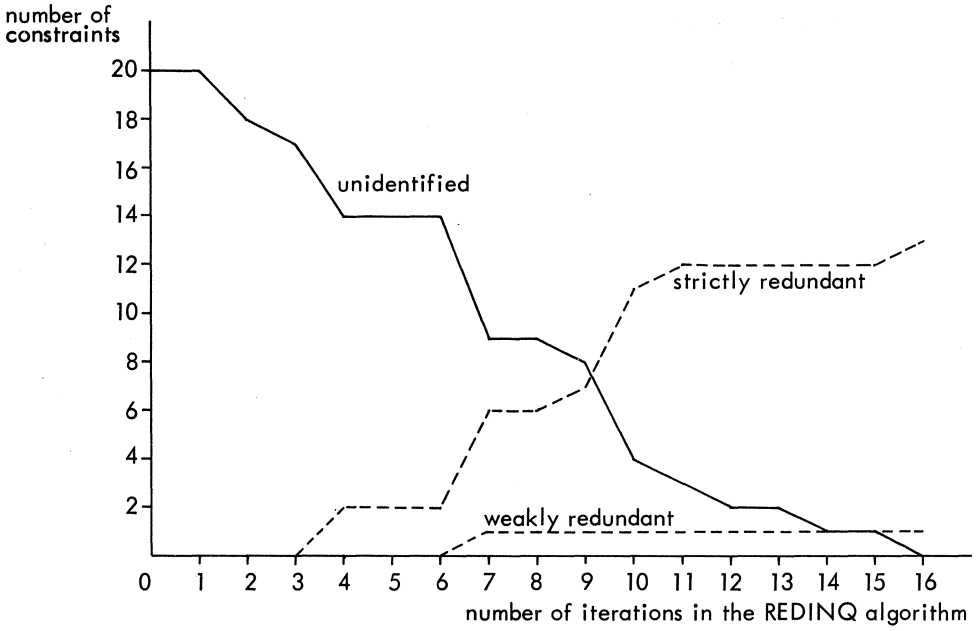

Figure 5.1. The performance of the REDINQ algorithm on testproblem B (37 constraints)

6. RELATED TOPICS

Until now we considered mainly systems of linear constraints. In this chapter we add a linear objective function and consider the *linear programming problem:*

$$(6.1) \qquad \begin{cases} \max c^T x \\ \text{subject to } Ax = a \\ \qquad\qquad Bx \leq b \end{cases}$$

It should be stressed that the concepts of a redundant constraint and an implicit equality are independent of the objective function. Therefore everything developed and proved in the preceding chapters applies to the constraint set of (6.1) as well.

6.1 *Nonbinding constraints*

Apart from redundant constraints which can always be omitted from a linear programming problem, there is another kind of constraints that can be omitted from the linear programming problem for some classes of objective functions. These constraints are called *nonbinding constraints*; some examples are sketched in figure 6.1.

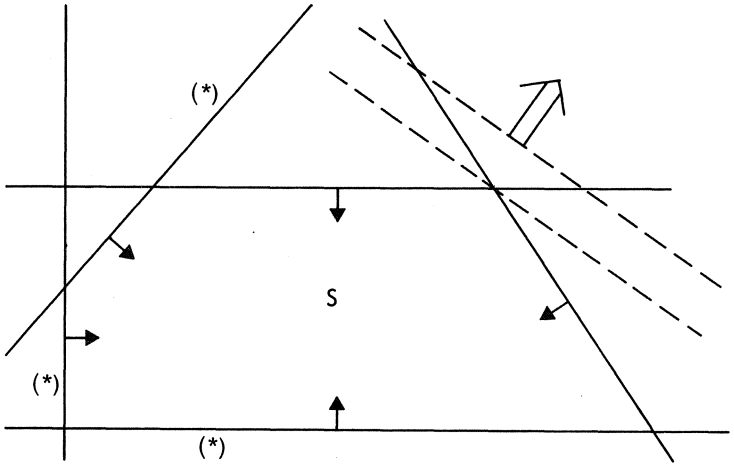

Figure 6.1. Nonbinding constraints; the objective function is indicated by a double arrow ($\Longrightarrow$) pointing in the direction of improvement; nonbinding constraints are indicated by an asterix (*)

In stating a formal definition of a nonbinding constraint we use the notation:

$$\hat{z} \equiv \max\{c^T x \mid x \in S\}$$
$$Z \equiv \{x \in S \mid c^T x = \hat{z}\}$$
$$Z_k \equiv \{x \in S_k \mid c^T x = \hat{z}\}$$
$$Z^k \equiv \{x \in S^k \mid c^T x = \hat{z}\}$$

Definition 6.1.1

The constraint $B_k x \leq b_k$ is *nonbinding* in the system (6.1) if and only if it is nonredundant and $Z = Z_k$.

Definition 6.1.2

The constraint $A_k x = a_k$ is *nonbinding* in the system (6.1) if and only if it is nonredundant and $Z = Z^k$.

Definition 6.1.3

A constraint is *binding* in the system (6.1) if and only if it is neither redundant nor nonbinding.

These definitions are not equivalent to the ones given in literature. Sometimes even other terms are used: *inactive* (Hoffman [1955]), *nondefining* (Llewellyn [1964]) and *relative redundant* (Zimmerman and Gal [1975]); for a detailed motivation and treatment we refer to Telgen [1977 C].

Both inequality and equality constraints can be nonbinding: examples of nonbinding inequality constraints were given in figure 6.1, a nonbinding equality constraint is sketched in figure 6.2.

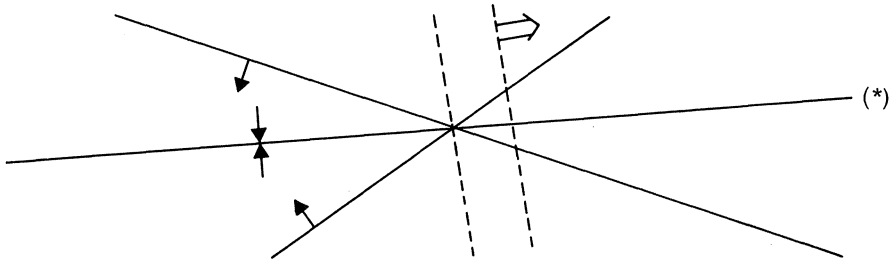

Figure 6.2. A nonbinding equality constraint (indicated by (*))

Note that any nonbinding constraint may be removed from the system: this does not affect the optimal solution(s). But not all nonbinding constraints may be removed at the same time, since by removing one another can become binding. This can be seen from figures 6.2 and 6.3.

Removing nonbinding constraints may even cause weakly redundant constraints to become binding. Consider figures 6.2 and 6.4.

In conclusion we have four types of constraints:

(a) strictly redundant constraints;

(b) weakly redundant constraints;

(c) nonbinding constraints;

(d) binding constraints.

We refer to (a) ∪ (b) as redundant constraints and to (c) ∪ (d) as nonredundant or *active* constraints (Thompson *et al.* [1966] and Zeleny [1974]

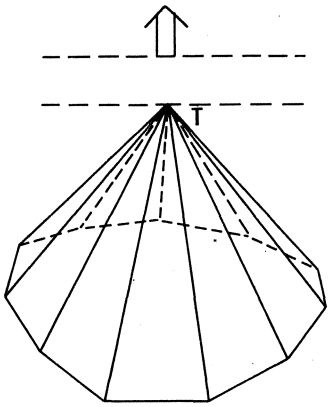

Figure 6.3. The interior of the pyramid (or wigwam) is the feasible region S. Each of the constraints all passing through the point T is nonbinding; however they may not be removed all at the same time

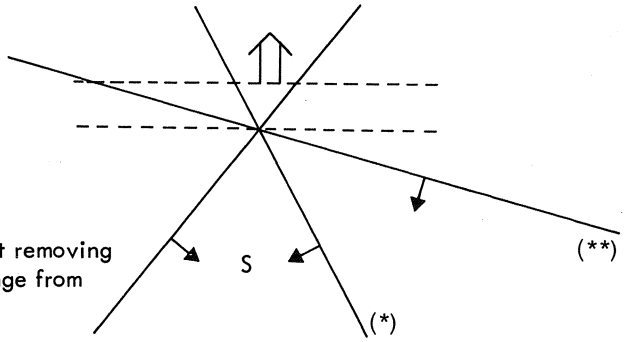

Figure 6.4.

Constraint (*) is nonbinding, but removing it causes constraint (**) to change from weakly redundant to binding

use the term: 'essential'). Further it is worth noting that (c) and (d) are not complementary.

There are no deterministic methods to identify nonbinding constraints without finding the optimal solution to the problem. However all heuristic methods to identify redundant constraints (described in section 4.4) can also be used to identify nonbinding constraints.

A probabilistic method to identify nonbinding constraints if a feasible solution x^o is available, is the following:

identify all redundant constraints in the system:

$$\begin{cases} Ax = a \\ Bx \leq b \\ c^T x \geq c^T x^o \end{cases}.$$

If there were nonbinding constraints in the original system, some of them may have been converted into redundant constraints. In this way nonbinding constraints "at the lower boundary of the feasible region" may be identified. Clearly if $c^T x^o = \hat{z}$ all nonbinding constraints will be identified by this method.

In general however it is not true that constraints "at the lower boundary of the feasible region" are nonbinding as Scolnik [1973] implicitly assumed. The popularity of this assumption is probably due to the fact that a constraint "at the lower boundary of the feasible region" can be recognized rather simply. In the problem (6.1) the constraint $B_k x \leq b_k$ is such a constraint if it forms an obtuse angle ϕ with the objective function, so $\cos \phi < 0$, but this implies $c^T B_k < 0$ which can easily be checked.

A counterexample to the hypothesis that a constraint "at the lower boundary of the feasible region" is nonbinding is provided by the problem:

$$(6.1.1) \qquad \begin{cases} \max \quad x_2 \\ \text{s.t.} \ -x_1 - x_2 \leq -2 \qquad (*) \\ \qquad x_1 + 3x_2 \leq 4 \\ \qquad x_1, x_2 \geq 0 \end{cases}$$

which is shown graphically in figure 6.5.

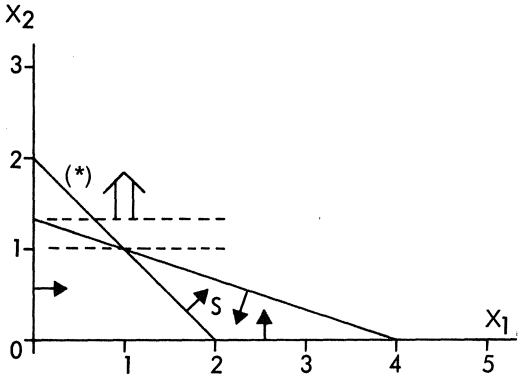

Figure 6.5.
Problem (6.1.1.)

According to $c^T B_k < 0$ the constraint marked by (*) is "at the lower boundary of the feasible region", but it is binding.

By identifying redundant constraints in both the primal and the dual problem and removing them, sometimes nonbinding constraints turn into redundant constraints (Thompson *et al.* [1966]). As an example consider the problem:

$$(6.1.2) \qquad \begin{cases} \max \; x_1 \\ \text{s.t. } x_1 + x_2 \le 2 \\ \qquad x_1 + 2x_2 \le 3 \qquad (*) \\ \qquad x_1, x_2 \ge 0 \end{cases}$$

which is shown graphically in figure 6.6.

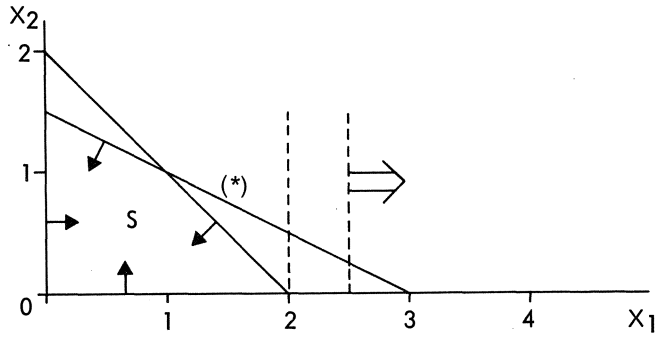

Figure 6.6.
Problem (6.1.2.)

It is easily seen that the constraint marked by (*) is nonbinding (and not redundant). In the dual problem

$$\begin{cases} \min \; 2y_1 + 3y_2 \\ \text{s.t. } y_1 + y_2 \ge 1 \\ \qquad y_1 + 2y_2 \ge 0 \qquad (+) \\ \qquad y_1, y_2 \ge 0 \end{cases}$$

it is trivial that the constraint marked by (+) is redundant. Removing this constraint from the dual problem and thus deleting x_2 in the primal problem, yields the new primal problem:

$$\begin{cases} \max \; x_1 \\ \text{s.t. } x_1 \le 2 \\ \qquad x_1 \le 3 \qquad (*) \\ \qquad x_1 \ge 0 \end{cases}$$

and now the constraint marked by (*) is clearly redundant.

Thus, by identifying and removing redundant constraints in both the primal and the dual problem some nonbinding constraints may be identified. It would be an interesting topic for further research to check whether this property can be used to solve the linear programming problem, as it did in our example.

Finally, by using the Tucker formulation of the linear programming problem (6.1):

$$\begin{cases} Ax = a \\ Bx \leq b \\ A^T u + B^T v = c \\ v \geq 0 \\ a^T u + b^T v \leq c^T x \end{cases}$$

some constraints will be converted from nonbinding into redundant constraints, so they may be identified as such.

6.2 Primal-dual relations

Considering linear programming problems instead of systems of linear constraints, allows us to study redundancy in the dual pair of linear programming problems. Of course everything developed so far for the primal problem can be applied to the dual problem too, thus enlarging the scope of the theory and methods considerably.

But what does it imply for the dual problem if a constraint is redundant in the primal problem and vice versa? Here we list some results on questions like this one assuming that a finite optimal solution to the linear programming problem exists (see Charnes *et al.* [1962] for the relation between boundedness of the feasible region and redundancy in the primal-dual context).

In any solution to the linear programming problem (and thus in the optimal solution too) a strictly redundant inequality in the primal problem has $u_k > 0$, which implies by the complementary slackness theorem (see *e.g.* Dantzig [1963]), that the corresponding dual variable $v_k = 0$ in the optimal solution. Hence this variable may be deleted from the dual problem. Another way to interprete this result is to say that $v_k \geq 0$ is an implicit equality in the dual problem.
By a similar reasoning we can show that implicit equalities in the primal problem correspond to redundant nonnegativity constraints on the dual variables.

For weakly redundant constraints and redundant equality constraints we cannot derive such a strong result. In these cases we know only that there exists an optimal solution to the dual problem, in which the values of the dual variables corresponding to these constraints are zero. This is easily seen from the fact that the problem has the same optimal solution with or without these redundant constraints. However we cannot guarantee that there does not exist another optimal solution with nonzero values for the corresponding dual variables.

By definition dropping nonbinding constraints from the problem does not change the set of optimal solutions. Therefore, just like for weakly redundant constraints, there exists an optimal solution to the dual problem such that the dual variables corresponding to nonbinding constraints in the primal problem, have a zero value.

Finally, binding constraints in the primal problem do not seem to have a clear-cut counterpart in the dual problem, since the corresponding

dual variables may have both zero and nonzero values. Only if we assume that both the primal and the dual problem have a unique optimal solution, we may say that the dual variables corresponding to binding constraints in the primal problem have a nonzero value in the optimal solution.

7. CONCLUSION

In the preceding chapters we developed a new and complete theory of redundancy. We introduced the concepts of an implicit equality and a minimal representation. We proved that a minimal representation is obtained by removing all implicit equalities and redundant constraints from a system.

Furthermore we introduced some new and general methods to identify both redundant constraints and implicit equalities. Preliminary computational experience shows that these methods perform relatively well.

What is the consequence of these developments? The question whether or not redundant constraints should be identified and removed from a given linear programming problem cannot be answered in general. It is useful to identify redundant constraints if the resulting simplification outweighs the effort spent in identifying them. However both the effort spent and the resulting simplification cannot be accurately estimated in advance. Therefore the fruitfulness of scanning for redundant constraints is in practice an open question.

An approach that might provide some more insight into this problem, is through computational complexity theory. That is the topic of the second part of this work.

The question whether or not a minimal representation of the feasible region should be determined cannot be answered in general too. Moreover since the minimal representation is not unique, there might be some preference for one minimal representation above another one *e.g.* based on computational arguments. Therefore one has to decide for each individual system whether or not to determine a minimal representation.

However in any case the theory and methods developed here enable us for theoretical purposes to assume that a system is in a minimal representation without loss of generality.

part 2

LINEAR PROGRAMS

8. INTRODUCTION

The development of linear programming started about thirty years ago when G.B. Dantzig and others formulated the *simplex method*. Various surveys confirm that it is still the operations research technique most widely used in practice. Its fields of application range from oil refinery management to hospital diet planning; problems with thousands of variables and constraints are solved routinely by sophisticated commercial codes.

Beyond any doubt the simplex method and linear programming have been eminently successful from a practical point of view. It is therefore perhaps surprising that various fundamental theoretical questions concerning linear programming have remained open for a very long time. Several of these questions are related to the inherent computational complexity of linear programming problems and the surprising success of the simplex method in solving these problems.

The recent spectacular development by L.G. Khachian of a new method for linear programming motivates the reexamination of these questions. Our starting point will be the theoretical quality of the simplex method. Generally, algorithms for combinatorial problems such as linear programming have been labeled *good* when the computational effort required to solve them is bounded by a polynomial function of problem size. In Chapter 9 we give a more precise formulation of this criterion; it has turned out to be a very satisfactory one, both from a theoretical and from a practical point of view, and problems for which such a good algorithm exists can properly be labeled *easy* ones. In the same section we recall that in spite of its practical performance the simplex method is not a good algorithm in this formal sense; for all the major variations on the method a class of problems can be constructed for which the number of simplex iterations increases exponentially with problem size. This result does not contradict the empirical observation that on the average the number of iterations in the simplex method is a linear function of problem size. It does underline the fact that no satisfactory explanation for this phenomenon has been put forward so far.

It follows that the simplex method cannot be invoked to justify the inclusion of linear programming among the formally easy combinatorial problems. Would it perhaps be possible to show that a polynomially bounded algorithm does not exist or is unlikely to exist in the case of linear programming? The most promising technique to establish such a result is

provided by the theory of *NP-completeness*. If it could be proved that linear programming belongs to the class of NP-complete problems, this would imply that the problem is computationally equivalent to a host of problems notorious for their computational intractability, such as 0-1 programming and the travelling salesman problem. A polynomially bounded algorithm for its solution could then be used to solve all these notoriously hard problems in polynomial time as well, and hence such an algorithm is very unlikely to exist.

However, there has always been strong circumstantial evidence against the possibility of settling the computational complexity of linear programming in this way. We briefly review this evidence in Chapter 10. Consequently, a certain effort was spent on exploring the possibility that linear programming might be easier than all the hard (NP-complete) problems, yet harder than all the easy (polynomially solvable) ones. In the course of this exploration a large list of problems was assembled that were computationally equivalent to linear programming and hence would share the above property. A few of these problems are reviewed in Chapter 11.

This was the situation when in 1979 Khachian published his *ellipsoidal algorithm*. Its most important theoretical property is that it provides a polynomially bounded solution method for linear programming, at last establishing the formal equivalent of the empirical fact that linear programming is indeed an easy problem. Perhaps not surprisingly, the ellipsoidal method is completely different from the simplex method; it solves a system of linear inequalities by constructing a series of increasingly smaller ellipsoids, whose centers converge to a feasible solution if one exists. This reflects an approach to the linear programming problem that is highly non-combinatorial and closer in spirit to ideas from nonlinear and nondifferentiable optimization. We describe the algorithm in some detail in Chapter 12.

It is already becoming apparent that Khachian's approach could have fascinating further applications within combinatorial optimization. These applications and other theoretical consequences are explored in Chapter 13. In the same chapter we discuss the practical relevance of the new method: is it likely to replace the simplex method in the long run or is Khachian's contribution merely a theoretical one? It is probably too early to give a definite answer to such questions, but in any case a whole new and extremely useful research area has been opened up, and many additional exciting results can be expected in the near future.

9. THE SIMPLEX METHOD

Let us start by introducing some concepts from the theory of computational complexity, without going into the details of deterministic and non-deterministic Turing machines and their language recognition capacities (the interested reader is referred to Aho *et al.* [1974]). These details are not required for the exposition below; a simplified treatment will suffice to explain the main implications of the theory.

An algorithm for a certain problem is termed *'good'* if it solves an instance of that problem in a number of computations that is bounded from above by a polynomial function of the *size of the problem* instance (Edmonds [1965]). The size of a problem instance is defined as the length of the input string, under any reasonable encoding of the problem data *e.g.* a binary one. Informally we say that an algorithm is 'good' if it runs in polynomial time.

This broad definition has attractive theoretical features: it is fairly insensitive to the choice of a particular theoretical model of computation and yet it captures the crucial difference between an efficient algorithm and one that demands an exponentially increasing computational effort. Moreover, the algorithms labelled good by virtue of the above definition have generally had polynomial bounds of low degree and work so well in practice that their positive qualification is amply deserved.

The current standard way to solve the linear programming problem

$$\text{(9.1)} \qquad \text{maximize} \qquad c^T x$$

$$\text{(9.2)} \qquad \text{subject to} \qquad Ax \leq b$$

$$\text{(9.3)} \qquad \text{and} \qquad x \geq 0$$

with $c \in R^n$, $x \in R^n$, $b \in R^m$ and $A \in R^{m \times n}$, is by means of the *simplex method*, developed by Dantzig and others in 1947. The simplex method constructs a path along the edges of the polytype defined by (9.2) and (9.3) in such a way that the consecutive extreme points have nondecreasing objective function values attached to them. In each iteration, the method examines if further improvement in the objective function is attainable by moving from the current extreme point to one of its neighbours; if so, one of these neighbours is selected by means of a pivot selection rule and a pivot

step is executed. To determine the theoretical quality of the simplex method it suffices to analyze the number of steps in the path, because the number of computations carried out for each extreme point is clearly polynomially bounded. Since the number of extreme points is bounded from above by $\binom{n}{m}$, an obvious upper bound* on the number of simplex iterations is $O(n^m)$. This bound is clearly exponential in m.

On the other hand, in practice the number of iterations in the simplex method usually ranges from m to 3m, as verified in many experiments (see *e.g.* Wolfe and Cutler [1963], Quandt and Kuhn [1964]). The difference between practical performance and theoretical upper bound has caused Gale [1969] to speak of 'a large and embarassing gap between what has been observed and what has been proved', which 'has stood as a challenge to workers in the field for twenty years now and remains, in my opinion, the principal open question in the theory of linear computation'. Three years later, in a landmark contribution, Klee and Minty [1972] showed that the above obvious upper bound on the number of iterations may actually be attained on certain classes of linear programming problems. We shall discuss their result in some detail.

The basic construction to show this is extremely simple: it uses the fact that a hypercube can be perturbed slightly so as to contain a path that visits all 2^n vertices and along which the last coordinate x_n is always increasing. If the pivot selection rule is simply one under which any neighbouring vertex that yields an improvement may be chosen, and if the objective function is taken equal to x_n, then we obtain the required counterexample against polynomially bounded behaviour.

The perturbation technique is illustrated for the cases n=2 and n=3 in Figures 1 and 3 respectively. Note that the 3-dimensional example is constructed by first forming the Cartesian product of the two-dimensional example and the interval [0,1], as in Figure 2. It is then further perturbed as in Figure 3 to obtain the required sequence of nondecreasing x_3 coordinates.

---

*The conjecture that a smaller upper bound of $\frac{2}{m}\binom{m+n}{2}$ existed (Saaty [1963]) was proven to be false (Quandt and Kuhn [1964]; Goldman and Kleinman [1964]).

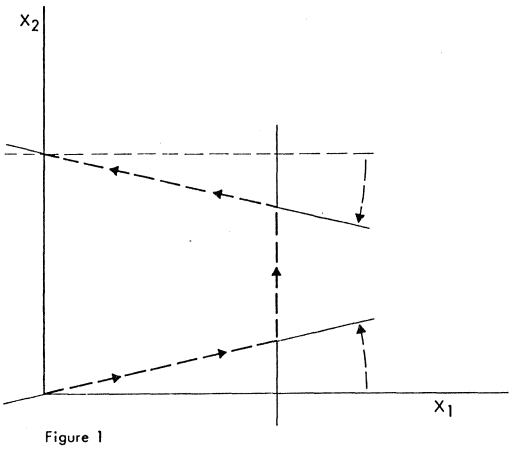

Figure 1

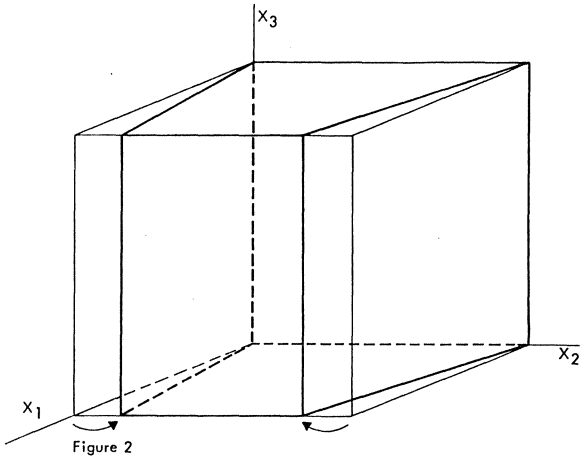

Figure 2

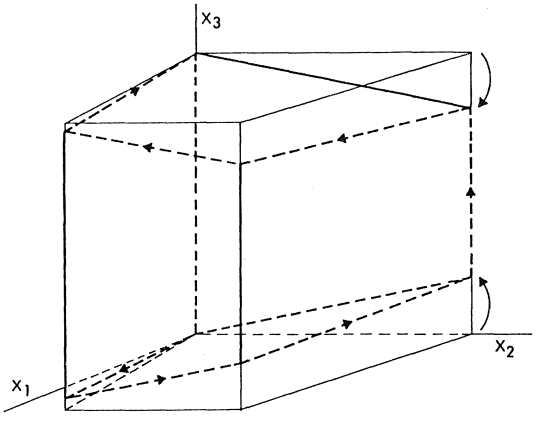

Figure 3

Obviously this particular counterexample does not work for a different choice of pivot selection rule. However, similar, albeit more complicated, constructions have been presented for almost every pivot selection rule that has been suggested so far.

For these cases we introduce the notion of a *reversible* simplex path, which is a path that will be followed in reverse order if we replace the 'max' operator by a 'min' operator and start in the optimal solution under the former. Define the maximum number of steps in a normal and a reversible simplex path for a linear programming problem with n variables and m constraints as M(n,m) and R(n,m) respectively, then clearly

$$M(n,m) \geq R(n,m) \tag{9.4}$$

To prove that M(n,m) is not polynomially bounded it suffices to show that R(n,m) is exponential. To this end Klee and Minty [1972] constructed a class of polyhedra on which

$$R(n+2,\ m+k+1) \geq k\ R(n,m) + k-1 \tag{9.5}$$

Then by induction on n we can prove that

$$\liminf_{n\to\infty} \frac{R(n,m)}{m^{[\frac{1}{2}n]}} \geq \frac{1}{2^{[\frac{1}{2}n]^2}} \tag{9.6}$$

The proof is as follows:

$$L = \liminf_{n\to\infty} \frac{R(n+2,\ m)}{m^{[\frac{1}{2}(n+2)]}} = \liminf_{n\to\infty} \frac{R(n+2,\ 2m+1)}{(2m+1)^{[\frac{1}{2}n]+1}}$$

from (9.5) with k=m it follows that:

$$L \geq \liminf_{n\to\infty} \frac{m\ R(n,m) + m-1}{(2m)\,(2m)^{[\frac{1}{2}n]}} \geq \frac{1}{2^{[\frac{1}{2}(n+1)]}} \liminf_{n\to\infty} \frac{R(n,m)}{m^{[n/2]}}$$

and by induction we have

$$L \geq \frac{1}{2^{[\frac{1}{2}(n+1)]}} \cdot \frac{1}{2^{[n/2]^2}} \geq \frac{1}{2^{[\frac{1}{2}(n+2)]^2}}$$

This completes the proof of (9.6) which in turn implies that $R(n,m) = \Omega(m^{[n/2]})$ and hence $M(n,m) = O(m^{[n/2]})$.

To prove that there is a class of linear programming problems on which (9.5) is true, we have to construct a polyhedron on which it holds. Klee and Minty [1972] use the polyhedron V and its perturbation W sketched in figure 9.4.

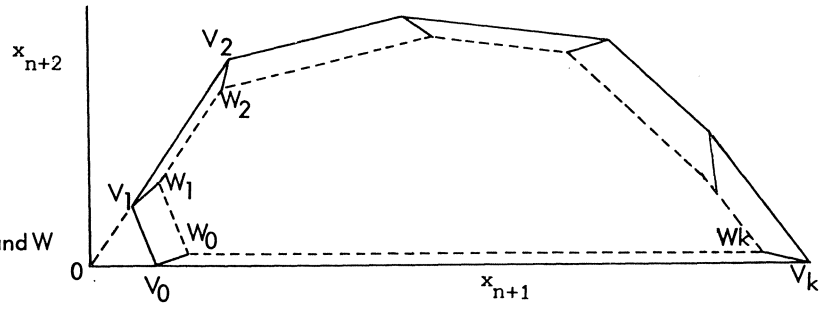

Figure 9.4.
The polytopes V and W

In this figure k should be considered a parameter denoting the number of faces of the polyhedra V and W. Furhter note that the x_{n+1} coordinates of v_i, i even, are larger than those of w_i, while the opposite is true for i odd. Finally the line segments (v_i,v_{i+1}) are parallel to (w_i,w_{i+1}) for all $i=1,\ldots,k-1$. Consider a polyhedron P with a reversible simplex path $p_0,\ldots,p_t$ of length R(n,m) for objective function c^Tx with $c^Tp_0=0$ and $c^Tp_t=1$. We form the Cartesian product V×P, which is perturbed to obtain the (combinatorially equivalent) polyhedron Q. The perturbation is accomplished by defining the x_{n+1} and x_{n+2} coordinates attributed to all points $p\epsilon P$ as

$$(1-c^Tp)v_i + (c^Tp)w_i.$$

This operation is illustrated in Figure 9.5 where P = [0,1].

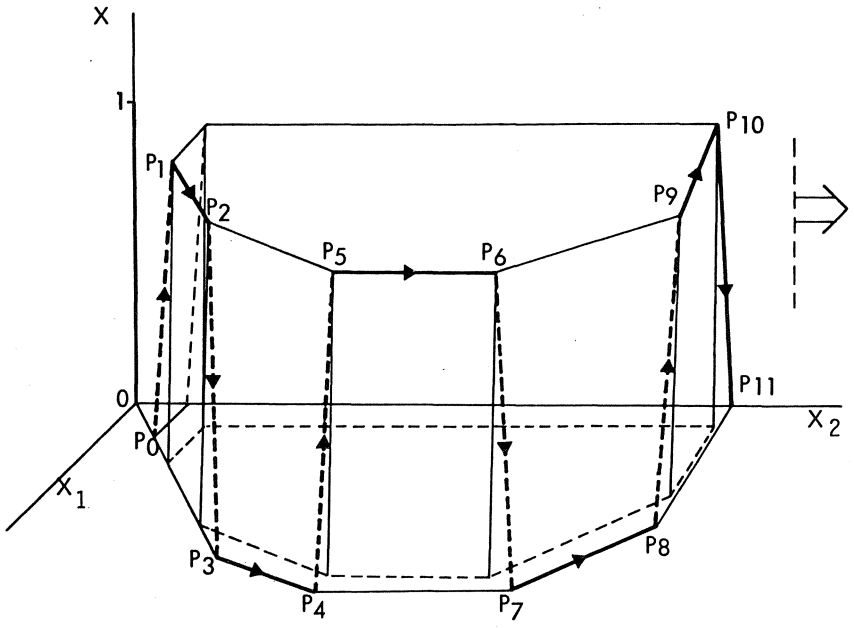

Figure 9.5. The perturbed Cartesian product Vx(0,1) on which the simplex method passes through almost all extreme points

Finally, by suitable choices of the scales of the variables, one can guarantee that the simplex method with the pivot selected to maximize the objective function value improvement per unit change in the variable introduced into the basis, starting in $\begin{pmatrix} P_0 \\ v_1 \end{pmatrix}$, follows the reversible simplex path on P while staying in the (perturbed) v_1 position of the x_{n+1} and x_{n+2} coordinates, i.e. follows the path

$$\begin{pmatrix} P_0 \\ v_1 \end{pmatrix}, \begin{pmatrix} P_1 \\ (1-c^T p_1)v_1 + c^T p_1 w_1) \end{pmatrix}, \ldots \begin{pmatrix} P_t \\ w_1 \end{pmatrix}.$$

Then the simplex method leads to $\begin{pmatrix} P_t \\ w_2 \end{pmatrix}$ and back again to $\begin{pmatrix} P_0 \\ v_2 \end{pmatrix}$ following the reversible simplex path. Since this process repeats itself for t = 1,2,...,k, inequality (9.) follows.

It is not directly clear how to translate this geometrical construction into algebraic terms. However a particularly elegant example of an exponential number of iterations in the simplex method was given by Avis and Chvatal [1978]. The simplex method requires 2^{n-1} steps on the problem:

$$\max \quad [10^{n-1} \quad 10^{n-2} \ \ldots\ldots\ 10^{1} \quad 1] \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

$$\text{s.t} \quad \begin{bmatrix} 1 & 0 & 0 \ \ldots\ldots\ 0 \\ 2.10^{1} & 1 & 0 \\ 2.10^{2} & 2.10^{1} & 1 \\ \vdots & \vdots & \vdots \\ 2.10^{n-1} & 2.10^{n-2} & \ldots\ldots\ 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} 1 \\ 10^{2} \\ 10^{4} \\ \vdots \\ 10^{2(n-1)} \end{bmatrix}$$

$$x_1, x_2, \ldots, x_n \geq 0$$

The construction by Klee and Minty [1972] leaves open the possibility that the simplex method with another pivot selection criterion is polynomially bounded. Jeroslow [1973] formulated a counterexample for one of the major candidates: selecting the pivot to maximize the improvement in the objective function value per iteration.

Instead of the polytopes V and W sketched in figure 9.4 slightly different polytopes V' and W' are obtained by introducing parallellograms in between every two trapezoids as sketched in figure 9.6.

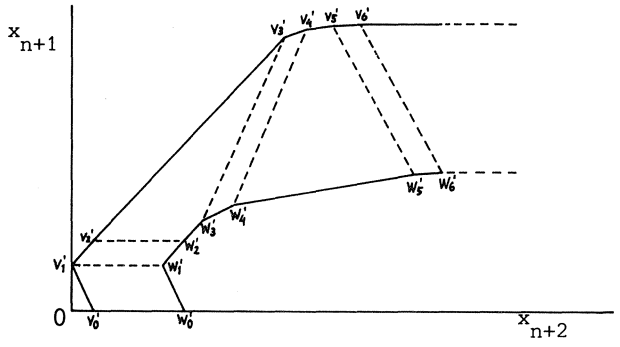

Figure 9.6.
The polytopes V' and W'

Then in the same way in which figure 9.5 was constructed from figure 9.4, Jeroslow constructed the cartesian product V'xP; for the purpose of illustration again we take P = [0,1].

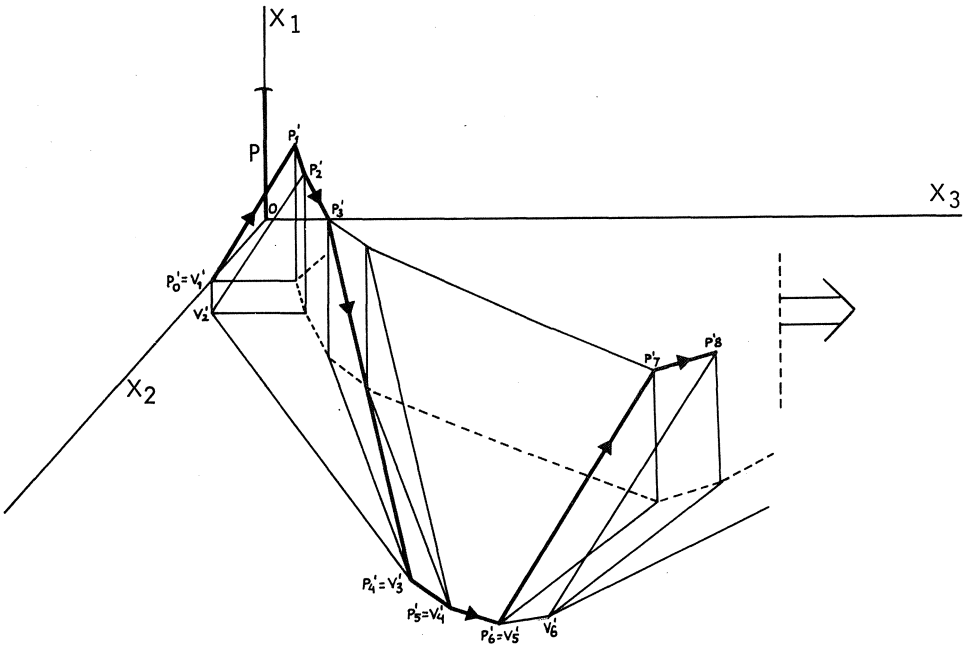

Figure 9.7. The perturbed cartesian product V' x [0,1]

From this figure one can verify that the path is followed as indicated by the heavily drawn lines, if the pivot is selected as as to maximize objective function improvement in each step. Thus we have shown that for this pivot selection rule too a polyhedron can be constructed such that almost all extreme points are visited. In fact Jeroslow [1973] proved that for this pivot selection rule (indicated by a prime)

$$R'(n+2,\ m+4k+3) \geq 2k\ R'(n,m) + 4k$$

This implies that

$$M'(n,m) = \Omega(m^{\lfloor n/2 \rfloor})$$

and this variant of the simplex method is not polynomially bounded either.

In view of relatively good performance (as far as the number of iterations is concerned) of the simplex method with the pivot selection rule of maximizing gradient in the space of all variables, as reported by Quandt and Kuhn [1964] and Crowder and Hattingh [1974] some hope has been held that this method might be polynomially bounded. However, Goldfarb and Sit [1978] were able to construct a counterexample to this conjecture as well; it is somewhat similar to the construction by Klee and Minty [1972], but requires larger perturbations.

Jeroslow [1973] indicated that it is possible to extend his result to still other pivot selection rules such as maximizing objective function improvement per t ($t \geq 1$) iterations.

A pivot selection rule for which no counterexample for polynomial behaviour has been constructed so far, is suggested in Zadeh [1980]. According to that tule, pivots are chosen to introduce variables into the basis that have been basic in the smallest number of previous extreme points.

However, as yet it has only been conjectured and not proved that there is no pivot selection rule for the simplex method that yields a polynomially bounded algorithm. Moreover practical linear programming problems seem to be of a type that can be solved efficiently by the simplex method (Liebling [1973]

None the less, we are forced to conclude that almost all the major variants of the simplex method exhibit exponential behaviour in the worst case, in spite of their excellent average case behaviour. No satisfactory explanation of this discrepancy has been put forward. In principle, it might be possible to prove *expected* polynomial time behaviour of the simplex method by specifying an appropriate probability distribution over all instances of the linear programming problem. This has indeed been attempted, given various definitions of *random polytopes*, without much success (cf. Borgwardt [1978], Sulanke and Wintgen [1972]). In this sense, the 'large and embarrassing gap' alluded to above still exists.

Recently, Dantzig [1980] showed that certain probabilistic assumptions about the behaviour of the simplex method can imply polynomially bounded expected running time, but unfortunately it has not been possible so far to relate all these assumptions to a probability distribution over problem instances.

The only conclusion that we can draw from the $O(n^m)$ bound on the number of simplex iterations is that the linear programming problem is solvable in polynomial time for every fixed constant value of m (or, by duality, of n). Thus, the two-dimensional problem can be solved in $O(m^2)$

time. However this is not optimal as was proved by Shamos and Hoey [1976]. They consider convex polygons as represented by their extreme points in consecutive order. This representation enables one to find the intersection of 2 convex polygons with k sides in O(k) time (Shamos [1975]). It is then easy to prove that the common intersection of m halfplanes can be found in O(m log m) time: let H_i i = 1, ..., m be the halfplanes, then

$$\bigcap_{i=1}^{m} H_i = (H_1 \cap \ldots \cap H_{m/2}) \cap (H_{m/2+1} \cap \ldots \cap H_m)$$

Since both terms in the parentheses are convex polygonal regions of at most m/2 sides, they can be intersected in O(m) time. If T(m) is the time required to form the intersection of n halfplanes this yields

$$T(m) = 2\ T(m/2) + O(m)$$

and therefore

$$T(m) = O(m \log m).$$

Shamos and Hoey [1976] also proved optimality of this result by showing that an algorithm for determining the common intersection of n halfplanes can sort numbers, for which problem an O(m log m) lower bound has been established. The construction is simple: given n real numbers $x_1 \ldots x_m$, let H_i be the halfplane containing the origin defined by a line of slope x_i, that is tangent to the unit circle. The intersection of these halfplanes is a convex polygon, whose successive edges are ordered by slope. Once the intersection polygon is formed, we can immediately read off the x_i in the proper order, so O(m log m) comparisons must have been performed.

Since the linear programming problem can be solved by calculating the objective function value in all (at most m) extreme points, which can be done in linear time, the two-dimensional linear programming problem can be solved in o(m log m) time.

One might be tempted to hope that extending the above method to higher dimensions will yield better upper bounds for linear programming than the simplex method does. However it turns out not to be efficient to determine all extreme points and then calculate the objective function values, since the number of extreme points grows exponentially with the number of variables.

1 . THE COMPLEXITY OF LINEAR PROGRAMMING

In view of the apparent impossibility to solve linear programming problems in polynomial time, it became more and more tempting to aim for a proof that a worst-case exponential solution method such as the simplex algorithm is likely to be unavoidable. A suitable theoretical framework for such a proof was provided by S.A. Cook [1971] and R.M. Karp [1972] in their pathbreaking work on *computational complexity theory*.

The theory of NP-completeness deals with the complexity of *recognition problems*, i.e. questions that require a yes/no answer. This, incidentally, is not a serious restriction: *optimization problems* such as linear programming ((9.1)-(9.3)) can be formulated as recognition porblems by asking for the existence of a feasible solution with objective function value at least equal to a given threshold y. A recognition problem is now said to belong to the class $\mathcal{P}$ of *easy* problems, if for any instance the answer to the yes/no question can be provided in polynomially bounded time. It belongs to the class $\mathcal{NP}$ if a proposed positive answer to the question can be verified for correctness in polynomially bounded time. Thus, the recognition version of linear programming trivially belongs to $\mathcal{NP}$: given $x \in R^n$, it is possible to verify in $O(nm)$ time, whether

$$(11.1) \qquad \begin{cases} c^T x \geq y \\ Ax \leq b \\ x \geq 0 \end{cases}$$

From the above definitions, it is obvious that $\mathcal{P} \subseteq \mathcal{NP}$. To compare the relative difficuly of problems in $\mathcal{NP}$, we next introduce the notion of *problem reducibility*. Problem P' is said to be *reducible* to problem P (notation: $P' \propto P$) if for any instance of P' an instance of P can be constructed in polynomial time such that solving the instance of P will solve the instance of P' as well. Informally, the reducibility of P' to P implies that P' can be considered as a special case of P, so that P is at least as hard as P'. P is now called *NP-complete* if $P' \propto P$ for every $P' \propto \mathcal{NP}$. In that case, P is at least as hard as any other problem in $\mathcal{NP}$. Thus, the NP-complete problems are the most difficult problems in $\mathcal{NP}$, and every problem in $\mathcal{NP}$ is a special case of such an NP-complete problem.

Suppose that it could be proved that linear programming, as formulat-

ed above, is an NP-complete problem. In that case, a polynomially bounded algorithm for its solution could be used to solve not only linear programming but every single problem in NP in polynomial time: for any instance of a problem in NP, one could first construct the corresponding linear programming instance and then solve the latter problem, with both steps being polynomially bounded. However, NP contains many notoriously difficult problems, none of which is likely to be polynomially solvable. The conclusion is that if any problem (and linear programming in particular) is NP-complete, it is unlikely to admit of a formally good algorithm.

One of the nice features of NP-completeness theory is that many of the problems that are notorious in practice for their computational intractability can indeed be proved to be NP-complete, including such classical ones as the 0-1 programming problem and the travelling salesman problem (see Garey and Johnson [1979] for a survey of these results and of the - surprisingly simple - proof techniques). As in the case of good algorithms, this yields overwhelming empirical justification for a theoretical concept: the NP-complete problems can properly be called the truly hard ones. By itself, this already argues against NP-completeness of linear programming. In spite of its theoretical deficiencies, the simplex method works so well that linear programming cannot be called hard from any practical point of view.

The scepsis about the possibility of proving NP-completeness for linear programming can be motivated from a more theoretical point of view as well. The recognition version of linear programming amounts to no more than asking for a feasible solution to a system of *linear inequalities* (11.1). The problem *complementary* to this one would amont to asking for verification that no such feasible solution exists. In view of the *Farkas lemma*, these two questions are computationally equivalent. Now, if linear programming would be NP-complete, a similar equivalence would hold between every other NP-complete problem and its complement. For all of these problems, however, the complements are not even known to belong to NP!

If NP-completeness of linear programming is unlikely and polynomial solvability hard to establish, is there then perhaps a third possibility?

Somewhat surprisingly, the answer to this question is positive: under the (as yet unproven!) assumption that $P \neq NP$, it has been possible to prove that there exists an infinite hierarchy of problems between the class P and the class of the NP-complete ones (Ladner [1975]). Linear programming appeared to be a natural candidate for this ubtermediate position. But as yet only

artificially constructed problems have been shown to be members of this intermediate class.

The classes P and NP are certainly not the only classes of interest. There are for example the class *PSPACE*, which contains all problems solvable in polynomial space, and the class *LOGSPACE*, which contains all problems solvable in space linear in the logarithm of problem size. It is not hard to see that *LOGSPACE* $\subseteq P$, but whether the inclusion is a proper one or not is not yet determined (see Garey and Johnson [1979] for a more detailed treatment). This is of some interest for the complexity of linear programming because Dobkin *et al.* [1979] showed that linear programming is *LOGSPACE*-hard for P, which means that, if linear programming is solvable in space $0 (\log^{k} n)$ for some fixed k, then all problems in P are solvable in space $0 (\log^{k'} n)$ for some fixed k'.

11. LP-EQUIVALENT PROBLEMS

In the course of investigating the computational complexity of linear programming, it became obvious that whatever would be true for linear programming would also be true for a class of problems computationally equivalent to linear programming (equivalent, in the sense of being mutually reducible to each other). We shall call these problems *LP-equivalent.**

The starting point for establishing LP-equivalent problems are the following two formulations of the linear programming problem:

LINEAR PROGRAMMING-OPTIMIZATION (LP-OPT):
Given: An integer (mxn)-matrix A, an integer m-vector b and an integer n-vector c.
Question: Find a rational n-vector x such that $Ax \leq b$ and $c^T x$ is maximal.

LINEAR PROGRAMMING-RECOGNITION (LP-REC):
Given: An integer (mxn)-matrix A, an integer m-vector b, an integer n-vector c and an integer z^0.
Question: Is there a rational n-vector x such that $Ax \leq b$ and $c^T x \geq z^0$?

First note that the integrality assumption is made to exclude irrational coefficients, since they would imply an infinitely large problem size and hence render our definition of a 'good' algorithm meaningless. Clearly rational coefficients are not excluded by this assumption, since the problem can always be rescaled so as to make these integer.

Second we have stated both the *optimization* problem and the *recognition* problem to facilitate proving LP-equivalence for a number of other problems. First we show that these problems themselves are equivalent:

Clearly LP-REC $\propto$ LP-OPT.

To prove LP-OPT $\propto$ LP-REC, suppose we have a 'good' algorithm for LP-REC; then LP-OPT could be solved in polynomial time as follows:

- First check whether $Ax \leq b$ admits of a feasible solution. If not, LP-OPT has no solution.

---

*Dobkin and Reiss [1978] use the confusing term LP-complete problems.

- Next, determine whether the system is unbounded by applying the LP-REC algorithm with $z^0 = (m.q)^{m+n+2} + 1$, where q is the largest absolute value of an integer from A, b or c.
  Since no finite optimal solution exists with objective function value larger than z^0 (Papadimitriou [1979]), feasibility of this system implies an unbounded solution to the LP-OPT problem.
- Finally, determine the objective function value of the optimal solution by a polynomially bounded search over rationals (Papadimitriou [1979], Reiss [1979]). In the same way the values of the variables can be determined.

Now we list some prominent LP-equivalent problems typical of many other problems:

LINEAR INEQUALITIES (LI):
Given: An integer (mxn)-matrix A and an integer m-vector b.
Question: Does a rational n-vector x exist such that $Ax \leq b$?

STRICT LINEAR INEQUALITIES (SLI):
Given: An integer (mxn)-matrix A and an integer m-vector b.
Question: Does a rational n-vector x exist, such that $Ax < b$?

LINEAR PROGRAMMING COMPLEMENT (LPC):
Given: An integer (mxn)-matrix A, an integer n-vector c, an integer m-vector b, and an integer z^0.
Question: Is there no rational n-vector x such that $Ax \leq b$ and $c^T x \geq z^0$?

STRICT REDUNDANCY (ST-RED)
Given: An integer (mxn)-matrix A and an integer m-vector b.
Question: Is the constraint $\sum_{j=1}^{n} a_{1j}x_j \leq b_1$ strictly redundant in the system $Ax \leq b$?

WEAK REDUNDANCY (W-RED):
Given: An integer (mxn)-matrix A and an integer m-vector b.
Question: Is the constraint $\sum_{j=1}^{n} a_{1j}x_j \leq b_1$ weakly redundant in the system $Ax \leq b$?

IMPLICIT EQUALITY (IEQ):
Given: An integer (mxn)-matrix A and an integer m-vector b
Question: Is the constraint $\sum_{j=1}^{n} a_{1j}x_j \leq b_1$ an implicit equality in the system $Ax \leq b$?

Geometrical counterparts to all of these problems can readily be obtained by replacing constraints by hyperplanes and using fundamental duality theory of linear programming. Perhaps not so obvious is the relation with some other geometrical problems:

EXTREME POINT (EP);
Given: A set of points $p^0, p^1, \ldots p^n$ in R^m.
Question: Is p^0 an extreme point of the convex hull of $p^0, p^1, \ldots p^n$?

BOUNDEDNESS (BND);
Given: A set of halfspaces $H_1, \ldots H_m$ in R^n.
Question: Is $H = \bigcap_{i=1}^{m} H_i$ bounded?

POINT-SET SEPARATION (P-S SEP);
Given: Points $p^0, p^1, p^2, \ldots p^n$ in R^m.
Question: Is p^0 *separable* from $p^1, p^2, \ldots, p^n$, *i.e.* does a hyperplane exist such that p^0 is on one side and $p^1, p^2, \ldots p^n$ on the other side of the hyperplane?

A still less obvious LP-equivalent problem is:

DIRECTED TWO COMMODITY FLOW (D2CF);
Given: A directed graph G = (V,E) in which two vertices are denoted sources and two vertices are denoted sinks, with capacities on all edges and an integer f.
Question: Is there a flow from sources to sinks, such that the capacities are not exceeded and the flow is greater than f?

We shall prove that all of the problems above are LP-equivalent. To illustrate our proofs the basic scheme of reductions is given in figure 11.1; an arrow pointing from problem P to problem Q indicates that we shall prove $P \propto Q$.

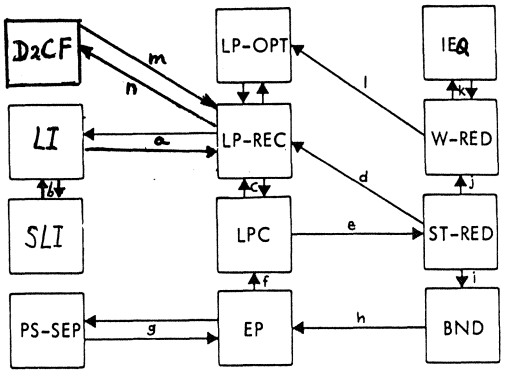

Figure 11.1. Reductions between LP-equivalent problems: symbols refer to the proof in the text

The following reductions suffice to prove LP-equivalence of all problems above:

(a) LP-REC ≡ LI:
trivial

(b) SLI ≡ LI:
it can be shown that there is a polynomially bounded scalar $\alpha(A,b)$ such that $Ax \leq b$ has a solution if and only if

$$Ax < b + \alpha(A,b)$$

has a solution (see Gacs and Lovasz [1979]).

(c) LP-REC ≡ LPC:
trivial, used in phase I of the simplex method.

(d) ST-RED $\propto$ LP-REC:
immediate from the definition of a strictly redundant constraint (Telgen [1977A]).

(e) LPC $\propto$ ST-RED:

In the LPC problem

$$\begin{cases} Ax \leq b \\ c^T x \geq z^0 \end{cases}$$

we may assume that $Ax \leq b$ is feasible (can be checked as LI which is LP equivalent).

Then LPC is infeasible if and only if the constraint $c^T x \leq z^0$ is strictly redundant in the system

$$\begin{cases} Ax \leq b \\ c^T x \leq z^0 \end{cases}$$

To see this note that $c^T x \leq z^0$ is strictly redundant if and only if $c^T x < z^0$ $\forall$ x with $Ax \leq b$.

(f) EP $\propto$ LPC:

The point p^0 is an extreme point in the convex hull of $p^0, p^1, \ldots, p^n$ if and only if there is no $\lambda = (\lambda_1, \ldots \lambda_n) \in R^n$ such that

$$\begin{cases} \sum_{j=1}^{n} \lambda_j p^j \geq p^0 \\ \lambda_j \geq 0 \quad \forall\, j = 1, \ldots, n \\ \sum_{j=1}^{n} \lambda_j \leq 1 \\ -\sum_{j=1}^{n} \lambda_j \leq -1 \end{cases}$$

(g) P-S SEP $\equiv$ EP:

The point p^0 is separable from $p^1, p^2, \ldots, p^n$ if and only if p^0 is an extreme point for the convex hull of $p^0, p^1, \ldots, p^n$.

(h) BND $\propto$ EP:

Denote every H_i as $\{x \in R^n \mid \sum_{j=1}^{n} a_{ij} x_j \leq b_i\}$,

then $H = \{x \in R^n \mid Ax \leq b\}$.

H is bounded if and only if

$$H' = \{x \in R^n \mid Ax \leq 0\}$$

is bounded, which is true if and only if the origin is an extreme point with respect to the points

$$(a_{i1}, \ldots, a_{in})^T \qquad \forall\, i = 1, \ldots, m$$

(i) ST-RED $\propto$ BND:

The constraint $\sum_{j=1}^{n} a_{1j}x_j \leq b_1$ is strictly redundant in the system $Ax \leq b$ if the system

$$\begin{cases} -\sum_{j=1}^{n} a_{1j}x_j \leq -b_1 \\ \sum_{j=1}^{n} a_{ij}x_j \leq b_i & \forall\ i = 2, \ldots, m \end{cases}$$

is infeasible.

Adding a maximizing (null) objective function and dualizing we obtain the problem:

$$\begin{cases} \min \quad -b_1y_1 + \sum_{i=2}^{m} b_iy_i \\ \text{s.t.} \; -a_{1j}y_1 + \sum_{i=2}^{m} a_{ij}y_i \geq 0 & \forall\ j = 1, \ldots, n \\ \qquad y_i \geq 0 & \forall\ i = 1, \ldots, m \end{cases}$$

Since this problem always has a feasible solution (y=0), the former system is infeasible if and only if the system

$$\begin{cases} -b_1y_1 + \sum_{i=2}^{m} b_iy_i \leq 0 \\ -a_{1j}y_1 + \sum_{i=2}^{m} a_{ij}y_i \geq 0 & \forall\ j = 1, \ldots, n \\ \qquad y_i \geq 0 & \forall\ i = 1, \ldots, m \end{cases}$$

is unbounded.

(j) ST-RED $\propto$ W-RED

The constraint $\sum_{j=1}^{n} a_{1j}x_j \leq b_1$ is strictly redundant in the system $Ax \leq b$ if and only if it is not weakly redundant and not binding. Checking the latter is an LI problem and since LI $\equiv$ ST-RED, this suffices to prove ST-RED $\propto$ W-RED.

(k) W-RED $\equiv$ IEQ:

The constraint $\sum_{j=1}^{n} a_{1j}x_j \leq b_1$ is weakly redundant in the system $Ax \leq b$ if and only if the constraint $\sum_{j=1}^{n} a_{1j}x_j \geq b_1$ is an implicit equality in the system

$$\begin{cases} \sum_{j=1}^{n} a_{1j}x_j \geq b_1 \\ \sum_{j=1}^{n} a_{ij}x_j \leq b_i \qquad \forall\ i = 2, \ldots, m \end{cases}$$

(l) W-RED $\propto$ LP-OPT:

Trivial from the definition of weakly redundant constraints.

(m) D2CF $\propto$ LP-REC:

see *e.g.* Ford and Fulkerson [1962].

(n) LP $\propto$ D2CF:

We shall sketch the fairly complex proof due to Itai [1977]. The starting point is the LINEAR EQUALITIES (LE) problem, which can be seen to be LP-equivalent by standard transformations.

An $[\ell,u]$ LE problem is defined as an LE problem with all coefficients being integers between ℓ and u. First, it is shown that LE $\propto$ $[-2,2]$ LE by bitwise decomposition of all constraints, *i.e.* a constraint $\sum_j a_{ij}x_j = b_i$ is rewritten as

$$\sum_j [\sum_k 2^k a_{ik}^j]x_j = \sum_k 2^k b_k^i$$

in which all a_{ik}^j and b_k^i are either -1, 0 or 1. Now we may replace this single constraint by a system of constraints in which the coefficients of all powers of two are equalled:

$$\sum_j a_{ik}^j x_j - 2(y_k - z_k) + (y_{k-1} - z_{k-1}) = b_k^i \qquad \forall\ k$$

The terms in y_k and z_k take into account the transfers from one equation to the next one. If M is the largest coefficent in the matrix the maximal number of constraints in the new system is log M. Since this is polynomial we have LE $\propto$ $[-2,2]$ LE.

It is easy to see that $[-2,2]$ LE $\propto$ $[-1,1]$ LE since we can replace variables x_j with coefficents of ± 2 by $\pm (x_j' + x_j')$ and add the extra constraint $x_j' - x_j'' = 0$.

The reduction to flow problems can be achieved by considering the problem:

HOMOLOGOUS FLOW (HOMFLOW);

Given: A graph G = (V,E), in which one vertex is denoted as source and one as sink, capacity constraints on the edges and some sets of homologous edges *i.e.* sets of edges, through which the

flow has to be equal.

Question: Is there a feasible flow in this network?

Itai [1977] proved [-1,1] LE $\propto$ HOMFLOW by considering the network given in the following figure:

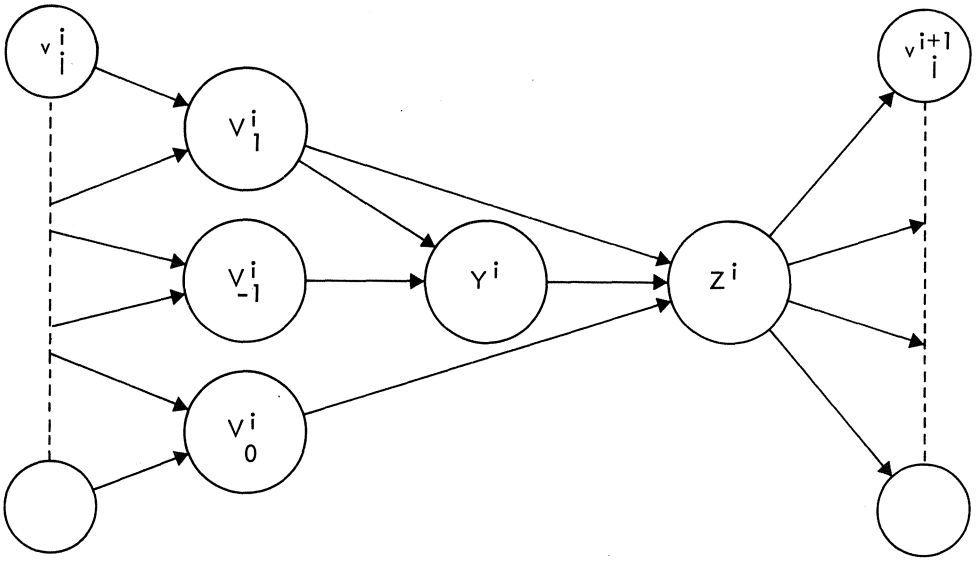

Figure 11.2. The network used in proving [-1, 1] LE $\propto$ HOMFLOW

In this figure a part of a network is shown which corresponds to one constraint i in the [-1,1] LE problem: for every variable j there is a vertex v^i_j which is connected with V^i_0, V^i_1 or V^i_{-1} according to the value of a_{ij}; other vertices and edges are defined as sketched in the figure. An additional vertex $s = z^0$ is introduced as the source of the network, while z^m is the sink.

The edges (V^i_1, y^i) and (V^i_{-1}, y^i) are homologous for all i, as well as the set of edges:

(z^0, v^1_j), (z^1, v^2_j), ..., (z^{k-1}, v^n_j) for all j.

Furthermore the edges (v^i_1, z^i) have a fixed minimal and maximal capacity of b_i for all i.

Then it is easily seen that a solution to HOMFLOW in this network, given by the flows through the edges (z^0, v^1_j) for all j, determines a solution to [-1,1] LE.

By introducing a second commodity and a special kind of edges we can get around the homology requirement. An edge is called *selective* if only one specific commodity may pass through it. Selective edges may be used to simulate homologous edges. If the edges (1,2) and (3,4) are homologous, we apply the construction given in the following figure.

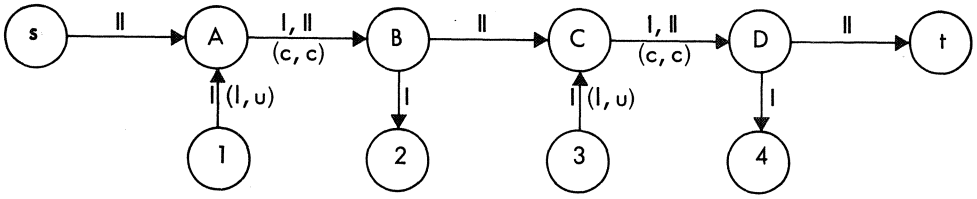

Figure 11.3. Simulating homologous flow with selective edges (c is a large constant)

The commodity that may pass through the edge is indicated above the edge, while the capacity is indicated below the edge (where necessary). Now it is easily verified that the flows from 1 to 2 and from 3 to 4 must be equal. The selective edges in turn can be simulated by introducing an extra source and sink for the selected commodity i and replacing the selective edge (1,2) with capacity (1,u) by the structure given in the following figure.

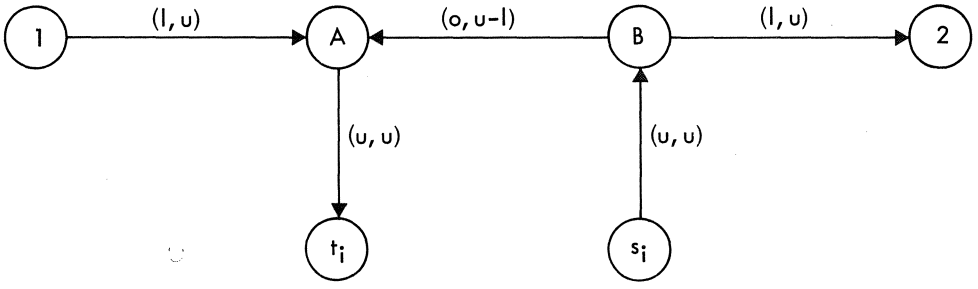

Figure 11.4. Simulating a selective edge (1, 2), through which only commodity i passes

In this way Itai [1977] showed that linear programming reduces to finding a feasible flow in a directed two commodity flow problem with constraints on the capacities of all edges; thus LP $\propto$ D2CF.

Itai [1977] noted that, since the reductions used never rely on the fact that 2 commodities are involved, the result can be extended to m commodities ($m \geq 2$) which means that the directed multi commodity flow

problem is also LP-equivalent.

Single commodity flow problems in a directed network are solvable quite efficiently and membership of P for the directed single commodity flow problem has been proved (Edmonds and Karp [1972]). Therefore it is interesting to ask which linear programming problems may be converted into a single commodity flow problem, such as the transportation and assignment problem, since that would enable us to solve those LP problems in polynomial time.

However it is not always straightforward to check whether a given linear programming problem has an underlying network structure; constraints and variables may have to be combined, added, redefined etc. to reveal the network structure. Usually the only clue as to how this should be done, can be obtained from considering the practical background of the problem.

Recently Bixby and Cunningham [1980] and Musalem [1979] developed algorithms to detect the possibility of conversion and to perform the conversion in polynomially bounded time. The method of Bixby and Cunningham [1980] tries to convert the incidence matrix of the rows to a graphic matroid. If it succeeds, the graphic matroid is used to scale the problem. Musalem [1979] scales the problem to a $(-1,+1,0)$ matrix and then builds a tree, edge by edge, to reveal the partial ordering related to the hidden network structure.

12. THE ELLIPSOIDAL METHOD

The announcement by Khachian [1979] of a polynomially bounded method for linear programming fell upon skeptical ears: in 1973, H.D. Scolnik surprised the participants at the Stanford Mathematical Programming Symposium with a similar claim, and it required quite an effort to detect the fatal flaws in his algorithm (Gay [1974], White [1974]). It took about three months for Khachian's work to reach Western researchers. Again three months later, P. Gacs and L. Lovasz [1979] supplied the proofs that were missing in the original manuscript and established the validity of the algorithm. The new developments attracted a lot of publicity; some amusing background information is provided in Lawler [1980].

In describing Khachian's approach it is most convenient to start from the strict linear inequalities problem: find x such that

$$(12.1) \qquad a_i^T x = \sum_{j=1}^{n} a_{ij} x_j < b_i \quad (i = 1,\ldots,m),$$

whose equivalence to linear programming was noted in the previous section. Khachian's method will be polynomial in the size of a problem instance, measured here by n, m and by the number L of bits needed to store the numerical problem data:

$$L = \sum_{i=1}^{m} \sum_{j=1}^{n} (\log|a_{ij}| + 1) + \sum_{i=1}^{m} (\log|b_i| + 1) + \log nm + 1.$$

It should be observed that running time bounds that are *data dependent* in that they involve the logarithms of numerical problem data occur for other problems as well (such as the linear transportation problem, Edmonds [1965]) and that they are perfectly acceptable from a theoretical point of view.

Now, for the strict linear inequalities problem (12.1), upper and lower bounds on the *volume* of the set of feasible solutions can be provided as follows:

(1) if (12.1) is feasible, a solution can be found within the *hypersphere* $\{x : |x| \leq 2^L\}$;

(2) if (12.1) is feasible, the volume of the set of solutions inside the *hypercube* $\{x : |x_j| \leq 2^L\}$ is at least $2^{-(n+1)L}$.

These results belong to folklore and can be proved using *Cramer's Rule* (see also Gacs and Lovasz [1979]). Note that observation (2) reflects the fact that the set of solutions to (12.1), if nonempty, must have a strictly positive volume.

Khachian's method can now be summarized as follows. In the k - th step of the method, we will have obtained a *(hyper)ellipsoid* such that if there exists a feasible solution to (12.1), it is contained in this ellipsoid. The center of the ellipsoid x_k is tested for feasibility. If x_k is feasible, we stop. If not, we select a constraint from (12.1) that is violated by x_k, and construct a hyperplane parallel to this constraint through x_k. This hyperplane cuts the ellipsoid into two halves, one of which certainly does not contain a feasible solution. We now construct a new ellipsoid that circumscribes the other semi-ellipsoid, and move on to the (k+1) - st step.

Observations (1) and (2) allow us to initialize the method with a unit hypersphere of radius $n^{\frac{1}{2}}2^L$ around the origin and to terminate the process if the volume of the new ellipsoid becomes smaller than $2^{-(n+1)L}$; in the latter case, no feasible solution exists. It follows that the number of steps will be determined by the ratio of the volume of two successive ellipsoids.

For a precise description, assume that the k - th ellipsoid is given as

$$\{x : (x - x_k)^T E_k^{-1} (x - x_k) \leq 1\}. \tag{12.2}$$

Suppose that the current center x_k of (12.2) violates the constraint $a_i^T x < b_i$. The hyperplane parallel to this constraint through x_k is given by $a_i^T x = a_i^T x_k$. To construct the (k+1) - st ellipsoid, consider a second hyperplane parallel to the violated constraint of the form $a_i^T x = a_i^T x_k + \theta$, with $\theta > 0$ chosen in such a way that the hyperplane is tangent to the ellipsoid at the feasible side of $a_i^T x = b_i$. The point of tangency x'_k is given by

$$x'_k = x_k - \frac{E_k a_i}{a_i^T E_k a_i}$$

The new ellipsoid is now uniquely determined by the following four requirements (Figure 12.1):

(a) its center x_{k+1} is located on the line connecting x_k to x'_k;

(b) it touches $a_i^T x = a_i^T x_k + \theta$ in x'_k;

(c) it intersects $a_i^T x = a_i^T x_k$ in the same lower-dimensional ellipsoid as old ellipsoid did;

(d) it has minimal volume over all ellipsoids satisfying (a), (b) and (c).

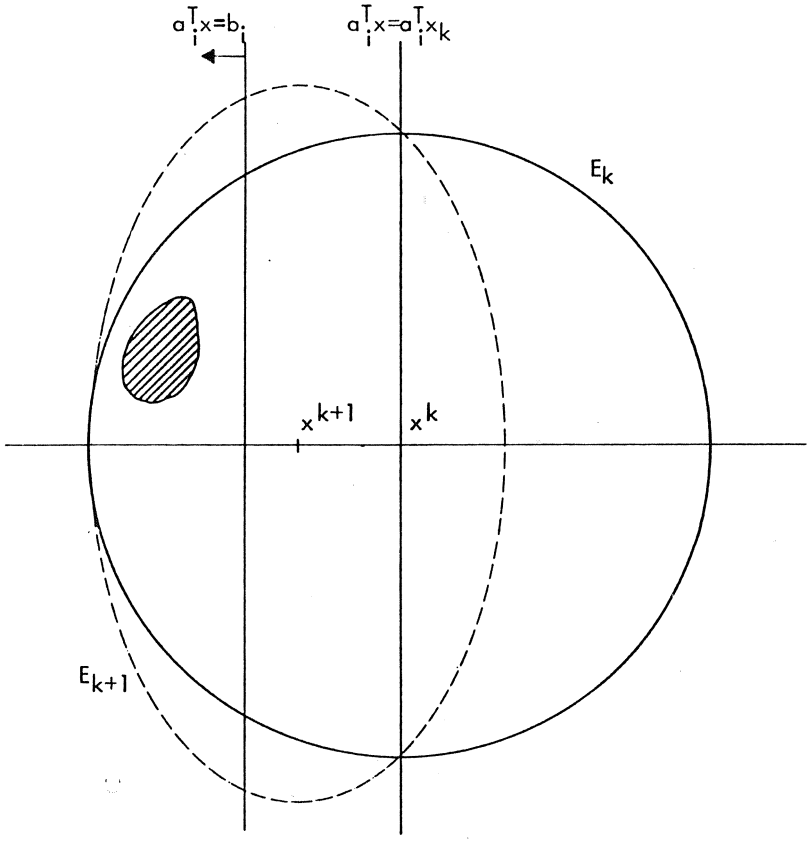

Figure 12.1. The ellipsoidal method.

By making use of the crucial property that problem (12.1) is invariant under affine transformations, it is easy to deduce update formulae expressing x_{k+1} and E_{k+1} in terms of x_k and E_k as a consequence of the above four requirements. The current ellipsoid is first transformed into a unit sphere around the origin; a_i can then be assumed to be a unit vector. After a

simple calculation, followed by the inverse of the above transformation, one obtains:

$$x_{k+1} = x_k - \frac{1}{n+1} \frac{E_k a_i}{\sqrt{a_i^T E_k a_i}} \tag{12.3}$$

$$E_{k+1} = \frac{n^2}{n^2-1} \left(E_k - \frac{2}{n+1} \frac{(E_k a_i)(E_k a_i)^T}{a_i^T E_k a_i} \right) \tag{12.4}$$

The affine transformation referred to above can now also be used to calculate the (affinely invariant) ratio of two successive volumes. Straightforward calculation yields that this ratio is equal to

$$\left(\frac{n^2}{n^2-1} \right)^{(n-1)/2} \frac{n}{n+1} \tag{12.5}$$

Since $n^2/(n^2-1) \leq e^{1/(n^2-1)}$ and $n/(n+1) \leq e^{-1/(n+1)}$, an upper bound on (12.5) is given by

$$e^{(n-1)/(2(n^2-1)) - 1/(n+1)} = e^{-1/(2(n+1))} \approx 1 - \frac{1}{2n} .$$

It is now an easy matter to verify that it takes at most $4(n+1)^2L$ iterations to reduce the volume of the initial hypersphere to less than $2^{-(n+1)L}$. In view of observation (2), this yields a polynomial bound on the number of iterations that can occur in the worst case.

Finally we note that there is one computational issue that we have ignored so far: the precision required to perform the calculations (cf. the square root appearing in (12.3)). Without going into technical details, we simply mention that all above statements about the algorithm remain correct if a precision of e^{-10nL} is maintained throughout the execution (Stone [1980]).

It is a result such as the latter one that immediately raises doubts about the practical usefulness of the algorithm. Maintaining the precision cited above would be an impossible task on a real computer, in spite of the polynomial bound on the number of bits that would be required. Similarly, the worst case bound on the number of iterations of $4(n+1)^2L$ seems impossibly large as well: if $n = 10^3$, $m = 10$ and each coefficient requires no more than 10 bits, about 10^8 iterations could be necessary! Obviously,

polynomial bounds can still be very large indeed (Dantzig [1979]).

Is there any hope that the ellipsoidal method will eventually turn out to be an empirically good one as was the case with its polynomially bounded predecessors? Or are its merits purely theoretical? This question, as well as the theoretical implications of Khachian's result, will be discussed in the final section.

13. CONCLUSION

There is no doubt that the ellipsoidal method is a remarkable one, if only for the fact that it represents an approach to linear programming which is radically different from the simplex method. A typical feature of the latter method is that it is very much a combinatorial one: it can easily be generalized to work over arbitrary ordered fields and has led to the study of purely combinatorial structures such as *oriented matroids* (Bland and Las Vergnas [1976]). The ellipsoidal method, on the other hand, crucially depends on the metric structure of R^n. It is this metric structure, combined with the convexity (rather than the linearity) of the constraints that makes the algorithm work. It is not surprising, therefore, that in a subsequent publication Khachian *et al.* [1979] showed that the method can be extended to solve *convex quadratic programming* problems in polynomial time as well.

The ellipsoidal method itself is closely related to methods of nonlinear programming, and in particular to methods of nondifferentiable optimization: the basic features of the approach can be traced back to earlier work in the latter field by N.Z. Shor [1970A, 1970B], and in particular the algorithm itself is a straightforward implementation of a technique proposed by A. Nemirovskii and D. Yudin [1979]. In its description, the method bears a resemblance to the early *conjugate gradient* methods that may be more than superficial; in spirit, it is close to the well known *projection methods* to solve systems of linear inequalities, in which in each iteration the current point is projected on or just over a violated constraint until feasibility is achieved (Agmon [1954] and Motzkin and Schoenberg [1954]). One crucial weakness of the latter methods has always been that the direction of projection is rather arbitrarily chosen to be orthogonal. Such a choice is not affinely invariant, which results in worst case exponential convergence (Telgen [1980]). The direction chosen in the ellipsoidal method apparently avoids this pitfall.

Although the ellipsoidal method is not combinatorial in nature, it nevertheless is likely to have very interesting theoretical implications for combinatorial optimization problems. This is due to the fact that all that is required to execute a step of the algorithm is identification of a single violated constraint, or more generally, the construction of a hyperplane separating the center of the current ellipsoid from the set of feasible solutions. It may be possible to construct such a hyperplane in polynomially

bounded time, even in cases where the polytope of feasible solutions is only defined implicitly and has an exponential number of facets. Such a situation occurs exactly for many combinatorial problems, for which a good *theoretical characterization* of the convex hull of feasible integer solutions is known. In such cases, the ellipsoidal method may yield the long suspected link between the existence of such a characterization and the existence of a polynomially bounded algorithm (albeit one whose immediate practical usefulness is doubtful). Several examples of a successful attack on combinatorial problems along those lines can be found in Grötschel *et al.* [1980]; see also Karp and Papadimitriou [1980].

It is less obvious whether the ellipsoidal method will yield new, computationally attractive methods to solve linear inequalities in 0-1 variables. In any case, this problem is NP-complete and a polynomial running time bound is extremely unlikely.

What about the practical usefulness of the ellipsoidal method as a tool to solve linear inequality systems or linear programming problems? As mentioned in the previous section, the outlook is gloomy at first glance. At the same time, some immediate improvements to the naive implementation described earlier are also apparent. For instance, rather than enclosing the entire feasible half of the old ellipsoid defined by the hyperplane $a_i^T x = a_i^T x_k$ by the new one, we can restrict ourselves to the part cut off by the original violated constraint $a_i^T x = b_i$ (Figure 13.1). The update formulae (12.3) and (12.4) are easily adapted to the use of such *deep cuts*:

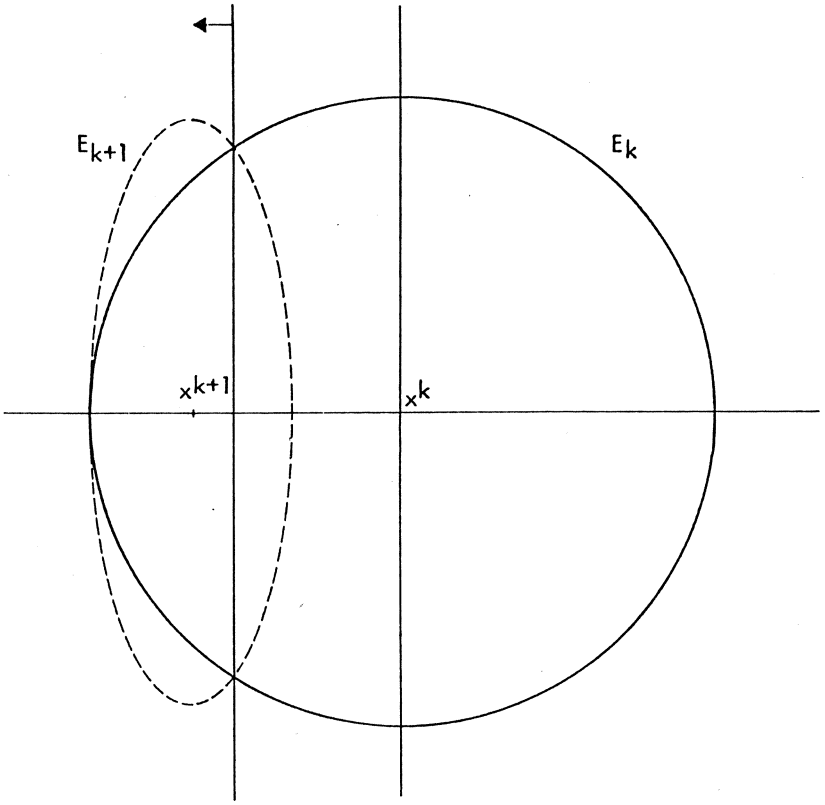

Figure 13.1. A "deep cut".

$$x_{k+1} = x_k - \frac{1-n\alpha_k}{n+1} \frac{E_k a_i}{\sqrt{a_i^T E_k a_i}}$$

$$E_{k+1} = \frac{n^2(1-\alpha_k^2)}{n^2-1} \left(E_k - \frac{2(1-n\alpha_k)}{(n+1)(1-\alpha_k)} \frac{(E_k a_i)(E_k a_i)^T}{a_i^T E_k a_i} \right)$$

with

$$\alpha_k = -\left(\frac{a_i^T x_k - b_i}{\sqrt{a_i^T E_k a_i}} \right) .$$

Although this change does not affect the worst case behavior of the method, it can be expected to yield a considerable improvement in practice.

By way of a second simple improvement, it is also very easy to verify if the current ellipsoid is contained completely in the feasible or in the infeasible halfspace defined by a constraint. In the former case, the constraint is redundant; in the latter case, there are no feasible solutions. Application of the latter idea in practice always allows termination long before the worst case number of iterations is reached. Further improvements may be achievable by using well chosen convex combinations of constraints rather than the original ones (cf. Goldfarb and Todd [1980]).

Although the theoretical precision required by the ellipsoidal method is clearly ridiculous and unnecessary from a practical point of view, the first computational experiments with the method did reveal that it suffered from numerical instability, due to the appearance of very elongated degenerate ellipsoids. Rescaling a problem, if done frequently, will not be of much help, but maintaining E_k in decomposed form has turned out to be quite effective. Two such *decompositions* are now available: in the simplest one, $E_k = J_k^T J_k$, with

$$J_{k+1} = n\sqrt{\frac{1-\alpha_k^2}{n^2-1}}\left(1 - \left[1 - \sqrt{\frac{(n-1)(1+\alpha_k)}{(n+1)(1-\alpha_k)}}\right]\frac{J_k a_i (J_k a_i)^T}{|J_k a_i|^2}\right) J_k .$$

An LDL^T decomposition, with L lower triangular and D diagonal, has also been developed and is given in Goldfarb and Todd [1980].

The straightforward improvements described above already represent substantial improvements in the computational performance of the ellipsoidal method as applied to linear inequality systems, but not yet to the point where the ellipsoidal method can successfully challenge the performance of simplex method on these problems. It is less obvious how to solve linear programming problems by Khachian's approach other than by attacking the complete set of Kuhn-Tucker conditions. A promising approach appears to be to introduce the constraint $c^T x \geq c^T x_k$ instead of a violated one as soon as the center of the ellipsoid becomes feasible (Goldfarb and Todd [1980]). It would then be possible, for instance, to move in the direction of the gradient and even to execute a few simplex steps, thus arriving at a truly hybrid approach. Ultimately, an ε-optimal solution can be guaranteed in polynomial time (Grötschel *et al.* [1980]).

The only conclusion that can be drawn at this point is that the

ellipsoidal method raises as many new questions as it solves old ones: On the theoretical side, one would like to know in addition to the points mentioned above if, for instance, data dependent running time bounds and the appearance of quadratic forms are truly unavoidable in solving linear programming problems efficiently. On the practical side, the challenge is obviously to apply the method in suitable practical circumstances, for instance when the huge initial hypersphere required theoretically can be replaced by a much smaller one by means of *ad hoc* arguments. Certainly, the field of a linear programming has received new impetus from these developments; many more exciting ones can be expected in the near future.

REFERENCES

AGMON, S. (1954), The relaxation method for linear inequalities, Canadian Journal of Mathematics 6, 382-392.

AHO, A.V., J.E. HOPCROFT and J.D. ULLMAN (1974), The design and analysis of computer algorithms, Addison-Wesley.

AMERONGEN, R.A.M. van (1977), Methoden voor vermogensherverdeling in elektrische energievoorzieningssystemen in (potentieel) overbelaste situaties, Afstudeerverslag T.H. Delft, afdeling elektrotechniek.

ANTOSIEWICZ, H.A. (ed.) (1955), Proceedings of the second symposium in linear programming, Washington D.C.

AVIS, D. and V. CHVATAL (1978), Notes on Bland's pivoting rule, Mathematical Programming Study, Vol. 8, 24-34.

BALINSKY, M.L. (1961), An algorithm for finding all vertices of convex polyhedral sets, Journal of Soc. for Industr. Appl. Math. 9 (1), 72-88.

BARTELS, R.H. and G.H. GOLUB (1969), The simplex method of linear programming using LU decomposition, Comm. ACM 12, 266-268.

BENDERS, J.F. (1962), Partitioning procedures for solving mixed variables programming problems, Numerische Mathematik 4, 238-252.

BIXBY, R.E. and W.H. CUNNINGHAM (1980), Converting linear programs to network problems, Mathematics of Operations Research 5 (3), 321-357.

BLAND, R.G. (1977), New finite pivoting rules for the simplex method, Mathematics of Operations Research 2, 103-107.

BLAND, R.G. and M. LAS VERGNAS (1976), Orientability of Matroids, Discussion paper 7633, CORE.

BONEH, A. and A. GOLAN (1979), Constraints redundancy and feasible region boundedness by random feasible points generator, Technical report, Technion, Haifa.

BOOT, J.C.G. (1962), On trivial and binding constraints in programming problems, Management Science 8 (4), 419-441.

BOOT, J.C.G. (1964), Quadratic programming, North-Holland.

BORGWARDT, K.H. (1978), Untersuchungen zur Asymptotik der mittleren Schrittzahl von Simplexverfahren in der linearen Optimierung, Operations

Research Verfahren 28, 332-345.

BREARLY, A.L., G. MITRA and H.P. WILLIAMS (1975), Analysis of mathematical programming problems prior to applying the simplex algorithm, Mathematical Programming 8, 54-83.

BROWN, G.G. (1977), Preprocessing optimization models, Transcript from a lecture held at University of Tennessee, Knoxville.

CHARNES, A., W.W. COOPER and G.L. THOMPSON (1962), Some properties of redundant constraints and extraneous variables in direct and dual linear programming problems, Operations Research 10 (5), 711-723.

COOK, S.A. (1971), The complexity of theorem proving procedures, in: Proceedings of the 3rd ACM annual symposium on the theory of computing, 151-158.

CROWDER, H. and J.M. HATTINGH (1975), Partially normalized pivot selection in linear programming, Mathematical Programming Study 4, 12-25.

DANTZIG, G.B. (1948), Programming in a linear structure, Comptroller USAF, Washington D.C.

DANTZIG, G.B. (1955), Upper bounds, secondary constraints and block-triangularity, Econometrika 23 (2), 174-183.

DANTZIG, G.B. (1963), Linear programming and extensions, Princeton.

DANTZIG, G.B. (1979), Comments on Khachian's algorithm for linear programming, Technical report SOL 79-22, Department of Operations Research, Stanford University.

DANTZIG, G.B. (1980), Expected number of steps of the simplex method for a linear program with a convexity constraint, Technical report SOL 80-3, Department of Operations Research, Stanford University.

DANTZIG, G.B. and P. WOLFE (1960), The decomposition principle for linear programs, Operations Research 8, 101-111.

DOBKIN, D., R.J. LIPTON and S. REISS (....), Linear programming is Log-space hard for P, Information Processing letters 8 (2), 96-97.

DOBKIN, D. and S.P. REISS (1978), The complexity of linear programming, Technical report no. 69, Yale University, Department of Computer Science.

DORFMAN, R., P. SAMUELSON and R.M. SOLOW (1958), Linear programming and

economic analysis, McGraw Hill.

ECKHARDT, U. (1971), Redundante Ungleichungen bei linearen Ungleichungssystemen, Unternehmensforschung 12, 279-286.

ECKHARDT, U. (1975), Theorems on the dimension of convex sets, Linear algebra and its applications 12, 63-76.

ECKHARDT, U. (1977), Semidefinite lineare Komplementärprobleme, Habilitationsschrift RWTH Aachen.

EDMONDS, J. (1965), Paths, trees and flowers, Canadian Journal of Mathematics 17, 449-467.

EDMONDS, J. and R.M. KARP (1972), Theoretical improvements in algorithm efficiency for network flow problems, Journal of the ACM 19, 248-264.

FARKAS, J. (1902), Über die Theorie der einfachen Ungleichungen, J. reine angew. Math. 124, 1-27.

FORD, L.R. and D.R. FULKERSON (1962), Flows in networks, Princeton University Press.

FORREST, J.J.H. and J.A. TOMLIN (1972), Updating triangular factors of the basis to maintain sparsity in the product-form simplex method, Mathematical Programming 2, 263-278.

GACS, P. and L. LOVASZ (1979), Khachian's algorithm for linear programming, Technical report STAN-CS-79-750, Computer Science Department, Stanford University.

GAL, T. (1975a), Redundancy reduction in the restrictions set given in the form of linear inequalities, Progress in Cybernetics and Systems Research, I, 177-179.

GAL, T. (1975b), Zur Identifikation redundanter Nebenbedingungen in linearen Programmen, Zeitschrift für Operations Research 19, 19-28.

GAL, T. (1975c), A note on redundancy and linear parametric programming, Operational Research Quarterly 26 (4), 735-742.

GAL, T. (1978), Redundancy in systems of linear inequalities revisited, Discussion paper no. 19, Fern-Universität, Hagen.

GAL, T. (1979), Postoptimal analysis, parametric programming and related

topics, McGraw Hill.

GALE, D. (1969), How to solve linear inequalities, American Mathematical Montly 76, 589-599.

GAREY, M.R. and D.S. JOHNSON (1979), Computers and intractability: a guide to the theory of NP-completeness, Freeman.

GARFINKEL, R.S. and G.L. NEMHAUSER (1972), Integer programming, John Wiley.

GAY, D.M. (1974), On Scolnik's proposed polynomial-time linear programming algorithm, SIGMAP Newsletter no. 15.

GOLDFARB, D. and W.Y. SIT (1978), Worst case behaviour of the steepest edge simplex method, to appear in Discrete Applied Mathematics.

GOLDFARB, D. and M.J. TODD (1980), Modifications and implementation of the Shor-Khachian method for linear programming, Technical report 446, School of Operations Research and Industrial Engineering, Cornell University.

GOLDMAN, A.J. and D. KLEINMAN (1964), Examples relating to the simplex method, Operations Research 12, 159-161.

GOMORY, R.E. (1958), Essentials of an algorithm for integer solutions to linear programs, Bull. Amer. Math. Soc. 64 (5), 275-278.

GREENBERG, H. (1975), An algorithm for determining redundant inequalities and all solutions to convex polyhedra, Numerische Mathematik 24, 19-26.

GRÖTSCHEL, M., L. LOVASZ and A. SCHRIJVER (1980), The ellipsoid method and its consequences in combinatorial optimization, Technical report, Bonn University.

HADLEY, G. (1962), Linear programming, Addison-Wesley.

HOFFMAN, A.J. (1955), How to solve a linear programming problem, in: H.A. Antosiewicz (ed.) (1955), 397-423.

HOLM, S. and D. KLEIN (1975), Size reduction of linear programs with special structure, working paper, Odense University.

ITAI, A. (1977), Two commodity flow, Technical report no. 103, Technion, Israel Institute of Technology, Computer Science Department.

JEROSLOW, R.G. (1973), The simplex algorithm with the pitvot rule of maximizing criterion improvement, Discrete Mathematics 4, 367-377.

JEROSLOW, R.G. (1975), Some relaxation methods for linear inequalities, Management Sciences Research, Report no. 366 R, Carnegie-Mellon University.

KANTOROVICH, L. (1939), Mathematical methods in the organization and planning of production, reprinted in Management Science (1960) 6, 366-422.

KARP, R.M. (1972), Reducibility among combinatorial problems, in: Miller and Thatcher (eds), Complexity of computer computations, Plenum Press.

KARP, R.M. and C.H. PAPADIMITRIOU (1980), On linear characterizations of combinatorial optimization problems.

KARWAN, M.J., J. TELGEN and S. ZIONTS (1981), Redundancy in Mathematical Programming, Springer Verlag.

KELLY, J.E. (1963), The cutting plane method for solving convex programs, J. Soc. Ind. Appl. Math. 8 (4), 703-712.

KHACHIAN, L.G. (1979), Polynomial algorithm for linear programming, Doklady Akademiia Nauk USSR, Mathematika 244 (5), 1093-1096. English translation in: Soviet Mathematics Doklady 20 (1) (1979), 191-194.

KHACHIAN, L.G., M.K. KOZLOV and S.P. TARASOV (1979), Polynomial Solvability of convex quadratic programming, Doklady Akademiia Nauk, USSR, 248 (5). English translation in: Soviet Mathematics Doklady 20 (5) (1979).

KLEE, V. and G.J. MINTY (1972), How good is the simplex algorithm? in: O. Shisha (ed.), Inequalities IV, Academic Press, 159-175.

KOOPMANS, T.C. (1951), Activity analysis of production and allocation, Cowles commission monograph 13, John Wiley.

KOTIAH, T.C.T. and D.I. STEINBERG (1977), Occurrences of cycling and other phenomena arising in a class of linear programming models, Comm. of the A.C.M. 20 (2), 107-112.

KUHN, H.W. and A.W. TUCKER (eds) (1956), Linear inequalities and related systems, Princeton.

KÜNZI, H.P. and H. TSCHACH (1967), Numerische Betrachtungen zur linearen Optimierung, Operations Research Verfahren, III, 270-285.

LADNER, R.E. (1975), On the structure of polynomial time reducibility, Journal of the ACM 22, 155-171.

LAWLER, E.L. (1976) Combinatorial optimization: networks and matroids, Holt, Rinehart and Winston.

LAWLER, E.L. (1980), The great mathematical sputnik of 1979, to appear in The Sciences.

LIEBLING, T.M. (1973), On the number of iterations of the simplex method, Operations Research Verfahren 17, 248-264.

LISY, J. (1971), Metody pro nalezini redundantnich omezeni v ulohach linearniho programovani, Ekonomicko Matematicky Obzor 7 (3), 285-298.

LLEWELLYN, R.W. (1964), Linear programming, Holt, Rinehart and Winston.

LUENBERGER, D.G. (1973), Introduction to linear and non-linear programming, Addison-Wesley.

MATTHEIS, T.H. (1973), An algorithm for determining irrelevant constraints and all vertices in systems of linear inequalities, Operations Research 21, 247-260.

MEYERMAN, G.L. (1966), Betekenis van een aantal cultuurtechnische factoren voor de ontwikkelingsmogelijkheden van veenkoloniale akkerbouw-bedrijven, Dissertation, Agricultural University Wageningen.

MEYERMAN, B.G. (1979), Private communication.

MOTZKIN, T.S. (1936), Beiträge zur Theorie der linearen Ungleichungen, Dissertation, Basel.

MOTZKIN, T.S. and I.J. SCHOENBERG (1954), The relaxation method for linear inequalities, Canadian Journal of Mathematics 6, 393-404.

MUSALEM, S. (1979), Converting linear models to network models, Ph.D. Dissertation, UCLA.

NEMIROVSKII, A. and D. YUDIN (1979), Complexity of problems and efficiency of methods for minimization (in Russian), Moscow.

NIJKAMP, P. and J. SPRONK (1978), Three cases in multiple criteria decision making: an interactive multiple goal programming approach, Report 7822 Centre for Research in Business Economics, Erasmus University Rotterdam.

ORCHARD-HAYS, W. (1968), Advanced linear programming computing techniques, McGraw Hill.

PAPADIMITRIOU, C.H. (1979), Efficient search for rationals, Information Processing Letters 8 (1), 1-4.

QUANDT, R.E. and H.W. KUHN (1964), On upper bounds for the number of iterations in solving linear programs, Operations Research 12, 161-165.

REISS, S.P. (1979), Rational search, Information Processing Letters 8 (2), 89-90.

ROBINSON, S.M. (1975), Stability theory for systems of inequalities, SIAM J. Numer. Anal. 12 (5), 754-769.

SAATY, T.L. (1963), A conjecture concerning the smallest bound on the number of iterations in linear programming, Operations Research 11, 151-153.

SCOLNIK, H.D. (1973), A new approach to linear programming, SIGMAP Newsletter, no. 14, 35-42.

SHAMOS, M.I. (1975), Geometric complexity, Proceedings of the 7th annual ACM SIGACT Symposium, 224-233.

SHAMOS, M.I. and D. HOEY (1976), Geometric intersection problems, in: Proceedings of the 17th annual symposium on foundations of Computer Science.

SHEFI, A. (1969), Reduction of linear inequality constraints and determination of all feasible extreme points, Dissertation, Stanford.

SHOR, N.Z. (1970a), Utilization of the operation of space dilatation in the minimization of convex functions, Kibernetika 6 (1), 6-12. English translation in: Cybernetics 6 (1) (1970), 7-15).

SHOR, N.Z. (1970b), Convergence rate of the gradient descent method with dilatation of the space, Kibernetika 6 (2), 80-85. English translation in: Cybernetics 6 (2) (1970), 102-108.

SIMMONNARD, M. (1966), Linear programming, translated by W.S. Jewell, Prentice-Hall.

SPRONK, J. and J. TELGEN (1979), A note on multiple objective programming and redundancy, Report no. 7906, Centre for Research in Business Economics, Erasmus University Rotterdam.

STONE, R.E. (1980), Khachiyan's algorithm with finite precision, Working paper 80-1, Department of Operations Research, Stanford University.

SULANKE, R. and P. WINTGEN (1972), Zufällige konvexe Polyeder in In-dimensionalen euklideschen Raum, Periodica Mathematica Hungaria 2, 215-221.

TELGEN, J. (1977a), On redundancy in systems of linear inequalities, Report 7718, Econometric Institute, Erasmus University Rotterdam.

TELGEN, J. (1977b), On R.W. Llewellyn's rules to identify redundant constraints in systems of linear inequalities, Report 7719, Econometric Institute, Erasmus University Rotterdam; also in Zeitschrift für Operations Research 23 (5) (1979), 197-206.

TELGEN, J. (1977c), Redundant and non-binding constraints in linear programming problems, Report 7720, Econometric Institute, Erasmus University Rotterdam.

TELGEN, J. (1979), Overbodige en niet-bindende restricties in lineaire programmeringsproblemen, Bedrijfskunde 51 (2), 168-173.

TELGEN, J. (1980), On relaxation methods for systems of linear inequalities, Working paper 92, College of Business Administration, University of Tennessee, Knoxville.

THOMPSON, G.L., F.M. TONGE and S. ZIONTS (1966), Techniques for removing non-binding constraints and extraneous variables from linear programming problems, Management Science 12 (7), 588-608.

TISCHER, H.J. (1968), Mathematische Verfahren zur Reduzierung der Zeilen- und Spaltenzahl linearer Optimierungsaufgaben, Zentralinstitut für Fertigungstechnik des Maschinenbaues, Karl Marx Stadt.

TSCHERNIKOW, S.N. (1966), Lineare Ungleichungen, VEB Deutscher Verlag der Wissenschaften, translated by H. Weinert.

WHEELWRIGHT, E.L. and B. McFARLANE (1971), The chinese road to socialism, Monthly Review Press.

WHITE, W.W. (1974), A commentary on the new approach to linear programming, SIGMAP Newsletter no. 15.

WOLFE, P. (1955), Reduction of systems of linear relations (abstract), in: H.A. Antosiewicz (ed.) (1955), 449-451.

WOLFE, P. and L. CUTLER (1963), Experiments in linear programming, in:

Graves and Wolfe (eds), Recent advances in mathematical programming.

ZADEH, N. (1980), What is the worst case behaviour of the simplex method?, Technical report 27, Department of Operations Research, Stanford University.

ZELENY, M. (1974), Linear multiobjective programming, Lecture Notes in Economics and Mathematical systems 95, Springer Verlag.

ZIMMERMAN, H.J. and T. (1975), Redundanz und ihre Bedeutung für betriebliche Optimierungsentscheidungen, Zeitschrift für Betriebswirtschaft 45 (4), 221-236.

ZIONTS, S. (1965), Size reduction techniques of linear programming and their application, Ph.D. thesis, Carnegie Institute of Technology.

SUBJECT INDEX

abstraction: 1
active constraint: 68
affine transformations: 105,106
algorithm: 8,41-58,75,78-112
automation: 9
average case: 88

basic feasible solution: 16-18,24,25, 41-43,45,46,48-50,52,56
basic solution: 10,15,16,24,25,51
basis: 10,12,29,46,55,56,58,85,88
binding constraint: 9,58,68,71,73,74
bounded: 38,39
boundedness: 95-98
branch and bound: 10

cardinality: 42,46
combinatorial optimization: 78,79,108
complementary problem: 91
complementary slackness: 73,74
computational complexity: 3,4,75,78-112
computer science: 3,4
conjugate gradient method: 108
crashing: 58
criterion function: 16
curtain: 58
cutting plane method: 10
cycling: 10,42

decomposition: 10,111
deep cut: 109-111
definitional constraint: 57
degenerate solution: 17,18,24,42,45,111
deterministic method: 54-56,61,69
dimension: 22,27,29,32,34,35,37-39
directed two commodity flow: 95,99-102
duality: 8,10,19,26,59,71,73,74,95,98
dynamic programming: 1

'easy' problem: 78,79,90
ellipsoidal method: 79,103-112
equality constraint: 6,7,20-22,35,36, 39,40
equivalent (polynomially): 91,93
essential constraint: 68,69
experiments: 8,59-65,75
extreme point: 56,80,81,87,89,95-97

face: 38
feasibility: 15,19,26,55,56,97,104,108
feasible region: 3,6,7,12,13,27,29, 54,70,73,103
feasible solution: 3,11,12,38,43,46, 57,69,79,93,103,104,109

game theory: 6
geometric programming: 1
geometry: 6
'good' algorithm: 78,91,93

halfplane: 89
halfspace: 6,111
'hard' problem: 79,91
heuristic method: 55,58-61,69
homologous edge: 99-101
homologous flow: 99,100
hyperplane: 6,58,95,108

implicit equality: 3,7,8,22-27,29-32, 34,35,38,40-44,51-54,58,73,75, 95-98
inactive constraint: 68
inconsistency: 9
inequality constraint: 6,7,12-20,22, 30,35,36,38,40
inessential constraint: 7
infeasible: 6,9,14,15,17,25,56,97,111
information theory: 7,10,11,59
instable inequality: 22,40
integer programming: 1,10,79,91
'interior point' lemma: 28-31,34
irrelevant constraint: 7
iteration: 10,41,42,45,46,50,58,63, 64,78,80-89,106

language recognition: 80
linear equalities: 99
linear inequalities: 79,91,94-99,108

linear manifold: 27,33-35,37
linear programming: 1,3,4,6,10,13,15, 16,20,26,41,45,51,54,55,59,61-65, 77-112
linear prog-complement: 91,94-97
linear prog-optimization: 93-96
linear prog-recognition: 91,93-96
LOGSPACE: 92
LP-equivalent: 92,93-103

main redundancy theorem: 27,30,34,35, 38-40,52

mathematical programming: 1-4
matroid: 102,108
minimal representation: 3,8,27-40, 52-54,70
minimal similar representation: 37-40,54
model: 1-4,9,59
MPSX: 61-64

near-cycling: 10
network flow: 60,78,99-102
nonbinding constraint: 8,67-74
nondefining constraint: 68
nondifferentiable optimization: 79,108
nonextremal variable: 38,40,54
nonlinear programming: 1,10,58,60, 79,108
NP: 90,91,92
NP-complete problem: 79,90,91,109
null variable: 38,40,54

objective function: 2,3,10,66-74, 80-89,94,98
operations research: 4,78
optimal solution: 8,11,16,58,62,68, 69,73,74
optimal value: 26

P: 90,92
pivot: 16,18,24,41,42,44-46,48-50, 58,80-89
point-set operation: 95-97
polygon: 89
polyhedron: 56,83,84,87
polynomially bounded: 78-112
polytope: 6,80,109
preprocessing: 59
PRIMAL: 62-69
probabilistic method: 54,56,57,59-61, 69
production planning: 9,59,62
projection method: 108
PSPACE: 92

quadratic programming: 108
queueing theory: 62

random polytopes: 88
recognition problem: 90
REDUCE: 61-64
reducible: 90,95
redundancy: 3,4,6-75,94-98
redundant constraint: 3,4,6-75,96, 98,111
redundant equality constraint: 20,21, 29-31,38,40,51,54
redundant inequality constraint: 13, 14-17,23,30,40,45-50,51,54-57, 96,98,111
relative redundant constraint: 68
reversible simplex path: 83,84,85

selective edge: 101
separable: 95
similarity transformation: 37-39
simplex method: 3,4,6,20,41-43,45,46, 49,61,62,64,78-89,90,91,96, 108,111
size of a problem: 3,59,78,80,92,103
slack: 11,12,16,22,45,55,58,62,73
solution path: 58
solvability: 6
sorting: 88
stability: 22
stochastic programming: 1
storage space: 10,58
strict linear inequalities: 94,96,103
strict redundancy: 13-16,18,25,49,50, 56,68,73,94-98
superfluous constraint: 7

tableau: 12,24,41,43-46,48-50,52, 57,62-64
test problems: 62-64
trivial constraint: 7
Tucker formulation: 72
Turing machine: 80
'turn-over' lemma: 14,15,55-57

unbounded: 9,38,39,73,94

weak redundancy: 13,14,16,56,61,68, 74,94-98
worst case: 88

AUTHOR INDEX

AGMON, S.: 108
AHO, A.V.: 80
AMERONGEN, R.A.M. VAN: 60
AVIS, D.: 85

BALINSKY: M.L.: 56
BARTELS, R.H.: 3
BENDERS, J.F.: 10
BIXBY, R.E.: 102
BLAND, R.G.: 42,108
BONEH, A.: 57,58,60
BOOT, J.C.G.: 6,7,14,55,60,61
BORGWARDT, K.H.: 88
BREARLY, A.L.: 57,59-61
BROWN, G.G.: 59,60

CHARNES, A.: 73
CHVATAL, V.: 85
COOK, S.A.: 90
COOPER, W.W.: 73
CROWDER, H.P.: 88
CUNNINGHAM, W.H.: 102
CUTLER, L.: 81

DANTZIG, G.B.: 3,6,10,12,22,58,62,73, 78,80,88,107
DOBKIN, D.: 92,93
DORFMAN, R.:3

ECKHARDT, U.: 16,28,40,57
EDMONDS, J.: 80,102,103

FARKAS, J.: 6,91
FORD, L.R.: 99
FORREST, J.J.H.: 3
FULKERSON, D.R.: 99

GACS, P.: 96,103,104
GAL, T.: 10,11,16-18,56,61,68
GALE, D.: 81
GAREY, M.R.: 91,92
GARFINKEL, R.S.: 10
GAY, D.M.: 103
GOLAN, A.: 57,58,60
GOLDFARB, D.: 88,111
GOLDMAN, A.J.: 81
GOLUB, G.H.: 3
GOMORY, R.E.: 10
GREENBERG, H.: 56
GRÖTSCHEL, M.: 109,111

HADLEY, G.: 4
HATTINGH, J.M.: 88
HOEY, D.: 89
HOFFMAN, A.J.: 60,68
HOLM, S.: 56
HOPCROFT, J.E.: 80

ITAI, A.: 99,100,101

JEROSLOW, R.G.: 86,87,88
JOHNSON, D.S.: 91,92

KANTOROVICH, L.: 3,6
KARP, R.M.: 90,102,109
KELLY, J.E.: 10
KHACHIAN, L.G.: 78,79,103,104,107, 108,111
KLEE, V.: 81,83,84,86,88
KLEIN, D.: 56
KLEINMAN, D.: 81
KOOPMANS, T.C.: 3
KOTIAH, T.C.T.: 62
KOZLOV, M.K.: 108
KUHN, H.W.: 6,81,88,111
KÜNZI, H.P.: 10

LADNER, R.E.: 91
LAS VERGNAS, M.: 108
LAWLER, E.L.: 103
LIEBLING, T.M.: 88
LIPTON, R.J.: 92
LISY, J.: 56
LLEWELLYN, R.W.: 57,68
LOVASZ, L.: 96,103,104,109,111
LUENBERGER, D.G.: 96,103,104,109,111

MATTHEIS, T.H.: 7,17,56
MCFARLANE, B.: v
MEYERMAN, B.G.: 61,62
MEYERMAN, G.L.: 62
MINTY, G.J.: 81,83,84,86,88
MITRA, G.: 57,59-61
MOTZKIN, TH.S.: 6,108
MUSALEM, S.: 102

NEMHAUSER, G.L.: 10
NEMIROVSKII, A.: 108
NIJKAMP, P. 62

ORCHARD-HAYS, W.: 3,58

PAPADIMITRIOU, C.H.: 94, 109

QUANDT, R.E.: 81,88

REISS, S.P.: 92,93,94
ROBINSON, S.M.: 22

SAATY, T.L.: 81
SAMUELSON, P.A.: 3
SCHOENBERG, I.J.: 108
SCHRIJVER, A.: 109,111
SCOLNIK, H.D.: 70,103
SHAMOS, M.I.: 89
SHEFI, A.: 37-40,54,56
SHOR, N.Z.: 108
SIMMONNARD, M.: 4
SIT, W.Y.: 88
SOLOW, R.M.: 3
SPRONK, J.: 62
STEINBERG, D.I.: 62
STIGLER: 62
STONE, R.E.: 106
SULANKE, R.: 88

TARASOV, S.P.: 108
TELGEN, J.: 11,18,54,57,61,68, 96, 108
THOMPSON, G.L.: 7,10,15,16,55,57-64, 68,71,73
TISCHER, H.J.: 57,59,62
TODD, M.J.: 111
TOMLIN, J.A.: 2
TONGE, F.M.: 7,10,15,16,55,57-64, 68,71
TSCHACH, H.: 10
TSCHERNIKOW, S.N.: 6
TUCKER, A.W.: 6,72,111

ULLMAN, J.D.: 80

WHEELWRIGHT, E.L.: *v*
WHITE, W.W.: 103
WILLIAMS, H.P.: 57,59-61
WINTGEN, P.: 88
WOLFE, P.: 10,54,81

YUDIN, D.: 108

ZADEH, N.: 88
ZELENY, M.: 7,57,68
ZIMMERMAN, H.J.: 11,68
ZIONTS, S.: 7,10,15,16,24,55,57-64 68,71

TITLES IN THE SERIES MATHEMATICAL CENTRE TRACTS

(An asterisk before the MCT number indicates that the tract is under preparation).

A leaflet containing an order form and abstracts of all publications mentioned below is available at the Mathematisch Centrum, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands. Orders should be sent to the same address.

MCT 1 T. VAN DER WALT, *Fixed and almost fixed points*, 1963. ISBN 90 6196 002 9.

MCT 2 A.R. BLOEMENA, *Sampling from a graph*, 1964. ISBN 90 6196 003 7.

MCT 3 G. DE LEVE, *Generalized Markovian decision processes, part I: Model and method*, 1964. ISBN 90 6196 004 5.

MCT 4 G. DE LEVE, *Generalized Markovian decision processes, part II: Probabilistic background*, 1964. ISBN 90 6196 005 3.

MCT 5 G. DE LEVE, H.C. TIJMS & P.J. WEEDA, *Generalized Markovian decision processes, Applications*, 1970. ISBN 90 6196 051 7.

MCT 6 M.A. MAURICE, *Compact ordered spaces*, 1964. ISBN 90 6196 006 1.

MCT 7 W.R. VAN ZWET, *Convex transformations of random variables*, 1964. ISBN 90 6196 007 X.

MCT 8 J.A. ZONNEVELD, *Automatic numerical integration*, 1964. ISBN 90 6196 008 8.

MCT 9 P.C. BAAYEN, *Universal morphisms*, 1964. ISBN 90 6196 009 6.

MCT 10 E.M. DE JAGER, *Applications of distrubutions in mathematical physics*, 1964. ISBN 90 6196 010 X.

MCT 11 A.B. PAALMAN-DE MIRANDA, *Topological semigroups*, 1964. ISBN 90 6196 011 8.

MCT 12 J.A.Th.M. VAN BERCKEL, H. BRANDT CORSTIUS, R.J. MOKKEN & A. VAN WIJNGAARDEN, *Formal properties of newspaper Dutch*, 1965. ISBN 90 6196 013 4.

MCT 13 H.A. LAUWERIER, *Asymptotic expansions*, 1966, out of print; replaced by MCT 54.

MCT 14 H.A. LAUWERIER, *Calculus of variations in mathematical physics*, 1966. ISBN 90 6196 020 7.

MCT 15 R. DOORNBOS, *Slippage tests*, 1966. ISBN 90 6196 021 5.

MCT 16 J.W. DE BAKKER, *Formal definition of programming languages with an application to the definition of ALGOL 60*, 1967. ISBN 90 6196 022 3.

MCT 17 R.P. VAN DE RIET, *Formula manipulation in ALGOL 60, part 1*, 1968. ISBN 90 6196 025 8.

MCT 18 R.P. VAN DE RIET, *Formula manipulation in ALGOL 60, part 2*, 1968. ISBN 90 6196 038 X.

MCT 19 J. VAN DER SLOT, *Some properties related to compactness*, 1968. ISBN 90 6196 026 6.

MCT 20 P.J. VAN DER HOUWEN, *Finite difference methods for solving partial differential equations*, 1968. ISBN 90 6196 027 4.

MCT 21 E. WATTEL, *The compactness operator in set theory and topology*, 1968. ISBN 90 6196 028 2.

MCT 22 T.J. DEKKER, *ALGOL 60 procedures in numerical algebra, part 1*, 1968. ISBN 90 6196 029 0.

MCT 23 T.J. DEKKER & W. HOFFMANN, *ALGOL 60 procedures in numerical algebra, part 2*, 1968. ISBN 90 6196 030 4.

MCT 24 J.W. DE BAKKER, *Recursive procedures*, 1971. ISBN 90 6196 060 6.

MCT 25 E.R. PAËRL, *Representations of the Lorentz group and projective geometry*, 1969. ISBN 90 6196 039 8.

MCT 26 EUROPEAN MEETING 1968, *Selected statistical papers, part I*, 1968. ISBN 90 6196 031 2.

MCT 27 EUROPEAN MEETING 1968, *Selected statistical papers, part II*, 1969. ISBN 90 6196 040 1.

MCT 28 J. OOSTERHOFF, *Combination of one-sided statistical tests*, 1969. ISBN 90 6196 041 X.

MCT 29 J. VERHOEFF, *Error detecting decimal codes*, 1969. ISBN 90 6196 042 8.

MCT 30 H. BRANDT CORSTIUS, *Exercises in computational linguistics*, 1970. ISBN 90 6196 052 5.

MCT 31 W. MOLENAAR, *Approximations to the Poisson, binomial and hypergeometric distribution functions*, 1970. ISBN 90 6196 053 3.

MCT 32 L. DE HAAN, *On regular variation and its application to the weak convergence of sample extremes*, 1970. ISBN 90 6196 054 1.

MCT 33 F.W. STEUTEL, *Preservation of infinite divisibility under mixing and related topics*, 1970. ISBN 90 6196 061 4.

MCT 34 I. JUHÁSZ, A. VERBEEK & N.S. KROONENBERG, *Cardinal functions in topology*, 1971. ISBN 90 6196 062 2.

MCT 35 M.H. VAN EMDEN, *An analysis of complexity*, 1971. ISBN 90 6196 063 0.

MCT 36 J. GRASMAN, *On the birth of boundary layers*, 1971. ISBN 90 6196 064 9.

MCT 37 J.W. DE BAKKER, G.A. BLAAUW, A.J.W. DUIJVESTIJN, E.W. DIJKSTRA, P.J. VAN DER HOUWEN, G.A.M. KAMSTEEG-KEMPER, F.E.J. KRUSEMAN ARETZ, W.L. VAN DER POEL, J.P. SCHAAP-KRUSEMAN, M.V. WILKES & G. ZOUTENDIJK, *MC-25 Informatica Symposium* 1971. ISBN 90 6196 065 7.

MCT 38 W.A. VERLOREN VAN THEMAAT, *Automatic analysis of Dutch compound words*, 1971. ISBN 90 6196 073 8.

MCT 39 H. BAVINCK, *Jacobi series and approximation*, 1972. ISBN 90 6196 074 6.

MCT 40 H.C. TIJMS, *Analysis of (s,S) inventory models*, 1972. ISBN 90 6196 075 4.

MCT 41 A. VERBEEK, *Superextensions of topological spaces*, 1972. ISBN 90 6196 076 2.

MCT 42 W. VERVAAT, *Success epochs in Bernoulli trials (with applications in number theory)*, 1972. ISBN 90 6196 077 0.

MCT 43 F.H. RUYMGAART, *Asymptotic theory of rank tests for independence*, 1973. ISBN 90 6196 081 9.

MCT 44 H. BART, *Meromorphic operator valued functions*, 1973. ISBN 90 6196 082 7.

MCT 45 A.A. BALKEMA, *Monotone transformations and limit laws* 1973. ISBN 90 6196 083 5.

MCT 46 R.P. VAN DE RIET, *ABC ALGOL, A portable language for formula manipulation systems, part 1: The language*, 1973. ISBN 90 6196 084 3.

MCT 47 R.P. VAN DE RIET, *ABC ALGOL, A portable language for formula manipulation systems, part 2: The compiler*, 1973. ISBN 90 6196 085 1.

MCT 48 F.E.J. KRUSEMAN ARETZ, P.J.W. TEN HAGEN & H.L. OUDSHOORN, *An ALGOL 60 compiler in ALGOL 60, Text of the MC-compiler for the EL-X8*, 1973. ISBN 90 6196 086 X.

MCT 49 H. KOK, *Connected orderable spaces*, 1974. ISBN 90 6196 088 6.

MCT 50 A. VAN WIJNGAARDEN, B.J. MAILLOUX, J.E.L. PECK, C.H.A. KOSTER, M. SINTZOFF, C.H. LINDSEY, L.G.L.T. MEERTENS & R.G. FISKER (eds), *Revised report on the algorithmic language ALGOL 68*, 1976. ISBN 90 6196 089 4.

MCT 51 A. HORDIJK, *Dynamic programming and Markov potential theory*, 1974. ISBN 90 6196 095 9.

MCT 52 P.C. BAAYEN (ed.), *Topological structures*, 1974. ISBN 90 6196 096 7.

MCT 53 M.J. FABER, *Metrizability in generalized ordered spaces*, 1974. ISBN 90 6196 097 5.

MCT 54 H.A. LAUWERIER, *Asymptotic analysis, part 1*, 1974. ISBN 90 6196 098 3.

MCT 55 M. HALL JR. & J.H. VAN LINT (eds), *Combinatorics, part 1: Theory of designs, finite geometry and coding theory*, 1974. ISBN 90 6196 099 1.

MCT 56 M. HALL JR. & J.H. VAN LINT (eds), *Combinatorics, part 2: Graph theory, foundations, partitions and combinatorial geometry*, 1974. ISBN 90 6196 100 9.

MCT 57 M. HALL JR. & J.H. VAN LINT (eds), *Combinatorics, part 3: Combinatorial group theory*, 1974. ISBN 90 6196 101 7.

MCT 58 W. ALBERS, *Asymptotic expansions and the deficiency concept in statistics*, 1975. ISBN 90 6196 102 5.

MCT 59 J.L. MIJNHEER, *Sample path properties of stable processes*, 1975. ISBN 90 6196 107 6.

MCT 60 F. GÖBEL, *Queueing models involving buffers*, 1975. ISBN 90 6196 108 4.

*MCT 61 P. VAN EMDE BOAS, *Abstract resource-bound classes, part 1*, ISBN 90 6196 109 2.

*MCT 62 P. VAN EMDE BOAS, *Abstract resource-bound classes, part 2*, ISBN 90 6196 110 6.

MCT 63 J.W. DE BAKKER (ed.), *Foundations of computer science*, 1975. ISBN 90 6196 111 4.

MCT 64 W.J. DE SCHIPPER, *Symmetric closed categories*, 1975. ISBN 90 6196 112 2.

MCT 65 J. DE VRIES, *Topological transformation groups 1 A categorical approach*, 1975. ISBN 90 6196 113 0.

MCT 66 H.G.J. PIJLS, *Locally convex algebras in spectral theory and eigenfunction expansions*, 1976. ISBN 90 6196 114 9.

*MCT 67 H.A. LAUWERIER, *Asymptotic analysis, part 2*, ISBN 90 6196 119 X.

MCT 68 P.P.N. DE GROEN, *Singularly perturbed differential operators of second order*, 1976. ISBN 90 6196 120 3.

MCT 69 J.K. LENSTRA, *Sequencing by enumerative methods*, 1977. ISBN 90 6196 125 4.

MCT 70 W.P. DE ROEVER JR., *Recursive program schemes: Semantics and proof theory*, 1976. ISBN 90 6196 127 0.

MCT 71 J.A.E.E. VAN NUNEN, *Contracting Markov decision processes*, 1976. ISBN 90 6196 129 7.

MCT 72 J.K.M. JANSEN, *Simple periodic and nonperiodic Lamé functions and their applications in the theory of conical waveguides*, 1977. ISBN 90 6196 130 0.

MCT 73 D.M.R. LEIVANT, *Absoluteness of intuitionistic logic*, 1979. ISBN 90 6196 122 X.

MCT 74 H.J.J. TE RIELE, *A theoretical and computational study of generalized aliquot sequences*, 1976. ISBN 90 6196 131 9.

MCT 75 A.E. BROUWER, *Treelike spaces and related connected topological spaces*, 1977. ISBN 90 6196 132 7.

MCT 76 M. REM, *Associations and the closure statement*, 1976. ISBN 90 6196 135 1.

MCT 77 W.C.M. KALLENBERG, *Asymptotic optimality of likelihood ratio tests in exponential families*, 1977. ISBN 90 6196 134 3.

MCT 78 E. DE JONGE & A.C.M. VAN ROOIJ, *Introduction to Riesz spaces*, 1977. ISBN 90 6196 133 5.

MCT 79 M.C.A. VAN ZUIJLEN, *Empirical distributions and rank statistics*, 1977. ISBN 90 6196 145 9.

MCT 80 P.W. HEMKER, *A numerical study of stiff two-point boundary problems*, 1977. ISBN 90 6196 146 7.

MCT 81 K.R. APT & J.W. DE BAKKER (eds), *Foundations of computer science II*, part 1, 1976. ISBN 90 6196 140 8.

MCT 82 K.R. APT & J.W. DE BAKKER (eds), *Foundations of computer science II*, part 2, 1976. ISBN 90 6196 141 6.

MCT 83 L.S. BENTHEM JUTTING, *Checking Landau's "Grundlagen" in the AUTOMATH system*, 1979. ISBN 90 6196 147 5.

MCT 84 H.L.L. BUSARD, *The translation of the elements of Euclid from the Arabic into Latin by Hermann of Carinthia (?) books vii-xii*, 1977. ISBN 90 6196 148 3.

MCT 85 J. VAN MILL, *Supercompactness and Wallman spaces*, 1977. ISBN 90 6196 151 3.

MCT 86 S.G. VAN DER MEULEN & M. VELDHORST, *Torrix I, A programming system for operations on vectors and matrices over arbitrary fields and of variable size*. 1978. ISBN 90 6196 152 1.

*MCT 87 S.G. VAN DER MEULEN & M. VELDHORST, *Torrix II*, ISBN 90 6196 153 X.

MCT 88 A. SCHRIJVER, *Matroids and linking systems*, 1977. ISBN 90 6196 154 8.

MCT 89 J.W. DE ROEVER, *Complex Fourier transformation and analytic functionals with unbounded carriers*, 1978. ISBN 90 6196 155 6.

MCT 90 L.P.J. GROENEWEGEN, *Characterization of optimal strategies in dynamic games*, 1981 . ISBN 90 6196 156 4.

MCT 91 J.M. GEYSEL, *Transcendence in fields of positive characteristic*, 1979. ISBN 90 6196 157 2.

MCT 92 P.J. WEEDA, *Finite generalized Markov programming*, 1979. ISBN 90 6196 158 0.

MCT 93 H.C. TIJMS & J. WESSELS (eds), *Markov decision theory*, 1977. ISBN 90 6196 160 2.

MCT 94 A. BIJLSMA, *Simultaneous approximations in transcendental number theory*, 1978. ISBN 90 6196 162 9.

MCT 95 K.M. VAN HEE, *Bayesian control of Markov chains*, 1978. ISBN 90 6196 163 7.

MCT 96 P.M.B. VITÁNYI, *Lindenmayer systems: Structure, languages, and growth functions*, 1980. ISBN 90 6196 164 5.

*MCT 97 A. FEDERGRUEN, *Markovian control problems; functional equations and algorithms*, . ISBN 90 6196 165 3.

MCT 98 R. GEEL, *Singular perturbations of hyperbolic type*, 1978. ISBN 90 6196 166 1.

MCT 99 J.K. LENSTRA, A.H.G. RINNOOY KAN & P. VAN EMDE BOAS, *Interfaces between computer science and operations research*, 1978. ISBN 90 6196 170 X.

MCT 100 P.C. BAAYEN, D. VAN DULST & J. OOSTERHOFF (eds), *Proceedings bicentennial congress of the Wiskundig Genootschap, part 1*, 1979. ISBN 90 6196 168 8.

MCT 101 P.C. BAAYEN, D. VAN DULST & J. OOSTERHOFF (eds), *Proceedings bicentennial congress of the Wiskundig Genootschap, part 2*, 1979. ISBN 90 6196 169 6.

MCT 102 D. VAN DULST, *Reflexive and superreflexive Banach spaces*, 1978. ISBN 90 6196 171 8.

MCT 103 K. VAN HARN, *Classifying infinitely divisible distributions by functional equations*, 1978. ISBN 90 6196 172 6.

MCT 104 J.M. VAN WOUWE, *Go-spaces and generalizations of metrizability*, 1979. ISBN 90 6196 173 4.

*MCT 105 R. HELMERS, *Edgeworth expansions for linear combinations of order statistics*, . ISBN 90 6196 174 2.

MCT 106 A. SCHRIJVER (ed.), *Packing and covering in combinatorics*, 1979. ISBN 90 6196 180 7.

MCT 107 C. DEN HEIJER, *The numerical solution of nonlinear operator equations by imbedding methods*, 1979. ISBN 90 6196 175 0.

MCT 108 J.W. DE BAKKER & J. VAN LEEUWEN (eds), *Foundations of computer science III*, part 1, 1979. ISBN 90 6196 176 9.

MCT 109 J.W. DE BAKKER & J. VAN LEEUWEN (eds), *Foundations of computer science III*, part 2, 1979. ISBN 90 6196 177 7.

MCT 110 J.C. VAN VLIET, *ALGOL 68 transput*, part I: *Historical review and discussion of the implementation model*, 1979. ISBN 90 6196 178 5.

MCT 111 J.C. VAN VLIET, *ALGOL 68 transput*, part II: *An implementation model*, 1979. ISBN 90 6196 179 3.

MCT 112 H.C.P. BERBEE, *Random walks with stationary increments and renewal theory*, 1979. ISBN 90 6196 182 3.

MCT 113 T.A.B. SNIJDERS, *Asymptotic optimality theory for testing problems with restricted alternatives*, 1979. ISBN 90 6196 183 1.

MCT 114 A.J.E.M. JANSSEN, *Application of the Wigner distribution to harmonic analysis of generalized stochastic processes*, 1979. ISBN 90 6196 184 X.

MCT 115 P.C. BAAYEN & J. VAN MILL (eds), *Topological Structures II*, part 1, 1979. ISBN 90 6196 185 5.

MCT 116 P.C. BAAYEN & J. VAN MILL (eds), *Topological Structures II*, part 2, 1979. ISBN 90 6196 186 6.

MCT 117 P.J.M. KALLENBERG, *Branching processes with continuous state space*, 1979. ISBN 90 6196 188 2.

MCT 118 P. GROENEBOOM, *Large deviations and asymptotic efficiencies*, 1980. ISBN 90 6196 190 4.

MCT 119 F. J. PETERS, *Sparse matrices and substructures, with a novel implementation of finite element algorithms*, 1980. ISBN 90 6196 192 0.

MCT 120 W.P.M. DE RUYTER, *On the asymptotic analysis of large-scale ocean circulation*, 1980. ISBN 90 6196 192 9.

MCT 121 W.H. HAEMERS, *Eigenvalue techniques in design and graph theory*, 1980. ISBN 90 6196 194 7.

MCT 122 J.C.P. BUS, *Numerical solution of systems of nonlinear equations*, 1980. ISBN 90 6196 195 5.

MCT 123 I. YUHÁSZ, *Cardinal functions in topology - ten years later*, 1980. ISBN 90 6196 196 3.

MCT 124 R.D. GILL, *Censoring and stochastic integrals*, 1980. ISBN 90 6196 197 1.

MCT 125 R. EISING, *2-D systems, an algebraic approach*, 1980. ISBN 90 6196 198 X.

MCT 126 G. VAN DER HOEK, *Reduction methods in nonlinear programming*, 1980. ISBN 90 6196 199 8.

MCT 127 J.W. KLOP, *Combinatory reduction systems*, 1980. ISBN 90 6196 200 5.

MCT 128 A.J.J. TALMAN, *Variable dimension fixed point algorithms and triangulations*, 1980. ISBN 90 6196 201 3.

MCT 129 G. VAN DER LAAN, *Simplicial fixed point algorithms*, 1980. ISBN 90 6196 202 1.

MCT 130 P.J.W. TEN HAGEN et al., *ILP Intermediate language for pictures*, 1980. ISBN 90 6196 204 8.

MCT 131 R.J.R. BACK, *Correctness preserving program refinements: Proof theory and applications*, 1980. ISBN 90 6196 207 2.

MCT 132 H.M. MULDER, *The interval function of a graph*, 1980. ISBN 90 6196 208 0.

MCT 133 C.A.J. KLAASSEN, *Statistical performance of location estimators*, 1981. ISBN 90 6196 209 9.

MCT 134 J.C. VAN VLIET & H. WUPPER (eds), *Proceedings international conference on ALGOL 68*, 1981. ISBN 90 6196 210 2.

MCT 135 J.A.G. GROENENDIJK, T.M.V. JANSSEN & M.J.B. STOKHOF (eds), *Formal methods in the study of language*, part I, 1981. ISBN 90 6196 211 0.

MCT 136 J.A.G. GROENENDIJK, T.M.V. JANSSEN & M.J.B. STOKHOF (eds), *Formal methods in the study of language*, part II, 1981. ISBN 90 6196 213 7.

MCT 137 J. TELGEN, *Redundancy and linear programs*, 1981. ISBN 90 6196 215 3.

MCT 138 H.A. LAUWERIER, *Mathematical models of epidemics*, 1981. ISBN 90 6196 216 1.

MCT 139 J. VAN DER WAL, *Stochastic dynamic programming, successive approximations and nearly optimal strategies for Markov decision processes and Markov games*, 1980. ISBN 90 6196 218 8.

MCT 140 J.H. VAN GELDROP, *A mathematical theory of pure exchange economies without the no-critical-point hypothesis*, 1981.
ISBN 90 6196 219 6.

MCT 141 G.E. WELTERS, *Abel-Jacobi isogenies for certain types of Fano threefolds*, 1981.
ISBN 90 6196 227 7.

MCT 142 H.R. BENNETT & D.J. LUTZER (eds), *Topology and order structures*, part 1, 1981.
ISBN 90 6196 228 5.

An asterisk before the number means "to appear".