SONDERDRUCK AUS

OPERATIONS RESEARCH
VERFAHREN XXVIII
METHODS OF OPERATIONS RESEARCH

8. KRABS W., 1975 : Optimierung und Approximation, Teubner, Stuttgart.

9. LEHMANN R. and W. OETTLI, 1975 : The theorem of the alternative, the key-theorem, and the vector-maximum problem, Math. Programming 8, p. 332-344.

10. LEVINE P. et J-Ch. POMEROL,1976 : C-closed mappings and Kuhn-Tucker vectors in convex programming, CORE D.P. N° 7620, Louvain.

11. OETTLI W., 1974 : A duality theorem for the non linear vector maximum problem, Progress in Operations Research Colloquia mathematica societatis Janos Bolyai 12, Eger (Hongrie), ed. by A. Prékopa, North-Holland, Amsterdam 1976.

12. POMEROL J-Ch., 1976 : Stability in convex programming, presented at the I Symposium über Operations Research, Heidelberg, 1976, forthcoming in Operations Research Verfahren, Anton Hain, Meisenheim.

13. SCHAEFER H.H., 1971 : Topological vector spaces, 3th. ed., Graduate Texts in Mathematics, Springer, New York.

Anschrift:
Groupe de Mathématiques Economiques
Université de Paris 6,
4 Place Jussieu
F-75230 Paris Cedex 5 (France)

Redundant Constraints in Linear Programming Problems

Jan Telgen, Rotterdam

Abstract

Redundant constraints are considered for their effects on general linear programming problems. It is conclused both for computational and for managerial reasons, that redundant constraints should not be incorporated in general LP problems.

However if redundant constraints are possibly present in LP problems it is still an open question whether or not the effort spent in identifying them is justified by the resulting computational simplifications.

We consider this question for a number of methods to identify redundant constraints. Among these methods are the ones described by Tischer, Boot, Thompson, Tonge and Zionts, Gal and Llewellyn. Some heuristic methods will be considered too.

It must be concluded that the identification of redundant constraints is a very interesting and promising aspect of linear programming.

Introduction

We consider the general linear programming problem

(1) $\max \; z = \underline{c}^T \underline{x}$

subject to

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

where A is an (mxn) matrix, $\underline{b}$ is an m-vector, and $\underline{c}$, $\underline{x}$ and $\underline{0}$ are n-vectors.

We define the feasible region S to be

$$(3) \qquad S = \left\{ \underline{x} \in R^n \;\middle|\; \begin{array}{l} A\underline{x} \leq \underline{b} \\ \underline{x} \geq \underline{0} \end{array} \right\}$$

and we assume $S \neq \emptyset$.

Redundant constraints are usually defined by the property that they may be removed from (2) without changing the feasible region. If we denote the removal of the k-th constraint by a subscript k, this means that the k-th constraint is redundant if

$$(4) \qquad S_k = S$$

Since we only want to make clear intuitively, what we mean by redundant constraints, we refer to Telgen (1977a) for more formal definitions.

We will illustrate redundancy by some figures

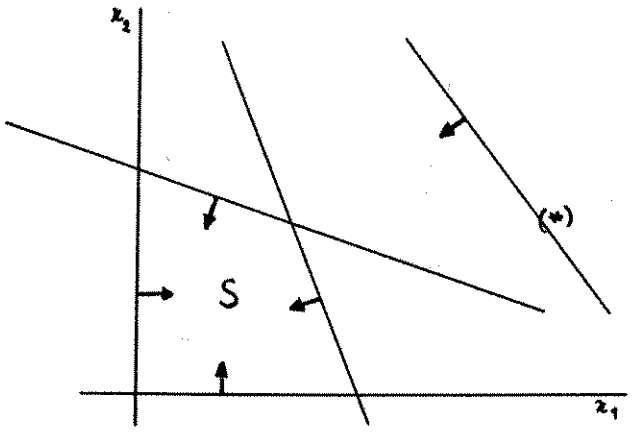

figure 1. The constraint (*) is redundant.

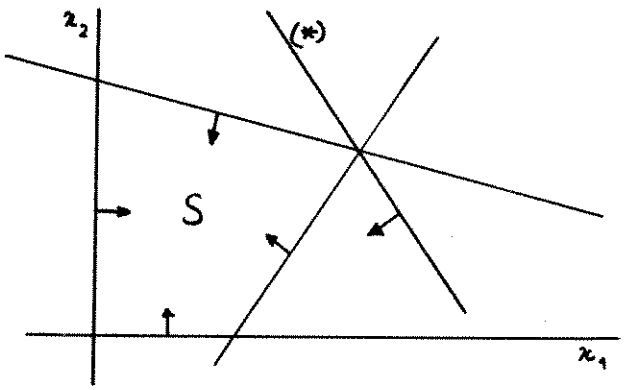

figure 2. The constraint (*) is redundant.

The future reference we call the redundant constraint in figure 1 strict redundant and the one in figure 2 weak redundant.

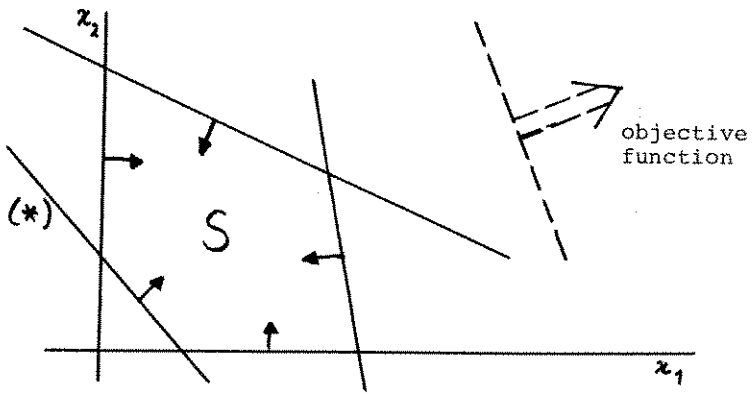

figure 3. The constraint (*) is not redundant.

One should be careful not to confuse redundant constraints with constraints like the one indicated in figure 3; the last one is called non-binding but is not redundant. Note that in figures 1 and 2 no objective function is specified, while in figure 3 it is essential. Since they lead to very much different problems, we will not consider non-binding constraints in this paper; for a detailed treatment of both definitions and methods we refer to Telgen (1977a)

In this paper we consider the influence of redundant constraints* on general linear programming problems and give a survey of existing methods to identify redundant constraints in linear programming problems. We emphasize on the practical applicability of these methods.

2. Existence of redundant constraints.

In a number of publications figures are reported on the percentage of redundant constraints in practical linear programming problems. Zionts (1965) reports 35 - 70%, Thompson, Tonge and Zionts (1966) report 35 - 50% and Tischer even mentiones problems with up to 80% redundant constraints. These percentages are even more astonishing if one bears in mind that they do not represent the actual percentage of redundant constraints in the problems studied, but the percentage of the constraints that were identified as being redundant by the imperfect and not exhaustive methods used by these authors.

However we are inclined to believe that not all practical linear programming problems contain as much redundant constraints as reported in these studies. The percentage of redundant constraints will very quite a lot, but still one may safely assume, that there is a considerable number of redundant constraints in almost all linear programming problems.

Let us now turn to the reasons, why redundant constraints are specified. They are mainly managerial, but may also be mathematical. We consider to be the most important:

(a) insufficient knowledge of the described system.
This may in turn be caused both by superficial consideration of the system or because the system under consideration is too big, too intricate and too difficult to conceive redundancy. So redundancy may be caused by human failure to perceive the system, whether or not on purpose.

---

*Of course, everything stated here for redundant constraints may also be applied to their dual equivalents, extraneous variables.

(b) implicite preference to specify more constraints to a smaller problem, of which the optimal solution may not meet all demands. This means that the maker of a linear programming model rather specifies redundant constraints than solves a slightly modified model in two runs. Apart from practical reasons (time etc), this may very well be based on psychological factors.

(c) some mathematical techniques require the specification of extra constraints, by which a number of constraints may become redundant: examples of these techniques are the cutting-plane algorithm for integer problems and the lower bound algorithm to identify non-binding constraints (see Telgen (1977a)).

(d) automation may cause the reason (b) to become more important. A large number of models are formulated to be solved a number of times with changing coefficients. Because of the changes in the coefficients some constraints may change from redundant to non-redundant but also the other way around. For example in production planning a capacity constraint may be redundant at the moment, but it is specified because it may become binding in future.

For more details about reasons for specification of redundant constraints we refer to Telgen (1977a).

The effects of specifying redundant constraints in linear programming problems, may be advantageous as well as disadvantageous. We consider two effects to be advantageous: first, specifying these constraints meets the managerial demands from reasons (b) and (d); second, specification of redundant constraints may conceptually simplify the model. Consider the problem:

$$(5) \quad \begin{cases} -x_1 - x_2 + x_3 + x_4 \leq 2 \\ x_1 + x_2 + x_3 + x_4 \leq 8 \\ -x_4 \leq -2 \end{cases}$$

The constraint

(6) $x_3 \leq 3$

is redundant in this system*, but may conceptually simplify it because an upper bound for x_3 is immediately known if this constraint is added to the system (5).

Note that no extra information is gained by specifying the redundant constraint (6), since it is implicitely contained in the constraints (5).

On the other hand there are many disadvantageous effects in specifying redundant constraints. A number of them are listed below:

(a) 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 to other procedures e.g. decomposition.

(b) redundant constraints necessitate more calculations per simplex iteration, for example in the determination of the variable to leave the basis.

(c) if redundant constraints are specified, in general there will be more basic solutions, so the simplex method may require more iterations in phase I.

(d) there will in general also be more degenerate basic solutions, which may cause numerical problems or cycling.

(e) redundant constraints have been noted by Thompson, Tonge and Zionts (1966) to increase the occurrence of a phenomenon called near-cycling. This means that in a number of iterations the value of the objective function changes only very little.

(f) the counterpart of an advantage of specifying redundant constraints is the disadvantageous fact that, just by being specified, redundant constraints make the impression of being non-redundant. This may in turn confuse the model and the user's view of the system.

*Add the first two constraints, divide by 2 and add the last constraint; the result is $x_3 \leq 3$.

While the last effect is a managerial one, most of the disadvantageous effects of specifying redundant constraints are mathematical.

3. Identification of redundant constraints.

From the preceding section we may conclude that redundant constraints should not be specified in linear programming problems. However, in general linear programming problems redundant constraints are specified. Therefore we have to answer the question whether identification and removal of redundant constraints is justified by the resulting simplifications in the linear programming problem. Because the removal of redundant constraints (once they have been identified) is a rather easy task, that may be neglected in this context, we just have to compare the identification and the resulting simplifications.

In the next section we consider some methods to identify redundant constraints and now we will turn to the resulting simplifications. The extent of these simplifications depends on two things.

First, it is very important how many redundant constraints are specified in the original model. Although some authors report rather large percentages of redundant constraints, we must stress that this percentage is very much problem dependent or at least depending on the kind of system being described. Secondly, the extent of the resulting simplifications depends on the purpose of specifying the model. It may make a lot of difference if the model is formulated just to gain insight into the problem, or to find the optimal solution. But in all cases it will be clear that some simplifications will result*, so it depends on the methods to identify redundant constraints, whether or not it is worthwhile to try to identify them.

---

*Even the knowledge of the fact that there are no redundant constraints specified, can be viewed as a simplification.

4. Methods to identify redundant constraints. 8

The identification of redundant constraints in systems of linear inequalities is handled from a theoretical point of view in Telgen (1977b). In the same paper a general method is developed to identify both strict and weak redundant constraints.

This method consists basically of the determination of

$$(7) \qquad \bar{u}_k = \min \{u_k \mid \underline{x} \in S_k\}$$

where

$$u_k = b_k - \sum_j a_{kj} x_j .$$

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.

All direct methods are based on the turn-over lemma, that can easily be derived from the definition of redundant constraints (Telgen (1977b)).

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

$$(8) \qquad A_k \underline{x} \leq \underline{b}_k$$

$$(9) \qquad \sum_{j=1}^{n} a_{kj} x_j > b_k$$

is infeasible".

For a formal proof we refer to Telgen (1977a).

In determining the feasibility of the system (8)-(9) one usually solves the linear programming problem

$$(10) \qquad \min \; g = b_k - \sum_{j=1}^{n} a_{kj} x_j$$

$$\text{s.t.} \qquad A_k \underline{x} \leq \underline{b}_k .$$

As soon as $g < 0$ the solution procedure is stopped because a feasible solution is found. Note that the problem (10) is exactly the same as the problem (7) solved by the general method.

Boot [1962] proposed to determine the feasibility of (8) - (9) by replacing the last constraint by

$$(11) \qquad \sum_{j=1}^{n} a_{kj} x_j = b_k + \varepsilon$$

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 (11) 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 (8)-(11) 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 (8)-(9) and (8)-(11) are not equivalent. If (8)-(11) is feasible, then (8)-(9) will also be feasible and the k-th constraint is not redundant. However the k-th constraint will be redundant if (8)-(9) is infeasible and that may be concluded only if (8)-(11) is infeasible for all $\varepsilon > 0$.*

Following an idea of Lisy [1971], Gal [1975a] presents a method to identify strict redundant constraints. Since this

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

method closely resembles the general method, we refer to Telgen [1977b].

Indirect methods to identify redundant constraints have been proposed frequently in literature. One of the earliest proposals is due to Llewellyn [1964]; in that 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 consitions 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

$$(12) \qquad x_p = \sum_{\substack{j=1 \\ j \neq p}}^{n} \tilde{a}_{kj} x_j + \tilde{b}_k$$

where a tilde indicates that this is with respect to some basis and where both $\tilde{a}_{kj} > 0 \quad \forall_j$ and $\tilde{b}_k \geq 0$. The nonnegativity of the variable x_p follows from the nonnegativity constraints on the other variables and the constraint (12). 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 $\tilde{a}_{kj}$ and $\tilde{b}_k$. Then constraint (12) can be interpreted as the objective row for (7), indicating that an optimal solution has been obtained with value $\tilde{b}_k$.

Thompson, Tonge and Zionts [1966]give a number of situations, where (12) 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 randomgenerator.

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

$$(13) \qquad \sum_{j=1}^{n} \tilde{a}_{kj} x_j = 0$$

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

Tischer [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, in fact this is a merely a simple way to solve (7), if only bounds are taken into consideration.

Heuristics 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 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.***

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

---

*Non-negativity as required by Thompson, Tonge and Zionts is not necessary.

**It is assumed that all x_j corresponding to non-zero a_{kj} are non-negative.

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

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 a zero value in the optimal solution. However, because there is no known simple way to ensure a convex solution path, these results should be used in a heuristic way only.

5. Conclusion

From a purely theoretical point of view it is not obvious whether or not identification and removal of redundant constraints in linear programming problems is worthwhile. Both linear programming and the identification of redundant constraints are in the same complexity class, which means that either one of these problems can be reduced to the other one. For a formal proof we refer to Telgen (1977b). However if a problem is to be solved that is more complex than general linear programming e.g. integer programming, then it will always be efficient from a theoretical point of view to try to identify redundant constraints.

From a practical point of view we may note that Tischer (1968) and Zionts (1965) report results that indicate that identification of redundant constraints at least in some instances has favourable effects. In both papers indirect methods were used in a standard simplex routine. However, it should be noted that all methods developed so far, are rather hard to implement in linear programming codes, in which the coefficients of the matrix are not updated after every iteration. This is certainly

the case for most practical linear programming codes, which do not use full tableau updating. By this fact indirect methods are not considered to be of great importance.

Therefore more attention should be paid to direct methods, of which we expect the general method of Telgen (1977b) to be the most powerful. Direct methods may be used prior to the solution procedure and therefore tableau updating by the solution procedure is not of major importance. However no empirical results are known yet.

In practice large commercial linear programming systems usually contain features like REDUCE, in which rather simple methods are applied to identify redundant constraints or redundant ranges on variables. For a survey we refer to Brearly, Mitra and Williams (1975).

Finally we must state that relatively little research is done in this area. There is hardly a reason for that, since the (very small number of) empirical results are rather in favour of identification of redundant constraints. Therefore we feel that the identification of redundant constraints is a very promising aspect of linear programming that may proove to be fruitful especially if good tricks are developed to implement the existing methods into non-tableau form linear programming codes.

References

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

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

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

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

J. Lisy (1971), Metody pro nalezeni redundantnich omezeni v ulohach linearniho programovani, Ekonomicko Matematicky Obzor 7, nr. 3, pp. 285-298.

R.W. Llewellyn (1964), Linear programming, Holt, Rinehart and Winston.

W. Orchard-Hays (1968), Advanced linear programming computing techniques, McGraw Hill.

J. Telgen (1977a), Redundant and non-binding constraints in linear programming problems, report 7720, Econometric Institute, Erasmus University Rotterdam.

J. Telgen (1977b), On redundancy in systems of linear inequalities, report 7718, Econometric Institute, Erasmus University Rotterdam.

J. Telgen (1977c), On RW Llewellyn's rules to identify redundant constraints: a detailed critique and some generalizations, 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.

J. Tischer (1968), Mathematische Verfahren zur Reduzierung der Zeilen- und Spaltenzahl linearer Optimierungsaufgaben, Zentralinstitut für Fertigungstechnik des Maschinenbaues; Karl Marx Stadt.

S. Zionts (1965), Size reduction techniques of linear programming and their application, Ph.D. thesis, Carnegie Institute of Technology.

Anschrift:
Econometric Institute
Erasmus University Rotterdam
NL Rotterdam

Linear Search Methods for Barrier Functions

J. Vermeylen and M. J. Rijckaert, Antwerp

Abstract : The paper discusses the difficulties encountered in implementing linear search methods in barrier type penalty methods. Seven different algorithms for one-dimensional optimization, adapted for the present purpose, are discussed together with their numerical performance on a set of 10 test problems.

1. Introduction.

A widespread and well known algorithm to solve the general non linear constrained programming problem :

$$\text{Minimize} \quad f(x) \qquad x \in \mathbb{R}^n \qquad (1)$$

$$\text{subject to} \quad g_j(x) \geq 0 \qquad j=1,\ldots,m \qquad (2)$$

is the method of penalty functions. The present paper will deal with one particular type of such functions, the inverse barrier function. (Although some of its conclusions can be extended to other type of penalty functions.)