CHAPTER 2

SURVEY OF LITERATURE ON REDUNDANCY IDENTIFICATION

2.1 INTRODUCTION

Structural redundancies in mathematical programming problems such as linear, integer and non-linear programming are nothing uncommon but the main attention in this research work has been directed towards linear programming. This study has been taken up due to the abundant applications of Linear Programming in the fields of engineering and science. Although the presence of embedded structural redundant constraint(s) / variable(s) in LPP does not alter the optimal solution of a given problem, they consume additional computational effort and occupy more memory. In large scale LPP's, these redundant constraint(s) / variable(s) innocently and inadvertently introduced can play havoc with the existing solution procedures and bring down the computational efficiency. Thus it is necessary to detect these embedded redundancies a priori before seeking optimal solution, and hence this places a heavy emphasis and demand on the development of efficient and effective identification techniques. This motivated the author to develop a few improved algorithms for a priori detection of redundancies by modifying some of the earlier researcher's algorithms.
2.2 TYPES OF CONSTRAINTS

Consider the Linear Programming model
Extremize $Z = CX$
subject to $AX \{\leq, =, \geq\} b; X \geq 0$ with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^n$ and $X \in \mathbb{R}^n$ (2.1)

Several researchers have defined redundant constraints for a given LP model in different ways. The definition for redundant constraint proposed by Natesan (2000) and definitions for strongly, weakly and active constraints enunciated by Karwan (1982) are given here.

2.2.1 Redundant Constraints

One or more constraints of $\sum_{j=1}^{n} a_{ij}x_j \{\leq, =, \geq\} b_i$
for $i=1,2,...,m; \ x_j \geq 0$ is redundant if and only if its associated surplus or artificial variable(s) $s_i / r_i$ are in every primary subsystem in the course of finding an optimal solution for the LP problem. In other words, if a surplus / artificial variable consistently stays in the basis in all the iterations with a value ($\geq 0$), then the associated constraint is redundant for the LP model.

2.2.2 Strongly Redundant Constraints

A constraint of $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ for $i=1,2,...,m; \ x_j \geq 0$ is said to be strongly redundant in system (2.1) if and only if its associated surplus variable $s_i > 0$. 
2.2.3 Weakly Redundant Constraints

A constraint of \( \sum_{j=1}^{n} a_i x_j \leq b_i \) for \( i=1,2,...,m; \ x_j \geq 0 \) is said to be weakly redundant in system (2.1) if and only if its associated surplus variable, \( s_i = 0 \).

The concepts of strong and weak redundancies are illustrated in figure 2.1. However, in the course of computations, it was found that strong redundancies appear more frequently than weak redundancies. A weak redundancy results in a degenerated solution.

2.2.4 Active Constraints

Constraints, which are not redundant, are called non-redundant or active constraints. In other words, constraints that play a vital role in the formation of a feasible set are referred to as active constraints.

2.3 REDUNDANT VARIABLES

A decision variable is redundant if and only if it has remained consistently as non-basic in the course of finding an optimal solution for a LP problem. By combining the primal-dual properties, redundant variables are also defined alternately as follows. For every primal variable there exists an associated dual constraint. If a dual constraint is redundant, then the associated primal variable is also redundant.
2.4 CONSEQUENCES OF REDUNDANCY

Whenever model builder models a large scale Linear Programming Problem the addition of an unintentional set of structural redundancies as constraints and variables due to inadvertency is not uncommon. Redundancy may occur in the formulation phase because of either bad source data or reservation in taking risk of omitting some relevant constraints while modeling a problem. The consequences of such embedded redundant constraint(s) / variable(s) given by Karwan (1982) are:

a) The dimension of the problem becomes large.

b) The sheer presence gives an impression that it has some influence on the model.
c) Selecting a Mathematical Programming model may be confusing.
d) More number of iterations and consequent computations are required.
e) Unnecessary repetitions of already processed information.
f) Can cause degeneracy.
g) Tends to conceal certain information and possibilities.
h) Might lead to different decisions.

Thus in a general mathematical programming problem, the unfavourable effects of the presence of redundancy usually outnumber the favourable ones.

2.5 NEED FOR REDUCED MODELS

It is very difficult to determine how much redundancy is present in a LP model until it is solved. Thus it is always a mute question whether it is worth spending time to identify redundancy when there are sophisticated computers to solve large scale LP problems in less time.

The existence of redundancies in a LP model can not only play havoc in solution procedures but also mislead good decisions, which can indirectly affect the organization. It is therefore, worth spending time on eliminating redundancy in a model instead of simply ignoring it. The cost of eliminating redundancy can be calculated by three options *viz.*

a) Ignore  b) identify redundancy  c) Identify and remove redundancy.

This paved a way for many researchers to develop algorithms for the identification of redundancies and remove them to get a reduced model. There always had been a need for developing simple algorithms to detect *a priori* redundant constraints at minimal computational effort. The main objective of these algorithms had been to remove all redundancies in a LP
model, which is not always possible in practice. This has led to many improvements in developing algorithms to identify as many redundant constraints as possible in a given problem.

2.6 EARLIER RESEARCH CONTRIBUTIONS

Early research contributions have proposed methods dating from 1955 to 2000 for minimizing the computational effort by identifying redundant constraints. The first paper that made reference to redundancy was primarily on some other topic and included results relating to redundancy which were obtained as a by-product. Dantzig (1955), for example, suggested using any available \textit{a priori} information in linear programming to predict the solution; then everything that is redundant with respect to this solution can be omitted from the solution process. A similar approach was followed by Sethi and Thompson (1982) who used mathematical information to make prediction on the solution and check their predictions at every stage.

Balinsky (1961) determines redundancy obtained as a by-product by using all extreme points of a convex polyhedron. However, the number of extreme points grows exponentially with the dimension of the problem and thus this approach is very cumbersome for very large-scale problems.

Basically, the same approach to redundancy identification was followed by Shefi (1969) (also by Luenberger (1973)), who developed a different algorithm for determining all extreme points. Apart from that he established certain minimality properties for systems of linear
constraints. Telgen (1979) generalized the above work into a minimal representation theory. Mattheiss (1973) also developed an algorithm to find all extreme points of a convex polyhedron. The objective was to minimize the number of required calculations by embedding the convex polyhedron in a space of higher dimension.

Boot (1962) published the first paper devoted entirely to redundancy identification and suggested checking the redundancy of an inequality by replacing the inequality sign of a constraint by a strict inequality in the reverse direction. If the resulting system is consistent, the constraint is non-redundant. A disadvantage of this method is that systems of linear constraints have to be first checked for feasibility in order to check a constraint for redundancy.

In a major contribution Zionts (1965) improved upon the implementation of Boot’s method, but not to the point where it attained practical value. Furthermore, a number of other approaches were developed therein, among which the geometric definition method was the most prominent. Zionts (1972) and Rubin (1972) extended certain concepts of redundancy to integer programming.

Lisy (1971) employed the rules developed by Zionts to identify all redundant constraints in a system of linear constraints. Gal (1975, 1978) elaborated on this approach by adding rules for situations in which constraints can be identified immediately as being non-redundant. Telgen (1979) further extended this approach by adding rules to deal with degeneracy and the possibility of redundant constraints passing through an extreme point. It was also shown that the rules defined by
Llewellyn (1964) and Zeleny (1974) are valid only for positive coefficients and other very special cases.

Greenberg (1975) developed a method for finding the extreme rays of convex polytopes. A flaw in Greenberg’s method in the case of degeneracy was pointed out by Sherman (1977) but later corrected in Dyer and Proll (1980). However, in a computational comparison, Dyer and Proll (1977) showed that Mattheiss method generally outperformed Greenberg’s method.

A totally different approach is given by Boneh and Golan (1979). This method consists of extending a ray from an interior point in a randomly chosen direction. Any constraint hit from the inside cannot be redundant. After a large number of trials, all constraints, which have not been hit so far, are declared to be redundant.

Tomin (1985) presented an algorithm for identifying and disposing of duplicate rows in LP problem, that is, the rows which are identical except for a scalar multiplier. It only detects redundancies of a particular type. In the same direction Bradley (1983) and Williams (1983) developed reduction procedures for linear and integer programming models. They applied this to large-scale optimization problems. This has been included in Mathematical Programming System (MPS) as presolvers. Bixby (1987) has also presented an algorithm for identifying duplicate rows in linear programming problem.

Many researchers like Mitchel (1986), Ye (1990), Goffin (1990) and Imbert (1992) have proposed a strategy for reducing the computational effort in this direction and solving linear programming
problems. Gondzio (1997) discussed the pre-solve procedure of detection and removal of different linear dependencies of rows and columns in a constraint matrix. Since it is not allowing apparent redundancy initially, it detects only a few redundancies and spends more computational efforts.

Imbert (1995) proposed a method for the reduction in the number of linear constraints using a lexicographic solved form. The lexicographic solved form is the basis of an incremental decision procedure, based on Simplex algorithm, for system of equalities and inequalities. It enjoys the two important properties of precluding syntactically the presence of implicit equalities and being preserved by pivoting under a lexicographic pivoting rule. This work shows that the lexicographic solved form helps to identify the new syntactic criterion for detecting redundant and non-redundant constraints. But this method uses more computational steps to split the set of linear inequalities as redundant and non-redundant. It detects 24 to 56 percent of the actual redundant constraints. Imbert (1995), Alziary (1994) and McAloon (1998) proposed similar approaches to eliminate the redundant linear arithmetic constraints.

Sometimes an optimization problem can be simplified to a form that is faster to solve. Indeed, sometimes it is convenient to state a problem in a way that admits some obvious simplifications, such as eliminating fixed variables and removing constraints that become redundant after simple bounds on the variables have been updated appropriately. Based on the work of Brearley (1975), Robert Fourer and David.M.Gay (1994) describe a pre-solver, discuss reconstruction of dual values for eliminated constraints and present some computational results.
Paulraj (1998) has developed a heuristic algorithm to identify \textit{a priori} redundant rows in a given problem with linear inequalities. Another heuristic model reduction approach proposed by Rhymend (1999) makes use of the matrix of intercepts. This algorithm identifies as many redundant constraints as possible for minimization and maximization objective with lower and upper bound constraints respectively.

Karwan (1982) edited a volume of lecture notes on Redundancy in Mathematical Programming, which gives the definitions and classification of various types of redundancy in mathematical programming, review of methods for handling redundancy and comprehensive tests of the various methods, together with extensions and further developments of the most promising methods. Some collection of articles and contributions to redundancy were also edited by Natesan (2000) for a short term course on New Approaches to Resource Management. This also suggests methods for \textit{a priori} identification of redundancies using the Simplex and the Multiplex algorithms.

The present work addresses a different strategy for identifying redundancies in LP model using constraint gradient matrix algorithm and matrix of intercepts algorithms proposed by Rhymend (1999). The conjunctive heuristic algorithm proposed in this work detects some of the redundancies, which escaped detection by the earlier algorithms developed by Paulraj (1998) and Rhymend (1999).
2.7 DESCRIPTION OF THE MODEL REDUCTION ALGORITHMS

Algorithms for an *a priori* identification and disposal of redundancies in a Linear Programming model have been developed to quicken the process of finding solution for a given LP problem and save computational effort. In this work the earlier researcher's heuristic model reduction algorithms, *viz* - Constraint Gradient Matrix and Matrix of Intercepts algorithms have been used as components to develop an integrated algorithm to identify those redundancies which escaped detection.

2.7.1 Constraint Gradient Matrix Algorithm

The following are the steps for *a priori* identification of redundancies using the constraint gradient matrix algorithm:

**Step 1:** A gradient matrix(Vgij) is constructed by taking the transpose of the coefficient matrix Amxn containing the decision and surplus variables as rows and columns respectively.

**Step 2:** Identify the largest element in each row, \( \beta_j = \max \{ Vgij \} \) for all j

**Step 3:** Score out the rows and columns corresponding to the entering and leaving variables. If a column has more than one largest element, score out those rows also.

**Step 4:** The constraints associated with the surplus variables in the unscored columns, if any, *ab initio* are assumed and predicted to be redundant.
**Step 5:** Remove these redundant constraints tentatively from the original model.

The application of the above steps results in a reduced model.

### 2.7.2 Matrix of Intercepts – Theta Max. Algorithm

The following are the steps of the algorithm.

**Step 1:** A matrix of intercepts of order nxm is constructed with the decision and surplus variables as rows and columns respectively.

\[ \theta_{ji} = \frac{b_i}{a_{ij}} ; \quad a_{ij} > 0 \text{ for all } i=1,2,...,m \text{ and } j=1,2,...,n \]

**Step 2:** Identify the largest element in each row

\[ \beta_j = \max \{ \theta_{ji} \} \text{ for all } j, \text{ while the objective is minimization and constraints are lower bound.} \]

**Step 3:** Score out the rows and columns corresponding to the entering and leaving variables. If a column has more than one largest value, score out those rows also.

**Step 4:** The constraints corresponding to the surplus variables in the unscored columns, if any, are assumed and predicted to be redundant.

**Step 5:** Remove these redundant constraints tentatively from the original model.

The application of the above steps results in a reduced model.
2.7.3 Matrix of Intercepts – Theta Min. Algorithm

The following are the steps of the algorithm.

**Step 1:** A matrix of intercepts of order nxm is constructed with the decision and surplus variables as rows and columns respectively.

\[ \theta_{ji} = \frac{b_i}{a_{ij}} \text{ ; } a_{ij} > 0 \text{ for all } i=1,2,\ldots,m \text{ and } j=1,2,\ldots,n \]

**Step 2:** Identify the smallest element in each row

\[ \beta_j = \min \{ \theta_{ji} \} \text{ for all } j, \text{ while the objective is maximization and constraints are upper bound.} \]

**Step 3:** Score out the rows and columns corresponding to the entering and leaving variables. If a column has more than one smallest value, score out those rows also.

**Step 4:** The constraints associated with the surplus variables in the unscored columns, if any, are assumed and predicted to be redundant.

**Step 5:** Remove these redundant constraints tentatively from the original model.

The application of the above steps results in a reduced model.

2.8 CONCLUSION

The literature survey hints that there exists a need for evolving a simple and efficient method of detecting the redundancies in a Linear Programming problem prior to solving the problem. Though there has been much work carried out to reduce the original dimension of the
problem, most of them are computationally expensive. The recently developed heuristic algorithms - Constraint Gradient Matrix algorithm and Matrix of Intercepts algorithm, have shown significant improvement in the identification of redundancies with minimal effort. But these individual algorithms, when applied to an LP model do not identify all redundancies leaving scope for further improvement. These algorithms taken together have shown some promise for the development of a conjunctive algorithm to identify more redundancies than what the individual algorithm did. The subsequent chapters describe the development methodology of an integrated approach for detecting the redundancies, which escaped detection earlier.