ECONOMETRIC INSTITUTE

REDUNDANT AND NON-BINDING CONSTRAINTS IN LINEAR PROGRAMMING PROBLEMS

JAN TELGEN

REPORT 7720/0

ERASMUS UNIVERSITY ROTTERDAM, P.O. BOX 1738, ROTTERDAM, THE NETHERLANDS

REDUNDANT AND NON-BINDING CONSTRAINTS IN LINEAR PROGRAMMING PROBLEMS

by

Jan Telgen

ABSTRACT

General mathematical programming problems may contain redundant and non-binding constraints. These are constraints, which can be removed from the problem without altering the feasible region or the optimal solution respectivily. Here we consider some more theoretical definitions and give reasons for selecting a special one. The emphasis is put on linear programming, but most of the material can be applied to any mathematical programming problem with linear constraints.

To identify redundant constraints several methods have been proposed. We give a survey and show that all these methods are variants of a general method (Telgen (1977a)).

No method is known to identify non-binding constraints directly; therefore we give some indirect ways to identify non-binding constraints. Finally, some remarks are made concerning the importance of the methods to identify redundant and non-binding constraints in practical linear programming problems, both from a managerial and from a computational point of view.

Contents

1. Definitions

We consider the general linear[0)] programming problem

$$\max z = \underline{c}^T \underline{x} \tag{1.1}$$

s.t.

$$\begin{cases} A\underline{x} \leq \underline{b} & (1.2) \\ \underline{x} \geq \underline{0} & (1.3) \end{cases}$$

where $\underline{c}$ and $\underline{x}$ are n-vectors, $\underline{b}$ is an m-vector and A is an $(m \times n)$ matrix; $\underline{0}$ is the n dimensional zero vector.
Alternatively we may write (1.2) - (1.3) as

$$\bar{A}\underline{x} \leq \bar{\underline{b}} \tag{1.4}$$

where $\bar{A} = \begin{bmatrix} A \\ \cdots \\ -I \end{bmatrix}$ and $\bar{\underline{b}} = \begin{bmatrix} \underline{b} \\ \cdots \\ \underline{0} \end{bmatrix}$

We define:

$$S = \{\underline{x} \in \mathbb{R}^n \mid A\underline{x} \leq \underline{b},\ \underline{x} \geq \underline{0}\} \tag{1.5}$$

$$= \{\underline{x} \in \mathbb{R}^n \mid \bar{A}\underline{x} \leq \bar{\underline{b}}\} \tag{1.6}$$

The removal of the k-th constraint from the problem is denoted by a subscript k, which can be attached to A, $\bar{A}$ or S, i.e.

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1, \ldots, m \qquad i \neq k \tag{1.7}$$

is equivalent to

$$A_k \underline{x} \leq \underline{b}_k \tag{1.8}$$

and

$$S_k = \{\underline{x} \in \mathbb{R}^n \mid \bar{A}_k \underline{x} \leq \bar{\underline{b}}_k\} \tag{1.9}$$

Note that in (1.8) k may range from 1 to m, while in (1.9) k may range from 1 to m+n, so non-negativity constraints are included as well in (1.9).

In the following figures some examples are given of redundant and non-binding constraints.

---

0 The restriction to linear programming problems is made only for simplicity of explanation. In fact, any mathematical programming problem with linear constraints may be considered. Then in the explanation that follows we have to replace the vector $\underline{c}$ by the gradient of the objective function (assuming the objective function is sufficiently differentiable).

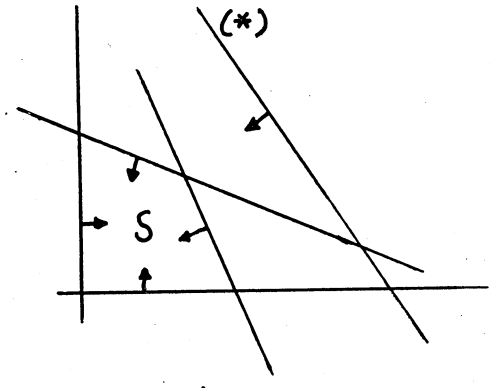


figure 1

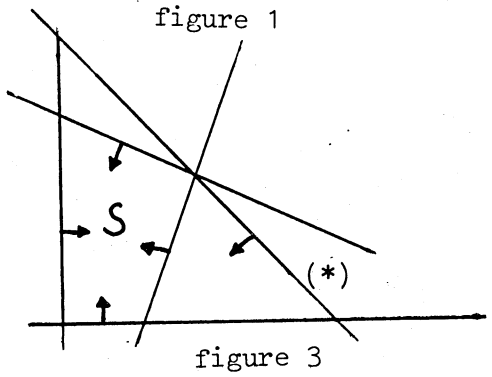


figure 3

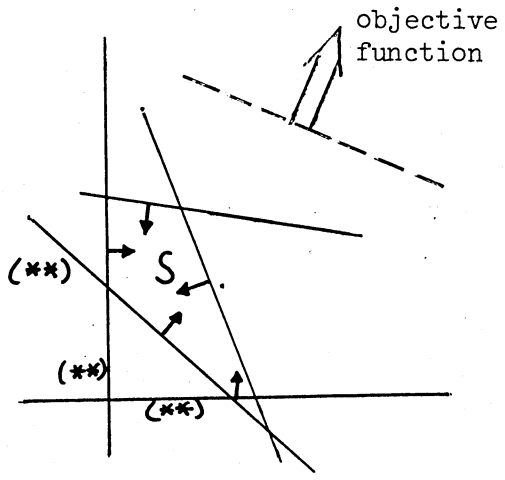


figure 2

In these figures constraints indicated with (*) are redundant and constraints indicated with (**) are non-binding. Note that in the figures with redundant constraints no objective function is specified, while the objective function plays an important role for non-binding constraints.

Formal definitions for redundant constraints as they appeared in literature are listed below:

Boot [1964]: The k-th constraint from $\bar{A}\underline{x} \leq \bar{b}$ is trivial if and only if

$$(1.10) \qquad \sum_{j=1}^{n} a_{kj}x_j \leq b_k \qquad \forall x \in S_k$$

Thompson, Tonge and Zionts [1966]: The k-th constraint from $A\underline{x} \leq \underline{b}$ is redundant if and only if

$$(1.11) \qquad S = S_k \qquad (k \leq m)$$

A similar definition is used by Eckhardt [1970].

Gal [1975a]: The k-th constraint from $\bar{A}\underline{x} \leq \bar{b}$ is strong[1)] redundant if and only if

$$(1.12) \qquad \sum_{j=1}^{n} a_{kj}x_j < b_k \qquad \forall\, x \in S$$

---

[1] The omission of the word "strong" in another paper (Gal [1975b]) is an error.

These definitions are not equivalent.
First, in the definition of Thompson, Tonge and Zionts [1966] the possibility for non-negativity constraints to be redundant (or $m+1 \leq k \leq m+n$) is excluded.
Second, if a constraint, that is otherwise not redundant, is stated twice, it should be redundant once. This is not the case in the definition of Gal [1975a].
Third, redundancy is a relative concept; a constraint can only be redundant with respect to a given system of constraints. Although Boot [1962] mentions this fact in a verbal definition, it is not sufficiently imposed in the mathematical definitions above.
Fourth, reconsidering the case in which an otherwise not redundant constraint is stated twice, both constraints are considered to be redundant by the definitions of Boot [1964] and Thompson, Tonge and Zionts [1966]. This can be prevented by a definition that is only applicable to systems in which no constraints are denoted as redundant yet.
In view of these considerations we prefer another definition, which can be used directly in a computational scheme.
Define

$$u_k = \bar{b}_k - \sum_{j=1}^{n} \bar{a}_{kj} x_j \qquad k \leq m+n \tag{1.13}$$

Then redundant constraints can be defined in the following way, which was suggested in a modified form in Gal [1975a]:

Definition: "The k-th constraint from $\bar{A}\underline{x} \leq \bar{b}$ is redundant with respect to the system $\bar{A}\underline{x} \leq \bar{b}$ if no other constraints[2)] have been denoted as redundant and if

$$\hat{u}_k = \min \{u_k \mid \underline{x} \in S_k\} \geq 0 \tag{1.14}$$

For computational purposes we distinguish between strict redundant constraints, for which $\hat{u}_k > 0$ and weak redundant constraints, for which $\hat{u}_k = 0$.
Constraints that are not redundant are called active.

---

[2] If more than one constraint is to be identified as being redundant, this can simply be achieved by removing the previous one from the system and again applying the definition to the new system.

Until now we did not pay any attention to the objective function of the linear programming problem. In fact it should be stressed, that the objective function does not appear in any definition of redundant constraints i.e. constraints are redundant independent of the objective function specified.

There is another kind of constraints that can be omitted from the linear programming problem without altering the optimal solution for some objective function. These constraints are called non-binding constraints. Definitions for non-binding constraints that appear in literature are as follows:

Thompson, Tonge and Zionts [1966]: "An active constraint that is satisfied as an equality at some optimum solution is a binding constraint. All other constraints are non-binding constraints."

Zimmerman and Gal [1975]: "A non-binding[3] constraint is an active constraint, that is not satisfied as an equality in some optimum solution".

There are two points that may be noted in these definitions; in the first place, they are in conflict about the question, whether or not redundant constraints are a subset of non-binding constraints (Thompson, Tonge and Zionts: yes, but Zimmerman and Gal: no). We prefer to follow Zimmerman and Gal, because the distinction between constraints that can be omitted from any problem, and constraints that can be omitted from some problems, is a useful one.

In the second place, in both definitions constraints that are satisfied as an equality at some optimal solution cannot be non-binding. This may cause some problems; consider the following figures:

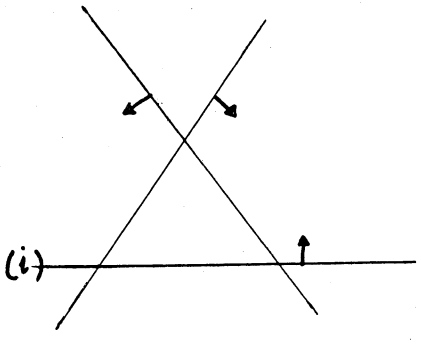
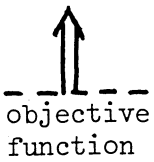


figure 4



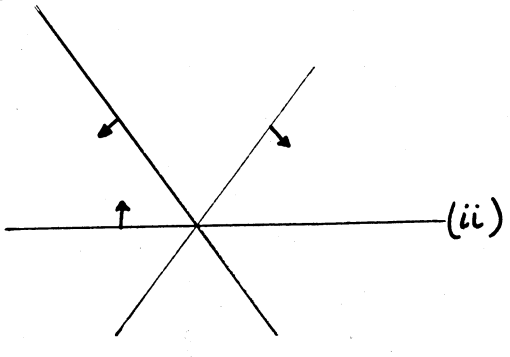


figure 5

Constraint (i) is non-binding, but constraint (ii) is not non-binding according to both definitions. Still, constraint (ii) is not redundant,

[3] Zimmerman and Gal use the word relative redundant, which may be confusing because these constraints are not redundant.

but may be omitted from the problem without changing the optimal solution. Because this is an unsatisfactory situation, we look for another definition, which is closer to the concept of possible removal of non-binding constraints and contains elements like:

$$\max\{\underline{c}^T\underline{x} \mid \underline{x} \in S\} = \max\{\underline{c}^T\underline{x} \mid x \in S_k\} = \hat{z} \tag{1.15}$$

if the k-th constraint is non-binding.
But this yields other difficulties as illustrated in the following figures:

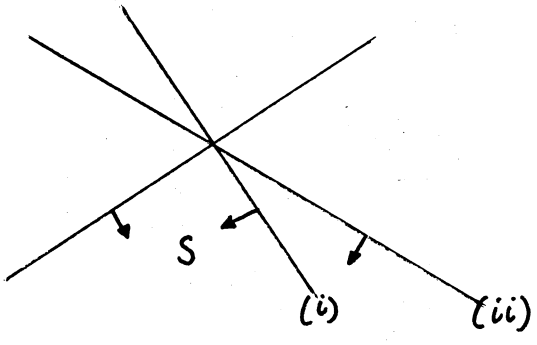


figure 6

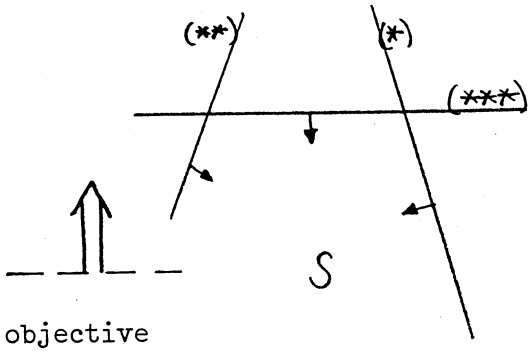


figure 7

Constraint (i) satisfies condition (1.15), but it should not be non-binding. Constraint (ii) is (weak) redundant, and if it is removed, constraint (i) is no longer non-binding. From this it follows that (weak) redundant constraints should be removed before non-binding constraints can be defined.

In figure 7 all points that satisfy constraint (***) as an equality have the same objective function value; the programming problem has multiple solutions. According to (1.15) both constraint (*) and constraint (**) are non-binding, but we would rather not consider these constraints as non-binding. Therefore we state the extra condition for non-bindingness:

$$\left\{\underline{x} \;\middle|\; \begin{array}{l}\underline{c}^T\underline{x} = \hat{z} \\ \underline{x} \in S\end{array}\right\} = \left\{\underline{x} \;\middle|\; \begin{array}{l}\underline{c}^T\underline{x} = \hat{z} \\ \underline{x} \in S_k\end{array}\right\} \tag{1.16}$$

Finally, just was was the case with redundant constraints, constraints can only be non-binding with respect to a specified set of constraints and objective functions. Consider the following 3-dimensional example.

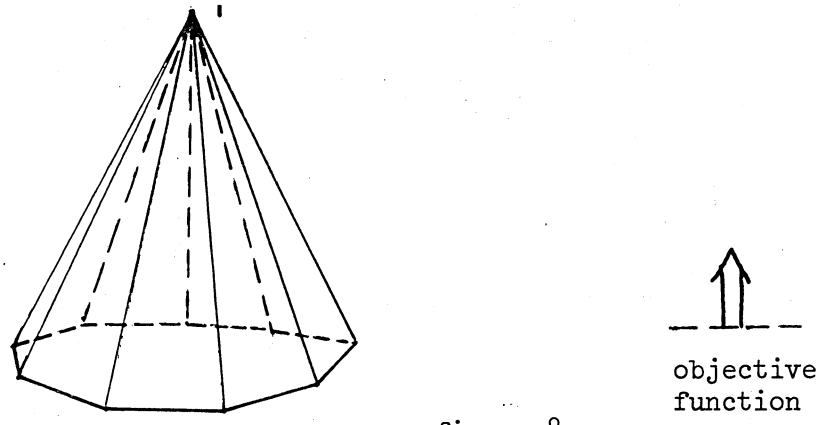


figure 8

Any of the constraints corresponding to these hyperplanes, all passing through the point T, may be omitted without changing the optimal solution, but all of them may not be omitted at the same time.

This leads to the following definition of non-binding constraints:

Definition: "The k-th constraint in the system $\bar{A}\underline{x} \leq \bar{b}$ is a non-binding constraint if it is an active constraint for which

$$\max \{\underline{c}^T\underline{x} \mid \underline{x} \in S\} = \max \{\underline{c}^T\underline{x} \mid \underline{x} \in S_k\} = \hat{z} \tag{1.17}$$

$$\left\{\underline{x} \mid \underline{c}^T\underline{x} = \hat{z},\ \underline{x} \in S\right\} = \left\{\underline{x} \mid \underline{c}^T\underline{x} = \hat{z},\ \underline{x} \in S_k\right\} \tag{1.18}$$

while no weak redundant constraints are specified in $\bar{A}\underline{x} \leq \bar{b}$ and no other constraints have yet been denoted as non-binding."[4)]

In conclusion, we have four types of constraints:

(a) strict redundant constraints;

(b) weak redundant constraints;

(c) non-binding constraints;

(d) binding constraints.

We refer to (a) + (b) as redundant constraints and to (c) + (d) as active constraints. Further, note that (c) and (d) are not complementary.

2. Dual equivalent concepts

Thompson, Tonge and Zionts [1966] introduced the expression "extraneous variables":

> extraneous variables are variables that are non-positive in every optimal solution.

Since we assumed that all variables are sign restricted (nonnegative), we might as well define extraneous variables to be zero in every optimal solution. As a direct result from complementary slackness extraneous variables and non-binding constraints are dual equivalents[5)] assuming a non-degenerate optimal solution.

---

[4] If more than one constraint is to be identified as non-binding, this can simply be done by removing the previous one from the system and applying the definition again to the new system.

[5] By a dual equivalent of concept A, we mean a concept that is equivalent to A in the dual of the problem for which A is defined.

Luenberger [1973] defines two other sets of variables:

- A variable is said to be a "null variable" if it has zero value in every solution of the system (1.2) - (1.3).

- A variable in the system (1.2) - (1.3) is "nonextremal" if its nonnegativity constraint is redundant in the system (1.2) - (1.3).

Now it is easy to see that redundant constraints and null variables are dual equivalents.

These results can be shown in the following figure.

	constraint	dual variable	slack variable
assuming non-degeneracy	binding		extraneous
	non binding	extraneous	
	redundant	null	nonextremal

So an adequate definition of these concepts leads to the development of formal relationships. Therefore in all following sections we can refer to redundant and non-binding constraints only. In that one should keep in mind, that all results apply equally well to their dual equivalents.

3. Identification of redundant constraints

The identification of redundant constraints in systems of linear inequalities is handled from a theoretical point of view in Telgen [1977a]. In the same paper a general method is developed to identify both strict and weak redundant constraints. This method is in fact mainly a direct application of the definition given in section 1, i.e., for all $k \leq m+n$ we determine

$$\hat{u}_k = \min\{u_k \mid \underline{x} \in S_k\} \tag{3.1}$$

Here we will show that all methods, that have been proposed in literature can be seen as variants of this general method.

We distinguish between three kinds of methods:

(a) direct methods: for a fixed constraint k it is determined, whether it is redundant or not;

(b) indirect methods: in certain situations redundant constraints may be recognized directly. Indirect methods consist of scanning a feasible solution for these situations. In this way redundant constraints may be identified by chance.

(c) heuristic methods: if certain conditions hold constraints may be identified as being redundant. However, not all conditions are easy to check and therefore the results should be validated afterwards.

A number of direct methods may be derived from the turn-over lemma, that follows directly from the definition of redundant constraints.

Turn-over lemma: "The k-th constraint from $\bar{A}\underline{x} \leq \bar{b}$ is redundant if and only if the system

$$\bar{A}_k \underline{x} \leq \bar{b}_k \tag{3.2}$$

$$\sum_{j=1}^{n} a_{kj} x_j > b_k \tag{3.3}$$

is infeasible".

Proof: ($\Rightarrow$) If the k-th constraint is redundant, then according to the definition of redundancy

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

Therefore $\bar{A}_k \underline{x} \leq \bar{b}_k \Rightarrow u_k \geq 0$

$$\Rightarrow \sum_{j=1}^{n} a_{kj} x_j \leq b_k$$

and thus (3.2) - (3.3) is infeasible.

($\Leftarrow$) If (3.2) - (3.3) is infeasible, there is no $\underline{x} \in S_k$ that satisfies $\sum_{j=1}^{n} a_{kj} x_j > b_k$, since $S_k \neq \emptyset$.

Therefore $u_k \geq 0 \qquad \forall\, x \in S_k$

and the k-th constraint must be redundant.

q.e.d.

In determining the feasibility of the system (3.2) - (3.3) one usually solves the linear programming problem

$$\min g = b_k - \sum_{j=1}^{n} a_{kj} x_j \tag{3.4}$$

$$\text{s.t.} \quad \bar{A}_k \underline{x} \leq \bar{b}_k \tag{3.5}$$

As soon as $g < 0$ the solution procedure is stopped because a feasible solution is found. Note that the problem (3.4) - (3.5) is exactly the same as the problem (3.1) solved by the general method.

Boot [1962] proposed to determine the feasibility of (3.2) - (3.3) by replacing (3.3) by

$$(3.6) \qquad \sum_{j=1}^{n} a_{kj} x_j = b_k + \varepsilon$$

substitute this equality for some x in all remaining inequalities and try to solve the resulting system of linear inequalities.

Thompson, Tonge and Zionts [1966] try to improve the computational performance of this method by considering constraint (3.6) as an inequality with a slack variable $-\varepsilon$. Then this slack variable is always kept in the basis with the same value. By this the computations for the elimination and substitution of a variable are unnecessary and fewer changes in the original problem (3.2) - (3.6) are caused.

But as well as in Boot's original scheme, for every constraint that is to be tested for being redundant, 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 is to be checked for being redundant.

It should be noted that (3.2) - (3.3) and (3.2) - (3.6) are not equivalent. If (3.2) - (3-6) is feasible, then (3.2) - (3.3) will also be feasible and the k-th constraint is not redundant. However, the k-th constraint will be redundant if (3.2) - (3.3) is infeasible and that may be concluded only if (3.2) - (3.6) is infeasible for all $\varepsilon > 0$.[6)]

Following an idea of Lisy [1971], Gal [1975a] presents a method to identify strict redundant constraints and some weak redundant constraints. Since this method closely resembles the general method, we refer to Telgen [1977a].

Eckhardt [1971] shows that for all redundant constraints k, there is some index $s \neq k$, some $\mu \geq 0$ and a basic feasible solution to (1.1) - (1.3) such that in the corresponding basic system:

$$(3.7) \qquad \begin{cases} \tilde{a}_{kj} \leq \mu \cdot \tilde{a}_{sj} & \forall\, j = 1, \ldots n \\ \tilde{b}_k \geq \mu \cdot \tilde{b}_s \end{cases}$$

---

6 The choice of a particular $\varepsilon > 0$, which will in general be rather small may also cause numerical difficulties.

The conditions (3.7) merely describe the situation in which it is possible, to perform one iteration and thereby obtaining another basic feasible solution in which the slack variable of the k-th constraint is basic, say in row t and all coefficients in this row t are nonpositive; then u_k will be at its minimal value and (3.1) is solved. Therefore Eckhardt's method may also be seen as a variant of the general method.

Indirect methods to identify redundant constraints have been proposed frequently in literature.
One of the earliest proposals is due to Llewellyn [1964]; some rules are given to identify constraints that are redundant by one other constraint and all non-negativity constraints. Since his rules are not as general as claimed and some more conditions should be imposed, we refer to Telgen [1977b] for a detailed treatment.
Thompson, Tonge and Zionts [1966] and Zionts [1965] introduce the concept of a definitional constraint. The k-th constraint and the p-th variable are called definitional if they can be written as

$$x_p = \sum_{j=1,\ j\neq p}^{n} \tilde{a}_{kj} x_j + \tilde{b}_k \tag{3.8}$$

where a tilde indicates that this is with respect to some basis and where both

$$\tilde{a}_{kj} \geq 0 \ \forall_j \quad \text{and} \quad \tilde{b}_k \geq 0.$$

The nonnegativity of the variable x_p follows from the nonnegativity constraints on the other variables and the constraint (3.8). Therefore the definitional variable x_p is a free variable. If x_p is the slack variable of the k-th constraint this means that the k-th constraint itself is redundant.
Now it is easy to see that x_p can be brought into the basis (if it is not already in) without changing the signs of a_{kj} and b_k. Then constraint (3.8) can be interpreted as the objective row for (3.1), indicating that an optimal solution has been obtained with value b_k.
Thompson, Tonge and Zionts [1966] give a number of situations, where (3.8) is not directly satisfied, but may be obtained in one iteration. Naturally, from these situations the same conclusions may be drawn without actually performing the iteration.

Moreover, Thompson, Tonge and Zionts [1966] describe a Monte Carlo technique in which one tries to construct situations as described above by a random generator.

Finally, Thompson, Tonge and Zionts [1966] note that in the case a constraint may be written as

$$\sum_{j=1}^{n} \tilde{a}_{kj} x_j = 0 \qquad (3.9)$$

where all $\tilde{a}_{kj}$ are of the same sign [7], then all x_j corresponding to non zero $\tilde{a}_{kj}$ will be zero in any feasible solution.[8]

Tisher [1968] gives an extensive list of very simple methods to identify redundant constraints. Among these are methods that check whether constraints are redundant given a number of upper and lower bounds, by simply replacing the variables in the constraints by their bounds. However, this is a merely a simple way to solve (3.1), if only bounds are taken into consideration.

Heuristic methods to identify redundant constraints may also be used on non-binding constraints. Therefore they will be considered in more detail in the next section.

4. Identification of non-binding constraints

There are no direct or indirect methods available to identify non-binding constraints. Therefore one has to resort to heuristic methods or to techniques that identify non-binding constraints as being redundant in a closely related system.

Heuristic methods are applications of rules, that are valid only if certain conditions are fulfilled. Because a priori checking of these conditions is not done, either because it is impossible or too laborious, the conditions should be checked and the rules validated a posteriori.

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 put in the basis and marked as being no candidate for leaving the basis. These slack variables or rather these constraints are placed behind some

---

[7] Non-negativity as required by Thompson, Tonge and Zionts is not necessary.

[8] It is assumed that all x_j corresponding to non-zero $\tilde{a}_{kj}$ are non-negative.

curtain, where they do not affect the solution procedure. Of course the same thing can be done in the dual formulation, where some variables may be placed behind the curtain, from where they are not allowed to enter the basis.[9)]

Apart from the fact that this technique may be profitable for the computing speed and the storage needed (everything behind the curtains does not have to be stored in the fast memory), it may be used to obtain a good starting solution (crashing) and in selecting a pivot. It should be noted that more curtains can be used together, separating variables with different probabilities to enter the basis. This may depend on the a priori information available, but also on the solution path being followed.

A solution path is called convex if any two-dimensional projection of the path, orthogonal to any hyperplane corresponding to a constraint is convex. Thompson, Tonge and Zionts [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 be on zero value in the optimal solution. However, because there is no known simple way to ensure a convex solution path, these results can be used in a heuristic way only.

Another way to identify (some) non-binding constraints is to construct a system of linear constraints that is closely related to the original one and has the same optimal solution.

In this sense it seems useful to introduce during the solution of a linear programming problem for a number of feasible solutions $\underline{x}^0$ the constraint

$$\underline{c}^T \underline{x} \geq c^T \underline{x}^0 \qquad (4.1)$$

By introducing this constraint one hopes to convert some originally non-binding constraints ("at the lower boundary of the feasible region") to redundant constraints. These redundant constraints may be identified by the techniques mentioned in the preceding section. After this identification constraint (4.1) can be removed again since it is certainly non-binding.

In general however one cannot say, that constraints "at the lower boundary of the feasible region", are non-binding as Scolnik [1973] implicitly

---

9 For a detailed treatment of these techniques we refer to Orchard-Hays [1968].

assumes. Probably this is due partly to the fact that a constraint"at the lower boundary of the feasible region" can be recognized rather simply. If the objective function is to be maximized and the constraints are in "$\leq$" form, then a constraint that is at the lower boundary of the feasible region forms a non-acute corner with the objective function, so this corner has a negative cosine:

$$\cos \phi = \frac{\sum_{j=1}^{n} a_{kj} c_j}{\left(\sum_{j=1}^{n} a_{kj}^2\right)\left(\sum_{j=1}^{n} c_j^2\right)} \tag{4.2}$$

but this implies that $\sum_{j=1}^{n} a_{kj} c_j$ should be negative and this is easily checked. However a counterexample may easily be formulated.
Consider the problem:

$$\begin{cases} \max & x_2 \\ \text{s.t.} & -x_1 - x_2 \leq 2 \\ & x_1 + 3x_2 \leq 4 \\ & x_1, x_2 \geq 0 \end{cases} \tag{4.3}$$

The first constraint is at the lower boundary of the feasible region (according to (4.2)) but it still is binding. This means that this rule may only be used in a heuristic way.

By considering both the dual and the primal problem, one can sometimes remove some columns from the problem, by which some non-binding constraints become redundant.
If, in the dual problem, a constraint is identified as being redundant, this constraint may be removed from the dual problem and the corresponding variable from the primal problem. This can cause some non-binding constraints to become redundant.
Consider the problem:

$$\begin{cases} \max & x_1 \\ \text{s.t.} & x_1 + x_2 \leq 2 \\ & x_1 + 2x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{cases} \tag{4.4}$$

it is easily seen that the second constraint is non-binding but not redundant. In the dual problem:

$$(4.5) \qquad \begin{aligned} \min \quad & 2y_1 + 3y_2 \\ \text{s.t.} \quad & y_1 + y_2 \geq 1 \\ & y_1 + 2y_2 \geq 0 \\ & y_1, y_2 \geq 0 \end{aligned}$$

it is trivial that the second constraint is redundant.
Removing the second variable from the primal problem, yields:

$$(4.6) \qquad \begin{aligned} \max \quad & x_1 \\ \text{s.t.} \quad & x_1 \leq 2 \\ & x_1 \leq 3 \\ & x_1 \geq 0 \end{aligned}$$

and now the second constraint is redundant. So by successively identifying redundant constraints in the primal and the dual problem and by removing them, some non-binding constraints may also be identified.

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

$$(4.7) \qquad \begin{cases} A\underline{x} \leq \underline{b} \\ \underline{x} \geq \underline{0} \\ A^T\underline{y} \geq \underline{c} \\ \underline{y} \geq \underline{0} \\ \underline{b}^T\underline{y} \leq \underline{c}^T\underline{x} \end{cases}$$

all non-binding constraints will be converted into redundant constraints, so they may be identified in this way.

5. Practical implications

From a computational point of view it is sufficient to specify only the binding constraints in linear programming problems. The reasons for specifying redundant constraints (the same applies partly to non-binding constraints), may be divided into two categories. First there are some managerial reasons among which we mention:

(a) insufficient knowledge of the described system, which causes not all relations and interactions to be taken into consideration. This may be caused by a superficial treatment of the problem, but also by some problem characteristics itself: the problem may be too big or too intricate to recognize redundant constraints.

(b) implicit preference for more constraints above the possibility of an optimal solution, that does not meet all practical demands. This means that one rather specifies another constraint, even if it may be redundant, than being confronted afterwards with the possibility of having overlooked a constraint. The importance of this reason is growing, because a lot of models are being developed that should be applicable a number of times and in which constraints are specified because they might become important in the future, but are not restrictive now.

There are also some mathematical reasons, for redundant constraints to be specified. We mention the cutting planes in integer programming and the addition of an extra constraint to identify non-binding constraints as described in section 4. Another example is the complete set of constraints for the transportation problem, from which one constraint may be considered to be redundant.

One will be inclined to think that extra constraints give extra information about the problem to be solved. This is not true for redundant constraints. Redundant constraints do not give any new information in that they do not exclude any possibilities (solutions), that would be admitted without these constraints.

However, redundant constraints may sometimes express the information, that is included in other constraints too, in a way, that may be easier from the point of view of getting some insight into the problem. Consider for example the problem

$$\begin{cases} x_1 + x_2 \leq 4 \\ x_1 \geq 0 \end{cases} \tag{5.1}$$

For this problem the constraint

$$x_2 \leq 4 \tag{5.2}$$

is redundant, but explicitely specifying (5.2) may give a bound for x_2 that may be rather difficult to compute from (5.1).

On the other hand the specification of redundant and non-binding constraints causes a number of problems:

(a) they use some space in the memory, that may be used better; it may even be so critical that one has to resort to other solution procedures e.g. decomposition.

(b) they increase the number of computations per iteration: it is not true that redundant constraints only cause an extra logical in the basis, but have no further influence.

(c) they cause an increase in the number of basic solutions, which may lead to a larger number of iterations, especially in phase I.

(d) they may cause degeneration to occur, which may even lead to cycling.

(e) Thompson, Tonge and Zionts (1966) report that the phenomenon of near-cycling (the objective function value is improved very little during a large number of iterations) is observed less frequently if redundant constraints are removed.

(f) dependent or nearly dependent constraints may cause some numerical troubles, which will be prevented if redundant constraints are removed, since (nearly) dependent constraints are often redundant.

Apart from these computational disadvantages, we observe that redundant and non-binding constraints enlarge the problem, because of which it may become rather complicated. This is also caused by the sheer fact that these constraints, by being specified, make the impression of being binding constraints.

The number of redundant constraints varies with the problem that is being considered. There is no such thing as a fixed percentage of the number of constraints that is redundant.

In literature there are some references to a certain portion of the constraints that is redundant: Zionts (1965) reports of problems with 35 to 50 % redundant and non-binding constraints; Thompson, Tonge and Zionts (1966) give some examples with 30 to 75 % reduction if both in the primal and in the dual problem redundant and non-binding constraints are removed; Tischer (1968) reports of reductions up to 80 %. All of these figures are conservative since the methods used by the reporters do not identify all redundant constraints.

However, not in all practical linear programming problems the reductions will be that dramatic, but the fact that reductions are almost always possible, is agreed upon by most practitioners (see e.g. Zionts (1965)).

6. Concluding remarks.

In literature there are not many references to practical tests on methods to identify redundant and non-binding constraints. We can only mention the reports of Zionts (1965), Thompson, Tonge and Zionts (1966) and Tischer (1968).

On the whole the reported results of the simplex method together with the indirect methods are better than the simplex method alone if the number of iterations is considered, but equally good if total computer time is considered.

The reported results for the heuristic methods are rather favorable, but depend very much on the problem under consideration.

From the preceding section we may conclude that in general it will be better to remove redundant (and non-binding) constraints from linear programming problems. However the effort spent in identifying them should not be larger than the savings achieved by deleting them.

Since this last topic is not considered to a far extent in existing literature we plan to explore it in some subsequent papers. This further research should concentrate on two points

(a) experiments with methods to identify redundant and non-binding constraints on practical problems.

(b) adapting the methods that have been proposed, to the linear programming codes that are used nowadays. Since these are based on the revised simplex method, in which the coefficients a_{ij} are not updated at every iteration, while the methods to identify redundant constraints use the updated values a_{ij}, this may be a major task.

In view of the last point and the results reported it is expected, that methods to identify redundant constraints may be of help in the pre-solution stage, where the problem is formulated. This expectation is supported by the fact that in present linear programming codes like MPSX 370 and APEX III, the routines for these methods,[10)] called REDUCE, are applied to the problem prior to the solution.

[10] Only rather simple cases, like the ones described by Tischer (1968) are covered by these routines. A description of the methods used in these routines can be found in Brearly, Mitra and Williams (1975).

References

J.C.G. Boot [1962]. On trivial and binding constraints in programming problems, Management Science, vol. 8, no. 4.

J.C.G. Boot [1964]. Quadratic programming, North Holland.

A.L. Brearly, G. Mitra and H.P. Williams[1975]. Analysis of mathematical programming problems prior to applying the simplex algorithm, Math. Prog., vol 8, pp 54-83.

G.B. Dantzig [1955]. Upper bounds, secondary constraints and block-triangularity, Ekonometrika 23, no. 2, pp. 174-183.

U. Eckhardt [1971]. Redundante Ungleichungen bei linearen Ungleichungs-systemen, Unternehmensforschung, vol. 12, pp. 279-286.

T. Gal [1975a]. Zur Identifikation redundanter Nebenbedingungen in linearen Programmen, Zeitschrift für Operations Research, Band 19, p. 19-28.

T. Gal [1975b]. Redundancy reduction in the restrictions set given in the form of linear inequalities, Progress in Cybernetics and Systems Research, volume I.

J. Lisy [1971]. Metody pro nalezeni redundantnich omezeni v ulohach linearniho programovani, Ekonomicko Matematicky Ob zor 7, nr. 3, pp. 285-298.

R.W. Llewellyn [1964]. Linear programming, Holt Rinehart and Winston.

D.G. Luenberger [1973]. Introduction to linear and non-linear programming, Addison-Wesley.

W.Orchard-Hays [1968]. Advanced linear programming computing techniques, McGraw Hill.

H.D. Scolnik [1973]. A new approach to linear programming, Sigmap Newsletter, no. 14.

J. Telgen [1977a]. On redundancy in systems of linear inequalities, report 7718, Econometric Institute, Erasmus University Rotterdam.

J. Telgen [1977b]. On RW Llewelly'n rules to identify redundant constraints in systems of linear inequalities, report 7719, Econometric Institute, Erasmus University Rotterdam.

G.L. Thompson, F.M. Tonge and S. Zionts [1966]. Techniques for removing non-binding constraints and extraneous variables from linear programming problems, Management Science, vol. 12, no. 7.

H.J. Tischer [1968]. Mathematische Verfahren zur Reduzierung der Zeilen- und Spaltenzahl linearer Optimierungsaufgaben, Zentralinstitut für Fertigungstechnik des Machinenbaues, Karl Marx Stadt.

H.J. Zimmerman and T. Gal [1975]. Redundanz und ihre Bedeutung für Betriebswirtschaft, vol. 45, no. 4.

S. Zionts [1965]. Size reduction techniques of linear programming and their application, Carnegie Institute of Technology.

Reports 1977

7700 List of Reprints, nos. 179-194; List of Reports, 1976.

7701/M "Triangular - Square - Pentagonal Numbers", by R.J. Stroeker.

7702/ES "The Exact MSE-Efficiency of the General Ridge Estimator relative to OLS", by R. Teekens and P.M.C. de Boer.

7703/ES "A Note on the Estimation of the Parameters of a Multiplicative Allocation Model", by R. Teekens and R. Jansen.

7704/ES "On the Notion of Probability: A Survey", by E. de Leede and J. Koerts.

7705/ES "A Mathematical Theory of Store Operation, by B. Nooteboom.

7706/ES "An Analysis of Efficiency in Retailing, by B. Nooteboom.

7707/S "A Note on Theil's Device for choosing a Blus Base", by C. Dubbelman.

7708/E "A General Market Model of Labour Income Distribution: An Outline", by W. H. Somermeyer.

7709/E "Further Results on Efficient Estimation of Income Distribution Parameters", by T. Kloek and H.K. van Dijk.

7710 "List of Reprints , nos. 195-199; Abstracts of Reports First Half 1977",.

7711/M "Degenerating Families of Linear Dynamical System I", by M. Hazewinkel.

7712/M "Twisted Lubin-Tate Formal Group Laws, Ramified Witt Vectors and (Ramified) Artin-Hasse Exponential Mappings", by M. Hazewinkel.

7713/EM "An Efficient Way in Programming 'Eaves' Fixed Point Algorithm", by R. Jansen and A.S. Louter.

7714/E "An Alternative Derivation and a Generalization of Nataf's Theorem"? by W.H. Somermeyer and J. van Daal.

7715/M "Application of Non-Linear Programming to Plane Geometry", by R. Stroeker.

7716/S "Stochastic Compactness of Sample Extremes" by L. de Haan and G. Ridder.

7717/ES "Cross-Section Studies of Efficiency in Retailing, Part I: Individual Grocers and Butchers", by B. Nooteboom.

7718/O "On Redundancy in Systems of Linear Inequalities", by J. Telgen.

7719/O "On R.W. Llewellyn's Rules to Identify Redundant Constraints; A Detailed Critique and Some Generalizations", by J. Telgen.

7720/O "Redundant and Non-Binding Constraints in Linear Programming Problems", by J. Telgen.