A COMPUTATIONALLY EFFICIENT PROCEDURE FOR DATA ENVELOPMENT ANALYSIS

Srinivasan Parthasarathy
Operational Research Group, Department of Management
London School of Economics and Political Science

Thesis submitted to the London School of Economics and Political Science, for the degree of Doctor of Philosophy

July 2010

THESES
F
9375

1263969

The author hereby declares that the work presented in this thesis is his own.

Srinivasan Parthasarathy

ABSTRACT

This thesis is the final outcome of a project carried out for the UK's Department for Education and Skills (DfES). They were interested in finding a fast algorithm for solving a Data Envelopment Analysis (DEA) model to compare the relative efficiency of 13216 primary schools in England based on 9 input-output factors. The standard approach for solving a DEA model comparing n units (such as primary schools) based on m factors, requires solving $2n$ linear programming (LP) problems, each with m constraints and at least n variables. At $m = 9$ and $n = 13216$, it was proving to be difficult.

The research reported in this thesis describes both theoretical and practical contributions to achieving faster computational performance. First we establish that in analysing any unit t only against some critically important units – we call them generators – we can either (a) complete its efficiency analysis, or (b) find a new generator. This is an important contribution to the theory of solution procedures of DEA. It leads to our new Generator Based Algorithm (GBA) which solves only n LPs of maximum size $(m \times k)$, where k is the number of generators. As k is a small percentage of n, GBA significantly improves computational performance in large datasets. Further, GBA is capable of solving all the commonly used DEA models including important extensions of the basic models such as weight restricted models.

In broad outline, the thesis describes four themes. First, it provides a comprehensive critical review of the extant literature on the computational aspects of DEA.

Second, the thesis introduces the new computationally efficient algorithm GBA. It solves the practical problem in 105 seconds. The commercial software used by the DfES, at best, took more than an hour and often took 3 to 5 hours making it impractical for model development work.

Third, the thesis presents results of comprehensive computational tests involving GBA, Jose Dula's BuildHull – the best available DEA algorithm in the literature – and the standard approach. Dula's published result showing that BuildHull consistently outperforms the standard approach is confirmed by our experiments. It is also shown that GBA is consistently better than BuildHull and is a viable tool for solving large scale DEA problems.

An interesting by-product of this work is a new closed-form solution to the important practical problem of finding strictly positive factor weights without explicit weight restrictions for what are known in the DEA literature as "extreme-efficient units". To date, the only other methods for achieving this require solving additional LPs or a pair of Mixed Integer Linear Programs.

ACKNOWLEDGEMENTS

I recall Professor Abraham Charnes' advice to aspiring PhD students to choose the teacher and not the topic. Even though I got to hear about this recently, I am glad that I followed it. I started my PhD with absolutely no knowledge on the topic but had the trust that I chose my teacher well. I am indebted to my supervisor, Professor Gautam Appa for giving me the opportunity to study for a PhD under his supervision with financial support. I am also indebted to him for his unfailing academic rigour and continuous encouragement. I cannot fail to mention his patience and attentive attitude towards me all these years for I tested him to his extreme emotions on numerous occasions. He was and remains a pillar in my life.

I am deeply indebted to Dr. Susan Powell for helping me with my writing style and for her attention to detail. I owe much to her patience and guidance in shaping this thesis.

I am also indebted to Professors Rajiv Banker and Ram Natarajan for inviting me to collaborate with them during my PhD study.

I am grateful to Professors Jose Dula and Emmanuel Thanassoulis for their patience and assistance with many of my persistent queries on various theoretical aspects of DEA.

I am grateful to Professor Ailsa Land for her feedback on the computational aspects of my thesis and to Mr. George Mitchell on the preface.

I would like to thank the OR department manager Brenda Mowlam, and the administrators Jenny Robinson, Richard Szadura and Lucy Underhill for their patience and help with all sorts of administrative requests over the years.

For their continued patience, help and friendship, I thank my fellow PhD students, Ionna Katrantzi, Kostas Papalamprou, Nikos Argyris, Nayat Horzoglou, Anastasia Kouvela, Dimitrios Karamanis, Kai Becker, Shweta Agarwal, Ana Barcus and Sumithra Sribashyam.

Thanks are due to my friends over many years - Cristiana, Anupama, Ernie, Leong, Lena, Emily, Diana, Arjen, Ofer, Joelle, Lida, Maha and many more – for making my life much more pleasant in many ways.

Lastly I thank my parents, athimber and athai, my sister and my brother Murari for their continued love and support towards me.

Financial support from the Department for Education (DfE) via grant EOR/SBU/2003/208 is gratefully acknowledged.

DEDICATION

The thesis is dedicated to my grandparents V. S. Ranganathan and R. Seetha.

PREFACE

I Motivation and Genesis

The issue addressed in the work recorded in this thesis is the computational efficiency in applying Data Envelopment Analysis (DEA) and how this may be improved.

In 2005, the Value for Money (VfM) unit of the Department for Education and Skills (DfES)[1] commissioned my supervisor, Professor Gautam Appa, to develop procedures for speeding up computation of DEA models for large scale datasets. The DfES had decided to use DEA to identify the well-run primary schools in England and to provide benchmarks based on these for the poorly-run ones. DEA divides the units (primary schools in this case) under investigation into efficient and not efficient and finds peers amongst the efficient ones which are relevant for setting targets for the inefficient ones. So in some sense DEA was suitable for their purpose. However, in carrying out DEA computations they encountered one big difficulty. The performance analysis software they were using for processing DEA datasets, PIM DEASoft-v2 (http://www.deasoftware.co.uk/), took too long to solve datasets with more than 5000 units and sometimes required multiple attempts to run them to completion. Professor Appa was given a grant to appoint a PhD student to review current computational methods and come up with improved ones. In September 2005, he drafted me as his PhD student and it was agreed that under grant EOR/SBU/2003/208 from the DfES, my research would review the extant literature on the different solution procedures to process DEA datasets and develop new techniques to realize improved computational efficiency.

It is worth noting that although the issues discussed in this thesis arose in connection with comparing the performance of primary schools, they could equally have arisen in many other contexts which involve large datasets (for example, in comparing the performance of branches of a bank, in financial

[1] It went through several changes of name; at the time of writing it is known as Department for Education (DfE).

applications such as portfolio analysis etc.). The methods developed herein therefore can be expected to have wide applications.

It is also important to note that there are certain applications in DEA that necessitate solving multiple DEA LPs for each unit and are computationally intensive even for medium scale datasets. These include the *bootstrapping* technique developed in Simar and Wilson (1998) and Simar and Wilson (2000), *outlier identification* technique developed in Wilson (1993), Kuosmanen and Post (1999), Simar (2003) and Banker and Chang (2006), various methods to estimate the *returns to scale* of units in DEA developed in Banker and Thrall (1992), Fare and Grosskopf (1994) and Banker et al (2004), and methods to estimate the productivity growth using the *malmquist productivity analysis* technique developed in Fare et al (1994), Ray and Desli (1997), Simar and Wilson (1999) and Banker et al (2010). The algorithm developed in this thesis is also expected to come useful in reducing the computational work in such intensive applications.

II Foundation and Development

Upon reviewing the literature on the computational aspects of DEA, only three strands of relevant research were identified. These (discussed in detail in Chapter 3) comprised of:

1. the DEA based pre-processing ideas of Ali (in Ali (1993) and Chen and Ali (2002)),
2. the hierarchical decomposition algorithm of Barr and Durchholz (1997), and
3. the BuildHull algorithm of Dula (1998).

Among these, only the works of Professor Jose Dula seemed significant as he had shown by extensive computational evidence that his BuildHull algorithm was much faster than all the others. In September 2006, the BuildHull algorithm proposed in Dula (1998) was presented to the VfM unit of the DfES along with suggestions to improve it. In pursuing some of these ideas further we came up with one that kept the main feature of BuildHull but cut down the number of LPs to be solved by half.

To evaluate any unit in DEA, it is sufficient to have data on its relevant peers from the dataset. Hence, comparing an unit with *all* the units in the dataset is unnecessary. This is especially important when a vast majority of units are inefficient and hence well known to have no relevance in the evaluation of any other unit. Most real life datasets have this feature. For example, the number of relevant efficient units in the primary school data was only 188 out of 13,216. We will call the relevant efficient units generators for now and define the term precisely later. The main strength of BuildHull comes from the fact that it identifies all the generators in a first pass where each unit is tested against already identified generators to see if it is inefficient in comparison. If it is, it can be discarded; if not, BuildHull has a way to find some hitherto undiscovered generator. So it requires n LPs to find all the generators (e.g., 188 schools) in a dataset and no LP will have more than $k+1$ units in it if there are k generators (here, 188 schools). For the primary schools problem, BuildHull will solve 13216 LPs with no more than 189 variables in each LP in the first phase. In phase 2, BuildHull will solve one LP for each unit which is not a generator, with only the k generators and the unit itself in each LP; so 13028 LPs with 189 variables in each.

The improvement we make is to find a way to work only with known generators but in such a way that at each step we either finish the analysis of unit t under investigation or find a new generator, thus needing to solve only n LPs with no more than k variables at any step. The main tool for achieving this is the use of super-efficiency model of DEA (introduced in Andersen and Petersen (1993)) and a ratio test to find a new generator.

In preparing the fine details of this Generator Based Algorithm (GBA) we had to deal with some technicalities. Thus, the super-efficiency model may lead to infeasible LPs and the ratio test could produce indeterminate ratios. Also, when choosing a new generator ties were encountered which had to be dealt with.

In the first instance we were able to deal with these finer points by assuming them away. So we assumed that the data entries were all positive, which took care of both the infeasibility and the indeterminate ratio problems. And ties were waved aside by assuming that there was a way to solve them. But eventually we had to solve these problems head on. It was found that indeterminate ratios were only problematic when there were ties and so can be handled by our tie breaking procedure. Fortunately we were able to find a closed form solution to the problem of breaking ties. It turns out that the same closed-form solution can also be used to find strictly positive optimal weights for all the factors for each generator.

III Organization of the thesis

Chapter 1 starts with an introduction to the DEA technique familiarising the reader with some of the important concepts by means of a graphical illustration, followed by a systematic investigation. We then give a brief history of the evolution of DEA.

Chapter 2 presents the standard DEA LP models, viz., constant and variable returns to scale models under input and output orientation and additive models, used in DEA applications. The chapter also provides a brief critical review of an important variant of these, namely, the super-efficiency models, which we employ in the construction of our new algorithm developed in chapter 4.

To reduce the computational strain in processing DEA datasets, various heuristics and alternative efficient algorithms have been developed in the literature over the years. Chapter 3 presents a comprehensive critical review of these, identifying the three main strands mentioned earlier.

Chapter 4 presents our main research contribution, the Generator Based Algorithm (GBA), for solving the input oriented CRS model under the assumption that the dataset is strictly positive. Every step is illustrated graphically for a small 2-input, one output case with 8 Decision Making Units (DMUs). Appendix 2 graphically illustrates the workings of the same example for Dula's BuildHull algorithm. In contrast to BuildHull, our algorithm avoids solving a second LP for the 8 DMUs.

Chapter 5 addresses the technical challenges in using GBA for general (not necessarily positive) datasets for all the standard DEA models described in chapter 2. For each specific DEA model, we present conditions under which the two main technical challenges of infeasibility and indeterminate ratios can or cannot occur when applying GBA.

The purpose of chapter 6 is to present ways to handle the technical challenge of infeasibility. We examine two different approaches to handle it, viz., clustering and penalty. Within the penalty method, we examine two different techniques, viz., employing a big penalty and a small penalty. The clustering technique works under restrictive conditions while the penalty methods can tackle infeasibility under all circumstances.

In chapter 7, we examine ways to deal with the remaining technical difficulties of GBA. We first show that within GBA the problem of indeterminate ratios is only relevant when there is a tie in the rule for finding a new generator. Then we provide novel closed-form solutions to resolve ties. Finally we extend these closed-form solutions to construct a strictly positive set of multiplier values for the generators.

Chapter 8 presents the computational results of processing datasets using various DEA models. While doing so, we compare the computational performance of GBA against BuildHull and the conventional solution procedure. For VRS models we compare the computational performance of GBA against the other two using the problem suite that Dula (1998, 2008, 2010) employed in his studies. As Dula does not provide any comparisons for CRS models we use a problem suite developed for this purpose. As the main contender for GBA is BuildHull, we give separate diagrams comparing just these two.

Finally, chapter 9 present directions for future research.

GLOSSARY OF THE TERMS AND SYMBOLS

DMU - Decision Making Unit.

(DMUs denote the plural form of DMU and not the sth DMU)

Units - Refers to DMUs.

PPS - Production Possibility Set.

RTS – Returns to Scale.

CRS - Constant Returns to Scale.

VRS - Variable Returns to Scale.

n - Number of observed decision making units.

m_1 - Number of input factors in the dataset.

m_2 - Number of output factors in the dataset.

(X,Y), $(\overline{X},\overline{Y})$, $(\tilde{X},\tilde{Y})$ - Activity of an unit with support over $R_+^{m_1+m_2}$.

X - Input component (vector) of the activity (X,Y) with support over $R_+^{m_1}$.

Y - Output component (vector) of the activity (X,Y) with support over $R_+^{m_2}$.

(X_j,Y_j) - Activity of an observed unit j.

(X_t,Y_t) - Activity of an observed unit (DMUt) that is currently being evaluated.

X – Matrix of inputs of all the observed units of size $m_1 \times n$.

Y – Matrix of outputs of all the observed units of size $m_2 \times n$.

λ_j or μ_j - Intensity variable of an observed unit j.

s^i - Input slack vector with support over $R_+^{m_1}$.

s^o - Output slack vector with support over $R_+^{m_2}$.

v - Input weight vector with support over $R_+^{m_1}$.

u - Output weight vector with support over $R_+^{m_2}$.

θ - Variable that depicts the input efficiency of DMUt.

ϕ - Variable that depicts the output inefficiency of DMUt.

0 - Zero vector with dimension decided by the context.

e - Vector of 1s with dimension decided by the context.

Cardinality – Number of observed DMUs in a DEA exercise.

Dimension – Number of inputs and outputs in a DEA exercise.

Density –Percentage of extreme-efficient units in a DEA exercise.

P-K efficient – Pareto-Koopmans efficient.

GBA – Generator Based Algorithm.

LP – Linear Programming.

MILP – Mixed Integer Linear Programming.

DGP – Data Generating Process.

EIE – Early Identification of Efficient units.

RBE – Restricted Basis Entry.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF CHARTS

LIST OF APPENDICES

1 INTRODUCTION TO DATA ENVELOPMENT ANALYSIS

1.1 Background

Data Envelopment Analysis (DEA) is a linear programming (LP) based technique that is used to determine the relative efficiency of homogeneous operating units responsible for converting inputs to outputs. The operating units, labelled in the DEA literature as Decision Making Units or DMUs, are similar in that they employ the same type of inputs to produce the same type of outputs. Conventionally, the purpose of a DEA exercise is to find the relative efficiency by which a DMU transforms its inputs to outputs when compared to other similar units. Relative efficiency, being a dimensionless scalar, does not require the various inputs and outputs to be measured in the same unit of measurement.

DEA is a non-parametric method in the sense that a functional form relating the inputs and outputs need not be specified apriori. It is also a frontier based method in that all the units are compared to the best practice units which also consume the same set of inputs to produce the same set of outputs.

Before we get into the theoretical framework of DEA, we will familiarise ourselves with some of the important concepts in DEA using a simple example. For ease of discussion, we will examine these concepts rather loosely and consign a rigorous treatise of them to section 1.2. Consider the following two inputs (*X1* and *X2*), one output (*Y1*), 9 DMU example portrayed in figure 1-1. The data for the example is provided in table 1-1.

DMU	***X1***	***X2***	***Y1***
A	2	9	1
B	4	6	1
C	6	3.5	1
D	10	2.5	1
E	12	2.5	1
F	2.5	13	1
G	9	5	1
H	5	6.5	1
I	8	3	1

Table 1-1 : 9 DMU, 3 factor data

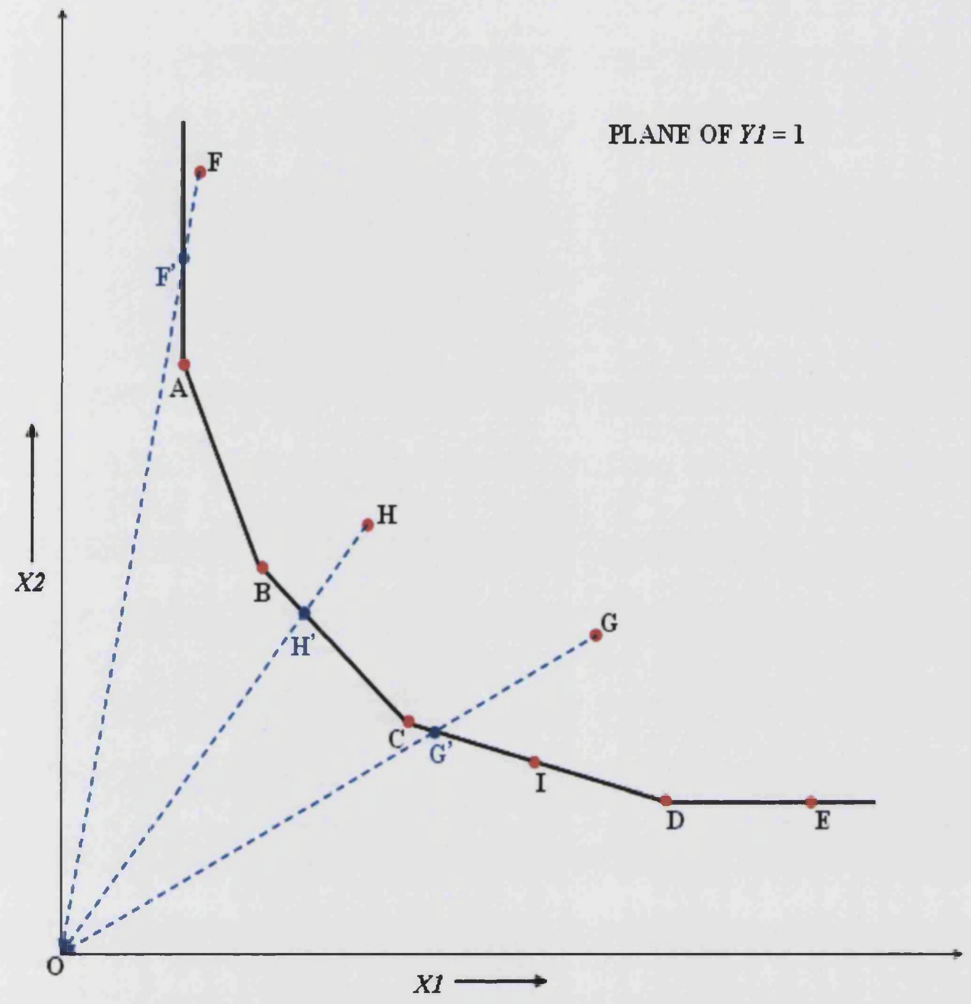


Figure 1-1 : 2-d representation of the data

To provide a context, one can think of the DMUs as different comparable universities operating in a city. Input 1 (*X1*) could represent the number of administrative staff and Input 2 (*X2*) could represent the number of research and teaching staff employed in an university. Output 1 (*Y1*) could be the number of registered students studying in the university.

We will assume that all the universities are operating in a constant returns to scale environment which means that a doubling (halving) of all the inputs (*X1* and *X2*) leads to a doubling (halving) of all the outputs (*Y1*). Hence, the number of registered students is scaled to 1 student and table 1-1 shows the number of administrative, and research and teaching staff required in different universities to manage a single student.

As the universities can be expected to have more control over their inputs than their outputs, we will assume that we are interested in knowing the input efficiency of these universities. This means that we are interested in knowing the minimum proportion of an university's current input usage that is sufficient, in comparison with other universities, to output a single student if it were to carry out its operation efficiently. If this minimum proportion is equal to 1 for an university then it is operating efficiently relative to the other universities.

The inputs-output values of the 9 universities represent coordinates of points in 3 dimensional (2 inputs + 1 output) space. As the output value is scaled to 1, one can represent all the 9 points in the $Y1 = 1$ plane using the $X1$ and $X2$ axes alone. In figure 1.1, the points (universities) A through I are plotted in red and the thick black line passing through the universities A, B, C, I, and D represents the efficient frontier. The extended frontier includes the vertical line north of A and the horizontal line east of D. Unit E lies on the extended frontier.

Universities A, B, C, I, D and E are lying on the extended frontier and are called boundary units. Given our empirical evidence, these universities cannot reduce their input usage any further proportionately and still be able to output 1 student. Hence, their input efficiency is 1. Universities F, G, and H are not lying on the extended frontier. These universities can reduce their input usage proportionately and still be able to output 1 student. Hence, their input efficiency is less than 1.

Among the boundary units, universities A, B, C, I and D are of special interest. For these universities, maintaining $Y1 = 1$, the usage of either of its input factors cannot be improved (decreased) any further without worsening (increasing) the usage of the other. Hence, universities A, B, C, I and D are said to satisfy the Pareto-Koopmans efficiency criterion. These units are the best practice units and other universities must hold them as benchmarks to improve their performance. University E, although a boundary unit with an input efficiency of 1, does not satisfy the Pareto-Koopmans efficiency criterion. This is because its input 1 usage can be reduced (improved) when compared to unit D without worsening its usage of input 2 while maintaining $Y1 = 1$. The peer unit or the benchmark for university E in order to improve its performance is university D.

The units that are lying on the extended frontier can be classified into three types of units, viz., extreme-efficient, efficient but not extreme, and weakly efficient. Universities A, B, C, and D lie on the efficient frontier and are called extreme-efficient units defined by the criterion that if one were to remove any one of these four units, the contour of the frontier will change. It is obvious that these extreme-efficient units (universities) satisfy the Pareto-Koopmans efficiency criterion. University I, which also satisfies the Pareto-Koopmans efficiency criterion is different. It can be expressed as a convex combination of units C and D, so its removal does not change the contour of the efficient frontier. Hence, university I is called an efficient but not extreme unit. Unit E, which has an input efficiency of 1, does not satisfy the notion of Pareto-Koopmans efficiency and is called a weakly efficient unit.

Lets us now consider the non-boundary units, F, G, and H. The input efficiency of university F is given by, $\theta_F = \frac{OF'}{OF} = \frac{10.59}{13.24} = 80\%$. This is the minimum proportion of inputs of university F that is sufficient to output a single student if it were to operate efficiently. The input 1 and input 2 usages at the boundary point F' are given by the input 1 and input 2 usages of university F scaled by 80%. In addition, the radial projection of unit F on the boundary, symbolized by F', is weakly efficient. This is because the point F' uses more of input 2 when compared to the extreme-efficient unit A. The slack (non-proportional or coordinate-wise inefficiency) present in unit F is given by the difference in the input 2 usage between points F' and A. The peer unit that university F must hold as benchmark in order to improve its performance is university A.

Let us consider unit G. The input efficiency of university G is given by $\theta_G = \frac{OG'}{OG} = \frac{7.1}{10.29} = 68.96\%$. The input 1 and input 2 usages at point G' are given by the input 1 and input 2 usages of university G scaled by 68.96%. The peer unit that university G must hold as benchmark in order to improve its performance is the virtual university G'. This virtual university G' can be obtained by a convex combination of the observed universities C and D. In particular, a combination of 94.8% of university C and 5.2% of university D synthesises this virtual university. The virtual university G' can also be obtained by a convex combination of the observed universities C and I. In particular, a combination of 89.6% of university C and 10.4% of university I can synthesise this virtual university. Hence, the peer units that university G must hold as benchmark to in order to improve its performance are the best practise universities C, D and I.

The input efficiency of university H is given by $\theta_H = \frac{OH'}{OH} = \frac{7.07}{8.2} = 86.27\%$. The input 1 and input 2 usages at point H' are given by the input 1 and input 2 usages of university H scaled by 86.27%. The peer unit that university H must hold as benchmark in order to improve its performance is the virtual university H'. This virtual university H' can be obtained by a convex combination of the observed universities B and C. In particular, a combination of 84.3% of university B and 15.7% of university C synthesises this virtual university. Hence, the peer units that university H must hold as benchmark in order to improve its performance are the best practise universities B and C.

Universities G and H are technically inefficient but their radial projection on the frontier does not contain any non-proportional inefficiencies. This is because their projection falls on the efficient frontier. This is in contrast to university F that does contain non-proportional inefficiencies. In the DEA jargon, universities G and H are technically inefficient but mix efficient units. University F, on the other hand, is both a technically inefficient and mix inefficient unit. University F is mix inefficient as its radial projection on the extended frontier contains non-proportional inefficiency.

In figure 1-1, the vertical line above A, lines A-B, B-C, C-D, and the horizontal line to the east of D are called facets. Facets provide the relative values (or weights) for the input and output factors for the units that are evaluated using that facet. For example, the line B-C provides the relative values for the input and output factors for the three universities, B, C and H, that are evaluated using that facet.

The input efficiency of an university can either be obtained geometrically as a ratio of radial distances or by computing the ratio of weighted outputs to weighted inputs. The equation of the line (facet) B-C is given by $0.114X1 + 0.091X2 = 1$. Thus the relative value or weight for the input 1 factor is 0.114 and weight for input 2 factor is 0.091. The weight for the single output $Y1$ is fixed at 1 as we have plotted the points in the plane of $Y1 = 1$. The input efficiencies of universities B, C and H that are evaluated at this facet can now be computed by,

$$\theta_B = \frac{\textit{weighted sum of outputs}}{\textit{weighted sum of inputs}} = \frac{(1\times1)}{(0.114\times4+0.091\times6)} = \frac{1}{1} = 100\%,$$

where, the input 1 and input 2 values for university B are 4 and 6 respectively.

Similarly, $\theta_C = \frac{(1\times1)}{(0.114\times6+0.091\times3.5)} = \frac{1}{1} = 100\%$, and

$$\theta_H = \frac{(1\times1)}{(0.114\times5+0.091\times6.5)} = \frac{1}{1.159} = 86.27\%.$$

1.2 Theoretical Framework

The DEA measures of technical efficiency as introduced in Charnes et al (1978) and Banker et al (1984) are operational extensions of the Debreu-Farrell measures referred as such after the works of Debreu (1951) and Farrell (1957). Debreu (1951) and Farrell (1957) introduced a measure of technical efficiency based on Koopmans' (1951) definition of technical efficiency. Given its likeness to the notion of Pareto optimality introduced by Pareto (1906), Koopmans' (1951) efficiency criterion is also referred to as Pareto-Koopmans efficiency criterion. We will examine the connection between the Debreu-Farrell measures and Koopmans' definition of technical efficiency and while doing so, discuss Shephard's (1953, 1970) important contributions to the topic. In order to do this, we will formally introduce concepts such as production technology, production possibility set, and input and output sets.

Throughout this section we are considering n observed decision making units with each unit utilising m_1 inputs to produce m_2 outputs. The inputs and outputs are non-negative with at least one positive component in any unit's input and output vector, i.e., for $j=1,\dots,n, X_j, Y_j \geq 0; X_j, Y_j \neq 0$;[2] where X_j represents the input vector of DMUj of dimension m_1 and Y_j represents the

[2] Inequality or equality symbol between vectors implies that the relationship holds for each component of the vectors.

output vector of DMUj of dimension m_2[3]. Also, (X_j, Y_j) denotes the observed activity of the jth DMU.

For any DEA exercise, the description of the Production Technology or Production Possibility Set, PPS, is paramount. Formally, the PPS is defined as the set of technologically feasible input and output activities $(X, Y) \in R_+^{m_1+m_2}$ represented as $T = \{(X,Y) \mid Y \geq 0 \textit{ can be produced from } X \geq 0, X \neq 0\}$. The components of an activity can be regarded as the coordinates of a point in the non-negative orthant of the $(m_1 + m_2)$ dimensional space. The PPS is assumed to satisfy some basic postulates that we discuss next.

1.2.1 Production Possibility Set under Constant Returns to Scale

All the n observed units are assumed to operate under a constant returns to scale environment. Formally, the CRS assumption implies that for every $(X,Y) \in T, (\alpha X, \alpha Y) \in T, \forall \alpha \geq 0$.

The postulates satisfied by the production possibility set under the constant returns to scale assumption are as follows.

1. *Observed unit postulate* – All the observed units (X_j, Y_j), $j = 1, \ldots, n$, belong to T.
2. *Free disposability or Inefficiency postulate* – For any $(X_j, Y_j) \in T$, all $(X'_j, Y_j) \in T$ where $X'_j \geq X_j$ and all $(X_j, Y'_j) \in T$ where $Y'_j \leq Y_j$.
3. *Ray unboundedness postulate* – For all non-negative scalars $\lambda_j \geq 0$, $\left(\sum_{j=1}^{n} \lambda_j X_j, \sum_{j=1}^{n} \lambda_j Y_j \right) \in T$.

[3] Depending on the context, X_j can be a column vector of dimension m_1 with X_{rj} denoting the rth input component of DMUj. Similarly, Y_j can be a column vector of dimension m_2 with Y_{sj} denoting the sth output component of DMUj.

The smallest polyhedral set that satisfies the above three postulates is the production possibility set under constant returns to scale assumption and can be represented as, $T_c = \left\{ (\widetilde{X},\widetilde{Y}) \middle| \, \widetilde{X} \geq \sum_{j=1}^{n} \lambda_j X_j, \widetilde{Y} \leq \sum_{j=1}^{n} \lambda_j Y_j, \ \lambda_j \geq 0 \right\}$.

1.2.2 Production Possibility Set under Variable Returns to Scale

DEA literature also looks at PPS under the assumption of variable returns to scale which allows for a production technology exhibiting increasing, decreasing and constant returns to scale. The postulates satisfied by the production possibility set under the variable returns to scale assumption are as follows.

1. *Observed unit postulate* – All the observed units (X_j, Y_j), $j = 1,\dots,n$, belong to T.

2. *Free disposability or Inefficiency postulate* – For any $(X_j, Y_j) \in T$, all $(X'_j, Y_j) \in T$ where $X'_j \geq X_j$ and all $(X_j, Y'_j) \in T$ where $Y'_j \leq Y_j$.

3. *Convexity postulate* - For all non-negative scalars $\lambda_j \geq 0$ such that $\sum_{j=1}^{n} \lambda_j = 1$, $\left(\sum_{j=1}^{n} \lambda_j X_j, \sum_{j=1}^{n} \lambda_j Y_j \right) \in T$.

By replacing the ray unboundedness postulate with the convexity postulate, the above production possibility set allows for different (increasing, decreasing and constant) returns to scale (RTS) to exist within the feasible set of input and output vectors. The smallest polyhedral set satisfying the above three postulates is the production possibility set under variable returns to scale assumption which can be represented as, $T_v = \left\{ (\overline{X},\overline{Y}) \middle| \, \overline{X} \geq \sum_{j=1}^{n} \lambda_j X_j, \overline{Y} \leq \sum_{j=1}^{n} \lambda_j Y_j, \sum_{j=1}^{n} \lambda_j = 1, \lambda_j \geq 0 \right\}$.

For a detailed description of the production possibility set under the assumptions of constant and variable returns to scale, see, Banker et al (1984), Banker (1984), and Charnes et al (1985).

1.2.3 Definitions and Measures of Technical Efficiency

Now that we have formally described the production technology or PPS, we can look into the connection between Koopmans' (1951) definition of technical efficiency and Debreu-Farrell measures of technical efficiency. We will also see how Shephard's (1953, 1970) works on the functional representation of the production technology under constant returns to scale provide an alternative approach to the Debreu-Farrell measures of technical efficiency.

Koopmans' (1951) definition of technical efficiency can be stated formally as $(X_j, Y_j) \in T$ is technically efficient iff $(X_k, Y_k) \notin T$ for $(-X_k, Y_k) \geq (-X_j, Y_j)$[4]; i.e., a technologically feasible unit satisfies the Koopmans' efficiency criterion iff it is not (weakly or strongly) dominated by another technologically feasible unit. In figure 1.1, universities A, B, C, I and D satisfy Koopmans' efficiency criterion. In contrast, the boundary unit E does not satisfy the notion as unit D's input-output activity weakly dominates unit E's activity, i.e., $(-X_D, Y_D) = (-10, -2.5, 1) \geq (-X_E, Y_E) = (-12, -2.5, 1)$[5].

The production technology $T = \{(X,Y) | \, Y \geq 0 \textit{ can be produced from } X \geq 0, X \neq 0\}$ can also be represented by the input sets $L(Y)$. $L(Y)$ can be defined as $L(Y) = \{X : (X,Y) \in T\}$. Further for every Y, there are input isoquants $I(Y) = \{X : X \in L(Y), \lambda X \notin L(Y), \lambda < 1\}$ and input efficient subsets given by $E(Y) = \{X : X \in L(Y), X' \notin L(Y), X' \leq X\}$ and the three sets satisfy $E(Y) \subseteq I(Y) \subseteq L(Y)$. In our example provided in table 1.1, the production possibility set is given by the region north-east of the piece-wise line segments joining observed units A-B-C-I-D and the line north of A and east of D. The input isoquant is the extended frontier given by the piece-wise line segments A-B-C-I-D and the line north of A and east of D. The input efficient subset is the efficient frontier shown by the piece-wise line segments A-B-C-I-D.

Shephard (1953) introduced the input distance function to provide a functional representation of the production technology under CRS. The input

[4] We assume that no two DMU's activity are identical.

[5] To avoid clutter, we allow for some abuse of notations representing column vectors as row vectors (or vice-versa) without representing it with a transpose symbol.

distance function is given by $D_I(X,Y) = \max\{\lambda : (X/\lambda) \in L(Y)\}$. So for $X \in L(Y)$, $D_I(X,Y) \geq 1$ and for $X \in I(Y)$, $D_I(X,Y) = 1$. Given standard assumptions on T_C presented earlier, the input distance function $D_I(X,Y)$ is non-increasing in Y and is non-decreasing, homogeneous of degree +1, and concave in X. In our example provided in table 1.1, units on the boundary of the PPS, viz., A, B, C, I, D and E, have an input distance function value of 1. The input distance function value of F is 1.25; i.e., the university's current input usage (*X1* and *X2*) has to be scaled down by 1.25 to become technically efficient. Similarly, the input distance function value of G is 1.45 and H is 1.16.

The Debreu-Farrell input-oriented measure of technical efficiency TE_I is simply the value of the function $TE_I = \min\{\theta : \theta X \in L(Y)\}$ and it follows that $TE_I(X,Y) = \dfrac{1}{D_I(X,Y)}$. For $X \in L(Y)$, $TE_I(X,Y) \leq 1$ and for $X \in I(Y)$, $TE_I(X,Y) = 1$. Once again, in our example provided in table 1.1, units on the boundary of the PPS, viz., A, B, C, I, D and E, have a Debreu-Farrell input-oriented technical efficiency measure of 1. The Debreu-Farrell input-oriented technical efficiency of F is 0.8; i.e., the university's current input usage (*X1* and *X2*) has to be reduced by 20% to become technically efficient. In other words, given the empirical evidence, 80% of university F's current input usage is sufficient to output a single student. Similarly, the input-oriented Debreu-Farrell technical efficiency measure of G is 68.96% and H is 86.27%.

The above exposition can be replicated in the output augmentation direction details of which are presented in Appendix 1.

The LP based DEA measures of technical efficiency under input and output orientations presented in the seminal Charnes et al (1978) and Banker et al (1984) articles are operational extensions of the Debreu-Farrell measures and are built on the description of the PPS presented earlier. Detailed descriptions of these LP models are presented in the next chapter. In the next section, we will examine the different concepts of efficiency that are of interest in the DEA methodology and relate them to the definitions and measures introduced in this section.

1.3 Concepts of Efficiency

In this section, we will discuss four different concepts of efficiency, namely, Pareto-Koopmans efficiency, technical efficiency, Debreu-Farrell efficiency and Mix efficiency.

Efficiency Concept 1 *Pareto-Koopmans Efficiency:* A unit is Pareto-Koopmans efficient iff it is not possible to improve an input or output factor of the unit without worsening some other factor.

In the simple 2-d example discussed at the beginning of this chapter, universities A, B, C, and D are Pareto-Koopmans efficient which are also extreme-efficient units. Any convex combination of two adjacent extreme-efficient units (that lie on a facet of the production possibility set) will also be Pareto-Koopmans efficient. For example, in figure 1-1, unit I can be obtained using a convex combination of the adjacent extreme-efficient units B and C. These Pareto-Koopmans efficient units that can be synthesised by a convex linear combination of some adjacent extreme-efficient units are designated as efficient but not extreme units. They are only of academic interest and almost absent in real data (see, Thrall, 1996b; Cooper et al, 2007).

Note that the Pareto-Koopmans efficiency criterion is more stringent than the Debreu-Farrell measures as the former requires absence of mix inefficiencies while the latter allows does not.

Among any set of observed units, a subset of units will always satisfy the Pareto-Koopmans efficiency criterion - for instance units A, B, C, I and D in our example. An additional subset of the units could just satisfy the Debreu-Farrell efficiency criterion – for instance, unit E in our example. Both these subsets of units are technically efficient which we define next.

Efficiency Concept 2 *Technical Efficiency*: The technical efficiency of a unit, when the orientation is input minimisation, is the minimum proportion of the unit's current input usage that is sufficient to produce its outputs.

Geometrically, this can be obtained from the ratio of the radial distance between the origin and the radial projection of the data point on the extended frontier to the radial distance between the origin and the data point of the unit.

For example, in figure 1.1, the technical efficiency of unit H is given by $\theta_H = \frac{OH'}{OH}$. For this reason, technical efficiency is sometimes referred to as radial efficiency. It is evident that as the dimensions of a DEA problem exceeds 3, the geometrical approach will become intractable. Consequently, the LP models developed in Charnes et al (1978) and Banker et al (1984) are used to obtain the technical efficiency of the units.

The notion of technical efficiency identifies only proportional reduction of inputs or expansion of outputs that are possible by the units' current operation. Non-proportional reduction or expansion, of inputs or outputs, to improve performance are identified by an input excess or output shortfall respectively, compared to the relevant Pareto-Koopmans efficient units defined in concept 1.

Efficiency Concept 3 *Weak or Debreu-Farrell efficiency:* Units that do not satisfy the Pareto-Koopmans efficiency criterion but are technically efficient are termed weakly-efficient or Debreu-Farrell efficient units. For example, in figure 1.1, units E and F' (which symbolizes the radial projection of unit F on the efficient frontier) are weakly efficient.

Efficiency Concept 4 *Mix efficiency:* Units whose radial projection do not satisfy the Pareto-Koopmans efficiency criterion are mix inefficient. In the context of input minimisation, a unit being mix efficient would imply that the proportion of its different input usages are efficient.

Regardless of whether a DMU is technically efficient or not, it can satisfy the notion of mix efficiency. For instance, a unit can be technically inefficient but be mix efficient. In figure 1-1, universities G and H are technically inefficient but mix efficient units. Units that satisfy the Debreu-Farrell notion of efficiency will be mix inefficient. In figure 1-1, university E satisfies the Debreu-Farrell efficiency criterion and is mix inefficient. In figure 1-1, university F is both technically inefficient as well as mix inefficient. Finally, it must be evident that units that satisfy the Pareto-Koopmans efficiency criterion are also mix efficient. In figure 1.1, units A, B, C, D, and I are both technically efficient and mix efficient.

1.4 A brief history of the evolution of DEA

A data enveloping frontier based method of measuring productive efficiency was introduced in the pioneering article by Farrell (1957), which was influenced by two seminal articles, viz., “analysis of production as an efficient combination of activities” by Koopmans (1951) and “coefficient of resource utilisation” by Debreu (1951). Shephard (1953), surprisingly not referred to in Farrell’s (1957, 1962) articles, provided functional representations of the production technology under CRS and introduced distance functions as a way of measuring the technical efficiency of the units. Twenty years later, Farrell’s (1957) method was given operational form by the seminal article of Charnes et al (1978). It is interesting to note that Forsund and Sarafoglou (2002) remark that the constant returns to scale model of Charnes et al (1978) was identical to the model introduced by Boles (1971) for measuring agricultural efficiency under the assumption of constant returns to scale. They also note that the variable returns to scale model of Banker et al (1984) was clearly stated in Afriat (1972) (for the single output case) and the general version stated and applied in Fare et al (1983).

The Charnes et al (1978) article concentrated on developing a linear programming based method to determine the efficiency of various DMUs all operating under a constant returns to scale environment. Efficiency was classified as technical or radial efficiency, and mix efficiency. Technical efficiency was the same as introduced in Farrell’s article but was extended to a more general multiple inputs multiple outputs setup. Following Farrell’s article, the orientation (input or output) for measuring efficiency and proportional reduction (expansion) of inputs (outputs) to meet the data enveloping frontier were incorporated in Charnes et al (1978). Farrell’s tricky ‘points at infinity’ concept was covered by the free disposability assumption (also referred to as the monotonocity assumption in the inputs and outputs) of inputs and outputs and the introduction of mix efficiency.

Building on the work of Charnes et al (1978), Banker et al (1984) developed LP models to determine the efficiency of units operating under variable returns to scale (thus allowing for increasing, decreasing and constant RTS) and introduced the notion and measure of scale efficiency.

The Charnes et al (1978) article and its variable returns to scale counterpart, Banker et al (1984), paved the way to measure relative efficiency of units using a method that is,

1. easily operational;
2. non-parametric;
3. units invariant;
4. unlike index number based approaches in that it does not require the unit's various input and output factors' prices to be available readily to measure their efficiency;
5. able to provide peer units for inefficient units and identify technical and mix inefficiency present in all the inputs and outputs of such units; and,
6. unlike a statistical regression line method using least squares principle, in that it compares all units to the best practice ones that operate in the same environment.

1.5 Conclusion

In this chapter, we attempted a gentle introduction to the DEA methodology. We connected Koopmans' (1951) definition of technical efficiency to Shephard's (1953, 1970) functional representation of the production technology and the Debreu-Farrell measures of technical efficiency, which ultimately lead to the seminal DEA articles in Charnes et al (1978) and Banker et al (1984). Once information on the scale properties of the production technology and orientation are determined, one can carry out a DEA exercise on the set of observed units using the relevant LP model introduced in Charnes et al (1978) and Banker et al (1984). In the next chapter, we will introduce the traditional LP models used in DEA.

2 LINEAR PROGRAMMING MODELS USED IN DEA

A DEA problem is typically characterised by the cardinality (number of DMUs), dimensions (number of inputs and outputs), and density (percentage of extreme-efficient units) present in the data. Before carrying out a DEA exercise on a set of observed units, one has to posit the returns to scale environment under which the units are operating. Equally important, one has to ascertain whether the DMUs have control over their inputs or over their outputs. For example, if our DMU is a bank, then it can easily control its inputs, say, the number of administrative and technical staff, while it cannot expect to have much control over its outputs, say, the number of customers. In another instance, if our DMU is a school, then it has little control over its inputs, say, the number of students with English as an additional language or whose parents are graduates, while it can expect to have more control over its outputs, say, the achievement of students upon exit from the school. The decision on whether a DMU can control its inputs or outputs decides the orientation (input minimisation or output maximisation) of the DEA exercise. In the former example, an input-oriented model seems more suitable while in the latter an output-oriented model seems more suitable. In some instances, it is possible that the DMUs have control over their inputs as well as outputs.

Once the returns to scale and orientation are determined, the DEA exercise is carried out on the observed set of units using the linear programs developed in the seminal articles, Charnes et al (1978) and Banker et al (1984), which are built from the appropriate production possibility sets described in sections 1.2.1 and 1.2.2 respectively. These LP based models determine the relative efficiency of the units in such a way that there is complete flexibility for each unit to choose non-negative weights for its various input and output factors to show itself in the best light when compared to other observed units.

In the previous chapter, we presented formal and informal discussion on production technology and the production possibility set (PPS), and definitions and measures of technical efficiency. In this chapter, we will present the standard linear programming models that are used in carrying out the efficiency analysis of the observed units in DEA. We will present the models for the constant returns to scale assumption followed by models when the returns to scale is variable. Subsequently, we will present the LP models for solving the constant returns to scale and variable returns to scale additive models. Additive models are non-oriented and non-radial in their operation and form an important class of DEA models. Finally, we will present an important variant of the standard models, viz., the super-efficiency models under both returns to scale assumptions. The Generators Based Algorithm (GBA) presented in chapter 4 for solving DEA models employs the super-efficiency models in its procedure.

2.1 LP models under Constant Returns to Scale assumption

The models presented here were introduced in Charnes et al (1978) and elaborated further in Charnes et al (1979), Charnes et al (1981) and Charnes and Cooper (1984) and are commonly referred to as the CCR models. Suppose we are evaluating DMUt with data (X_t, Y_t) relative to all the DMUs (including itself) and their possible non-negative linear combinations. Here, X_t represents the input vector of DMUt of dimension m_1 and Y_t represents the output vector of DMUt of dimension m_2. We are interested in finding the proportion by which the inputs of DMUt can be reduced while producing at least the same amount Y_t of its outputs. All the DMUs are assumed to operate in a constant returns to scale environment and the data is assumed to be non-negative. The linear programming model to determine the relative efficiency of DMUt, built on the description of the CRS production possibility set presented in section 1.2.1, is as below:

Minimise θ_t

subject to,

$$\theta_t X_t - \sum_{j=1}^{n} \lambda_j X_j \geq 0 \qquad \textbf{(LP-1)}$$

$$0 \ + \sum_{j=1}^{n} \lambda_j Y_j \geq Y_t$$

$$\theta_t \ \textit{free}; \ \lambda_j \geq 0, j=1,\ldots,n$$

where, θ_t represents the efficiency score of DMUt and λ_j the intensity variable of DMUj, j=1,...,n. LP-1 is called the envelopment form of the input-oriented (minimisation) constant returns to scale model.

It is easy to see that LP-1 is bounded at an upper limit of 1 as DMUt can always compare with itself. Moreover, given that $X_j, Y_j \neq 0$, the solution to LP-1 can never be trivial, i.e.,$\theta_t^* \neq 0$, to satisfy both the set of constraints simultaneously, so ensuring $0 < \theta_t^* \leq 1$.

The dual to the above model is called the multiplier form of the input-oriented constant returns to scale model and is presented below:

$$\begin{aligned}
&\textit{Maximise } uY_t \\
&\textit{subject to,} \\
&vX_t + 0 = 1 \qquad\qquad\qquad\qquad \textbf{(LP-2)} \\
&uY_j - vX_j \leq 0;\ j = 1,\dots,n \\
&u, v \geq 0
\end{aligned}$$

where, v are the weights or dual values corresponding to the m_1 input factors and u are the dual values corresponding to the m_2 output factors. It follows that the efficiency score of DMUt is given by $\theta_t^* = \frac{u^*Y_t}{v^*X_t}$, where the input value of DMUt is normalised to 1, i.e., $v^*X_t = 1$[6].

Corresponding to the input-oriented version of the constant returns to scale model, there is an output-oriented (maximisation) version built using the same description of the CRS production possibility set as in section 1.2.1. In this version, we are interested in finding the maximum proportion of DMUt's outputs that can be produced by a non-negative linear combination of all the DMUs using no more than X_t amounts of inputs. The envelopment form of the output-oriented constant returns to scale model is presented below:

[6] * denotes value at the optimal solution.

Maximise ϕ_t
subject to,

$$0 \quad + \sum_{j=1}^{n} \mu_j X_j \le X_t \qquad \textbf{(LP-3)}$$

$$\phi_t Y_t - \sum_{j=1}^{n} \mu_j Y_j \le 0$$

$$\phi_t \textit{ free}; \ \mu_j \ge 0, j = 1,\ldots,n$$

where, the reciprocal of ϕ_t gives the efficiency score of DMUt and μ_j the intensity variable of DMUj, $j = 1,\ldots,n$. The dual to the above model is the multiplier form of the output-oriented constant returns to scale model which we provide below:

Minimise vX_t
subject to,

$$uY_t \quad + 0 \quad = 1 \qquad \textbf{(LP-4)}$$

$$-uY_j + vX_j \ge 0; j = 1,\ldots,n$$

$$u, v \ge 0$$

where, v are the weights or dual values corresponding to the m_1 input factors and u are the dual values corresponding to the m_2 output factors.

For more on the relationship between the input and output oriented CCR models, see Cooper et al (2000).

The LPs presented above can only identify radial or proportional inefficiencies that may be present in the DMUs. To identify mix or non-proportional inefficiencies in the DMUs, a second LP needs to be solved for each DMU. In the second phase LP, for each DMU, we maximise the slacks that may be present in its inputs and outputs compared to other DMUs subjected to the condition that the optimal $\theta_t^* \left(\phi_t^*\right)$ realised in the first phase LP is maintained. The (non-oriented, non-radial) LP solved in the second phase for DMUt is presented below:

$$\begin{aligned}
&\textit{Maximise } e^i s^i + e^o s^o \\
&\textit{subject to,} \\
&\sum_{j=1}^{n} \lambda_j X_j + s^i = \theta_t^* X_t \qquad \textbf{(LP-5)} \\
&\sum_{j=1}^{n} \lambda_j Y_j - s^o = Y_t \\
&s^i, s^o \geq 0; \lambda_j \geq 0, j = 1,\ldots,n
\end{aligned}$$

where, s^i, s^o are the input and output slack vectors of dimension m_1 and m_2 respectively; e^i and e^0 are vectors of 1's, also of dimension m_1 and m_2 respectively. Model LP-5 is sometimes referred to as the *max-slack model* and the optimal solution to it as the *max-slack solution.* In the second phase LP, we need not compare DMUt with all the DMUs as in the first phase. Rather, one can compare it with only that subset of DMUs that had an efficiency score of 1 in phase 1 (i.e., only the set of technically efficient units). If the max-slack solution for DMUt is 0, then the unit is mix efficient. If the max-slack solution is greater than 0, the unit is mix inefficient. The correct peers for DMUt are the units that are in the optimal basis of LP-5 (or its equivalent based on LP-3) rather than LP-1 or LP-3 as only units in the optimal basis of LP-5 are guaranteed to satisfy the Pareto-Koopmans efficiency criterion (Cooper et al, 2000). The standard two-phase approach to solving a DEA exercise under the CRS assumption involves solving LP-1 or LP-3 along with LP-5 (or its equivalent based on LP-3) for each DMU in phase 1 and 2 respectively.

2.2 LP models under Variable Returns to Scale assumption

The models presented here were introduced in Banker et al (1984) and are commonly referred to as the BCC models. These models are built on the PPS described in section 1.2.2. If to LP-1, LP-3, and LP-5, one adds the convexity constraint on the intensity variables, i.e., $\sum_{j=1}^{n} \lambda_j = 1$ or $\sum_{j=1}^{n} \mu_j = 1$ to the set of constraints, we get the envelopment form of the corresponding variable returns to scale model. Adding this constraint to the primal model changes the objective function and introduces an additional variable to one set of constraints in the corresponding dual model. For example, the envelopment form of the input-oriented version of the variable returns to scale model is presented below:

Minimise θ_t
subject to,

$$\begin{aligned} &\theta_t X_t - \sum_{j=1}^{n} \lambda_j X_j \geq 0 \\ &0 \ + \sum_{j=1}^{n} \lambda_j Y_j \geq Y_t \\ &0 \ + \sum_{j=1}^{n} \lambda_j = 1 \\ &\theta_t \ free; \ \lambda_j \geq 0, j = 1,\dots,n \end{aligned} \qquad \textbf{(LP-6)}$$

where, θ_t represents the efficiency score of DMUt and λ_j the intensity variable of DMUj, $j = 1,\dots,n$.

The dual to the above LP is called the multiplier form of the input-oriented variable returns to scale model and is presented below:

Maximise $uY_t + u_0$
subject to,

$$\begin{aligned} &0 \ + vX_t \ + 0 \ = 1 \\ &uY_j - vX_j + u_0 \leq 0; \ j = 1,\dots,n \\ &u_0 free; \ u, v \geq 0 \end{aligned} \qquad \textbf{(LP-7)}$$

where, v are the weights or dual values corresponding to the m_1 input factors, u are the dual values corresponding to the m_2 output factors, and u_0 is the dual value associated with the convexity constraint.

The envelopment and multiplier form VRS models for the output-oriented case are presented below in LP-8 and LP-9 respectively.

Maximise η_t
subject to,

$$\begin{aligned} &0 \ + \sum_{j=1}^{n} \rho_j X_j \leq X_t \\ &\eta_t Y_t - \sum_{j=1}^{n} \rho_j Y_j \leq 0 \\ &0 \ + \sum_{j=1}^{n} \rho_j = 1 \\ &\eta_t \ free; \ \rho_j \geq 0, j = 1,\dots,n \end{aligned} \qquad \textbf{(LP-8)}$$

$$\begin{aligned}
&\textit{Minimise } vX_t + v_0 \\
&\textit{subject to,} \\
&uY_t \quad + 0 \quad = 1 \qquad\qquad\qquad\qquad \textbf{(LP-9)} \\
&-uY_j + vX_j + v_0 \geq 0; j = 1,\ldots,n \\
&u, v \geq 0; v_0 \textit{ free}
\end{aligned}$$

In LP-8, the reciprocal of η_t gives the efficiency score of DMUt and ρ_j the intensity variable of DMU*j*, $j = 1,\ldots,n$. And in LP-9, v_0 is the dual value associated with the convexity constraint in LP-8.

For more on the relationship between the different constant returns to scale and variable returns to scale models, see Banker et al (1984) and Cooper et al (2000).

2.3 Additive models under CRS and VRS assumptions

Additive models were first introduced in the DEA literature by Charnes et al (1985). Additive models closely resemble the max-slack model (LP-5 or its VRS equivalent) and are non-oriented and non-radial in the sense that the corresponding LP problem aims to maximise the total sum of the input and output slacks of DMUt and not necessarily radially.

The standard additive CRS model solved to evaluate DMUt is shown below.

$$\begin{aligned}
&\textit{Maximise } e^i s^i + e^o s^o \\
&\textit{subject to,} \\
&\sum_{j=1}^{n} \lambda_j X_j + s^i = X_t \qquad\qquad\qquad\qquad \textbf{(LP-10)} \\
&\sum_{j=1}^{n} \lambda_j Y_j - s^o = Y_t \\
&s^i, s^o \geq 0; \lambda_j \geq 0, j = 1,\ldots,n
\end{aligned}$$

where, s^i, s^o are the input and output slack vectors respectively; e^i and e^0 are conformable vectors of 1's.

The dual to the above model is shown below.

$$\begin{aligned}&\textit{Minimise } vX_t + uY_t\\&\textit{subject to,}\\&vX_j + uY_j \geq 0; j = 1,...,n \qquad\qquad \textbf{(LP-11)}\\&v \geq +1\\&u \leq -1\end{aligned}$$

The VRS counterpart of LP-10 has an additional convexity constraint to the constraint set, viz., $\sum_{j=1}^{n} \lambda_j = 1$.

As we maximise the input and output slacks simultaneously, the units in the optimal basis of LP-10 (and its VRS counterpart) will always be Pareto-Koopmans efficient unlike in the case of the oriented models (Charnes et al, 1985). An additional advantage of the additive model under the VRS assumption over its oriented counterparts is that it is translation invariant w.r.t both inputs and outputs and hence can also handle negative values for the inputs and outputs factors (Ali & Seiford, 1990). In spite of the apparent advantage of additive models in providing Pareto-Koopmans efficient targets by solving a single LP problem and thus circumventing the need to solve a second LP unlike oriented models, they have some well established shortcomings. Coelli (1998) and Aparicio et al (2007) have pointed out that the target points provided by the optimal solution of the additive models may not be representative of DMUt as we *maximise* its inputs and outputs slacks. Hence, their argument is that the additive models should not be used for benchmarking purposes. In other words, as we maximise simultaneously the inputs and outputs slacks present in DMUt, the target points or peers for DMUt might be further away from it and thus less similar functionally. Also, unlike oriented models, the standard additive models introduced in Charnes et al (1985) are not units of measurement invariant. Lastly, unlike in the oriented models, the ratio of output value to input value does not have a natural interpretation and hence no meaningful discussion of additive models based technical efficiency scores for the units is possible. For the above reasons, additive models are commonly used for classification purposes (i.e., to classify whether a unit is Pareto-Koopmans efficient or otherwise) rather than for efficiency evaluation and benchmarking.

2.4 Introduction to Super-Efficiency model

The first published work on super-efficiency models is by Andersen and Petersen (1993) in the context of ranking DMUs that are extreme-efficient under constant returns to scale assumption. Inefficient DMUs have an objective function value of $0 < \theta_j^* < 1$ and hence, have a natural ranking based on their efficiency scores. In contrast, all the efficient DMUs are on the boundary of the production possibility set and possess a score of 1. Hence, this tie needs to be broken in some way if we are to rank them. Andersen and Petersen (1993) suggest that by using the super-efficiency models, one can rank the extreme-efficient DMUs as their objective function value (in the super-efficiency model) is no longer bounded at the upper value of 1. We will presently see the LPs employed in the super-efficiency models followed by an illustration and conclude this section with a brief review of the extant literature on super-efficiency models.

2.4.1 Linear Programs employed in super-efficiency models

The standard envelopment forms of the super-efficiency models resemble the LPs as set forth in LP-1, LP-3, LP-5, LP-6, LP-8 and LP-10 with the only difference that the DMU under evaluation, DMUt, is not included in the coefficient matrix. In other words, when evaluating DMUt, it is compared with all other DMUs and their non-negative or convex combinations except itself. The standard envelopment form of the super-efficiency CRS model when the orientation is input minimisation can be seen below.

Minimise θ_t
subject to,

$$\theta_t X_t - \sum_{\substack{j=1 \\ j \neq t}}^{n} \lambda_j X_j \geq 0 \qquad \textbf{(SE LP-1)}$$

$$0 \;\; + \sum_{\substack{j=1 \\ j \neq t}}^{n} \lambda_j Y_j \geq Y_t$$

θ_t *free*; $\lambda_j \geq 0, j = 1,\dots,n; j \neq t$

Similarly, the standard envelopment forms of the super-efficiency LP for the output-oriented CRS case, the second phase max-slack model under CRS assumption and additive CRS model can be seen in SE LP-2, SE LP-3 and SE LP-4 below.

Maximise ϕ_t
subject to,

$$0 \quad + \sum_{\substack{j=1 \\ j\neq t}}^{n} \mu_j X_j \le X_t \qquad \textbf{(SE LP-2)}$$

$$\phi_t Y_t - \sum_{\substack{j=1 \\ j\neq t}}^{n} \mu_j Y_j \le 0$$

ϕ_t *free*; $\mu_j \ge 0, j = 1,\dots,n; j \neq t$

Maximise $es^i + es^o$
subject to,

$$\sum_{\substack{j=1 \\ j\neq t}}^{n} \lambda_j X_j + s^i = \theta_t^* X_t \qquad \textbf{(SE LP-3)}$$

$$\sum_{\substack{j=1 \\ j\neq t}}^{n} \lambda_j Y_j - s^o = Y_t$$

$s^i, s^o \ge 0; \lambda_j \ge 0, j = 1,\dots,n; j \neq t$

Maximise $e^i s^i + e^o s^o$
subject to,

$$\sum_{\substack{j=1 \\ j\neq t}}^{n} \lambda_j X_j + s^i = X_t \qquad \textbf{(SE LP-4)}$$

$$\sum_{\substack{j=1 \\ j\neq t}}^{n} \lambda_j Y_j - s^o = Y_t$$

$s^i, s^o \ge 0; \lambda_j \ge 0, j = 1,\dots,n; j \neq t$

The VRS counterparts of SE LP-1, SE LP-2, SE LP-3 and SE LP-4 have the additional convexity constraint added to their constraint sets, i.e., $\sum_{\substack{j=1 \\ j\neq t}}^{n} \lambda_j = 1$ to the constraint set of SE LP-1, SE LP-3, and SE LP-4 and $\sum_{\substack{j=1 \\ j\neq t}}^{n} \mu_j = 1$ to the constraint set of SE LP-2.

In the input-oriented case, the optimal objective function value of SE LP-1, θ_t^*, gives the input saving that a particular extreme-efficient DMU exhibits when compared to other DMUs. The more the θ_t^* value is than 1 for an extreme-efficient unit, the greater is the input saving present in the unit. In other words, an extreme-efficient unit can proportionately increase its current input usage by $(\theta_t^* - 1) \times 100\%$ and still remain technically efficient. Only extreme-efficient DMUs can have θ_t^* greater than 1 when using SE LP-1 (or its VRS counterpart) and hence, one can rank them based on their super-efficiency score. Other boundary DMUs still have an efficiency (and super-efficiency) score of 1 and ties exist among them when ranking. The efficiency scores of the non-extreme efficient units (units that are not extreme-efficient, i.e., inefficient, weakly efficient, and efficient but not extreme units) is the same regardless of whether we solve LP-1 or SE LP-1, as removal of a non-extreme efficient unit from the coefficient matrix does not affect the contour of the PPS (see, Charnes et al, 1991).

2.4.2 An illustration of the super-efficiency model

We will illustrate the above statements using the example provided in table 1-1 and referring to the diagram provided in figure 2-1 below.

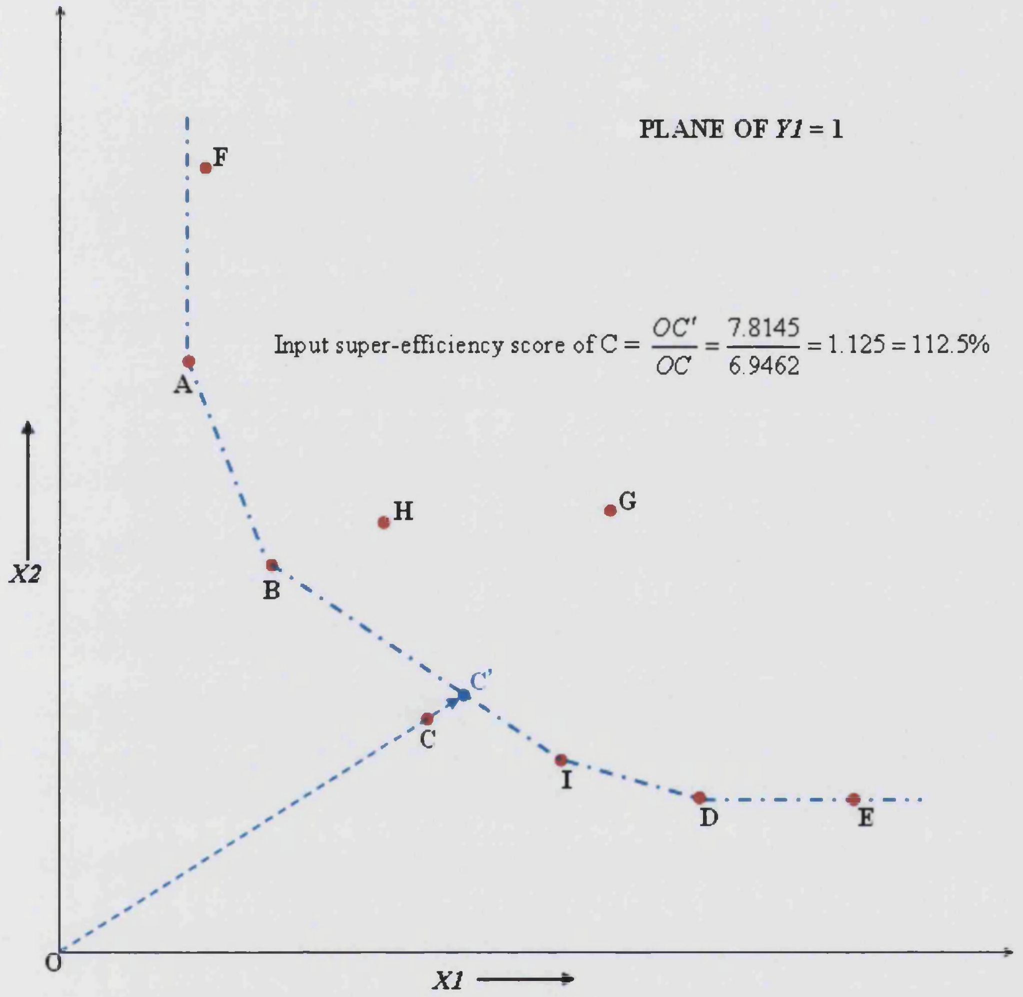

PLANE OF $Y1 = 1$
Input super-efficiency score of C = $\frac{OC'}{OC} = \frac{7.8145}{6.9462} = 1.125 = 112.5\%$
F
A
H
G
B
C'
C
I
D
E
X2
O
X1

Figure 2-1 : Super-efficiency evaluation of DMU C

The boundary units are A, B, C, I, D, and E and they all posses a technical efficiency of 1. The non-boundary units can be ranked based on their technical efficiency score. Suppose we desire to rank the boundary units based on their super-efficiency scores. Consider extreme-efficient unit C. The production possibility set when unit C is evaluated using the super-efficiency model, SE LP-1, is shown by the dash-dot line connecting units A, B, I, D and the vertical line north of A and the horizontal line east of D using piece-wise line segments. Unit C, being extreme-efficient, lies outside the (partial) PPS spanned by the units A, B, I, and D, and hence its super-efficiency score $\theta_C^* > 1$; the score can be geometrically given by $\frac{OC'}{OC} = \frac{7.8145}{6.9462} = 1.125 = 112.5\%$. This means that unit C can proportionately increase its input usage 1.125 times and still remain technically efficient. Carrying on in the above fashion, the super-efficiency score of A is 1.3455, B is 1.0217, and D is 1.0667. The super-efficiency scores of the boundary units that are not extreme-efficient, i.e., units I and E, are 1. Hence, based on the super-efficiency scores, unit A performs better than C, which performs better than D, and B performs the least best among the extreme-efficient units.

2.4.3 A brief literature review on super-efficiency models

One can see from Thrall (1996b) as well as Banker and Chang (2006) that the idea of super-efficiency was introduced much earlier in the article by Banker and Gifford (1989), an article that was then submitted to Management Science and remains unpublished. Much work has been done on super-efficiency models since the first published article in 1993. For example, Charnes et al (1996), Zhu (1996) and Seiford and Zhu (1998a) use them for studying the stability of efficiency classifications; Rousseau and Semple (1995) use them for carrying out two-person ratio efficiency games; Wilson (1995) uses them for detecting influential observations in the data set; Thrall (1996b) uses them to identify extreme-efficient units. Recently, based on their simulation results, Banker and Chang (2006) argue that the super-efficiency model can be used to identify outliers in the data set but should not be used for ranking extreme-efficient units.

Andersen and Petersen (1993) fail to recognize that as DMUt is compared with everyone else except itself, the resultant LP problem, SE-LP-1, can become infeasible. It is understood from Thrall (1996b) and Banker and Chang (2006) that this was already thought out in Banker and Gifford (1989). Thrall (1996b) is the first article to connect infeasibility of super-efficiency LPs to extreme-efficient DMUs in the input-oriented CRS case. In particular, he noted that if DMUt's super-efficiency LP (SE LP-1) results in infeasibility, then it must be an extreme-efficient unit. However, apart from pointing out the limitations of Thrall's (1996b) work in terms of orientation and RTS assumption, Dula and Hickman (1997) and Seiford and Zhu (1999) are the only two articles to carry out extensive studies on oriented super-efficiency models, providing necessary and sufficient conditions for their infeasibility. In the additive model case, Seiford and Zhu (1998b) point out that if DMUt is extreme-efficient then solving SE LP-4 or its VRS counterpart will result in infeasibility. Importantly, Thrall (1996b), Zhu (1996) and Dula and Hickman (1997) proved that infeasibility occurs in the input-oriented CRS super-efficiency model due to a certain pattern of zeroes in the data, while under VRS assumption, LP infeasibility can arise regardless of whether the data has zeroes or otherwise.

Among the two articles that provided an extensive examination of the infeasibility issue in the oriented super-efficiency models, Seiford and Zhu's (1999) discussion was based on the assumption that the data is strictly positive. The assumption of strictly positive data is strong and "unnatural" in real data as shown in Thompson et al (1993)[7]. The basis of Seiford and Zhu's (1999) assumption of strictly positive data is that extreme-efficient DMUs are translation invariant with respect to inputs and outputs; this is also the basis of the results developed in Ali and Seiford (1990). Although this is a valid argument, one has to note that in general, the input (output)-oriented VRS model is translation invariant with respect to outputs (inputs) and not inputs (outputs) as shown in Pastor (1996). Pastor (1996) has also shown that the input (output)-oriented VRS model is only 'classification invariant' if the inputs (outputs) are affinely scaled and hence translating the data to become strictly positive will be tantamount to solving a different DEA problem. Hence, the super-efficiency score of the translated model for extreme-efficient units will be different from the original model and cannot be used for ranking, identifying outliers etc. Importantly, Thrall (1996a) showed that the dual solutions under translation are not invariant in the oriented and non-oriented (CRS and) VRS models. In particular he showed that translating the inputs while solving the output-oriented VRS model can change the returns to scale status of a DMU from DRS to IRS. The upshot is that as yet there is no satisfactory way of translating the data that preserves the attributes of the units before translation. Notably, Seiford and Zhu (1999) do not cite Thrall (1996a). Hence, the connection between returns to scale status of an extreme-efficient unit and infeasibility of its corresponding VRS

[7] Thompson et al (1993) also rightly question assigning arbitrary small positive values to zero valued data or of deleting DMUs or factors that have zero value.

super-efficiency LP developed by them must be validated for a dataset that needs translation.

While Dula and Hickman (1997) along with Seiford and Zhu (1998b, 1999) provide conditions for identification of infeasible LPs under different circumstances, they do not offer ways to overcome them. Attempts at tackling this problem of ranking extreme-efficient units using super-efficiency models were made in Xue and Harker (2002), Lovell and Rouse (2003), and Chen (2005). However in a recent review, Cook et al (2008) shows that all these papers have some unresolved problems. Cook et al (2008) also develops a new super-efficiency based LP model similar to Lovell and Rouse (2003) to overcome infeasibility under VRS assumption. Perhaps heeding to the caution of Banker and Chang (2006), no new work on this subject has been published since Cook et al (2008).

2.5 Conclusion

In this chapter we presented the standard LPs, oriented and non-oriented, used in a DEA exercise and also provided a brief discussion on an important variant of the standard LPs, namely, the super-efficiency models which we will make use of in our presentation of GBA in chapter 4. The conventional computational scheme to process a DEA dataset is achieved by means of a two phase approach. To carry out a DEA exercise using the standard input-oriented envelopment form model for a n DMU, $m\left(=m_1+m_2\right)$ factor problem under the assumption of constant returns to scale, we need to run $2\times n$ LPs, with n LPs solved in each phase. The maximum size of an LP solved in the first phase is $m\times(n+1)$, and in the second phase $m\times(n+m)$. Under the assumption of variable returns to scale, the number of LPs solved is also $2\times n$. The maximum size of an LP solved in the first phase is $(m+1)\times(n+1)$ while in the second phase it is $(m+1)\times(n+m)$. To reduce the computational strain in solving DEA models, some heuristics and alternative effective algorithms have been developed in the DEA literature over the years. A critical examination of these approaches is presented in the next chapter.

3 COMPUTATIONAL ISSUES IN SOLVING DEA MODELS – A CRITICAL LITERATURE REVIEW

It is overwhelming to realise the surge of articles addressing various issues and developments in DEA since the seminal articles of Charnes et al (1978) and Banker et al (1984). For example, Gattoufi et al (2004) list no less than 1800 works in DEA circa 2004 in their bibliography of works in DEA. Recently, Emrouznejad et al (2008) list more than 4000 articles since the seminal article of Charnes et al (1978). Given this swell of articles on DEA, it is surprising that there are only 3 major strands of the literature devoted to computational issues in DEA.

While solving an LP problem is not computationally taxing, DEA requires the solution of at least n LPs. If n is very large, improving the computational performance of processing a DEA dataset becomes relevant. While established computational constructs for efficient computation of any LP such as data pre-processing, advanced starting basis, anti-cycling rules, and candidate list are applicable in the standard DEA solution procedure, the properties and structure of the DEA LP models can be further exploited.

The initial works of Ali (Ali, 1993; Chen & Ali, 2002) develop some basic ideas which are useful in pre-processing the data. No new algorithm for processing a DEA dataset is presented in the papers; rather, various pre-processing and LP accelerating techniques in the DEA context are developed and tested. Further, the papers examine the obvious but useful categorising of DMUs into frontier and non-frontier sets that can lead to a saving in the amount of computational work done for the LPs solved. Frontier DMUs are those that satisfy the Pareto-Koopmans (P-K) efficiency criteria. We note in passing that some of the ideas presented in Ali (1993) were originally developed in Sueyoshi and Chang (1989) and Sueyoshi (1990) but were not cited by Ali (1993).

The second major contribution is the parallel and hierarchical decomposition procedure developed by Barr and Durchholz (1997). In the first phase of their method, the data is decomposed into several smaller sub-problems, consisting of subsets of DMUs and can be seen as separate, smaller DEA problems. The non-frontier DMUs are eliminated in each sub-problem. The remaining DMUs are collated in the next stage and decomposed into several new sub-problems. This procedure is iterated in the first phase until all the non-frontier DMUs are eliminated. In each of the sub-problems at any stage, standard DEA LPs are employed to eliminate non-frontier units. At the end of the first phase, all the frontier DMUs are identified. In the second phase, the non-frontier DMUs are scored using only the data of the frontier DMUs.

The third group of articles are authored by Dula on his own or with co-authors (Dula & Helgason, 1996; Dula, 1998; Dula et al, 1998; Dula & Thrall, 2001) and provide a totally different approach. Instead of trying to eliminate non-frontier DMUs, Dula's BuildHull algorithm, as presented in Dula (1998), identifies DMUs that are extreme-efficient in the first phase using n LPs in such a way that the number of intensity variables in any LP solved is never more than the number of extreme-efficient DMUs. In the second phase, using only the data of the extreme-efficient units, all other DMUs are scored using standard DEA LPs. By carrying out extensive computational testing, Dula (1998) has shown that his BuildHull algorithm is superior to the hierarchical decomposition procedure of Barr and Durchholz (1997) and the standard two-phase approach, especially in the case of large scale problems.

Two other recent papers, viz., Chen and Cho (2009) and Korhonen and Siitari (2009), have also examined computational issues in solving DEA models and presented faster procedures under certain special conditions. Chen and Cho's (2009) algorithm targets large scale DEA problems. Their algorithm involves clustering DMUs based on their input and output values and evaluating DMUs within a cluster using the standard DEA LP models. If the optimal dual values for DMUt obtained within its cluster also satisfies the duality condition for all the n units in the dataset, then the efficiency analysis for DMUt is complete and the unit is discarded from further analysis. Else, the cluster size is increased by some user-defined factor until the duality condition is satisfied for all the DMUs in the dataset. The authors claim that their procedure is suitable for solving large scale DEA datasets when the density is high and the number of inputs and outputs is small. Neither of these conditions applies in real datasets used during this research or shown in past research by Barr and Durchholz (1997). Also, the algorithm involves random elements in establishing clusters within a dataset and in expanding the cluster when the optimal dual solution to a DMU within its cluster does not satisfy the duality condition for all the DMUs.

Korhonen and Siitari (2009) provide a similar approach to Barr and Durchholz (1997) by decomposing the dataset using the dimensions instead of the cardinality as in the hierarchical decomposition procedure and show that their approach is useful in datasets when the dimension is low. As both recent papers (Chen & Cho, 2009; Korhonen & Siitari, 2009) apply under restrictive assumptions and are not supported by extensive computational results, we regard these as interesting evolving ideas and confine our attention to the three major strands.

In subsequent sections, we will examine each of the three major strands, viz., Ali's work on Pre-processing and LP acceleration techniques, Barr and Durchholz's work on Hierarchical decomposition procedure, and Dula's work on BuildHull algorithm, in detail.

3.1 Ali's contributions: pre-processing and LP acceleration techniques

Ali's contributions in Ali (1993) and Chen and Ali (2002) can be listed under four themes, viz., restricted basis entry, candidate list, pre-processing techniques, and advanced starting basis.

Restricted basis entry and *Candidate list* are based on the well established theorem that for any DEA LP model, only the frontier DMUs can appear in an optimal basis in which all the optimal dual values (weights) are strictly positive (see, Charnes et al, 1985). The LPs employed in Ali (1993) use non-Archimedean infinitesimal $\varepsilon > 0$, wherein, the weights for the input and output factors are constrained to be strictly positive. This then ensures that only P-K efficient DMUs can appear as comparator units in an optimal basis of any LP solved. This implies that, if λ_p^* (or μ_p^*) is in the optimal basis of any DMU's LP, as per the definition of P-K efficient units, DMUp's efficiency score is 1 and its max-slack solution is 0. Thus, the solution to an LP for any DMU helps in identifying and updating the frontier and non-frontier DMU sets. In particular, all the DMUs in an optimal basis of an LP are frontier DMUs. A DMU belongs to the non-frontier set if its efficiency score is less than 1. The frontier DMUs identified thus far form the *candidate list* of DMUs. In its turn, this helps to reduce the amount of work required in subsequent LPs in checking the optimality conditions. As the DMUs belonging to the non-frontier set can never be in the optimal basis for any DMU, they can be ignored from optimality and basis entry checks. Thus Ali (1993) helps to identify cases where *restricted basis entry (RBE)* is permitted. DMUs in the non-frontier set are restricted, and starting with DMUs known so far in the frontier set as the *candidate list*, the optimal basis is obtained. This is shown to help a great deal in practice as it impacts on the work done for every new LP solved.

Ali (1993) also discusses some simple data *pre-processing techniques* like dominance criteria which were first introduced in Sueyoshi and Chang (1989) and Sueyoshi (1990) to identify DMUs belonging to the non-frontier set. An observed DMUj is dominated by another observed DMUk if $X_k \leq X_j$ and $Y_k \geq Y_j$. Any dominated DMU belongs to the non-frontier set. More rules are discussed in Ali (1993) for early identification of frontier units which can be achieved by some simple data analysis. For example, early identification of efficient units is possible in the variable returns to scale models due to the presence of the convexity constraint $\sum_{j=1}^{n} \lambda_j = 1$ or $\sum_{j=1}^{n} \mu_j = 1$ which warrants that any DMU that uses the unique minimum of a particular input and/or produces the unique maximum of a particular output is bound to belong to the frontier set.

In addition, Ali (1993) proves that any DMU that has the unique maximum value of the ratio of simple aggregation of outputs to inputs must belong to the frontier set (of constant and variable returns to scale models). Chen and Ali (2002) take this further to show that any DMU having the maximum value of the ratio of weighted sum of outputs to weighted sum of inputs using some selection of non-negative weights is on the constant and variable returns to scale extended frontiers. Algebraically, if $R_j = \frac{uY_j}{vX_j}, u, v \geq 0; j = 1, \ldots, n$ and $\underset{j=1}{\overset{n}{ArgMax}} \{R_j\} =$ DMUg, then DMUg is technically efficient and on the CRS and VRS extended frontier. Also, DMUs having the minimum weighted sum of (a subset or whole of) inputs $\underset{j=1}{\overset{n}{ArgMin}} \{vX_j\}$ or the maximum weighted sum of (a subset or whole of) outputs $\underset{j=1}{\overset{n}{ArgMax}}\{uY_j\}$ is on the variable returns to scale extended frontier, where $u,v \geq 0$. Note that any DMU that is efficient under the assumption of constant returns to scale is also efficient under variable returns to scale though the converse is not necessarily true. Under VRS, Chen and Ali (2002) employs the ratio $R_j = \frac{uY_j + u_0}{vX_j},\ u,v \geq 0;\ u_0\ \mathit{free};\ j = 1,\ldots,n$ for the input-oriented case and $R_j = \frac{uY_j}{vX_j + v_0},\ u,v \geq 0;\ v_0\ \mathit{free};\ j = 1,\ldots,n$ for the output-oriented case to identify units on the frontier. In all the above cases, the subset of inputs and outputs and combination of non-negative weights is arbitrarily chosen to find frontier DMUs. It is not hard to foresee that one could end up identifying the same frontier DMUs for different subsets and weights.

In addition, Sueyoshi (1990) and Ali (1993) suggest using the ordered list of DMUs based on the ratio values of simple aggregation of outputs to inputs, i.e., $\overline{R}_j = \frac{\sum_{l=1}^{m_2} Y_{lj}}{\sum_{k=1}^{m_1} X_{kj}},\ j = 1,\ldots,n$, in descending order to identify frontier DMUs earlier. In their experience, instead of randomly evaluating DMUs, one can evaluate units based on the descending order of the value of $\overline{R}_j$ to identify frontier DMUs earlier. These DMUs can then be added to the candidate list.

An important LP acceleration technique that was discussed in Ali (1993) is the *advanced starting basis*. The advanced starting basis technique uses the optimal basis of DMUj as a starting basis for the next unsolved DMU considered for evaluation. Ali (1993) shows that employing this technique reduces the number of iterations to achieve optimality.

Ali (1993) reports that computational testing with a 533 DMU, 7 factor real-world dataset gave the most significant reduction of 84% in computational time with restricted basis entry and early identification of efficient units making major contributions to the reduction as they create a "ratchet-like" effect for the LPs solved subsequently.

3.1.1 Limitations of Ali (1993), and Chen and Ali (2002)

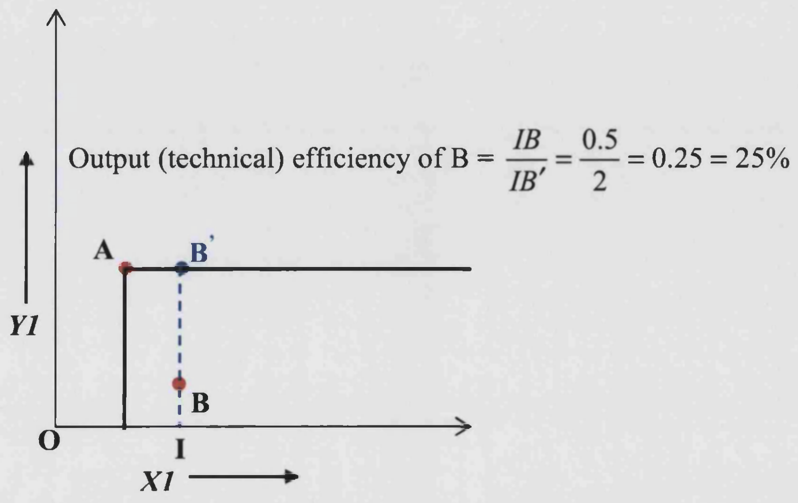
The serious drawbacks of Ali (1993), and of Chen and Ali (2002), are that they fail to identify or acknowledge the following five issues that can be encountered in using their computational constructs, namely, the issues of incorrect identification, tied ratio, indeterminate ratio, randomness, and binary weights.

The first issue of *incorrect identification* occurs when applying random weights to identify frontier units under the assumption of variable returns to scale. Because the ratio employed in the case of output-oriented variable returns to scale model involves the dual value of the convexity constraint, v_0 (see, LP-8 and LP-9 in chapter 2), which is unconstrained in sign, the resulting ratio values for *all* the DMUs could become negative for some selection of random weights. This could lead to incorrect identification of frontier units as explained using a small example shown below.

Consider the example as in table 3-1 and illustrated in figure 3-1 with 2 DMUs A and B each consuming a single input *X1* to produce a single output *Y1*.

DMU	*X1*	*Y1*
A	1	2
B	1.5	0.5

Table 3-1 : Data to show incorrect identification



Output (technical) efficiency of B = $\frac{IB}{IB'} = \frac{0.5}{2} = 0.25 = 25\%$
A
B'
B
Y1
O
I
X1

Figure 3-1 : Figure to show incorrect identification

Under the assumption of variable returns to scale, if the orientation is output maximisation, the ratio used by Chen and Ali (2002) to evaluate DMUs to identify frontier units is $R_j = \frac{u_1 Y_j}{v_1 X_j + v_0}$; where, u_1 and v_1 are the (random) output and input weights respectively which are constrained to be non-negative, and v_0 is the dual value of the convexity constraint and is a free variable. For a set of random weights where $u_1 = 1$, $v_1 = 1$, and v_0=-2, the ratios for the DMUs are, $R_A = \frac{(1\times 2)}{(1\times 1)+(-2)} = -2$ and $R_B = \frac{(1\times 0.5)}{(1\times 1.5)+(-2)} = -1$. According to Ali (1993), and Chen and Ali (2002), the DMU that produces the unique maximum of the ratios for some selection of random weights is a frontier unit. Hence, it will identify DMU B to be a frontier unit. However, this is incorrect as DMU B is an inefficient unit with efficiency score of 25%. This can be seen from figure 3-1. Unit A is the only frontier DMU, the horizontal and vertical thick lines through unit A shows the production frontier, and the region to the south-east of unit A shows the production possibility set of the data. DMU B is strictly inside the production possibility set and is inefficient with an output efficiency of 25%.

It is obvious to see that such an anomaly occurs because of the application of unconstrained random weights which can cause the ratios to be negative. It is surprising that even a recent work on this topic by Dula and Lopez (2009) failed to recognise this possibility[8] for misidentification. One way to ensure that this does not happen is to use the minimum of the ratios to identify frontier units instead of the maximum of the ratios when all the ratios are negative. An alternative simpler way is to let $v_0 = 0$ when applying random weights to identify frontier units.

The second issue of *tied ratio* occurs when upon applying a random set of weights to the input and output factors, the ratio is not uniquely maximised. Either we discard the iteration and apply a new set of weights which is a waste of computational time or we use the information in some fashion. For example, all the DMUs that are tied for the maximum ratio value, if finite, must be on the boundary of the production possibility set.

[8] On 11 August 2008, Professor Dula in a private communication acknowledged the error in Chen and Ali (2002) - "*...I think you have a point. After going to their paper I see how your example may contradict their claim...This could be a nifty paper for you!*".

The third issue of *indeterminate ratio* occurs when the data has some zero valued inputs for some DMUs and we apply non-negative weights, with possibly some zero input weights; on evaluating the ratios, it is possible for a ratio or ratios whose denominator is 0 and hence *indeterminate.* Again, one can ignore the iteration and apply a new set of weights adding to the computational burden or use the information in some fashion. We will see later, in chapter 7, how indeterminate ratios are connected to tied ratios and present ways to resolve them.

The fourth and fifth issues are marginal. It is clear that some pre-processing techniques have a *random* component in their application, and depending on the data set and the random numbers generated for the weights, one could waste time identifying the same set of units as frontier DMUs or having tied or indeterminate ratios. As a result, applying the standard two-phase algorithm for each DMU could, computationally, be equally or even more efficient for some datasets.

The fifth issue of *binary weights* (weights that are either 0 or 1) arises when one applies 0 weights to some factors, as in Chen and Ali (2002), thus effectively ignoring the performance of DMUs in those dimensions and resulting in partial productivity indices as opposed to full productivity indices. By applying 0 weights to some factors, one fails to differentiate between weakly efficient and P-K efficient units. This may lead to incorrectly computing the mix inefficiencies that may be present in some DMUs. One way to counter this issue is to consider only strictly positive weights for the input and output factors.

An additional issue with Ali (1993), and Chen and Ali (2002) is that the computational testing carried out was not extensive. Although large for computing power available at the time, the largest dataset considered for evaluating the impact of the computational constructs developed in their papers is a 533 DMU, 7 factor problem. It is hard to envisage the impact of their computational constructs in solving large datasets running to thousands of DMUs with varying densities and dimensions. Such extensive testing is carried out in Barr and Durchholz (1997) and Dula (1998), and also in chapter 8 here.

3.2 Barr and Durchholz (1997) contribution: Hierarchical Decomposition procedure

The second significant contribution in the DEA literature addressing computational issues is by Barr and Durchholz (1997). The paper, targeting large scale DEA problems, starts with a survey of runtime speeding-up techniques listing all those discussed by Ali (1993). The paper then considers degeneracy and cycling issues in DEA and claims that in their experience with large scale DEA problems, stalling and cycling can be avoided by simple scaling of the problem data and in case of lack of progress invoking a lexicographic ordering procedure. The issue of cycling in DEA LPs was also discussed in Ali (1994) albeit briefly.

Computational tests using pre-processing and LP acceleration techniques on a 8700 DMU data set indicate significant computational savings similar to the ones observed by Ali (1993) with reduced basis entry and early identification of efficient DMUs making considerable impact. However, they found that some of Ali's (1993) ideas (re-optimization or advanced starting bases) are of uneven value.

The paper then extends these speeding-up techniques to a parallel machine architecture, in particular, to a multiple-instruction multiple-data (MIMD) environment. Such computing systems contain multiple, independently executable, processors that can operate simultaneously on different data sets. They conclude that mapping of the large scale DEA solving process (involving solving many separate LP sub-problems) to a tightly coupled MIMD architecture would greatly exploit a parallel processing design and enable significant run-time savings. A parallel computing environment will definitely alleviate the computational burden of a DEA run but at the cost of additional processors and the set-ups required. Simply, one can think of having n processors, one for each DMU, and solve any problem in a few seconds. However, in this thesis we will skirt the idea of employing parallel processors and focus on alternative efficient solution procedures on single processors.

The paper then discusses the simple yet effective technique of hierarchical decomposition wherein a large problem is decomposed into a series of smaller ones of approximately equal cardinality that are mutually exclusive and collectively exhaustive. The practical insight that an inefficient (non-frontier) DMU in a sub-problem will be inefficient in the entire problem helps in reducing the size of the LP solved in subsequent stages. However, an efficient (frontier) DMU in a sub-problem may or may not be efficient in the entire problem and needs to be considered again in later stages. The speeding-up procedures described earlier (viz., dominance criteria, early identification of efficient units etc.) are applied first to the original problem and then to the sub-problems. DMUs are classified into frontier and non-frontier sets and solved with much reduced LPs.

The Hierarchical Decomposition procedure is based on three user-defined parameters, viz., b, β, and γ; b is the cardinality of each sub-problem which after the initial stage will change; β is the factor by which b changes in the next iteration; and γ is the cut-off point at which the number of sub-problems collapses to one. At the first iteration, *n* LPs are solved, one for each DMU within its sub-problem. This decomposition procedure which is similar to the well-known divide and conquer algorithm is shown to work efficiently for data sets where the density is low which is typically the case for large scale DEA problems. However, the choice of the user-defined parameters is decided by prior simulation which determine the trade-off between the cardinality and the number of sub-problems solved.

Significant computational savings were observed both in a single as well as in a parallel processing environment when the hierarchical decomposition procedure was employed. Although one can expect this procedure to improve the computational behaviour in solving large scale DEA problems, the need to consider some DMUs (namely, efficient ones in a sub-problem) again and again in the later stages is clearly uneconomical. Dula (1998) has shown in his extensive computational testing that the hierarchical decomposition procedure performs worse in a single-processor environment than the standard two-phase procedure when the cardinality of the problem is small or even medium sized which is not altogether surprising. Also, the cardinality and number of sub-problems need to be optimally determined using prior simulation as this affects the number of stages in which one carries out a DEA run which in essence can affect the effectiveness of the hierarchical decomposition procedure.

3.3 Dula's BuildHull algorithm

The next significant published paper addressing computational issues in DEA is by Dula and Thrall (2001). To evaluate it we need to consider Dula's earlier unpublished manuscript in 1998 which is based on two of his even earlier works (Dula & Helgason, 1996; Dula et al, 1998). Notably, Dula et al (1998) improves on LP based algorithms for finding the extreme rays of the conical hull of a finite set of vectors whose generated cone is pointed. Building on this groundwork, Dula (1998) gives the BuildHull algorithm for solving large scale DEA problems as a direct application of the Dula et al (1998) work.

Dula (1998) discusses a new solution procedure, akin in some ways to the two-phase hierarchical decomposition procedure of Barr and Durchholz (1997) but totally different in its philosophy in that it attempts to find in the first phase all the extreme-efficient DMUs. These are then shown to be the same as the minimum cardinality set that forms the 'frame'(meaning, the DMUs needed to generate the DEA frontier) of the production possibility set. Irrespective of the DEA model used, i.e., oriented or non-oriented and under any returns to scale assumption, the extreme-efficient units are identified by their geometrical properties using one small LP for each DMU and some additional computations. Then in the second phase all other DMUs are scored through standard DEA LPs but with only the data of the extreme-efficient units. The main advantage of the BuildHull algorithm is that the size of the LP remains relatively small in both phases and does not exceed $m\times(k+1)$ at any iteration under the assumption of constant returns to scale, where, m is the number of factors in the data and k is the number of extreme-efficient units present in the data. Similar results exists for the VRS and additive models.

It is important to note that typically, $k << n$; i.e., the percentage of extreme-efficient units in any 'real' data set is relatively small. In an application to the state of Texas' southwest district banks containing 8748 banks and 9 factors (6 inputs + 3 outputs), Barr and Durchholz (1997) report that no more than 1% were extreme-efficient. The U.K.'s Department for Education (DfE) provided data set of secondary schools with 9 factors (8 inputs + 1 output) had only 111 extreme-efficient units out of 1258 schools (8.8%), while in another data set with 10 factors, out of 1200 non-sixth form (NSF) and 1653 sixth form

(SF) schools, only 188 NSF (15.67%) and 232 SF (14.03%) schools were extreme-efficient. In a dataset for primary schools in England provided by the DfE, only 188 out of 13216 DMUs (1.42%) were extreme-efficient.

Dula's (1998) BuildHull algorithm uses two different LPs, one in each phase. In the first phase, under the assumption of constant returns to scale, the LP solved for DMUt is provided below:

$$
\begin{aligned}
&\textit{Minimise } \omega_t \\
&\textit{subject to,} \\
&\omega_t X_m - \sum_{j\in J_l} \lambda_j X_j \geq -X_t && \textbf{(BH LP-1)} \\
&\omega_t Y_m + \sum_{j\in J_l} \lambda_j Y_j \geq Y_t \\
&\omega_t \geq 0, \lambda_j \geq 0, j = 1,\ldots,n
\end{aligned}
$$

where, (X_t, Y_t) is the input-output vector of DMUt, the DMU under evaluation; (X_j, Y_j) is the input-output vector of DMUj; J_l is the set of currently identified extreme-efficient units; (X_m, Y_m) is the input-output vector of the average DMU used in each LP run in phase 1 of the BuildHull algorithm defined by $X_m = -\frac{1}{|J_l|}\sum_{j\in J_l}(-X_j - e), Y_m = -\frac{1}{|J_l|}\sum_{j\in J_l}(Y_j - e)$. Here, e is a vector of 1's of appropriate dimension and $|J_l|$ represents the cardinality of the set J_l .

The dual to the above LP is provided below:

$$
\begin{aligned}
&\textit{Maximise } -vX_t + uY_t \\
&\textit{subject to,} \\
&uY_m + vX_m \leq 1 && \textbf{(BH LP-2)} \\
&uY_j - vX_j \leq 0, \forall j \in J_l \\
&u, v \geq 0
\end{aligned}
$$

where, u, v denote the set of dual values corresponding to the first and second set constraints of BH LP-1 respectively.

Note that all the data points $(-X_j, Y_j)$ in J_l are constrained to a single orthant while the average vector, by negating the sign of inputs and outputs, is positioned in a different orthant. It is relatively straightforward to see that if the data point of DMUt, $(-X_t, Y_t)$, is outside the space spanned by (a non-negative linear combination of) the extreme-efficient units in set J_l, it is not possible to describe it using a non-negative linear combination of them. In that case, the objective function value of BH LP-1 will be strictly greater than 0 as we need some positive multiple of the average vector which is not in the orthant of the data points in set J_l to describe it. Else, $\omega_t^* = 0$, and DMUt is not extreme-efficient; we can discard DMUt and consider the next DMU whose status is unresolved. When $\omega_t^* > 0$, the BuildHull algorithm employs a slightly complicated ratio test to identify a new extreme-efficient unit (which could be DMUt) among the units that are outside the space spanned by the units in set J_l. Set J_l is then updated and a new DMU from the status unresolved set is considered in the next iteration. The algorithm proceeds until the status of all the DMUs is determined. In the second phase, standard DEA LPs as in LP-1 or LP-3 are employed to correctly score the other units using only the data of the extreme-efficient units in the final set J_l.

The algorithm is best initiated with some extreme-efficient units in set J_l which can be easily done using heuristics discussed in section 3.1. See, Dula (1998) for more details on the working of the BuildHull algorithm. An illustrated summary of Dula's work (presented in Appa and Parthasarathy (2006a)) can be found in Appendix 2. See, also, Dula (2007, 2010) for a recent and more elegant version of the BuildHull algorithm presented with some illustrations.

3.4 Algorithmic characteristics of competing solution procedures

Under the assumption of constant returns to scale, using the standard envelopment form LPs, LP-1 or LP-3, the size of any LP solved is $m \times (n+1)$ in the first phase. Restricted basis entry can reduce the size of the LP solved as we come closer to n. For example, for the last unit analysed, the LP will have only the frontier units and the unit under consideration, the inefficient units having been eliminated. So the largest LP solved is $m \times (n+1)$ and the smallest size is $m \times (k+1)$, where k is the number of extreme-efficient units present in the data. Assuming that we do not perform any data pre-processing to identify efficient units, the number of LPs solved in the first phase is n.

If we are interested in computing the correct mix inefficiencies present in all the DMUs and in indentifying Pareto-Koopmans efficient peers for the inefficient units, we need to solve a second LP, LP-5, for each DMU, making a total of $2n$ LPs. However, if in the first phase, an inefficient DMU's projection is on an extreme-efficient DMU or the weights for the input and output factors for DMUt obtained in phase-1 are strictly positive, its correct technical and mix inefficiencies are already computed and a second LP is not required for that DMU. Hence, the minimum and maximum number of LPs to be solved in the second phase are 0 and n respectively. For a DMUt that requires a second LP, we need to compare it with only those DMUs that were technically efficient at DMUt's optimal weights in its first phase LP. Hence, the minimum and maximum number of comparator units in the second phase LP for DMUt are 1 and n respectively.

Using the BuildHull algorithm, the number of LPs to be solved reduces from the standard $2n$ LPs to $2n-k$ LPs. In the first phase, where n LPs are solved, the size of the LP solved gradually increases from $m\times 2$ to $m\times(k+1)$ with average size $m\times\left(\left(\frac{k+1}{2}\right)+1\right)$. In the second phase, $n-k$ LPs are solved with fixed size $m\times(k+1)$.

The number of LPs solved using the hierarchical decomposition procedure can easily exceed $2n$ as it depends on the number of stages one goes through in the first phase, which in turn is affected by the number and size of the sub-problems. The typical size of the LP solved using the hierarchical decomposition procedure in the first phase is less than $m\times(n+1)$ but can be greater than $m\times(k+1)$ depending on the density of the dataset and the size of the sub-problems. The size and the number of LPs solved in the first phase are variable and user-defined, while in the second phase, they are the same as in using the BuildHull algorithm.

BuildHull algorithm is shown to be computationally superior to other existing algorithms for solving large scale DEA models. This was underscored by extensive computational testing carried out in Dula (1998) by comparing it with the hierarchical decomposition procedure and the standard two-phase algorithm for different data sets with varying dimensions, densities, and cardinalities. BuildHull consistently outperformed both the hierarchical decomposition procedure and the standard two-phase algorithm. See also Dula and Thrall (2001) and Dula (2007) for additional results.

However, the major weakness of the BuildHull algorithm is that, for all the non extreme-efficient units, two LPs need to be solved, one in each phase resulting in a total of $2n-k$ LPs. This is because of the limitation of the LP used in the first phase which can only identify whether a particular unit is extreme-efficient or not, and if not, provides no supplementary information resulting in an additional LP of size $m\times(k+1)$ for each non extreme-efficient unit in the second phase. This major weakness of the BuildHull algorithm is overcome in the Generator Based Algorithm (GBA) described in chapter 4. The GBA is similar in spirit to BuildHull where the size of any LP solved remains at most $m\times(k+1)$ but the second phase is not required, thereby reducing the number of LPs solved by almost half to n.

3.5 Conclusion

In this chapter, we critically reviewed the three main strands of the DEA literature that examined computational aspects of DEA. Chapter 4 presents details of the GBA for solving the input-oriented CRS model. Extensions to other returns to scale and orientations, and ways to handle the technical challenges in applying GBA to process a dataset using various DEA models are discussed in chapters 5, 6 and 7.

4 GENERATOR BASED ALGORITHM FOR SOLVING THE INPUT-ORIENTED CRS MODEL

In this chapter, we present a new algorithm called the Generator Based Algorithm (GBA) for solving the input-oriented constant returns to scale (CRS) model. GBA is related to Dula's (1998) BuildHull algorithm which was designed for solving large scale DEA models. The largest LP solved in Dula is $m \times (k+1)$, where k is the number of extreme-efficient units in the datasets, but the drawback is that two LPs have to be solved for each of the $n-k$ non extreme-efficient DMUs. We overcome this weakness by using the super-efficiency model of Andersen and Petersen (1993) and a computationally economical procedure for evaluating DMUs.

The chapter is organised as follows. First, we restate the standard input-oriented CRS DEA model and define three characteristics of extreme-efficient units which we label as generators. Then we present a modified version of the input-oriented CRS super-efficiency model that will be employed in GBA. This is followed by the algorithmic procedure of GBA with an illustrative example. Finally, we list the advantages of GBA over the existing algorithms.

4.1 Background and definitions

Consider a DEA problem with $m (= m_1 + m_2)$ factors and n DMUs. We will make two assumptions on the $(m \times n)$ dataset. First, that no two DMUs' activity are proportional to each other; i.e., there are no two DMUs j, k in the dataset such that $(X_j, Y_j) = a(X_k, Y_k), a > 0$[9]. Secondly, that the dataset is strictly positive.

Suppose the orientation of the DEA exercise is input minimisation. The standard approach to compute the technical efficiency of DMUt under CRS is to solve the following $m \times (n+1)$ LP problem presented as LP-1 in chapter 2.

[9] This is an assumption that is commonly made in the literature (see, Charnes et al, 1991; Barr & Durchholz, 1997; Dula, 1998; Dula, 2008).

Minimise θ_t
subject to,

$$\theta_t X_t - \sum_{j=1}^{n} \lambda_j X_j \geq 0 \qquad \textbf{(LP-1)}$$

$$0 \;\; + \sum_{j=1}^{n} \lambda_j Y_j \geq Y_t$$

θ_t *free*; $\lambda_j \geq 0, j \; in \; 1,\ldots,n$

The dual to LP-1, described as LP-2 in chapter 2 is presented below.

Maximise uY_t
subject to,

$vX_t + 0 \; = 1$ **(LP-2)**

$uY_j - vX_j \leq 0;\; j = 1 \ldots n$

$u, v \geq 0$

Before presenting the modified super-efficiency model that will be employed in the GBA and the algorithmic description of GBA, we will follow Charnes and Cooper (1984), Charnes et al (1985) and Charnes et al (1991) in presenting three characteristics of extreme-efficient DMUs that are relevant for our discussion on GBA.

Characteristic 1: DMUt is extreme-efficient iff its omission will change the efficient frontier of the DEA problem.

Characteristic 2: DMUt is an extreme-efficient unit iff at every optimal solution to LP-1, $\theta_t^* = \lambda_t^* = 1;\; \lambda_j^* = 0,\; for\, j \;=\; 1,\ldots,n, j \neq t$.

Characteristic 3: DMUt is extreme-efficient iff it can be shown to achieve an unique maximum value for the ratio of weighted outputs to weighted inputs for some strictly positive weights.

If DMUt satisfies any one of the above three characteristics, then Charnes and Cooper (1984), Charnes et al (1985) and Charnes et al (1991) have shown that it can be proven to:

1. satisfy the other two characteristics;
2. be an extreme-efficient unit.

We call extreme-efficient DMUs *generators* to designate the fact that only they are required to generate the efficient frontier. It follows that weakly efficient DMUs (for example, unit E in figure 1-1) and efficient but not extreme

DMUs (for example, unit I in figure 1-1) are not generators. In other words, the set of generators is the minimal subset of the dataset to describe the Production Possibility Set. Dula (1998, 2007, 2010) refers to this minimal subset as the *frame* of the dataset.

4.2 Modified input-oriented CRS super-efficiency model

The LP we solve at each stage for DMUt in GBA is the one corresponding to the super-efficiency model but formulated with only the generators identified so far. Suppose set $N = \{1, 2, ..., n\}$ represents the n units in the dataset, set GEN represents the generators identified up to now, set U represents the set of status unresolved units, and DMUt is being investigated. Then the LP solved in GBA is presented below:

$$\begin{aligned}
&\textit{Minimise } \theta'_t \\
&\textit{subject to,} \\
&\theta'_t X_t - \sum_{j\in GEN} \gamma_j X_j \geq 0 && \textbf{(GBA LP-1)} \\
&\;\;0 \;+ \sum_{j\in GEN} \gamma_j Y_j \geq Y_t \\
&\theta'_t \textit{ free};\; \gamma_j \geq 0, j \in GEN
\end{aligned}$$

The dual to the above LP is provided below:

$$\begin{aligned}
&\textit{Max } u'Y_t \\
&\textit{subject to,} \\
&v'X_t = 1 && \textbf{(GBA LP-2)} \\
&u'Y_j - v'X_j \leq 0, \forall j \in GEN \\
&u', v' \geq 0
\end{aligned}$$

The notation used here follows closely the notation used in defining LP-1 and LP-2 in chapter 2 with θ'_t and γ_j replacing θ_t and λ_j of LP-1 and (v', u') replacing (v, u) of LP-2.

GBA LP-1 defines the super-efficiency model for DMUt in which we score it against the DMUs in set GEN without allowing it to compare with itself. The production possibility set (PPS) generated by the data of units in set GEN is called the partial PPS. Since the data is strictly positive and the returns to scale is constant, GBA LP-1 will be feasible for any DMUt (Thrall, 1996b; Zhu, 1996; Dula & Hickman, 1997).

Let the optimal solution to GBA LP-1 be represented by γ_j^* and $\theta_t'^*$ and the corresponding dual solution by $\pi_t'^* = (v'^*, u'^*)$ where, v'^* are the optimal dual values or weights for the inputs and u'^* for the outputs. Let the objective function value at the optimal solution be $z_t'^*$. Here, $z_t'^* = \theta_t'^*$ as the γ_j of the generators have a coefficient of 0 in the objective function. At the optimal solution to GBA LP-1, $z_t'^* = \theta_t'^* > 0$, $v'^* X_t = 1$, and the strong duality theorem dictates that $u'^* Y_t = z_t'^*$. There are two possible outcomes of interest for the optimal objective function value $z_t'^*$, viz., $0 < z_t'^* \le 1$, which implies DMUt is situated inside the partial PPS, and $z_t'^* > 1$, which occurs when DMUt is outside the partial PPS.

4.3 Generator Based Algorithm

We now proceed to describe our algorithm. In 4.3.1 and 4.3.2 we provide basic tools needed in the full description of GBA given in 4.3.3.

4.3.1 Ratio R_j

Using the optimal dual values, $\pi_t'^* = (v'^*, u'^*)$, compute for all $j \in U$, *including* DMUt, ratios of weighted sum of outputs to weighted sum of inputs defined as $R_j = \dfrac{u'^* Y_j}{v'^* X_j}, \forall j \in U$. We will presently show how these R_j values are used either to find a new generator or to settle the efficiency analysis of DMUt.

We note that at any optimal solution to GBA LP-1, $v'^* X_t = 1$ and $\underset{j \in GEN}{Max}\{R_j\} = 1$ because $R_j \le 1, j \in GEN$ is implied by the constraints

$u'Y_j - v'X_j \leq 0, \forall j \in GEN$ [10]. We will investigate two possible scenarios, namely, $\underset{j\in U}{Max}\{R_j\} \leq 1$ and $\underset{j\in U}{Max}\{R_j\} > 1$ as case 1 and 2 respectively.

Case 1: $\underset{j\in U}{Max}\{R_j\} \leq 1$.

This case can happen only if $z_t'^* \leq 1$ which implies that either $z_t'^* < 1$ or $z_t'^* = 1$. In such a situation, we claim that DMUt is a non-generator and also that all the peers to score it correctly are in set GEN. Before proving this claim, we illustrate what happens geometrically.

When $\underset{j\in U}{Max}\{R_j\} \leq 1$, the dual values $\pi_t'^* = (v'^*, u'^*)$ define a supporting hyperplane of both the partial PPS and the full PPS. To see this, consider the hyperplane *h* given by $u'^*Y_j - v'^*X_j = 0$. The input-output vector (X_s, Y_s) for generators $s \in \text{GEN}$ with $\gamma_s^* > 0$ or $R_s = 1$ lie on *h*. It follows that these generators $s \in GEN$ have $R_s = \dfrac{u'^*Y_s}{v'^*X_s} = 1$ and act as peers for DMUt. Finally, as $\underset{j\in U}{Max}\{R_j\} \leq 1$, both the full and the partial PPS lie in the half space $u'^*Y_j - v'^*X_j \leq 0, \forall j \in N$. In fact *h* here is not only a supporting hyperplane but its intersection with the partial PPS defines a facet of the full and partial PPS. As per our claim, in this case, DMUt's efficiency analysis is resolved. Figure 4-4 illustrates this case when DMU F is evaluated with its only peer DMU A already identified as a generator in set GEN. Note that in all the figures in this chapter, the partial frontier corresponding to the relevant set GEN is shown in dotted lines while the *non-overlapping* part of the actual frontier is shown in bold.

We now introduce the following two lemmas to prove our claim by showing that the solution obtained for GBA LP-1 is also an optimal solution for LP-1.

[10] Note that the condition $\underset{j\in GEN}{Max}\{R_j\} = 1$ dictates that $\underset{j\in N/U}{Max}\{R_j\} = 1$.

Lemma 4.1: If $z_t'^* < 1$ and $\underset{j \in U}{Max}\{R_j\} \le 1$, DMUt is inefficient with score $\theta_t'^*$, input-output weights given by $\pi_t'^*$ and peers given by $j \mid \gamma_j^* > 0$.

Proof: Consider LP-1 for DMUt. We adapt the optimal solution of GBA LP-1 to define an optimal solution for LP-1 as follows.
The variables of LP-1 are λ_j and θ_t and the dual variables are $\pi_t = (v, u)$. Assign for these variables values in the following fashion: $\theta_t = \theta_t'^*$, $\pi_t = \pi_t'^*$, $\lambda_j = \gamma_j^*$ for $j \in GEN$ and $\lambda_j = 0$ for $j \notin GEN$. Then the adapted solution can be seen to be feasible and optimal for LP-1 and LP-2 for DMUt. **Q.E.D.**

Lemma 4.2: If $z_t'^* = 1$ and $\underset{j \in U}{Max}\{R_j\} = 1$, DMUt is on the overall frontier with $\theta_t'^* = 1$ but not a generator.

Proof: $z_t'^* = \theta_t'^* = 1$ means that DMUt is on the frontier of the partial PPS described by model GBA LP-1 with $\gamma_j^* > 0$ for at least one $j \in$ GEN. Suppose $\gamma_s^* > 0$. Note that $s \neq t$. Once again consider LP-1 for DMUt. The same construction as in the proof to lemma 4.1 shows that the optimal values obtained in GBA LP-1 lead to an optimal solution to LP-1. As $\theta_t \left(= \theta_t'^*\right) = 1$ in LP-1, DMUt is on the *overall* frontier. DMUt is not an extreme-efficient unit as it does not satisfy Characteristic 2 ($\lambda_s = \gamma_s^* > 0$ for $s \neq t$). **Q.E.D.**

Note that if $z_t'^* = 1$ then $\underset{j \in U}{Max}\{R_j\}$ cannot be less than 1.

Case 2: $\underset{j \in U}{Max}\{R_j\} > 1$.

This case can happen under two circumstances. First, when $z_t'^* \le 1$ but $\underset{j \in U}{Max}\{R_j\} > 1$ and second when $z_t'^* > 1$.

In the first situation, we can classify DMUt as a non-generator but do not have all the peers in set GEN to score it correctly. As $\underset{j \in U}{Max}\{R_j\} > 1$, there are some DMUs in U that are outside the current partial PPS, at least one among which is a peer for DMUt.

In the second situation, DMUt is outside the partial PPS generated by the DMUs in GEN as $R_t > 1$ and its status is still unknown. Although DMUt is outside the partial PPS, it may or may not be a generator itself.

In both situations, we apply procedure FindNewGen described below to find a new generator among the units in U. Before doing so, we illustrate geometrically what happens in this case.

When $\underset{j \in U}{Max}\{R_j\} > 1$ and $z_t'^* \leq 1$, the hyperplane h given by $u'^*Y_j - v'^*X_j = 0$ supports the partial PPS but not the full one. The input-output vector (X_s, Y_s) for generators $s \in GEN$ with $\gamma_s^* > 0$ or $R_s = 1$ lie on h. In h we have a separating hyperplane that has DMUt and the entire partial PPS on one side of it (satisfying $u'^*Y_j - v'^*X_j \leq 0$) and all the DMUs with $R_j > 1$ (including some of its peers) on the other side of it. Figure 4-3 illustrates this case when evaluating DMU G.

When $z_t'^* > 1$, the hyperplane h given by $u'^*Y_j - v'^*X_j = 0$ separates the partial PPS for which $u'^*Y_j - v'^*X_j \leq 0$ or $R_j \leq 1$ from those DMUs, *including* DMUt, for which $R_j > 1$ or $u'^*Y_j - v'^*X_j > 0$. Figures 4-2 and 4-5 illustrate this case when evaluating DMUs D and B respectively.

4.3.2 Procedure FindNewGen

Upon evaluating the ratio values, R_j, compute $\underset{j \in U}{ArgMax}\{R_j\}$. We present three different scenarios that can arise upon computing $\underset{j \in U}{ArgMax}\{R_j\}$ and ways to proceed with them.

1. Suppose, $\underset{j \in U}{ArgMax}\{R_j\}$ = DMUf and it is unique. We prove in lemma 4.3 below that in this case DMUf is a generator. In particular, if f = t, then DMUt is a generator. Append it to set GEN and evaluate the next DMU in set U in the subsequent iteration.

2. Suppose, $\underset{j \in U}{ArgMax}\{R_j\}$ = DMUf, is unique and f ≠ t. In this case, append DMUf to set GEN. The status of DMUt is still unresolved; one can now evaluate DMUt again in the next iteration against the augmented set of generators in GEN. Or, one can append it at the end of set U and choose the next DMU in U in the subsequent iteration. The latter practice, though computationally inefficient, will be followed in section 4.4 of this chapter for purely illustrative purposes.

3. If there is a tie for $\underset{j \in U}{ArgMax}\{R_j\}$, there are different ways of breaking it and they are discussed in chapter 7. The final outcome is a new generator among the tied units which is appended to set GEN. If the status of DMUt is still unresolved, it is added to the end of the queue in set U and the next DMU in U is evaluated in the next iteration.

Lemma 4.3: If the weights $\pi_t'^*$ obtained by solving GBA LP-1 lead to $\underset{j \in U}{Max}\{R_j\}>1$ and $\underset{j \in U}{ArgMax}\{R_j\}$= DMUf is unique, then DMUf is a generator.

Proof: Let $\underset{j \in U}{Max}\{R_j\}=\dfrac{u'^* Y_f}{v'^* X_f}=d$[11]. Let $\hat{v}=dv'^*$ and $\hat{u}=u'^*$ so that $R_f=1$ at $(\hat{v},\hat{u})$. Now scale the input weights by the input value $\alpha=\hat{v}X_f$ so that the input value of DMUf is 1. To maintain the ratio of output value to input value for DMUf as 1, one has to scale the output weights by α as well; i.e., $\tilde{v}=\dfrac{\hat{v}}{\alpha}, \tilde{u}=\dfrac{\hat{u}}{\alpha}$. At $(\tilde{v},\tilde{u})$, $R_f=1$, $R_j<1$, for all j ≠ f, and $(\tilde{v},\tilde{u})$ provide a feasible solution to the dual of LP-1, i.e., LP-2, for DMUf. This implies that $-\tilde{v}X_f+\tilde{u}Y_f=0$ and $-\tilde{v}X_j+\tilde{u}Y_j<0$ for all $j \neq f$. The corresponding primal solution for LP-1, $\lambda_j=0$ for all $j \neq f$ and $\lambda_f=1$, is feasible with the same objective function value of 1 as the dual LP. The primal solution is unique as

[11] As we have assumed that the data is strictly positive, $v'^* X_f \neq 0$ and indeterminate ratios cannot occur guaranteeing 'd' to be a finite value. The technical challenge of dealing with indeterminate ratios are discussed in chapter 5 and chapter 7.

$-\tilde{v}X_j + \tilde{u}Y_j \neq 0$ for all $j \neq f$. Hence, the primal and dual feasible solutions are also optimal and DMUf is a generator as it satisfies Characteristic 2. **Q.E.D.**

4.3.3 Description of GBA

We present our new algorithm to solve the input-oriented CRS model under the assumption of strictly positive data below.

```
PROCEDURE GBA
Step 0: Initialisation
set $GEN = \{\phi\}$, $U = \{1...n\}$, $TU = \{\phi\}$;
where,
GEN is the set of generators;
$U$ is the set of status unresolved DMUs;
$TU$ is the set of tied units for $ArgMax$ in $R_j, j \in U$ at $\pi^*$;
    0.1   Identify, using heuristics, a generator from set $U$ and move it to GEN.
End Initialisation.
Step 1: Iteration. While $U \neq \{\phi\}$, do:
    1.1   Select the first DMU from $U$, DMUt, and solve GBA LP-1 for it; let the
          optimal weights be $\pi^* = (v'^*, u'^*)$;
    1.2   Evaluate $R_j$ at $\pi^*$ for $j \in U$;
    1.3   If $\underset{j \in U}{Max}\{R_j\} \leq 1$, do:
          1.3.1 Record the optimal weights, peers and slacks for DMUt;
          1.3.2 Remove DMUt from $U$ and go to Step 1.1;
    1.4   If $\underset{j \in U}{Max}\{R_j\} > 1$, do:
          1.4.1 Compute $\underset{j \in U}{ArgMax}\{R_j\}$ and $TU$;
                1.4.1.1 If $|TU| > 1$, go to Step 1.4.2;
```

1.4.1.2 If $|TU| = 1$ and $\underset{j \in U}{ArgMax}\{R_j\} = DMUf$, record the optimal weights for DMUf; Move DMUf to set GEN; Go to Step 1.1;

1.4.2 Resolve the tie in $ArgMax$ in some fashion. Identify one generator, $DMUq \in TU$ and record the weights for it;

1.4.2.1 Move DMUq to set GEN; Go to Step 1.1;

End Procedure.

Numeric issues are important to acknowledge here. They can arise even while processing datasets using the standard algorithm, BuildHull or an enhanced version of the algorithms. For GBA they are critical while computing reduced cost and ratio values as all the values are computed to a fixed precision of 7 decimal points. While acknowledging that there can be pathological cases where computing values to a fixed precision could lead to inaccurate identification of generators, slacks and peers, the thesis does not elaborate on them. For a detailed discussion on the numeric issues in DEA, see Ali (1994) and Ali et al (1995).

4.4 An illustration of GBA

We illustrate our algorithm graphically using a small example. The example used below was also used in our report to the DfE on Dula's work. Appendix 2 contains this report giving a diagrammatic view of Dula's algorithm BuildHull applied to the same problem[12].Consider the following example with 8 DMU, 2 inputs, and 1 output.

[12] Some of the notations used in Appendix 2 are slightly different but fully explained. It is hoped that this does not cause a problem.

DMU	*X1*	*X2*	*Y1*
A	2	4	1
B	2.5	2.5	1
C	4	1.5	1
D	8	1	1
E	10	1	1
F	3	7	1
G	7	4	1
H	4	3	1

Table 4-1 : DEA data for GBA illustration

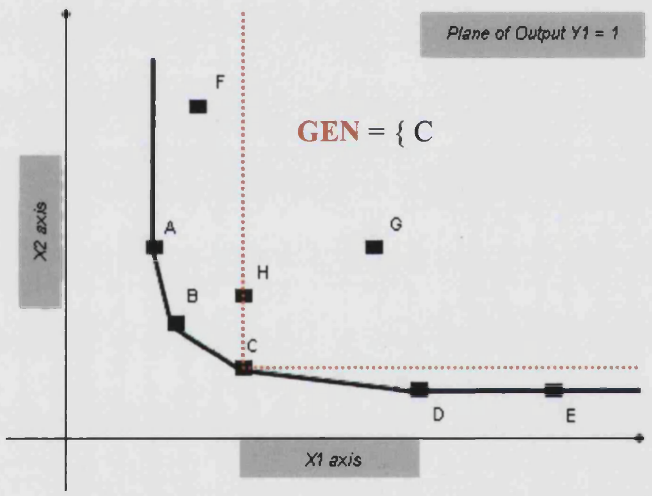


Figure 4-1 : Step 1 of GBA

Figure 4-1 illustrates GBA for the $m \times n = 3 \times 8$ DEA problem of example 4-1 starting with GEN = {C}. The full (extended) frontier is shown in bold (the vertical line north of A, A-B-C-D-E and the horizontal line east of E). The partial frontier with GEN = {C} is shown in dotted lines (the vertical line north of H, H-C and the horizontal line east of C). The only overlap between the two is at point C, the only generator in set GEN.

We have started our procedure by identifying C as a generator using heuristics (for e.g., by using random weights 1, 2, and 7, for input 1, input 2, and output 1 respectively) leading to $\underset{j \in N}{Max}\{R_j\} = 1$ and $\underset{j \in N}{ArgMax}\{R_j\} = \{C\}$.

To illustrate different cases we set U = {D, G, F, B, A, H, E} and examine DMUs in the order in which they are listed in set U .

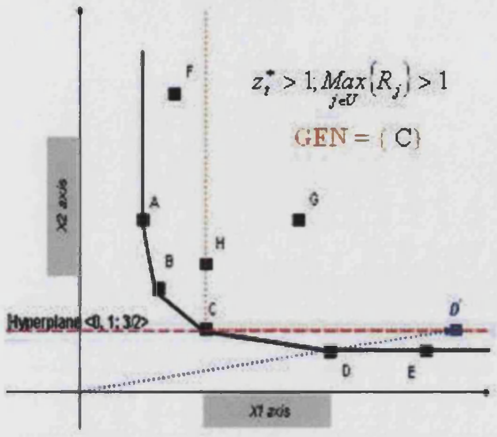


Figure 4-2 : Step 2 of GBA

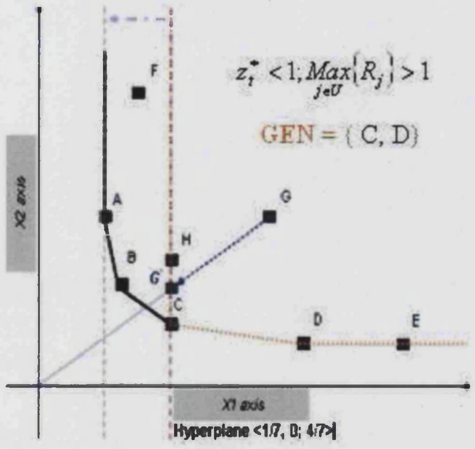


Figure 4-3 : Step 3 of GBA

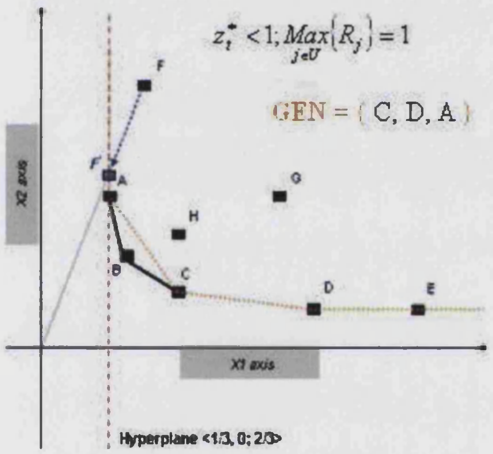


Figure 4-4 : Step 4 of GBA

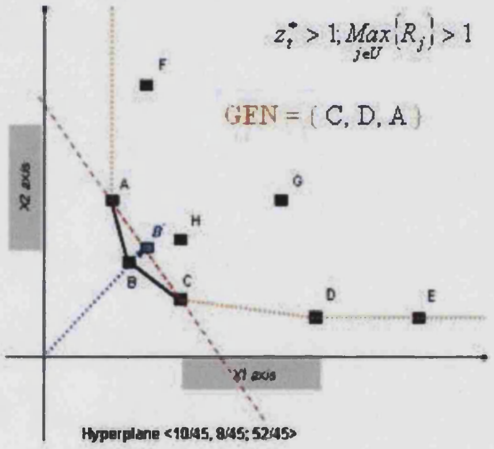

$z_t^* > 1; \underset{j\in U}{Max}\{R_j\} > 1$
GEN = { C, D, A }
X2 axis
X1 axis
F
A
G
B'
H
B
C
D
E
Hyperplane <10/45, 8/45; 52/45>

Figure 4-5 : Step 5 of GBA

In step 2 (see figure 4-2), DMUt = D is evaluated using GBA LP-1 defining a 3x2 LP problem. D is outside the partial PPS, and we have the following results, $z_t'^* = 3/2$, $\pi_t'^* = (0, 1, 3/2)$, $\underset{j\in U}{Max}\{R_j\}=1.5>1$.

As shown in figure 4-2, $\pi_t'^*$ defines the hyperplane $\left(\frac{3}{2}\times Y_1\right)-\left(0\times X_1+1\times X_2\right)=0$ passing through generator C for which $R_j=1$ separating D from the partial PPS. The ratio R_j is maximised for both D and E so that $\underset{j\in U}{ArgMax}\{R_j\}=\{D,E\}$ at this $\pi_t'^*$. Assume for now that using some tie-breaking rule, we have identified DMU D as the new generator.

Now, GEN = {C, D} and U = {G, F, B, A, H, E}.

We proceed to the next iteration where DMUt = G is evaluated using a 3x3 LP. See figure 4-3. G is inside the partial PPS and we have the following results, $z_t'^{*} = 4/7$, $\pi_t'^{*} = (1/7, 0, 4/7)$, $\underset{j\in U}{Max}\{R_j\} = 2 > 1$.

Though the hyperplane $\left(\frac{4}{7}\times Y_1\right)-\left(\frac{1}{7}\times X_1 + 0\times X_2\right)=0$ supports the partial PPS at C, it doesn't support the full PPS as A and B are DMUs in set U that are outside the partial PPS with $R_j > 1$ at this $\pi_t'^{*}$. As $\underset{j\in U}{ArgMax}\{R_j\}=\{A\}$, it implies that DMU A is a generator. So DMU A is removed from U and appended to GEN while G is put at the back of the queue in U.

Now GEN = {C, D, A}, and set U = {F, B, H, E, G}.

In the next iteration (see figure 4-4) DMUt = F is evaluated using a 3x4 LP. We have the following results, $z_t'^{*} = 2/3$, $\pi_t'^{*} = (1/3, 0, 2/3)$, $\underset{j\in U}{Max}\{R_j\}=1$.

As seen in figure 4-4, the hyperplane $\left(\frac{2}{3}\times Y_1\right)-\left(\frac{1}{3}\times X_1 + 0\times X_2\right)=0$ supports the partial PPS *and* the full PPS at A, one of the generators in the set GEN. By lemma 4.1, we have computed the correct efficiency score of F and also its proper set of peers, input-output weights and slacks. F can now be discarded from further analysis.

So now GEN = {C, D, A} and U = {B, H, E, G}.

In the next iteration (see figure 4-5) DMUt = B is evaluated using a 3x4 LP. B is outside the partial PPS and we have the following results, $z_t'^{*} = 52/45$, $\pi_t'^{*} = (10/45, 8/45, 52/45)$, $\underset{j\in U}{Max}\{R_j\}=1.1556 > 1$, $\underset{j\in U}{ArgMax}\{R_j\}=\{B\}$.

As shown in figure 4-5, $\pi_t'^{*}$ defines the hyperplane $\left(\frac{52}{45}\times Y_1\right)-\left(\frac{10}{45}\times X_1 + \frac{8}{45}\times X_2\right)=0$ passing through generators A and C for which $R_j = 1$ separating B from the partial PPS. As B achieves the unique maximum value of R_j for $j \in U$ at $\pi_t'^{*}$, it is another generator.

So at this stage, *GEN* = {C, D, A, B} and U = {E, G, H}. Now that we have identified all the generators in the dataset, every iteration henceforth will involve solving a 3x5 LP. Its solution will give the correct efficiency score, input-output weights, and proper set of peers and slacks for DMUt. For example, the efficiency of E upon solving GBA LP-1 with GEN = {C, D, A, B} is 100% and its only peer is unit D. Its input-output weights are (0, 1, 1) and it has 2 units of slack in *X1*. The efficiency of G upon solving GBA LP-1 with GEN = {C, D, A, B} is 48.08% and its peers are units B and C. Its input-output weights are (1/13, 3/26, 25/52). Lastly, the efficiency of H upon solving GBA LP-1 with GEN = {C, D, A, B} is 73.53% and its peers are units B and C. Its input-output weights are (1/13, 3/26, 25/52). Units G and H are technically inefficient but mix efficient while unit E is technically efficient but mix inefficient.

To summarise, after finding one generator, we had to solve 7 LPs of maximum size 3×5 with additional algebra for the evaluation of ratio values and $\underset{j\in U}{ArgMaxX}\{R_j\}$ to solve a DEA exercise for a 8 DMU 3 factor problem.

4.5 Ratios and Reduced cost values

Based on the proof of lemma 4.3, in procedure FindNewGen we employed $R_j = \frac{u'^{*}Y_j}{v'^{*}X_j}, j\in U$, to identify a new generator. It is important to realise why the reduced cost values[13], i.e., $RC_j = u'^{*}Y_j - v'^{*}X_j, j\in U$, could not be used for the same purpose. As we are dealing with the CRS case, any data vector can be scaled by a positive scalar without changing the PPS and therefore, the set of generators. While R_j remains unchanged under the scaling operation, RC_j changes for DMUj, thus making $\underset{j\in U}{Max}\{RC_j\}$ arbitrary. The following example illustrates this.

Consider a data set with two DMUs A and B each consuming two inputs (*X1* and *X2*) to produce a single output (*Y1*) in a CRS environment.

[13] Note that in the input-oriented case, $u'^{*}Y_j - v'^{*}X_j$ gives the negative of the reduced cost as used in the standard LP terminology. In the output-oriented case, $u'^{*}Y_j - v'^{*}X_j$ gives the reduced cost value according to the standard LP terminology. For the sake of simplicity, we will dub $u'^{*}Y_j - v'^{*}X_j$, the reduced cost of DMUj.

DMU	*X1*	*X2*	*Y1*
A	5/4	1	1
B	1/2	1/3	1/2

Table 4-2 : Reduced cost and ratio values

W.l.o.g., assume that we are provided with the following set of dual values $\pi_t'^* = (v'^*, u'^*) = (1, \frac{3}{2}, 4)$. Now, the reduced cost values of the DMUs in U are, $RC_A = u'^*Y_j - v'^*X_j = 4 - 2.75 = 1.25$ and $RC_B = u'^*Y_j - v'^*X_j = 2 - 1 = 1$; so $\underset{j\in U}{Max}\{RC_j\} = 1.25$ and $\underset{j\in U}{ArgMax}\{RC_j\} = A$. Suppose we scale unit B's data vector by a factor of 2, $\underset{j\in U}{Max}\{RC_j\} = 2$ and $\underset{j\in U}{ArgMax}\{RC_j\} = B$. Therefore RC_j cannot be used to decide whether DMUj is a generator. As we are dealing with the CRS case, only the ratios at the specified $\pi_t'^*$ can identify generators. The ratios for the DMUs are $R_A = \frac{u'^*Y_A}{v'^*X_A} = \frac{4}{2.75} = 1.45$ and $R_B = \frac{u'^*Y_B}{v'^*X_B} = \frac{2}{1} = 2$. Hence, only DMU B can be classified as a generator at this $\pi_t'^*$ as it produces the unique maximum of the ratio values. Note that the significance of reduced costs is different under VRS assumption and we will discuss it in chapter 5.

4.6 Advantages of GBA

Chapter 8 gives a detailed account of the excellent relative computational performance of GBA in comparison with BuildHull and the conventional two-phase solution procedure. The overarching reason is that GBA solves only n small LPs. Phase 2 is not required and the size of the LPs solved progresses from $m\times 2$ to $m\times(k+1)$. The downside is the work needed to compute the R_js and $\underset{j\in U}{ArgMaxX}\{R_j\}$ at each iteration after solving an LP. However, as shown in chapter 8, this is more than compensated for by not having to solve n extra LPs and keeping the size of the LPs small.

4.6 Conclusion

This chapter presented a new algorithm, the GBA, for solving the input-oriented envelopment form DEA model under the assumption of constant returns to scale and positive data. Before every iteration we have a list of as yet unresolved DMUs. An iteration consists of an LP with columns of data of all the generators found so far and a new DMU from the list. At each iteration either another generator is discovered or the status of the DMU under evaluation is resolved, i.e., its efficiency score, input-output weights, peers, and slacks are obtained. The GBA algorithm identifies all the extreme-efficient units in the dataset and the correct efficiency scores, peers and slacks for all the non-extreme efficient units.

Theoretically we showed that for solving a DEA exercise involving m factors and n DMUs, GBA requires the solution of utmost n LP problems. In addition, the maximum size of any LP solved is $m \times (k+1)$ where k is the number of extreme-efficient DMUs which we call generators. Technical complications that can arise when using the GBA under CRS, extensions to the VRS and other standard models under different orientations will be examined in chapters 5, 6 and 7. Computational experience is presented in chapter 8.

5 TECHNICAL CHALLENGES AND EXTENSION OF GBA TO OTHER DEA MODELS

GBA was presented in Chapter 4 for the input-oriented CRS model only, and under the restrictive assumption of positive data. Moreover some technical problems were not fully addressed. In this chapter we extend GBA to oriented and non-oriented CRS and VRS models without the assumption of strictly positive data and present the technical problems in each of them.

The technical problems occur at step 1.1, 1.2 and 1.4.2 of the algorithm described in section 4.3.3 and are relevant to any extension of GBA to other models. Here is a list of the technical problems.

i. At step 1.1 we solve the relevant LP for DMUt and then use the optimal solution in what follows. What if the LP is infeasible?

ii. At step 1.2 we evaluate the ratio $R_j, \forall j \in U$. Can R_j be indeterminate because its denominator is zero?

iii. At step 1.4.2 we talk about resolving ties in $ArgMax\{R_j\}$ in some fashion. Are ties always an issue and how do we resolve them?

It is clear that upon encountering any of the above three challenges in GBA, they have to be resolved without proceeding to the next iteration. The first step to resolving these issues is to understand the conditions under which they can or cannot occur in the various DEA models when applying GBA. In the rest of this chapter, we describe when infeasibility and indeterminate ratios can and cannot occur for different DEA models. Ways to resolve LP infeasibility is presented in chapter 6 and indeterminate ratios are resolved in chapter 7. The last of these is taken up in chapter 7 where a new closed-form method is developed to resolve ties.

For the sake of completeness we mention one final technical problem, viz., finding a generator to initialise the algorithm. In chapter 3, heuristics developed by Ali (1993) and Chen and Ali (2002) to identify some generators in a dataset were described in detail. Given the various limitations within these heuristics, we will only use them to identify one generator to initialise GBA.

More on heuristics is discussed in chapter 8 which deals with computational comparisons of GBA with extant algorithms.

5.1 GBA for the general input-oriented CRS model

In this section, we discuss the two technical challenges, viz., LP infeasibility and indeterminate ratios, that can arise when applying GBA for this model without the assumption of positive data.

5.1.1 LP infeasibility

Zeroes naturally occur in many multi input-output datasets (see, Thompson et al, 1993). For example, with the DfE's DEA model for primary schools in England, the percentage of students with English as an Additional Language (EAL), is an input factor which does have zero value for some schools. Let us consider the following examples to illustrate LP infeasibility in the presence of zeroes in the data.

DMU	*X1*	*X2*	*Y1*
A	10	2	24
B	15	0	24

Table 5-1 : Infeasibility due to 0 inputs

Using random weights (2, 2, 1) for input 1, input 2, and output 1 respectively, we can see that DMU A is a generator. Suppose, we now evaluate DMU B with DMU A in set GEN; the resultant LP would be,

$$\begin{aligned}
&\textit{Minimise } \theta'_B \\
&\textit{subject to,} \\
&\theta'_B 15 - \gamma_A 10 \geq 0 \\
&\theta'_B 0 - \gamma_A 2 \geq 0 \\
&0 + \gamma_A 24 \geq 24 \\
&\theta'_B, \gamma_A \geq 0
\end{aligned}$$

Clearly, the second constraint can only be satisfied for $\gamma_A = 0$; but then, the third constraint cannot be satisfied leading to infeasibility. Now, consider the example below.

DMU	*X1*	*Y1*	*Y2*
A	10	2	24
B	10	0	30

Table 5-2 : Infeasibility due to 0 outputs

Using random weights (3, 1, 1) for input 1, output 1, and output 2 respectively, we can see that DMU B is a generator. Suppose, we now evaluate DMU A with DMU B in set GEN; the resultant LP would be,

Minimise θ'_A
subject to,
$\theta'_A 10 \; - \gamma_B 10 \geq 0$
$0 \;\; + \;\; \gamma_B 0 \geq 2$
$0 \;\; + \;\; \gamma_B 30 \geq 24$
$\theta'_A, \; \gamma_B \geq 0$

It is clear that the second constraint cannot be satisfied, leading to infeasibility.

Presence of zero data entries is a *necessary* condition for infeasibility to occur in this model (see, Thrall, 1996b; Zhu, 1996; Dula & Hickman, 1997). In the example in table 5-1, DMU B has a unique zero for a particular input factor, while in example in table 5-2, due to a unique zero in a particular output factor for DMU B, the resultant LP for DMU A becomes infeasible. It follows that infeasibility *can* occur when there are zeroes in inputs and/or outputs. As discussed in chapter 2, in the standard super-efficiency model, an infeasible solution for DMUt corresponds to it being a generator. But as GBA works with partial PPS, the resultant LP can become infeasible even when evaluating a non-generator. For example, if to the data in table 5-1, one adds DMU C with data (13, 0, 24) for input 1, input 2, and output 1 respectively, it is easy to see that DMU B cannot be a generator (in fact, it becomes inefficient). However, the LP solved does not change if B is evaluated against A. It is still infeasible. Similarly, if in table 5-2, one adds DMU C with data (10, 3, 25) for input 1, output 1, and output 2 respectively, it is easy to see that DMU A cannot be a generator (and is again inefficient) even though it results in an infeasible LP when evaluated against B.

5.1.2 Indeterminate ratios

In step 1.2 of the algorithm, once the LP is solved, we evaluate $R_j = \frac{u'^* Y_j}{v'^* X_j}$ for all the DMUs in U. If the dataset contains some zeroes, then indeterminate ratio can occur. This is illustrated using the example below.

DMU	*X1*	*X2*	*X3*	*Y1*
A	10	2	8	24
B	12	0	7	38
C	0	8	5	28

Table 5-3 : Indeterminate ratio

Using random weights (2, 3, 2, 1) for input 1, input 2, input 3, and output 1 respectively, we can see that DMU B is a generator. Suppose, we evaluate DMU A with DMU B in set GEN; then the resultant LP would be,

$$
\begin{aligned}
&\textit{Minimise } \theta'_A \\
&\textit{subject to,} \\
&\theta'_A 10 - \gamma_B 12 \geq 0 \\
&\theta'_A 2 - \gamma_B 0 \geq 0 \\
&\theta'_A 8 - \gamma_B 7 \geq 0 \\
&0 + \gamma_B 38 \geq 24 \\
&\theta'_A, \gamma_B \geq 0
\end{aligned}
$$

The (unique) optimal solution to this LP is $\theta'^*_A = 0.757895$, $v'^* = (0.1, 0, 0)$, $u'^* = (0.031579)$. The ratio R_C at this π'^*_t is indeterminate. Incidentally, if we had chosen to evaluate DMU C (instead of DMU A) against DMU B, it is easy to see that the resultant LP will be infeasible.

Before we proceed to the next section, we show why indeterminate ratios cannot occur when the data is strictly positive. In other words, similar to the technical challenge of infeasibility, presence of zero data entries is a necessary condition for this challenge to occur. This can be seen from GBA LP-2 which is copied below.

$$\begin{aligned}
&\textit{Maximise } u'Y_t \\
&\textit{subject to,} \\
&v'X_t = 1 \qquad\qquad\qquad\qquad \textbf{(GBA LP-2)} \\
&u'Y_j - v'X_j \leq 0, \forall j \in GEN \\
&u', v' \geq 0
\end{aligned}$$

The first constraint will be satisfied as an equality at any optimal solution, i.e., $v'^{*}X_t = 1$. Since, the data is strictly positive, $X_j, Y_j > 0$ and as $v'^{*}X_t = 1$, $v'^{*} \neq 0$. Hence, the ratios $R_j = \dfrac{u'^{*}\ Y_j}{v'^{*}X_j}, \forall j \in U$ cannot become indeterminate.

5.2 GBA for the output-oriented CRS model

The LP to be used in GBA to solve the output-oriented CRS model to evaluate DMUt is shown below.

$$\begin{aligned}
&\textit{Maximise } \eta'_t \\
&\textit{subject to,} \\
&0 + \sum_{j \in GEN} \rho_j X_j \leq X_t \qquad\qquad \textbf{(GBA LP-3)} \\
&\eta'_t Y_t - \sum_{j \in GEN} \rho_j Y_j \leq 0 \\
&\eta'_t \textit{ free}; \ \rho_j \geq 0, j \in GEN
\end{aligned}$$

The optimal objective function value $\eta_t'^{*}$ is ≥ 0 as in the corresponding super-efficiency model. The dual to the above LP is shown below:

$$\begin{aligned}
&\textit{Minimise } v'X_t \\
&\textit{subject to,} \\
&u'Y_t = 1 \qquad\qquad\qquad\qquad \textbf{(GBA LP-4)} \\
&-u'Y_j + v'X_j \geq 0, \forall j \in GEN \\
&u', v' \geq 0
\end{aligned}$$

where, u', v' are the weights of the output and input factors respectively.

As in the standard CRS models presented in chapter 2, there is a useful relationship between the solutions of input and output-oriented super-efficiency CRS models and their modified versions in GBA LP-1 and GBA LP-3. This relationship makes it possible to derive the solutions to one model from the other.

The procedure for solving the output-oriented CRS model using GBA is similar to the procedure described for its input-oriented counterpart. Hence with no modification to the GBA procedure presented in section 4.3.3, one can carry out a DEA exercise using the output-oriented CRS model.

5.2.1 LP infeasibility

Regardless of whether the data is strictly positive or not, the output-oriented CRS super-efficiency model (and so also GBA LP-3) is *never* infeasible because $\eta_t'^{*} = 0$, $\sum_{j \in GEN} \rho_j^{*} = 0$ is a feasible solution for any DMUt (see, Zhu, 1996; Dula & Hickman, 1997).

5.2.2 Indeterminate ratios

When the dataset contains some zeroes, indeterminate ratios can arise as shown in the following example.

DMU	*X1*	*Y1*	*Y2*
A	10	2	24
B	12	0	36
C	16	8	28

Table 5-4 : All zero solution

Using random weights (3, 1, 1) for input 1, input 2, and output 1 respectively, we can see that DMU B is a generator. Suppose, we now evaluate

DMU A against DMU B in set GEN; the resulting LP and its dual are shown below.

Primal LP:

Maximise η'_A
subject to,
$0 \; + \; \rho_B 12 \leq 10$
$2\eta'_A \; - \; \rho_B 0 \leq 0$
$24\eta'_A \; - \; \rho_B 36 \leq 0$
η'_A *free*; $\rho_B \geq 0$

Dual LP:

Minimise $10v'_1$
subject to,
$2u'_1 + 24u'_2 = 1$
$12v'_1 - 0u'_1 - 36u'_2 \geq 0$
$v'_1, u'_1, u'_2 \geq 0$

Clearly, the second constraint (in the primal LP) dictates that the trivial "all zero" solution $\eta'^*_A = \rho^*_B = 0$ is the only feasible solution and hence optimal, with dual values $v'^* = (0)$ and $u'^* = (0.5, 0)$.

In the case when all the generator(s) in set GEN have zero value for an output factor s and $Y_{st} > 0$, if $X_t > 0$, it follows that $v'^* = 0$ because of the strong duality theorem. (For our example in 5-4, DMU B has Y1 = 0; also, Y1 for A is 2 and X1 is positive for A. Hence, $v'^* = 0$). Therefore, R_j is indeterminate for all $j \in U$. In addition, for $j \in GEN$ the dual constraints dictate that both $v'^* X_j = 0$ and $u'^* Y_j = 0$ leading to $R_j = \frac{0}{0}$. Unit B in our example illustrates this.

It can be shown that, when the output-oriented CRS model has an "all zero" optimal solution for DMUt, the corresponding input-oriented model is infeasible. We show in lemma 5.1 that an "all-zero" solution cannot be an optimal solution to GBA LP-3 when the data is strictly positive. This proviso can be seen as analogous to the input-oriented CRS model where LP infeasibility cannot occur when the data is strictly positive.

Lemma 5.1: When the data is strictly positive, an "all-zero" solution cannot be an optimal solution to GBA LP-3.

Proof: This can be seen from the primal-dual relationship between the LPs (GBA LP-3 and GBA LP-4). Suppose that $\eta_t'^*$ is 0. By the strong duality theorem, $v'^*X_t = 0$ and since $X_j, Y_j > 0, \forall j$, this can only happen if $v'^* = 0$. The output weights cannot be trivial as $u'^*Y_t = 1$ and hence $u'^*Y_j > 0$ for any j in GEN. This means that the second constraint $-u'Y_j + v'X_j \geq 0, \forall j \in GEN$ cannot be satisfied. Hence, by contradiction, an "all-zero" solution cannot be an optimal solution to GBA LP-3. **Q.E.D.**

Corollary to lemma 5.1: When the data is strictly positive, indeterminate ratios cannot occur.

Proof: We know from lemma 5.1 that when the data is strictly positive $v'^* \neq 0$. Given that $X_j > 0$, it follows that $v'^*X_j \neq 0$. Hence the ratios evaluated as $R_j = \dfrac{u'^*Y_j}{v'^*X_j}, j \in U$, cannot be indeterminate. **Q.E.D.**

5.3 GBA for the input-oriented VRS model

The standard LP for computing the efficiency of DMUt using the input-oriented VRS model is shown below.

$$\begin{aligned}
&\textit{Minimise } \theta_t \\
&\textit{subject to,} \\
&\theta_t X_t - \sum_{j=1}^{n} \lambda_j X_j \geq 0 \\
&0 \ + \sum_{j=1}^{n} \lambda_j Y_j \geq Y_t \qquad \textbf{(LP-6)} \\
&0 \ + \sum_{j=1}^{n} \lambda_j = 1 \\
&\theta_t \textit{ free}; \ \lambda_j \geq 0, j \textit{ in } 1,\dots,n
\end{aligned}$$

The dual to LP-6, described as LP-7 in chapter 2 is presented below.

$$\begin{aligned}
&\textit{Maximise } uY_t + u_0 \\
&\textit{subject to,} \\
&vX_t + 0 \ = 1 \qquad \textbf{(LP-7)} \\
&uY_j - vX_j + u_0 \leq 0; \ j = 1 \dots n \\
&u, v \geq 0, \ u_0 \textit{ free}
\end{aligned}$$

The LP to be used with GBA for solving the input-oriented VRS model is shown below.

$$\begin{aligned}
&\textit{Minimise } \theta_t' \\
&\textit{subject to,} \\
&\theta_t' X_t - \sum_{j \in GEN} \gamma_j X_j \geq 0 \\
&0 \ + \sum_{j \in GEN} \gamma_j Y_j \geq Y_t \qquad \textbf{(GBA LP-5)} \\
&0 \ + \sum_{j \in GEN} \gamma_j = 1 \\
&\theta_t' \textit{ free}; \ \gamma_j \geq 0, j \in GEN
\end{aligned}$$

The corresponding dual LP to GBA LP-5 is shown below:

$$\begin{aligned}
&\textit{Maximise } \ u'Y_t + u_0 \\
&\textit{subject to,} \\
&v'X_t = 1 \qquad \textbf{(GBA LP-6)} \\
&u'Y_j - v'X_j + u_0 \leq 0, \forall j \in GEN \\
&u', v' \geq 0; u_0 \textit{ free}
\end{aligned}$$

where, u' and v' are the weights for the outputs and inputs factors respectively and u_0 is the dual value associated with the convexity constraint.

Let the optimal solution to GBA LP-5 be represented by γ_j^* and $\theta_t'^*$ and the dual solution by $\pi_t'^* = (v'^*, u'^*, u_0^*)$. Let the objective function value at the optimal solution be $z_t'^*$. Here, $z_t'^* = \theta_t'^*$. At the optimal solution, $z_t'^* = \theta_t'^* > 0$, $v'^* X_t = 1$, and the strong duality theorem dictates that $u'^* Y_t + u_0^* = z_t'^*$. There are two possible outcomes of interest for $z_t'^*$, viz., $0 < z_t'^* \le 1$ or $z_t'^* > 1$.

GBA has to be slightly modified to take into account the dual value of the convexity constraint u_0 when defining R_j. For this purpose, we define $R_j = \dfrac{u'^* Y_j + u_0^*}{v'^* X_j}, \forall j \in U$, and apply procedure GBA of section 4.3.3 without further changes.

5.3.1 Using the reduced costs RC$_j$ instead of the R$_j$ values in GBA

In section 4.5, we showed why it is not possible to use the reduced cost values RC_j to identify generators in the CRS case. The VRS case is fundamentally different because arbitrary scaling of a data vector is not allowed. So here it is possible to use $Max\{RC_j\}$ value in FindNewGen to identify a generator. For this purpose, define RC_j at $\pi_t'^* = (v'^*, u'^*, u_0^*)$ as $RC_j = (u'^* Y_j + u_0^*) - v'^* X_j, \forall j \in U$. We will presently see how the GBA procedure needs to be modified when the reduced costs, RC_j, of the units in U are used instead of their R_j values.

Note that there are only two possibilities for RC_j upon solving GBA LP-5 for DMUt, viz., $\underset{j \in U}{Max}\{RC_j\} \le 0$ and $\underset{j \in U}{Max}\{RC_j\} > 0$. It follows from our proof to the CRS case that in the former situation, the efficiency analysis for DMUt is complete. In the latter case, we need to identify a new generator among the units in U. In lemma 5.2 we show that the unit that achieves the maximum of the reduced cost values $RC_j = (u'^* Y_j + u_0^*) - v'^* X_j, \forall j \in U$, if unique, is another generator.

Lemma 5.2: If the weights $\pi_t'^{*} = (v'^{*}, u'^{*}, u_0^{*})$ obtained by solving GBA LP-5 lead to $\underset{j \in U}{Max}\{RC_j\} > 0$ and $\underset{j \in U}{ArgMax}\{RC_j\} =$ DMUf is unique, then DMUf is a generator.

We show that using $\pi_t'^{*}$, it is possible to construct weights π'' for which LP-6 has a unique optimal solution at DMUf.

Proof: Let $\underset{j \in U}{Max}\{RC_j\} = (u'^{*}Y_j + u_0^{*}) - v'^{*}X_j = d > 0$ for $j=f$. We know that $\underset{j \in GEN}{Max}\{RC_j\} = 0$.

Let $\hat{u}_0 = (u_0^{*} - d)$. Then, $\underset{j \in U}{Max}\{RC_j\} = (u'^{*}Y_j + \hat{u}_0) - v'^{*}X_j = 0$ for $j=f$ and $\underset{j \in N/f}{Max}\{RC_j\} < 0$. This implies that $R_f = 1$ and $R_j < 1$ for all $j \neq f$. By suitable scaling, a feasible solution to LP-7 for DMUf can be derived. Scale the input weights by the input value, $\alpha = v'^{*}X_f$ [14], such that the input value of DMUf is 1. To maintain $R_f = 1$, one has to scale the output weights along with $\hat{u}_0$ by α as well; i.e., let $v'' = \frac{v'^{*}}{\alpha}, u'' = \frac{u'^{*}}{\alpha}$, $\hat{u}_0'' = \frac{\hat{u}_0}{\alpha}$. At $\pi'' = (v'', u'', \hat{u}_0'')$, $R_f = 1$, $R_j < 1$, for $j \neq f$. Also, π'' provide a feasible solution to the dual of LP-6, i.e., LP-7, for DMUf. This implies, $(u''Y_f + \hat{u}_o'') - v''X_f = 0$ and $(u''Y_j + \hat{u}_o'') - v''X_j < 0$ for $j \neq f$. The corresponding primal solution for LP-6, $\lambda_j = 0$ for $j \neq f$ and $\lambda_f = 1$, is feasible with the same objective function value of 1 as the dual LP. The primal solution is unique as $(u''Y_j + \hat{u}_o'') - v''X_j \neq 0$ for $j \neq f$. Hence, the primal and dual feasible solutions are also optimal and DMUf is a generator by Characteristic 2. **Q.E.D.**

[14] We assume here that $v'^{*}X_f \neq 0$. If $v'^{*}X_f = 0$, then using the closed-form solution introduced in chapter 7, we can arrive at a positive set of weights for DMUf such that at this new set of weights, $\underset{j \in U}{ArgMax}\{RC_j\}$=DMUf is unique. We can then apply the proof presented here to prove that DMUf is a generator.

From lemma 5.2 it follows that if $\underset{j \in U}{ArgMax}\{RC_j\} = f$ is unique and $f = t$, then DMUt is a generator. In this case, we append DMUt to set GEN and proceed to the next iteration. If $f \neq t$, the status of DMUt is still unresolved; one can now evaluate DMUt again against the augmented set of generators in the next iteration. If there is a tie for $\underset{j \in U}{ArgMax}\{RC_j\}$, there are different ways of breaking it and they are discussed in chapter 7. The final outcome is a new generator.

5.3.2 LP infeasibility

As the VRS model is enclosed in the CRS model, the technical challenges that need to be taken care of in the CRS case also apply for its VRS counterpart. Furthermore, the VRS model has additional complications. First, GBA LP-5 for DMUt can be infeasible even when the data is strictly positive. This can be seen from the example in table 5-5 below.

DMU	***X1***	***Y1***	***Y2***
A	5	6	2
B	5	1	7
C	5	5	5

Table 5-5 : LP infeasibility with positive data

All the 3 DMUs consume 5 units of input 1 to produce two outputs. We know that DMUs that use minimum of any inputs or produce maximum of any outputs if unique are generators. Since all the DMUs consume the same amount of input 1, we have DMU A (unique max for *Y1*) and DMU B (unique max for *Y2*) as our starting subset of generators. Suppose we evaluate DMU C, the resultant GBA LP is shown below.

Minimise θ'_C
subject to,
$\theta'_C 5 - \gamma_A 5 - \gamma_B 5 \geq 0$
$0 + \gamma_A 6 + \gamma_B 1 \geq 5$
$0 + \gamma_A 2 + \gamma_B 7 \geq 5$
$0 + \gamma_A 1 + \gamma_B 1 = 1$
θ'_C *free*; $\gamma_A, \gamma_B \geq 0$

This is infeasible. If we combine constraints 2 and 3 we get, $0 + \gamma_A 8 + \gamma_B 8 \geq 10$ which is impossible for $\gamma_A + \gamma_B = 1$.

When the input-oriented CRS model is infeasible, the corresponding VRS model is also infeasible as both have the same LP formulation with the VRS model having an additional convexity constraint. And we now know that the input-oriented VRS model can become infeasible *even* with strictly positive data. Clearly, infeasibility is a more common issue when applying GBA to solve the input-oriented VRS model.

5.3.3 Indeterminate ratios

If we chose to use RC_j in procedure FindNewGen, the question of indeterminate ratios does not arise because reduces costs are not ratios. If we do opt for R_j as defined at the start of this section, indeterminate ratios can arise as shown in the example below.

DMU	*X1*	*X2*	*X3*	*Y1*
A	10	2	8	24
B	12	0	7	38
C	0	8	5	28

Table 5-6 : Indeterminate ratios

Using random weights (2, 3, 2, 1) for input 1, input 2, input 3, and output 1 respectively, and $u_0 = 0$, we can identify DMU B as a generator. Suppose, we evaluate DMU A against DMU B in set GEN; the resultant LP is shown below.

Minimise θ'_A
subject to,

$$\begin{aligned}
&\theta'_A 10 - \gamma_B 12 \geq 0\\
&\theta'_A 2 - \gamma_B 0 \geq 0\\
&\theta'_A 8 - \gamma_B 7 \geq 0\\
&0 + \gamma_B 38 \geq 24\\
&0 + \gamma_B = 1\\
&\theta'_A \ free; \gamma_B \geq 0
\end{aligned}$$

The solution to the above LP is $\theta'^*_A = 1.2$, $v'^* = (0.1, 0, 0)$, $u'^* = (0)$, $u^*_0 = 1.2$. The ratio R_C at this π'^*_t is indeterminate. Incidentally, if we had chosen to evaluate DMU C against DMU B, the resultant LP is infeasible.

As in the CRS case, indeterminate ratios cannot occur when the data is strictly positive. This can be seen from GBA LP-6. The first constraint needs to be satisfied as an equality at any optimal solution, i.e., $v'^* X_t = 1$. Hence, $v'^* \neq 0$. With $X_j > 0$, $v'^* X_j > 0$ so that the ratios $R_j = \dfrac{u'^* Y_j + u^*_0}{v'^* X_j}, \forall j \in U$ cannot result in an indeterminate form.

5.4 GBA for the output-oriented VRS model

The LP to be used in conjunction with GBA for solving the output-oriented VRS model is shown below.

Maximise η'_t
subject to,

$$\begin{aligned}
&0 + \sum_{j \in GEN} \rho_j X_j \leq X_t\\
&\eta'_t Y_t - \sum_{j \in GEN} \rho_j Y_j \leq 0 \qquad\qquad \textbf{(GBA LP-7)}\\
&0 + \sum_{j \in GEN} \rho_j = 1\\
&\eta'_t \ free; \ \rho_j \geq 0, j \in GEN
\end{aligned}$$

The output-oriented model aims to find the maximum multiple of outputs possible with the inputs currently consumed by DMUt. Hence, the optimal objective function value is $\eta_t'^* \geq 1$ when evaluating a DMUt that is inside the partial PPS. When evaluating a DMUt that is outside the partial PPS, if GBA LP-7 is feasible, the objective function value is bounded between 0 and 1, i.e., $0 \leq \eta_t'^* < 1$. This implies that in certain cases a "zero" solution, $\eta_t'^* = 0,\ \sum_{j \in GEN} \rho_j^* = 1$, is a feasible solution for DMUt. We will revisit the "zero solution" to GBA LP-7 towards the end of this section.

The corresponding dual to GBA LP-7 is shown below.

$$\begin{aligned}
&\textit{Minimise } v'X_t + v_0 \\
&\textit{subject to,} \\
&u'Y_t = 1 \qquad\qquad\qquad\qquad \textbf{(GBA LP-8)} \\
&-u'Y_j + v'X_j + v_0 \geq 0, \forall j \in GEN \\
&u', v' \geq 0; v_0 \textit{ free}
\end{aligned}$$

If we chose to use R_j for this model, we will have to redefine it to take account of v_0^*. Specifically, the ratios to be evaluated are $R_j = \dfrac{u'^* Y_j}{v'^* X_j + v_0^*}, \forall j \in U$. GBA as described in 4.3.3 can be applied without further modifications.

In addition, as in the input-oriented case, we can (in place of the ratio values) use the reduced cost values at $\pi_t'^* = (v'^*, u'^*, v_0^*)$ to decide whether DMUt's efficiency analysis is complete or that we need to identify a new generator. So instead of computing ratio values, we can compute the reduced cost values $RC_j = u'^* Y_j - (v'^* X_j + v_0^*), \forall j \in U$ to identify a new generator. The rest of the procedure is the same as in applying GBA to solve the input-oriented VRS model using RC_j.

5.4.1 Infeasibility

As in the input-oriented case, the output-oriented VRS model can become infeasible even when the data is strictly positive. This can be seen from the example below.

DMU	*X1*	*X2*	*Y1*
A	6	2	5
B	1	7	5
D	3	4	5

Table 5-7 : LP infeasibility with positive data

All DMUs produce 5 units of output 1 by consuming varying amounts of inputs 1 and 2. Since all DMUs produce the same quantity of output 1, if we look at the input factors, we have DMU A (unique min for input 2) and DMU B (unique min for input 1) as our starting subset of generators. Suppose, we now evaluate DMU D, the corresponding LP is shown below:

Maximise η'_D
subject to,

$$\begin{aligned}
&0 \;+ \rho_A 6 + \rho_B 1 \le 3 \\
&0 \;+ \rho_A 2 + \rho_B 7 \le 4 \\
&5\eta'_D - \rho_A 5 - \rho_B 5 \le 0 \\
&\qquad \rho_A + \rho_B = 1 \\
&\eta'_D \textit{ free};\ \rho_A, \rho_B \ge 0
\end{aligned}$$

This is infeasible. If we combine constraints 1 and 2, we have $\rho_A 8 + \rho_B 8 \le 7$. This is impossible for $\rho_A + \rho_B = 1$.

5.4.2 Indeterminate ratios

As in the input-oriented case, if we chose to use RC_j in procedure FindNewGen, the question of indeterminate ratios does not arise. If we do opt for R_j as defined at the start of this section, indeterminate ratios can arise. As observed earlier, any technical challenge with the output-oriented CRS model will also be reflected here and we saw in section 5.2.2 that indeterminate ratios was one such challenge when the dataset contains zeroes.

Interestingly, unlike other models, indeterminate ratios can occur in this model even when the data is strictly positive. This is illustrated using the example below.

DMU	***X1***	***X2***	***Y1***
A	2	0.5	3
B	4.5	4.5	7
C	0.125	2	4
D	3.25	2.5	5

Table 5-8 : Indeterminate ratios with positive data

The above data is strictly positive and units A, B and C are generators. Suppose, at a particular iteration we had identified DMUs A and B to be generators. If we evaluate DMU D against A and B in GEN, the optimal solution to GBA LP-7 is $\eta_A'^* = 1$, $v'^* = (8,0)$, $u'^* = (5), v_0^* = -1$. The ratio R_C at this $\pi_t'^*$ is indeterminate. Incidentally, if we had chosen to evaluate DMU C against the generators A and B, it is easy to see that the resultant LP will be infeasible.

As indicated at the beginning of this section, we will now examine the condition under which a "zero solution" to GBA LP-7 can occur. Specifically, we will show using lemma 5.3 that when the data is strictly positive, a "zero" solution cannot occur while solving the output-oriented VRS model using GBA.

Lemma 5.3: A "zero" optimal solution does not exist for GBA LP-7 when the dataset is strictly positive.

Proof: When GBA LP-7 is feasible, the optimal objective function value is bounded below at 0, i.e., $\eta_t'^* \geq 0$. To prove that $\eta_t'^* \neq 0$ when the data is strictly positive, we restate GBA LP-7 below.

$$
\begin{aligned}
&Max\ \ \eta_t' + \sum_{j\in GEN} \rho_j 0 \\
&subject\ to, \\
&\ \ 0\ +\ \sum_{j\in GEN} \rho_j X_j \le X_t \\
&\eta_t' Y_t\ -\ \sum_{j\in GEN} \rho_j Y_j \le 0 \\
&0\ \ +\ \sum_{j\in GEN} \rho_j = 1 \\
&\eta_t'\ free;\ \rho_j \ge 0, j \in GEN
\end{aligned}
\qquad \textbf{(GBA LP-7)}
$$

Without loss of generality, choose a DMUg (observed or obtained through a convex combination of some generators) in set GEN as a comparator unit for DMUt. Let the input and output vector of DMUg be represented by (X_g, Y_g). Assume that GBA LP-7 for DMUt is feasible; a necessary condition for this is $X_g \le X_t$ as indicated from the input constraints. Using simple algebra, we can see from the output constraints that a lower-limit on η_t is given by $\eta_t \ge w$, where $w = \underset{s=1}{\overset{m_2}{Min}} \left\{ \frac{Y_{sg}}{Y_{st}} \right\}$. As the dataset is strictly positive, $w > 0$ and since the sense of the objective function is maximisation, $\eta_t^* > 0$. **Q.E.D.**

Lemma 5.3 along with the section on infeasibility discussed in section 5.4.1 shows that when the dataset is strictly positive, either GBA LP-7 for DMUt is infeasible or else $\eta_t^* > 0$.

5.5 GBA for the CRS additive model

The super-efficiency form of the additive CRS model employed in GBA to evaluate DMUt with respect to the set of generators in GEN is shown below.

$$
\begin{aligned}
&Maximise\quad e^i s^i + e^o s^o + 0\lambda_j \\
&subject\ \ to, \\
&s^i\ +\ 0\ \ +\ \sum_{j\in GEN} \lambda_j X_j = X_t \\
&0\ \ -\ s^o\ +\ \sum_{j\in GEN} \lambda_j Y_j = Y_t \\
&s^i, s^o, \lambda_j \ge 0, j \in GEN
\end{aligned}
\qquad \textbf{(AGBA LP-1)}
$$

where, s^i, s^o are the input and output slack vectors of dimension m_1 and m_2 respectively; e^i and e^0 are conformable vectors of 1's of dimension m_1 and m_2 respectively. The dual to the LP problem AGBA LP-1 is shown below.

$$\begin{aligned}&\textit{Minimise} \quad vX_t + uY_t\\&\textit{subject to},\\&vX_j + uY_j \geq 0, \forall j \in GEN \qquad\qquad \textbf{(AGBA LP-2)}\\&v \geq +1\\&u \leq -1\end{aligned}$$

As we are solving a CRS model, it is not appropriate to use RC_j here. So GBA for this model parallels to the oriented CRS cases although the ratio R_j requires a minor modification as $u \leq -1$.

We note that if DMUt is outside the partial PPS, it will be identified by $v^*X_t + u^*Y_t < 0$. In other words, $1 < \frac{-u^*Y_t}{v^*X_t} = R_t$ which essentially is our original ratio test for the oriented CRS models as $u \leq 0$ and hence, $-u \geq 0$ and $R_t > 1$. For the units $j \in GEN$ that are in the optimal basis of AGBA LP-1, $v^*X_j + u^*Y_j = 0$ or $R_j = 1$. Hence, except for the LP problem solved, the procedure for solving the CRS additive model using GBA essentially remains the same as in 4.3.3 with $R_j = \frac{-u^*Y_j}{v^*X_j}$.

5.5.1 Infeasibility

It is interesting to note that even when the dataset is strictly positive, when using GBA for this model, the resultant LP can become infeasible. As we aim to maximise both the inputs and outputs slacks of DMUt and as the slacks are constrained to be non-negative (i.e., $s^i, s^o \geq 0$), if DMUt lies outside the partial PPS, the corresponding LP problem will be infeasible; obviously, this can happen regardless of whether the dataset is strictly positive or otherwise. In other words, DMUt's activity must be dominated by some of the units in set GEN for

AGBA LP-1 to be feasible. To illustrate this point, let us consider the following example.

DMU	*X1*	*Y1*	*Y2*
A	1	2	3
B	1	6	2

Table 5-9 : LP infeasibility with positive data

Both A and B are extreme-efficient and our data is strictly positive. Suppose our set GEN consists of unit A and we evaluate B[15]. For the oriented case, the corresponding oriented GBA models are feasible as the data is strictly positive. However, AGBA LP-1 for DMU B is infeasible as DMU B is not dominated by A.

5.5.2 Indeterminate ratios

The issue of indeterminate ratios does not arise when employing GBA to solve the additive CRS model as the input weights are constrained to be strictly positive, i.e., $v \geq +1$.

5.6 GBA for solving the VRS additive model

The standard additive VRS model is shown below.

$$\begin{array}{ll}
\textit{Maximise} & e^i s^i + e^o s^o + 0\lambda_j \\
\textit{subject to}, & \\
s^i + 0 + \sum_{j=1}^{n} \lambda_j X_j = X_t & \\
0 - s^o + \sum_{j=1}^{n} \lambda_j Y_j = Y_t & \qquad \textbf{(VALP-1)} \\
0 + 0 + \sum_{j=1}^{n} \lambda_j = 1 & \\
s^i, s^o, \lambda_j \geq 0, j \ in \ 1,\dots, n &
\end{array}$$

[15] Note that this is equivalent to solving the standard additive CRS super-efficiency model (LP-10 in chapter 2) for unit B as there are only two units in our dataset. Hence, the result presented here also applies for the standard additive CRS super-efficiency model.

where, s^i, s^o are the input and output slack vectors respectively of dimension m_1 and m_2 respectively; e^i and e^0 are conformable vectors of 1's of dimension m_1 and m_2 respectively.

The dual to the above model is shown below.

$$\begin{aligned}
&\textit{Minimise} \quad vX_t + uY_t + \beta \\
&\textit{subject to}, \\
&vX_j + uY_j + \beta \geq 0, \forall j \\
&v \geq +1 \\
&u \leq -1 \\
&\beta \ \textit{free}
\end{aligned} \qquad \textbf{(VALP-2)}$$

where, β is the dual value of the convexity constraint and is unconstrained in sign.

The super-efficiency form of the additive VRS model employed in GBA to evaluate DMUt with respect to the set of generators in GEN is shown below.

$$\begin{aligned}
&\textit{Maximise} \quad e^i s^i + e^o s^o + 0\lambda_j \\
&\textit{subject to}, \\
&s^i + 0 + \sum_{j\in GEN} \lambda_j X_j = X_t \\
&0 - s^o + \sum_{j\in GEN} \lambda_j Y_j = Y_t \\
&0 + 0 + \sum_{j\in GEN} \lambda_j = 1 \\
&s^i, s^o, \lambda_j \geq 0, j \in GEN
\end{aligned} \qquad \textbf{(AGBA LP-3)}$$

The dual to AGBA LP-3 is shown below.

$$\begin{aligned}
&\textit{Minimise} \quad vX_t + uY_t + \beta \\
&\textit{subject to}, \\
&vX_j + uY_j + \beta \geq 0, \forall j \in GEN \\
&v \geq +1 \\
&u \leq -1 \\
&\beta \ \textit{free}
\end{aligned} \qquad \textbf{(AGBA LP-4)}$$

As in the oriented VRS models, we can use the reduced cost values at $\pi_t'^* = (v'^*, u'^*, v_0^*)$ to decide whether DMUt's efficiency analysis is complete or that we need to identify a new generator. If an unit, DMUt is outside the partial

PPS, it will be identified by $v^*X_t + u^*Y_t + \beta^* < 0$ or $-v^*X_t - u^*Y_t - \beta^* > 0$. As $u \le 0, -u \ge 0$ and hence its reduced cost $RC_t = -v^*X_f - u^*Y_f - \beta^* > 0$. It follows that when $Max\{RC_j\} > 0, j \in U$, one can compute $ArgMax\{RC_j\}, j \in U$ to identify a new generator. Also, when $Max\{RC_j\} \le 0, j \in U$, the efficiency analysis for DMUt is resolved.

If we chose to use R_j for this model, we will have to take account of β^*. First we note that for the units j in set GEN that are in the optimal basis of AGBA LP-3, $v^*X_j + u^*Y_j + \beta^* = 0$ or $\dfrac{-u^*Y_j - \beta^*}{v^*X_j} = \dfrac{-u^*Y_j}{v^*X_j + \beta^*} = 1$. For a DMUt that lies outside the partial PPS, $RC_t > 0$; in other words, $1 < \dfrac{-u^*Y_t - \beta^*}{v^*X_t} = R_t$ or $1 < \dfrac{-u^*Y_t}{v^*X_t + \beta^*} = R_t$. This implies that the ratio values given by $R_j = \dfrac{-u^*Y_j - \beta^*}{v^*X_j}$ or $R_j = \dfrac{-u^*Y_j}{v^*X_j + \beta^*}$ can be used in GBA for the VRS additive model.

5.6.1 Infeasibility

The GBA additive VRS model will be infeasible when the corresponding CRS model is infeasible. As indicated in section 5.5.1, the additive CRS GBA model can become infeasible even when the data is strictly positive. Hence, the GBA additive VRS model can also become infeasible regardless of whether the data is strictly positive or otherwise.

5.6.2 Indeterminate ratios

If we chose to use RC_j in procedure FindNewGen, the question of indeterminate ratios does not arise. Even if we do opt for the ratio value $R_j = \dfrac{-u^*Y_j - \beta^*}{v^*X_j}$ in FindNewGen, indeterminate ratios cannot occur as $v \ge +1$.

5.7 Conclusion

In this chapter we extended the application of GBA to solve the oriented and non-oriented CRS and VRS models without the assumption of strictly positive data. We also set forth the technical challenges of LP infeasibility and indeterminate ratio that one can encounter when applying GBA to solve these models. Importantly, we have shown in this chapter that GBA can handle DEA models with weight restrictions by way of considering CRS and VRS additive models in which the weights are constrained to be non-zero. The upshot is that GBA can handle DEA models with additional constraints in terms of additional weight restrictions as long as the resulting model is linear.

Ways to resolve the technicalities are presented in chapters 6 and 7.

6 WAYS TO RESOLVE THE LP INFEASIBILITY ISSUE IN GBA[16]

The purpose of this chapter is to present ways to handle the principal technical challenge of GBA, viz., LP infeasibility, for solving all the DEA models discussed in chapter 5 except for the output-oriented CRS model which is never infeasible as shown in section 5.2.1. The main tools developed are two penalty methods and a model specific clustering technique for the DfE dataset. The clustering technique presented in section 6.1.1 works when zeroes exist only in the *input* factors of a CRS dataset while the penalty methods can tackle infeasibility under all circumstances. Within the penalty methods, only the big penalty method can guarantee that whenever possible, DMUt is evaluated only against the units in set GEN adhering to the GBA procedure introduced in chapter 4. The small penalty method can only guarantee that DMUt will not evaluate with itself if it lies inside the partial PPS. Both the penalties invoke FindNewGen appropriately and are valid for use within GBA. For ease of computation, the small penalty method is to be preferred to avoid LP infeasibility in GBA.

6.1 Infeasibility in the input-oriented CRS model

The input-oriented CRS model can become infeasible only when there are zeroes in the data. In this section, we will examine two different approaches to handle infeasibility when using GBA for this model, viz., clustering and penalty.

6.1.1 Natural Clustering for the input oriented CRS case

The DfE model of primary schools in England contained zeros only in one input factor. While working with this dataset, we developed the following clustering approach.

[16] Dr. Argyris had developed alternative LP models to handle the infeasibility issue in GBA. These models are not discussed in the thesis and are presented in Appa et al (2006b).

When zeroes exist *only* in the input factors of a dataset, a simple method to avoid infeasibility is to cluster DMUs based on the input factors having a zero value for some DMUs and treat each cluster as a separate DEA problem. The number of clusters depends on the number of combinations of input factors having a zero value; in other words, the number of clusters depends on the *different pattern* of zeroes in the input factors that exist in a dataset. If r is the number of input factors that have a zero value for some DMUs $(r \le m_1)$, the number of theoretically possible input combinations/clusters is given by $(2^r - 1)$. For a given problem, some of these clusters could be empty. We will describe the clustering technique using the following example.

DMU	*X1*	*X2*	*X3*	*X4*	*Y1*	*Y2*
A	4	0	0	0	3	6
B	6	0	0	0	14	8
C	7	1	0	0	8	9
D	3	9	0	0	7	11
E	4	0	6	0	15	2
F	6	0	0	8	13	5
G	8	0	1	2	10	10
H	4	4	9	9	4	6
I	8	2	0	5	14	12
J	5	6	7	3	12	12

Table 6-1 : Clustering DMUs based on zero valued input factors

The steps to be followed in employing the clustering technique are as follows:

i. **Define the clusters:**

Identify the r input factors that have a zero value for some DMUs and the corresponding DMUs that have them. Cluster the DMUs in terms of the right combination of zero containing input factors that they uniquely share. This can be achieved as follows.

Let C be the set of inputs belonging to cluster c. Then DMUj$\in c$ iff$X_{rj} = 0$ when $r \in C$ and $X_{rj} > 0$ otherwise. This ensures that each DMU with zeros for some inputs belongs to a unique cluster c. For our example in 6-1, we can see that only DMUs H and J have strictly positive (input) data and so all other DMUs have to be clustered based on the pattern of zero valued input factors. Three inputs, viz., inputs 2, 3, and 4 have zero value for some DMUs while input 1 is strictly positive for all DMUs. Hence, $r = 3$ and the possible number of clusters are$\left(2^r - 1\right) = \left(2^3 - 1\right) = 7$. However, there is no DMU with zeros for input 4 only. So there are six clusters in all and details of them can be seen in table 6-2 below.

Input Cluster c	$\{X2, X3, X4\}$	$\{X3, X4\}$	$\{X2, X4\}$	$\{X2, X3\}$	$\{X2\}$	$\{X3\}$	$\{X4\}$
DMUs with $X_{rj} > 0$ iff $r \notin C$	A,B	C,D	E	F	G	I	None

Table 6-2 : Clusters in example 6-1

ii. Define the order in which clusters will be analysed:

The idea behind clustering is that in analysing the DMUs within a cluster, the zero valued inputs can be ignored because all of them share the zero value.

This leaves the clustered DEA problems with strictly positive data so that the resultant GBA LP cannot be infeasible[17].

To apply GBA with a clustering technique that achieves this we need to establish a hierarchy among clusters. This is done with the notion of a parent cluster. Cluster c_i is a parent of cluster c_j if $C_j \subset C_i$ where, C_i and C_j are the sets of inputs belonging to cluster c_i and c_j respectively. In example 6-1, cluster $\{X2, X3, X4\}$ is the parent cluster for all the other 5 clusters, while $\{X3, X4\}$ is a parent cluster for $\{X3\}$ but not for $\{X2\}$. Note that a cluster can have more than one parent and also that a cluster with a parent can also be a parent of some other cluster. Figure 6.1 below illustrates this hierarchy. In it, each cluster is represented as a node and a directed arc $c_i \rightarrow c_j$ represents the fact that cluster c_i is a parent of cluster c_j. Also, the number of layers in a clustered problem depends on the cardinality of the clusters. For our example in 6.1, cluster$\{X2, X3, X4\}$ is of cardinality 3 and is placed in layer 3. Clusters $\{X3, X4\}, \{X2, X4\}$ and $\{X2, X3\}$ are of cardinality 2 and are placed in layer 2. Finally, clusters$\{X2\}, \{X3\}$ and $\{X4\}$ are of cardinality 1 and are placed in layer1.

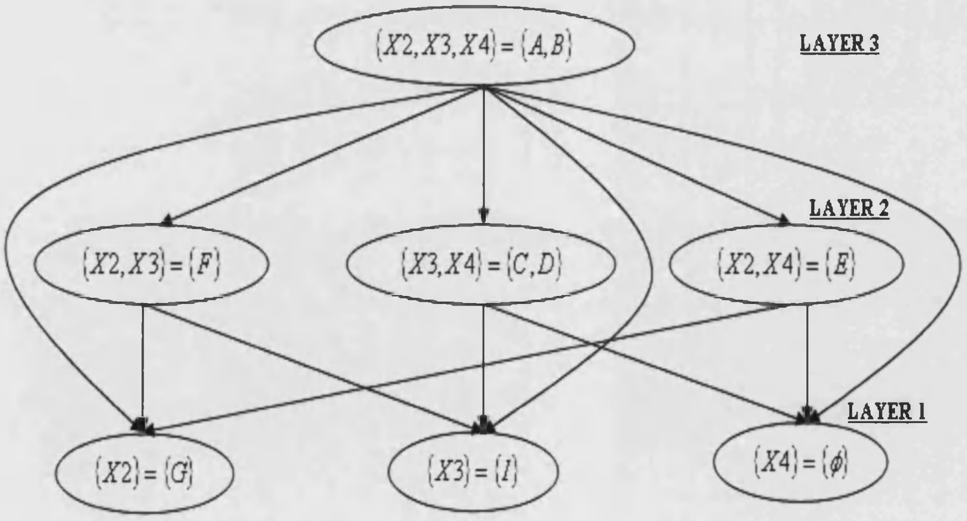


Figure 6-1 : Underlying hierarchical structure of example 6-1

[17] Clustering in this fashion cannot lead to infeasibility even if the zero valued input factors within clusters are retained. The reason being that the units within a cluster share the *same pattern of zeros* in them.

iii. Apply GBA to DMUs within a cluster ignoring the cluster defining inputs:

We apply GBA on the clustered DMUs first, taking one cluster at a time in a hierarchical order that ensures that a cluster is analysed only after all its parent clusters have been analysed. In other words, if R is the set of clusters remaining to be analysed, choose the next cluster c from R in such a way that there is no cluster $c' \in R$ that is a parent of c.

Applying this rule to example 6-1 implies that cluster $\{X2, X3, X4\}$ in layer 3 of figure 6.1 containing DMUs A and B is the first cluster to be solved after which we can solve any one of the three clusters in layer 2. Suppose the second cluster to be solved is $\{X3, X4\}$ and the third one is $\{X2, X4\}$. Then the fourth can be $\{X2, X3\}$ or $\{X4\}$ because neither has a parent in R at this stage.

As all the DMUs within a cluster have zero value for the same set of inputs, GBA can be applied to them after blocking the cluster defining inputs. So, for example 6-1, the first cluster with three inputs will give rise to a 1 input 2 outputs DEA problem (by blocking inputs 2, 3, and 4). Similarly, while evaluating cluster $\{X2, X4\}$ after the parent node cluster is evaluated, we can block inputs 2 and 4.

iv. Define set GEN before applying GBA to each cluster:

Start applying GBA to a cluster without any parents - $\{X2, X3, X4\}$ in our example. For any such cluster, set GEN is empty at the initial stage of applying GBA. We can apply the usual heuristics to identify a generator among them. On the other hand, for any cluster with one or more parents, the union of GENs found in all the parent clusters defines the initial set GEN. This ensures that all the DMUs in the initial set GEN for cluster c also have zeros in the input factors contained in set C defining cluster c.

An alternative simpler way to define GEN for a cluster c is to append all the generators from the previous layers to set GEN. However, if we do this it is no longer valid to disregard input factors of set C in applying GBA to cluster c as some of the units in GEN will not share the same set of zeroes compared to set C. For example, while evaluating cluster $\{X2\}$, the only generators that need to be appended are from its parent nodes, viz., $\{X2, X3, X4\}$, $\{X2, X3\}$ and $\{X2, X4\}$. We can then block input 2 while evaluating DMU G in cluster $\{X2\}$. However, if we also append the generators (if any) from cluster $\{X3, X4\}$ while evaluating cluster $\{X2\}$, we cannot block input 2 as cluster $\{X3, X4\}$ does not contain DMUs that have a zero value for input 2. So this simpler method for creating GEN will end up with larger LPs.

v. GBA for the non-clustered DMUs:

After completing the efficiency analysis for all the clustered DMUs in the hierarchy described above, the generators identified from all the clusters will now form the starting set of generators when applying GBA to the DMUs that have strictly positive data - in our example for DMUs H and J. Apply GBA to this set of non-clustered DMUs to complete the analysis.

vi. Finer details of the clustering technique:

A simple way to program the technique is to divide the clusters into r layers based on the cardinality of the set of input factors in cluster c_k. Let set K defining the layers have numbers $\{1,\ldots,r\}$. Then for $\chi \in K$, cluster c_k is placed in layer χ if $|C_k| = \chi$, where C_k is the set of inputs belonging to cluster c_k and $|C_k|$ denotes the cardinality of the set C_k. Now we apply GBA first to clusters in layer r (provided there are any), followed by layer $r-1$ and so on. The advantage is that there cannot be a parent for any cluster from its own or lower levels. Figure 6.1 illustrates the levels and hierarchy for example 6-1. Within each level there is still some choice in choosing the cluster to be evaluated which can be resolved arbitrarily. Efficiency analysis of clusters in the same layer can be carried out independently with only their parent generators forming the starting set of generators.

For the clusters at the highest level the following characteristics hold:

- Given that zero is the best possible input value a DMU can have, if the cluster(s) in the highest layer have a single DMU, then it must be a generator.

- If cluster(s) in the highest layer has more than one unit, as in example 6-1, we apply GBA within them to identify some generators as there must be at least one among them.
- We can complete the efficiency analysis of all the DMUs within these cluster(s) without reference to generators from outside the cluster as for the non-generators in them, only the generators in them can act as peers.

vii. Illustration of the clustering technique for example 6-1:

The first cluster to be looked at is the cluster $\{X2, X3, X4\}$ in layer 3 containing DMUs A and B. Upon evaluating cluster $\{X2, X3, X4\}$ using GBA, we find that both A and B are generators. The maximum size of the GBA LP solved at this stage is (3×2).

Now, we move to the clusters in layer 2 and there are three clusters to choose from. Suppose, we choose arbitrarily cluster $\{X3, X4\}$ containing units C and D. $GEN = \{A, B\}$ at this stage. While evaluating this cluster by blocking inputs 3 and 4, we find that unit D is a generator and unit C is inefficient with peers A, B and D. The maximum size of the GBA LP solved at this stage is (4×4). Suppose, we next choose cluster $\{X2, X3\}$ containing unit F. $GEN = \{A, B\}$ and the size of the GBA LP solved is (4×3). While evaluating F by blocking inputs 2 and 3, we find that the unit is inefficient and its only peer is DMU B. We then move to the only remaining cluster $\{X2, X4\}$ in level 2 containing DMU E. Given that $GEN = \{A, B\}$, the size of the GBA LP solved is again (4×3). While evaluating E by blocking inputs 2 and 4, we find that unit E is a generator. As we have evaluated all the clusters in layer 2, we can move to clusters in layer 1.

Suppose we arbitrarily choose cluster $\{X2\}$. The starting set of generators is $GEN = \{A, B, E\}$ given by the union of generators found in analysing all its parents, viz., $\{X2, X3, X4\}$, $\{X2, X3\}$ and $\{X2, X4\}$. Evaluating unit G by blocking input 2, we find that the unit is inefficient and its peers are units A and B. The size of the GBA LP solved at this stage is (5×4).

Now analyse cluster $\{X3\}$. Set $GEN = \{A, B, D\}$ and upon evaluating DMU I by blocking input 3, we find that the unit is inefficient and its peers are units A, B and D. The size of the GBA LP solved is again (5×4).

Now that the efficiency analysis of all the clustered DMUs is completed, we can move to the set of units with strictly positive data. The generators identified from *all* the clusters will now form the starting set of generators when applying GBA to the DMUs H and J. So, $GEN = \{A, B, D, E\}$, and we find that unit J is a generator and unit H is inefficient and its peers are units A and J. The maximum size of GBA LP solved at this stage is (6×6).

6.1.1.1 Pros and Cons of the Clustering technique

Natural clustering is a simple technique to overcome the GBA LP infeasibility issue when the DEA dataset has few input factors that take 0 values. In the case of the DfE's data on primary schools, only one of the school's input factors, namely, % of students taking English as an Additional Language (EAL) has zero values. This means that we are dealing with just one cluster, viz., the cluster of DMUs having zero value for the input factor EAL. After completing the efficiency analysis of all the units in this cluster (by appropriately blocking the EAL factor), we can move to the cluster of DMUs having strictly positive data with a list of generators. This is similar to the hierarchical decomposition principle introduced by Barr and Durchholz (1997) in that we decompose the original problem into smaller sub-problems; however, the sub-problems are trimmer as we block the (common) input factors having zero values within each cluster and the efficiency analysis does not require a second phase. Hence, the clustering method can be seen as a hybrid of the decomposition techniques described in Barr and Durchholz (1997) and Korhonen and Siitari (2009).

The drawbacks of the natural clustering technique are obvious. First, the number of clusters can quickly explode as there is a combinatorial escalation. Secondly, the hierarchical order to evaluate clusters as presented for the example above, requires the parent generators to be stored and recalled properly in different layers. This could be tedious and taxing on the run time although parallel processing as discussed in Barr and Durchholz (1997) could possibly alleviate it. However, the issue of recalling generators appropriately is not a necessity of the clustering technique, i.e., one can instead have all the generators identified so far (and not just from its parents) from all the previous layers in set GEN when evaluating any cluster.

In addition, we showed in section 5.1.1, that GBA LP infeasibility can occur in the input-oriented CRS model when there are zero values in the output factors. Clustering as explained above will not work in the case of zeroes in the output factors. This is because, in case of inputs factors, zero is the best possible value a DMU can have while it is the worst value for any output factor (assuming that the data is non-negative). Whether a DMU with a zero value for certain output factors is a generator or not depends on the input-output correspondence of its other factors relative to all other DMUs. An example elucidating this was shown in table 5-2.

In the next section, we present the penalty method by which the GBA LP infeasibility issue can be resolved regardless of which factors contain zeroes. This is achieved by introducing DMUt into the coefficient matrix along side the generators but penalising its use.

6.1.2 Penalty methods for the input oriented CRS case

6.1.2.1 Big Penalty or Big-M method

The Big-M method avoids GBA LP infeasibility by introducing DMUt into the coefficient matrix of generators. Its usage carries a very large penalty M thereby ensuring that DMUt compares with itself iff there is no feasible solution using only the set of generators in GEN. In other words, the modified GBA applied to DMUt behaves like the original GBA super-efficiency model (GBA LP-1) when there is a feasible solution to it and compares with itself when there isn't. This technique was first employed by Banker and Chang (2006) in the output-oriented VRS super-efficiency model for the purpose of identifying outliers. Here, we attempt to quantify M in the case of the input-oriented CRS model.

Recall that GBA LP-1 is defined as follows.

$$\begin{aligned}
&\textit{Minimise } \theta_t' \\
&\textit{subject to,} \\
&\theta_t' X_t - \sum_{j \in GEN} \gamma_j X_j \geq 0 && \textbf{(GBA LP-1)} \\
&\;0 \;+ \sum_{j \in GEN} \gamma_j Y_j \geq Y_t \\
&\theta_t' \textit{ free};\; \gamma_j \geq 0,\; j \in GEN
\end{aligned}$$

The modified GBA LP, MGBA LP-1, to avoid infeasibility in the input-oriented CRS super-efficiency model is shown below.

$$\begin{aligned}
&\textit{Minimise } \theta_t' + \sum_{j \in GEN} \lambda_j 0 + \lambda_t M \\
&\textit{subject to,} \\
&\theta_t' X_t - \sum_{j \in GEN}^{n} \lambda_j X_j - \lambda_t X_t \geq 0 && \textbf{(MGBA LP-1)} \\
&\;0 \;+ \sum_{j \in GEN}^{n} \lambda_j Y_j + \lambda_t Y_t \geq Y_t \\
&\theta_t' \textit{ free};\; \lambda_j,\, \lambda_t \geq 0; j \in GEN
\end{aligned}$$

The only difference between MGBA LP-1 and GBA LP-1 is that, along with the list of generators, DMUt is included in the coefficient matrix in MGBA LP-1 with penalty M in the objective function. This results in an additional column in MGBA LP-1 compared to GBA LP-1.

The dual to MGBA LP-1 is presented below.

$$\begin{aligned}
&\textit{Maximise } u'Y_t \\
&\textit{subject to,} \\
&v'X_t = 1 \\
&u'Y_j - v'X_j \leq 0, \forall j \in GEN && \textbf{(MGBA LP-2)} \\
&u'Y_t - v'X_t \leq M \\
&u', v' \geq 0
\end{aligned}$$

Obviously, MGBA LP-2 has an extra row compared to GBA LP-2.

The following two lemmas are important in proving theorem 6.1 which quantifies M.

Lemma 6.1: Applied to DMUt, GBA LP-1 results in infeasibility only if DMUt lies strictly outside the partial PPS spanned by the generators in set GEN.

Before proving lemma 6.1, we illustrate in figure 6-2 what an infeasible solution to GBA LP-1 looks like. In section 5.1.1, we had presented a two DMU DEA problem in table 5-1, wherein the GBA LP for DMU B is infeasible when evaluated against the generator DMU A. The corresponding illustration is shown in figure 6-2 below.

The partial PPS while evaluating DMU B is the region to the north-east of the generator DMU A and the boundary of the partial PPS is shown using dashed lines to the east and north of A. The radial projection direction for DMU B is along the X1 axis as its value for X2 is 0. The corresponding GBA LP is infeasible as it is not possible to reach the boundary of the partial PPS by means of this projection direction.

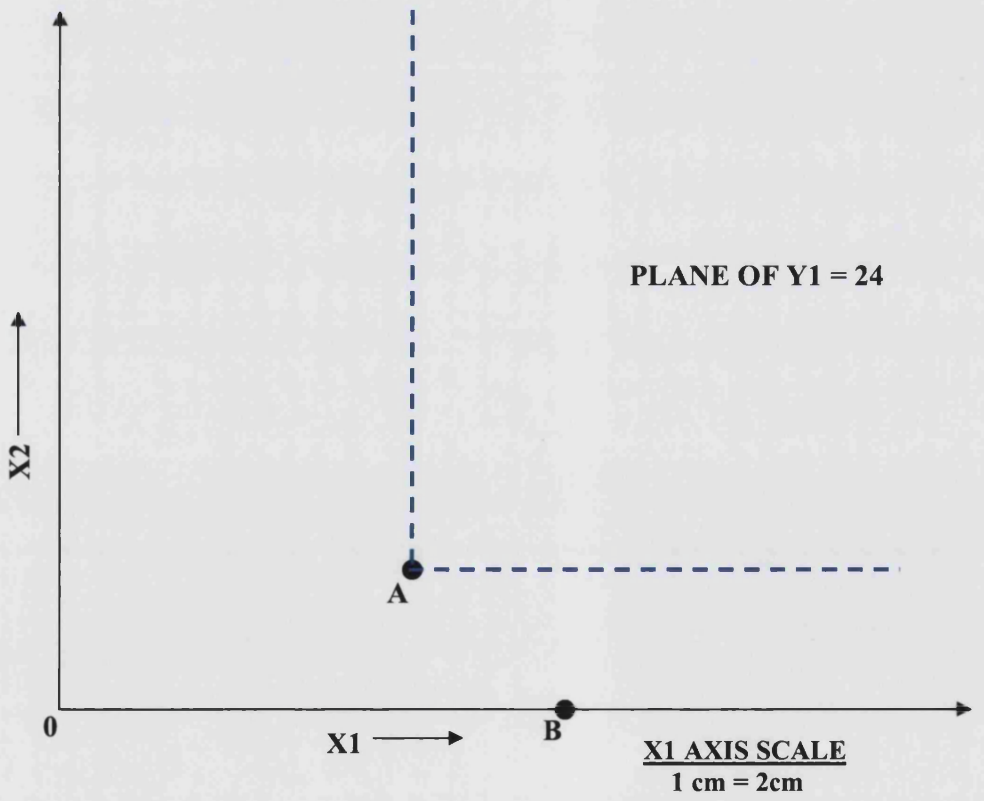


Figure 6-2 : Graphical illustration of infeasibility

Proof: For there to be a feasible solution to DMUt, its radial projection must lie on the boundary of the partial PPS. Suppose DMUt is not outside the partial PPS.

In this case, a linear combination of some of the generators dominates DMUt leading to $0 < \theta_t'^* \leq 1$. So unless a DMUt lies outside the partial PPS, it cannot effect an infeasible solution to GBA LP-1. **Q.E.D.**

Note that if DMUt is outside the partial PPS, GBA LP-1 can be feasible with $\theta_t'^* > 1$, or it may be infeasible as illustrated in figure 6-2.

We now proceed to establish a value for M. Let $X_{\max}$ be the largest input value, $X_{\min}^{+}$ the smallest *positive* input value, $Y_{\max}$ the largest output value and $Y_{\min}^{+}$ the smallest *positive* output value among all the inputs and outputs of all the DMUs in the dataset.

Formally,

$X_{\max} = Max\{X_{rj}\}$, for $r = 1,\dots,m_1; j = 1,\dots,n$;

$X_{\min}^{+} = Min\{X_{rj}\} > 0$, for $r = 1,\dots,m_1; j = 1,\dots,n$;

$Y_{\max} = Max\{Y_{sj}\}$, for $s = 1,\dots,m_2; j = 1,\dots,n$;

$Y_{\min}^{+} = Min\{Y_{sj}\} > 0$, for $s = 1,\dots,m_2; j = 1,\dots,n$;

Lemma 6.2: The maximum finite value that $\theta_t'^*$ can take as a solution to GBA LP-1 is given by $\frac{X_{\max}}{X_{\min}^{+}} \times \frac{Y_{\max}}{Y_{\min}^{+}}$.

Proof: DMUt can be positioned inside or outside the partial PPS. Lemma 6.1 demonstrates that for any DMUt that is inside the partial PPS, GBA LP-1 is feasible with a maximum finite value of $\theta_t'^* = 1$. Should $\theta_t'^*$ take a finite value when DMUt lies outside the partial PPS, GBA LP-1 has to be feasible, i.e., all the constraints must be satisfied simultaneously. If γ_j^* and $\theta_t'^*$ satisfy the most restrictive of the input and output constraints in GBA LP-1, they must satisfy all the m constraints. Hence by focusing on what could, *hypothetically,* be the most restrictive situation, we can work out the maximum value of $\theta_t'^*$.

The "extreme" situation with the m_1 input constraints occurs when $X^+_{\min} = X_{r't}$ and $X_{\max} = X_{r'g}$, $g \in GEN$ for some input factor r'; i.e., DMUt has the smallest positive input value in the dataset for a certain input factor r' and the overall maximum input value happens to occur for a DMUg in the set GEN, also for the same factor r'. The "extreme" situation with the m_2 output constraints occurs when $Y^+_{\min} = Y_{s'g}, g \in GEN$ and $Y_{\max} = Y_{s't}$ for some output factor s'; i.e., DMUt has the maximum output value for a certain output factor s' and the overall smallest positive output value occurs for DMUg in set GEN for the same output factor.[18]

Algebraically, this reduces to the following.

$$\gamma_j \, Y_{s'g} \geq Y_{s't} \Rightarrow \gamma_j \geq \frac{Y_{s't}}{Y_{s'g}};$$

$$\theta_t' X_{r't} \geq \gamma_j X_{r'g} \Rightarrow \theta_t' \geq \frac{X_{r'g}}{X_{r't}} \times \gamma_j;$$

$$\Rightarrow \theta_t' \geq \frac{X_{r'g}}{X_{r't}} \times \frac{Y_{s't}}{Y_{s'g}}; \; where, \; X_{r't} = X^+_{\min}, Y_{s't} = Y_{\max}, X_{r'g} = X_{\max}, Y_{s'g} = Y^+_{\min}.$$

Since the sense of the objective function is minimisation, the largest possible θ'^* value is given by $\underset{j=1}{\overset{n}{Max}} \{\theta_j'^*\} = \frac{X_{\max}}{X^+_{\min}} \times \frac{Y_{\max}}{Y^+_{\min}}$. **Q.E.D.**

Now we show that whenever there is a feasible solution to GBA LP-1 there is always a cheaper solution to MGBA LP-1 than the one with $\lambda_t = 1$. We show in Appendix 3 that it not possible to have an optimal solution to MGBA LP-1 with $0 < \lambda_t^* < 1$, i.e., either $\lambda_t^* = 0$ or $\lambda_t^* = 1$. This is valid for the penalty methods used throughout this chapter.

[18] Note that both the "extreme" conditions can occur simultaneously such that DMUt does not dominate (any of) the generator(s) in set GEN.

Theorem 6.1: In MGBA LP-1 the penalty value of $M = \frac{X_{\max}}{X^{+}_{\min}} \times \frac{Y_{\max}}{Y^{+}_{\min}}$ ensures that DMUt evaluates with itself only if the corresponding GBA LP-1 is infeasible.

Proof: The proof can be constructed by comparison with the following two cases where GBA LP-1 is feasible for DMUt.

Case 1: DMUt is inside the partial PPS.

There is always a feasible solution with $\lambda_t^* = 0$ and $\theta_t^* \leq 1$ which is cheaper than the solution with $\lambda_t^* = 1$ for which the overall objective function value is $M + \theta_t^*$.

Case 2: DMUt is outside the partial PPS but GBA LP-1 is feasible.

By lemma 6.2 we know that the maximum value $\theta_t'^*$ can take for any DMUt is $\frac{X_{\max}}{X^{+}_{\min}} \times \frac{Y_{\max}}{Y^{+}_{\min}}$. Since the λ_j's of the generators in the coefficient matrix have a 0 value in the objective function, if $\lambda_t^* = 0$, the maximum possible objective function value in this case is $\frac{X_{\max}}{X^{+}_{\min}} \times \frac{Y_{\max}}{Y^{+}_{\min}} = M$. In contrast, suppose DMUt tries to evaluate with itself in MGBA LP-1 even though GBA LP-1 for DMUt is feasible. It is easy to see that $\theta_t'^* = \lambda_t^* = 1$ in this case. However, $\lambda_t^* = 1$ has a penalty of M in the objective function so that the overall objective function value becomes M + 1. So the solution with $\lambda_t^* = 0$ is cheaper. **Q.E.D.**

It is a simple task to arrive at the value of M from the data and after computing it, we can use MGBA LP-1 to carry out the efficiency analysis for all the DMUs using GBA as described in 4.3.3[19]. Whenever DMUt evaluates with

[19] Note that the value of $M = \frac{X_{\max}}{X^{+}_{\min}} \times \frac{Y_{\max}}{Y^{+}_{\min}}$ needs to be computed only once from the data at the beginning of the analysis.

itself because there is no other feasible solution, the objective function value will be greater than 1 indicating that the unit lies outside the current partial PPS. In such a case, as with the usual GBA procedure, we would invoke the FindNewGen procedure to identify a new generator among the units in U. Hence, the GBA procedure is unaffected except for the LP being solved for DMUt.

Note that one can obtain a tighter bound M' on M if instead of identifying the smallest and largest input and output values across the whole dataset, one looks at the smallest and largest values within each input and output factor[20]. However, it is obvious that $M \geq M'$ and calculating M is easier than M′ as it circumvents searching for the maximum 'ratio' value in each input and output factor. Hence, we retain the penalty for including DMUt in the coefficient matrix as $M = \frac{X_{\max}}{X^{+}_{\min}} \times \frac{Y_{\max}}{Y^{+}_{\min}}$ even though it may not be the tight most bound.

6.1.2.2 Small-m method

A valid *data independent* penalty for λ_t can be any positive value, say, 0.1. However, at this penalty, we cannot guarantee that DMUt will evaluate with itself iff the corresponding GBA LP-1 is infeasible. We can only guarantee that, even at this small penalty, DMUt will *not* evaluate with itself if it lies inside the partial PPS. For the purpose of GBA this is enough because all that is needed at steps 1.3 and 1.4 is a way to decide when to invoke procedure FindNewGen which is required whenever DMUt is outside the partial PPS.

The reason that this penalty is valid for our use is that when DMUt evaluates with itself at a small penalty of say 0.1, its overall objective function value is 1.1 > 1 indicating that the unit lies outside the current partial PPS. In such a case, as with the usual GBA procedure, we would apply the FindNewGen procedure to identify a new generator among the units in set U. Hence, the GBA procedure is unaffected except for the LP solved for DMUt.

[20] A tighter penalty to be computed is given by,

$$M' = \overset{m_1}{\underset{i=1}{Max}} \left\{ \frac{X_{ij}}{X_{ik}}, X_{ik} > 0; j = 1,\dots,n; k = 1,\dots,n; j \neq k \right\} \times \overset{m_2}{\underset{s=1}{Max}} \left\{ \frac{Y_{sc}}{Y_{sd}}, Y_{sd} > 0; c = 1,\dots,n; d = 1,\dots,n; c \neq d \right\}$$

6.2 Infeasibility in the input-oriented VRS model

6.2.1 Big-M method

As seen in the previous chapter in section 5.3, regardless of whether the data is strictly positive or not, the input-oriented VRS model (GBA LP-5) can become infeasible. We can handle the infeasibility issue by employing the Big-M method developed in the previous section. Theorem 6.2 helps us to quantify the value of M for the input-oriented VRS model which turns out be the same as M that was quantified for the CRS counterpart. Before showing these results, we present below the modified version of GBA LP-5, the Big-M incorporated GBA super-efficiency model, to be used to solve the input-oriented VRS model.

$$\begin{aligned}
&\textit{Minimise } \theta_t' + \sum_{j\in GEN}\lambda_j 0 \quad + \lambda_t M \\
&\textit{subject to,} \\
&\theta_t' X_t - \sum_{j\in GEN}\lambda_j X_j - \lambda_t X_t \geq 0 \\
&\ 0 \ + \sum_{j\in GEN}\lambda_j Y_j + \lambda_t Y_t \geq Y_t \qquad\qquad \textbf{(MGBA LP-3)} \\
&0 \ + \sum_{j\in GEN}\lambda_j + \lambda_t = 1 \\
&\theta_t' \ \textit{free};\ \lambda_j, \lambda_t \geq 0; j \in GEN
\end{aligned}$$

The only difference between MGBA LP-3 and MGBA LP-1 for the CRS counterpart is the additional convexity constraint present in the VRS model. Compared to GBA LP-5, MGBA LP-3 has one additional column.

Theorem 6.2: In MGBA LP-3, the penalty value of $M = \frac{X_{\max}}{X^+_{\min}} \times \frac{Y_{\max}}{Y^+_{\min}}$ ensures that DMUt evaluates with itself only if the corresponding GBA LP-5 is infeasible.

Proof: Without loss of generality, choose a DMUg (actual or obtained through a convex combination of some generators) in set GEN as a comparator unit for DMUt. Let the input and output vector of DMUt and DMUg be represented by

(X_t, Y_t) and (X_g, Y_g) respectively. Assume that GBA LP-5 for DMUt is feasible; a necessary condition for this is $Y_g \geq Y_t$ as indicated by the output constraints. Now using simple algebra, we can see from the set of input constraints that an upper-limit on θ_t' is given by $\theta_t' \leq w$, where, $w = \underset{r=1}{\overset{m_1}{Max}}\left\{\frac{X_{rg}}{X_{rt}}\right\}$ [21]. Given that the sense of the objective criterion is minimisation, it is easy to see that the maximum θ'^* value that was quantified for the input-oriented CRS model, viz., $\underset{j=1}{\overset{n}{Max}}\{\theta_j'^*\} = \frac{X_{\max}}{X_{\min}^+} \times \frac{Y_{\max}}{Y_{\min}^+}$, will also hold for its VRS counterpart as $w \leq \frac{X_{\max}}{X_{\min}^+} \times \frac{Y_{\max}}{Y_{\min}^+}$.

It is now straight-forward to see that the proof for theorem 6.1 can be easily adapted to prove theorem 6.2. That is, at the penalty value of $M = \frac{X_{\max}}{X_{\min}^+} \times \frac{Y_{\max}}{Y_{\min}^+}$, DMUt will evaluate with itself only if the corresponding GBA LP-5 is infeasible. **Q.E.D.**

So to avoid infeasibility in applying GBA in the input-oriented VRS case, calculate the penalty M from the data by finding $\frac{X_{\max}}{X_{\min}^+} \times \frac{Y_{\max}}{Y_{\min}^+}$ and use MGBA LP-3 to solve the model using GBA. The GBA procedure described in chapter 5 for solving VRS models essentially remains the same except for the LP being solved for DMUt.

6.2.2 Small-m method

Here also, the penalty can take any positive value, say, 0.1 and everything works out exactly as it did for the CRS case. Thus, at this penalty, we cannot guarantee that DMUt will evaluate with itself iff the corresponding GBA LP-5 is

[21] As we assume that GBA LP-5 is feasible, if $X_{rt} = 0$, then $X_{rg} = 0$. Hence, while computing w, we ignore $\frac{0}{0}$. The upshot is that w is determinate.

infeasible. We can only guarantee that, DMUt will *not* evaluate with itself if it lies inside the partial PPS. Unlike the Big-M method, DMUt may compare with itself if it lies outside the partial PPS even if the corresponding GBA LP-5 is feasible. The conclusion is that even at this data independent penalty value for λ_t, the GBA procedure is unaffected except for the LP solved for DMUt.

6.3 Infeasibility in the output-oriented VRS model

As seen in section 5.4.1, regardless of whether the data is strictly positive or otherwise, the output-oriented VRS super-efficiency model, GBA LP-7, can become infeasible. To handle infeasibility in this case we can employ the Big-M technique but a penalty value has to be calculated in this context. Banker and Chang (2006) article introduced a modified super-efficiency model with penalty for DMUt calculated in the context of identifying outliers in the data. They used an output-oriented VRS model for this purpose with a penalty of '–2' for DMUt. Although this penalty is also valid for our purposes, in this section we will try to develop an improved penalty meaning a better lower bound and if possible the *infimum*, for DMUt in the context of GBA which ensures that DMUt compares with itself iff the corresponding GBA LP-7 is infeasible. For arriving at this value, we present below the modified version of GBA LP-7.

$$\begin{aligned}
&\textit{Maximise}\ \ \eta_t' + \sum_{j\in GEN} \rho_j 0 + \rho_t M \\
&\textit{subject to,} \\
&\ \ 0 + \sum_{j\in GEN} \rho_j X_j + \rho_t X_t \le X_t \\
&\eta_t' Y_t - \sum_{j\in GEN} \rho_j Y_j - \rho_t Y_t \le 0 \qquad\qquad \textbf{(MGBA LP-4)} \\
&\ \ 0 \ \ + \sum_{j\in GEN} \rho_j + \rho_t = 1 \\
&\eta_t'\ \textit{free};\ \ \rho_j, \rho_t \ge 0;\ j \in GEN
\end{aligned}$$

We start with the simplest case wherein the data is strictly positive. The following theorem uses lemma 5.3.

Theorem 6.3: When the data is *strictly positive*, the penalty at which any DMUt can be brought into the coefficient matrix of the set of generators such that it evaluates with itself iff the corresponding GBA LP-7 is infeasible is M = -1.

Proof: From lemma 5.3, we know that when GBA LP-7 for DMUt is feasible, $\eta_t'^* > 0$. So there is a solution to MGBA LP-4 with $\rho_t = 0$. Since the coefficients of ρ_j in the objective function are 0, the overall objective function value in MGBA LP-4 in this case is $\eta_t'^* > 0$. In contrast, when DMUt compares with itself upon solving MGBA LP-4, we can see that $\eta_t'^* = 1$ and $\rho_t^* = 1$ and assuming a penalty of $M = -1$ for ρ_t, the overall objective function value becomes 0. Since the sense of the objective function is maximisation, it is clear that the former option is better. So $\rho_t^* = 0$ unless GBA LP-7 is infeasible for DMUt. **Q.E.D.**

In fact, M = -1 is the best lower bound possible when the data is strictly positive to ensure that DMUt evaluates with itself iff the corresponding GBA LP-7 is infeasible. Any value of the penalty in the range $0 < M < -1$, *although valid in the context of evaluating units using GBA*, is problematic. This is because when DMUt lies outside the partial PPS, if a feasible solution exists, $\eta_t'^* < 1$.

The next task is to compute a value for M when the data has some zero values in them. Fortunately, M = -1 will also hold in this case but with a rider. The only awkward case is when the objective function value is 0 and there are multiple optimal solutions to GBA LP-7 for DMUt. This case is illustrated using the example in table 6-3 below.

DMU	***X1***	***Y1***	***Y2***
A	10	2	24
B	10	0	36
C	16	8	28

Table 6-3 : Penalty in case of the output-oriented VRS model

Using random weights (3.6, 1, 1) for input X1, outputs Y1 and Y2 respectively and $v_0 = 0$, we can see that DMU B is a generator. Suppose, we now evaluate DMU A against DMU B in set GEN. The resulting MGBA LP-4 for DMU A is shown below:

Maximise $\eta'_A + \rho_B 0 - \rho_A 1$
subject to,

$$0 + \rho_B 10 + \rho_A 10 \le 10$$
$$\eta'_A 2 - \rho_B 0 - \rho_A 2 \le 0$$
$$\eta'_A 24 - \rho_B 36 - \rho_A 24 \le 0$$
$$\rho_B + \rho_A = 1$$
$$\eta'_A, \rho_B, \rho_A \ge 0$$

Clearly, an optimal solution to the above LP is $\eta'^*_A = 1, \rho^*_B = 0, \rho^*_A = 1$ with overall objective function value 0. The optimal weights are $v'^* = 0$ and $u'^* = (0.5, 0)$ with $v^*_0 = 0$. Another possible optimal solution to the LP is $\eta'^*_A = 0, \rho^*_B = 1, \rho^*_A = 0$ with the same optimal weights. The objective function value in this case is also 0. At either solution, because of the strong duality theorem, $v'^* X_A + v^*_0 = 0$ and R_A is indeterminate. For both solutions, the reduced cost value of A, $RC_A = 1 > 0$ indicating that the unit lies outside the partial PPS.

This is a special case as there are multiple optimal primal solutions which allow a DMUt that is outside the current partial PPS to evaluate with itself at the penalty of -1 even when there is an alternative reference set among the generators in GEN to compare with. However, procedure FindNewGen is invoked appropriately as $RC_A > 0$.

It is pertinent to note that if we had introduced a penalty of any value less than -1 for DMUt, then in the above example, the only optimal solution to the LP would be $\eta'^*_A = 0, \rho^*_B = 1, \rho^*_A = 0$. Hence, it must be clear from the example that at the penalty of say '-2', a DMUt would evaluate with itself iff the corresponding GBA LP-7 is infeasible.

6.4 Infeasibility in the CRS and VRS additive models

As the sense of the objective function in AGBA LP-1 (presented in chapter 5) is maximisation and we are maximising the total slacks present in DMUt, the penalty (coefficient of λ_t in the objective function) can be any negative value. We assign a small negative penalty of say -1 to λ_t so that when

DMUt evaluates with itself, its reduced cost value is +1 indicating that it lies outside the partial PPS and FindNewGen is invoked.

The aforesaid discussion on penalty for the GBA additive CRS model holds for the VRS model as well; i.e., the penalty for DMUt can be any negative value and we assign a value of -1 for a clear indication of the location of DMUt w.r.t the partial PPS.

6.5 Conclusion

In this chapter we presented ways to handle the principal technical challenge of LP infeasibility in all the DEA models except the output-oriented CRS model which is never infeasible. In chapter 7, we will discuss the remaining two technical challenges, namely, tied ratios and indeterminate ratios. We will also present a novel closed-form solution approach to obtain a positive set of multiplier values for the generators in the dataset.

7 CLOSED-FORM SOLUTIONS TO RESOLVE TIES AND CONSTRUCT NON-ZERO MULTIPLIER VALUES

In the previous chapter we saw how to deal with the principal technical challenge of infeasible LPs when applying GBA. In this chapter, we will examine the issues of ties and indeterminate ratios.

We saw in chapter 4 and 5 that ties and indeterminate ratios are problematic as in both cases one cannot identify a new generator immediately upon applying the FindNewGen procedure. In this chapter we present novel closed-form solutions to resolve both ties and indeterminate ratios in the case of oriented CRS and VRS models[22]. Closely connected to these, we also present closed-form solutions to obtain positive multiplier values for the generators under CRS and VRS assumptions.

7.1 Indeterminate ratios

7.1.1 Dula's ratio

One way to avoid indeterminate ratios in the CRS case is to use the ratio test discussed in Dula (1998). To identify generators at the optimal weights (v^*, u^*), Dula (1998) introduced the ratio, $R_j^D = \dfrac{u^*Y_j - v^*X_j}{eX_j}, \forall j \in U^+$; where $j \in U^+$ if $RC_j = u^* Y_j - v^* X_j > 0$. Since $\underset{j \in N/U}{Max}\{RC_j\} = 0$, it follows that if $j \in U^+$ then $j \in U$. So, the set $U^+ \subseteq U$ contains those status unresolved units that lie strictly outside the current partial PPS. The vector e is a vector of 1's of dimension m_1 and eX_j denotes the sum of inputs of DMUj. As $eX_j > 0, \forall j$, evaluating R_j^D cannot lead to indeterminate ratio for any DMUj. Dula (1998) has shown that using R_j^D guarantees the identification of a new generator. In

[22] Ties or indeterminate ratios do not pose a problem in the case of CRS and VRS additive models as the weights in the additive models are constrained to be non-zero.

particular, Dula (1998) proved that if $\underset{j \in U^{+}}{ArgMax}\{R_j^D\}$=DMUf and is unique, then DMUf is a new generator[23].

Dula's ratio R_j^D can be naturally extended to the case of the input and output oriented VRS models. Assuming that the optimal weights at a particular iteration are (v^*, u^*, u_0^*) and (v^*, u^*, v_0^*) for the input and output oriented VRS models respectively, the original and Dula's ratios in each case are presented in table 7.1 below.

Models	**Original Ratio**	**Dula's Ratio**
Input-Oriented VRS	$R_j = \frac{u^* Y_j + u_0^*}{v^* X_j}, \forall j \in U$	$R_j^D = \frac{u^* Y_j + u_0^* - v^* X_j}{e X_j}, \forall j \in U^+$
Output-Oriented VRS	$R_j = \frac{u^* Y_j}{v^* X_j + v_0^*}, \forall j \in U$	$R_j^D = \frac{u^* Y_j - (v^* X_j + v_0^*)}{e X_j}, \forall j \in U^+$

Table 7-1 : Original and Dula's ratio for the oriented VRS models

As in the CRS case, there are two ratios, viz., R_j and R_j^D, that can be evaluated at an optimal solution to the corresponding VRS GBA LP, and one can potentially[24] identify two different generators with one LP solution although only one is guaranteed.

In GBA, we do not employ Dula's ratio R_j^D for the following reason. While overcoming the problem of indeterminate ratios, R_j^D does not resolve the issue of tied ratios for which Dula proposes solving a second LP. We will show

[23] As shown in Dula (1998), the ratio $R_j^D = \frac{u^* Y_j - v^* X_j}{e X_j}$ can be reduced to $R_j'^D = \frac{u^* Y_j}{v' X_j}$; where, $v' = v^* + R_{\max}^D$ and $R_{\max}^D = Max\{R_j^D\}, j \in U^+$. Hence, the optimal weights for the generator DMUf employing his ratio is (v', u^*) and not (v^*, u^*).

[24] If $\underset{j \in U^{+}}{ArgMax}\{R_j^D\} = f$ and $\underset{j \in U}{ArgMax}\{R_j\} = f'$ where $f \neq f'$, then both f and f' are generators assuming that the ArgMax is unique in both cases.

in the next section that an indeterminate ratio value of R_j does not pose a problem in itself.

7.1.2 Indeterminate ratios and their association with tied ratios

In section 4.5 we showed why $ArgMax\{RC_j\}$ cannot be used to identify a generator for the CRS models. However, RC_j values can always be used to decide when to invoke FindNewGen. This is because one of the dual LP constraints is the reduced cost constraint given by $RC_j = uY_j - vX_j + \beta \leq 0, \forall j \in GEN$, where, β (dual value of the convexity constraint) is unconstrained in sign under VRS and $\beta = 0$ under CRS[25]. We showed in chapter 4 and 5 that the efficiency analysis of DMUt is complete if the dual LP constraints are satisfied for all the n units in the dataset, i.e., if the condition $RC_j = u^*Y_j - v^*X_j + \beta^* \leq 0, \forall j$ at (v^*, u^*, β^*) is satisfied. If the above condition is not satisfied, then the status of DMUt remains unresolved and we apply procedure FindNewGen to identify a new generator.

Suppose, $\underset{j \in U}{Max}\{RC_j\} > 0$ and $RC_a > 0$ for some unit $a \in U$. Then $a \in U^+$, i.e., unit a lies outside the partial PPS. So, the reduced cost values RC_j also help us to determine the position of the units in U w.r.t. the partial PPS.

Based on this we connect the problematic case of indeterminate ratios to ties in the value of $\underset{j \in U}{ArgMax}\{R_j\}$. Note that if for a DMUa, its ratio value is indeterminate, then $R_a = \infty$; this is the best possible value for R_j.

Let us first consider the CRS case[26]. We use the RC_j values of the units in U to ascertain if the efficiency analysis of DMUt is complete. If it is not, i.e., if $RC_a > 0$ for some $a \in U$, the ratios, $R_j = \frac{u^*Y_j}{v^*X_j}, j \in U$, are evaluated. If

[25] To avoid clutter, we simply let β to represent the dual value of the convexity constraint in the VRS models irrespective of the orientation.

[26] We know that in the CRS case the ratio R_j remains the same regardless of the orientation.

$\underset{j \in U}{ArgMax}\{R_j\} = DMUf$ is unique, regardless of whether $Max\{R_j\} = R_f$ is finite or otherwise, DMUf is a generator. Else, when $\underset{j \in U}{ArgMax}\{R_j\}$ is not unique, regardless of whether $\underset{j \in U}{Max}\{R_j\}$ is finite or otherwise, we are required to break this tie in $\underset{j \in U}{ArgMax}\{R_j\}$ to identify a generator.

As to the VRS models, we saw in section 5.3.1 that RC_j can be used to decide when to invoke FindNewGen and $\underset{j \in U}{ArgMax}\{RC_j\}$ can be used to identify a new generator. So the only issue here is how to break ties in $\underset{j \in U}{ArgMax}\{RC_j\}$.

In conclusion, indeterminate ratios in themselves do not pose a problem and can be dealt with in the same fashion as dealing with tied ratios. However, even when we use RC_j and do not have the problem of indeterminate ratios, ties have to be resolved to identify a new generator. We will see in the next section how to resolve tied ratio or tied reduced cost values using a novel closed-form solution approach.

7.2 Ways to resolve ties

From the previous section, we know that the issue of tied ratio or tied reduced cost values can only occur among DMUs that are strictly outside the current partial PPS. The task is to identify one generator among the tied units. Tie breaking plays a crucial role in solving DEA models using faster algorithms like GBA and BuildHull. Although, Dula (1998) in his extensive experimental study on solving DEA models remarked that ties do not occur commonly while using BuildHull, it was found in our experiments using GBA and BuildHull that ties did occur both in real and simulated datasets[27].

[27] In chapter 8, where we present the computational results, we will see that ties involving several hundred units occurred while solving a 'real' dataset using BuildHull and GBA.

7.2.1 Dula's approach to resolve ties

Dula (1998) proposed a simple approach to identify one generator among the tied units. This approach is to consider the tied units as a separate smaller DEA problem and apply BuildHull or any other algorithm for solving the smaller DEA problem to resolve the status of the tied units. The generators among the tied units are then appended to set GEN while DMUt (if not resolved) is evaluated again in the next iteration. Although the approach is entirely valid in identifying a new generator among the tied units, it has two drawbacks. First, we would need to solve at least one additional LP to identify a generator among the tied units. Secondly, the weights obtained by solving the additional LP to identify a generator among the tied units may not be globally valid to prove its status as a generator. This is because the new weights that prove that one among the tied units is a generator cannot guarantee the same when all the units in the dataset are considered. In other words, the new weights have a local validity but may not possess a global validity. The second drawback is illustrated in figure 7-1, the data for which are provided in table 7-2 below.

DMU	*X1*	*Y1*	*Y2*
A	1	10	8
B	1	7.5	11.5
C	1	7	11.5
D	1	6	11.5
E	1	5.5	11.5
F	1	5	11.5
G	1	3	11.5

Table 7-2 : Data for Dula's tie-breaking approach

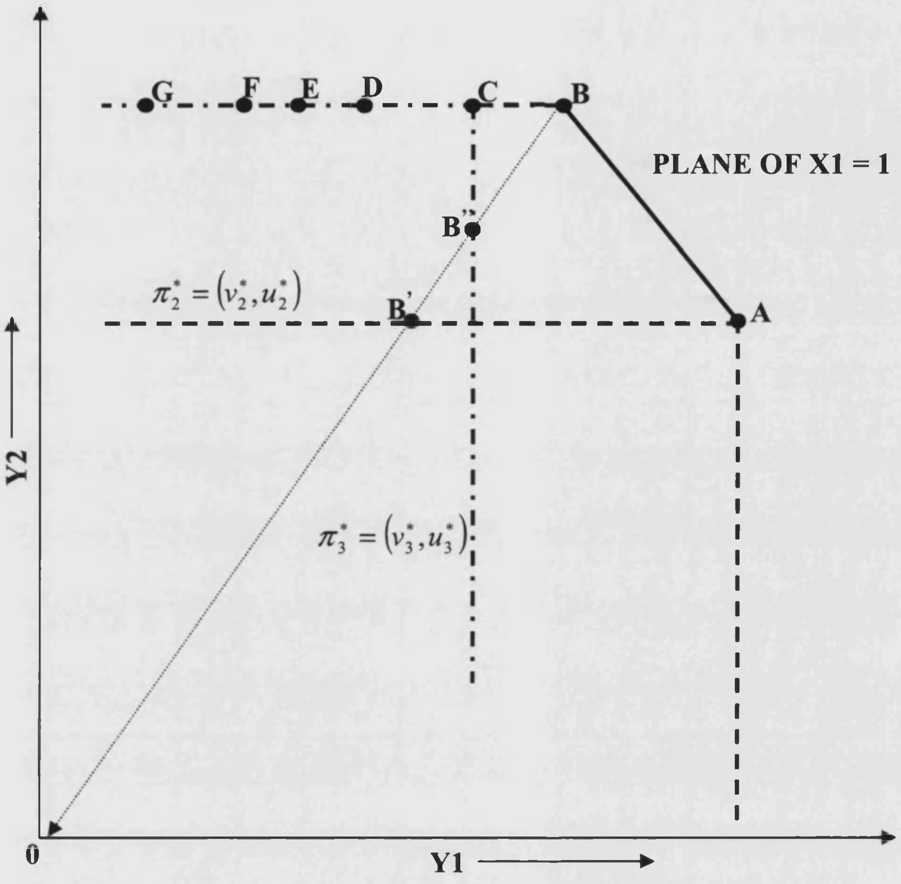


Figure 7-1 : Graphical illustration of Dula's tie-breaking approach

Consider the above DEA problem with 2 outputs (Y1 and Y2) and a single standardised input (X1). There are seven efficient units in total and two generators overall, namely, A and B. Suppose at a particular iteration, DMUt = B and GEN = {A}. The current partial PPS is the region to the south-west of A and the boundary of the PPS is described by dashed lines to the west and south of A. Upon evaluating B against A, the radial projection of B falls on the boundary line at B' to the west of A defined by the set of optimal weights $\pi_2^* = (v_2^*, u_2^*) = (0.695, 0, 0.0869)$. Dula's ratio R_j^D would modify the set of optimal weights to $\pi_2' = (v_2', u_2^*) = (1, 0, 0.0869)$ and at π_2', $Max\{R_j'^D\} = 1$ and $\underset{j \in N}{ArgMax}\{R_j'^D\} = \{B, C, D, E, F, G\}$. Let the set of tied units be denoted by $TU = \{B, C, D, E, F, G\}$. Following Dula's approach, we will consider the units in TU as a separate DEA problem. If we apply GBA to this DEA problem to identify a generator, we need to choose one unit to be GEN and another to be DMUt. Suppose we arbitrarily choose DMUt to be B and GEN to be C[28]. The partial PPS is the region to the south-west of C and the boundary of the PPS is described by dashed-dot lines to the west and south of C. Upon evaluating B against C, the radial projection of B falls on the boundary line to the south of C at B'' defined by the set of optimal weights $\pi_3^* = (v_3^*, u_3^*) = (0.933, 0.133, 0)$. Dula's ratio R_j^D would once again modify these set of weights to $\pi_3' = (v_3', u_3^*) = (1, 0.133, 0)$. At π_3', $\underset{j \in TU}{Max}\{R_j'^D\} = 1$ and $\underset{j \in TU}{ArgMax}\{R_j'^D\} = \{B\}$. Hence, applied locally to the units in TU, π_3' is a valid set of weights to prove that unit B is a generator[29]. But when we apply the weights π_3' to all the n units in the dataset, $\underset{j \in N}{Max}\{R_j'^D\} = 1.33$ and $\underset{j \in N}{ArgMax}\{R_j'^D\} = \{A\}$. Hence, π_3' does not have a global validity to prove that unit B is a generator. In other words, weights π_3', although valid for the 6 DMU problem, are not valid dual values for the original 7 DMU problem.

7.2.2 Closed-form solutions to resolve ties

Before we present the closed-form solutions, it is important to recognize its main limitation. By using the closed-form solutions, we cannot guarantee to *only* identify generators from the tied units; rather, the approach can only guarantee to identify Pareto-Koopmans (P-K) efficient units among the tied units. We know that both extreme-efficient and efficient but not extreme units are P-K efficient. In other words, the closed-form solution may not guarantee that the set of the generators is the minimal subset as it could include Pareto-Koopmans efficient units that are not generators. Although the GBA originally introduced in chapter 4 evaluates units only against the set of extreme-efficient units (generators), it is understood that as long as the units are evaluated only against P-K efficient units, we will obtain the correct efficiency scores, slacks and set of peers for all the units. P-K efficient units other than generators are

[28] Note that some other arbitrary choice of DMUt and GEN could lead to another tie upon evaluating the LP. For example, choice of DMUt={C} and GEN={B} leads to another tie.
[29] This also dictates that DMU B must be a generator for the original 7 DMU problem.

superfluous but their inclusion in GEN, though unnecessary, is un-harmful. Moreover, it is well-known (see, Thrall, 1996b; Cooper et al, 2007) that efficient but not extreme units rarely occur in real datasets. This was also the case in our experience with solving several hundred real and simulated datasets and in earlier works in Barr and Durchholz (1997) and Dula (1998). Because of the (almost) total absence of efficient but not extreme units in a dataset, using the closed-form solutions, we will expect to identify a generator among the tied units. This was indeed the case in our extensive experiments.

In this section we propose closed-form solutions that not only identify P-K efficient units among the tied units, thereby circumventing the need to solve additional LPs, but also such that the new set of weights are strictly positive and globally valid. First, we will consider resolving ties for the VRS case and subsequently the CRS case. The reason for this order is that the closed-form solution for the CRS case builds upon the solution for the VRS case. Orientation does not affect the logic of the procedure and the closed-form solutions are valid for the additive models as well. However, tied ratio or tied reduced cost values do not pose a problem when applying GBA (or BuildHull) to solve CRS or VRS additive models as all the units tied at the maximum value are guaranteed to be P-K efficient. This is a consequence of the fact that the weights from the additive model are constrained to be non-zero.

We note in passing that a naïve option upon encountering a tie is to move DMUt to U and choose another DMU to be evaluated. However, this is obviously a waste of computational time, and importantly does not guarantee that one would not encounter a tie again while solving for any other DMU in U including unit t when it is selected again. By employing our closed-form solutions, we are able to guarantee that among the tied units we will identify one P-K efficient unit (in practice, generators) and that the new set of weights are strictly positive and globally valid. Because of its conclusive nature, we employ the closed-form solutions to resolve ties in GBA.

7.2.2.1 Closed-form solution to resolve ties under VRS

Suppose we had arrived at the optimal set of weights $\pi^* = (v^*, u^*, \beta^*)$ where, β^* is the dual value of the convexity constraint. Let π_f^* represent the weights of the input and output factors alone. So $\pi_f^* = (v^*, u^*)$ and $\pi^* = (\pi_f^*, \beta^*)$. If $\pi_f^* > 0$, we know that every tied unit is P-K efficient; we can move all the tied units to set GEN and proceed to evaluate DMUt again (if not resolved) against the augmented set GEN. The problem of ties arises only when π_f^* is not strictly positive.

The closed-form solution computes a set of weights $\bar{\pi}$ with $\bar{\pi}_f > 0$ from π^* that finds a P-K efficient unit among the tied units. The weights $\bar{\pi}$ are constructed by adding a carefully scaled version π^S of π^* to unary weights (all input and output weights are equal to 1 with the dual value of the convexity constraint β^1 fixed at 0) π^1. So $\bar{\pi}_f = \pi_f^S + \pi_f^1$ which is obviously positive. More importantly the scaling used in constructing π^S is such that using $\bar{\pi}$ to find a generator breaks the tie. To elaborate, let $TU = \{t_1, t_2, ..., t_k\}$ be the set of k tied units at the original optimal weights π^*, i.e., $\underset{j \in N}{ArgMax}\{RC_j\}_{\pi^*} = \{t_1, t_2, ..., t_k\}$[30]. Then weights $\bar{\pi}$ with $\bar{\pi}_f > 0$ ensure that a unit p satisfies $\underset{j \in N}{ArgMax}\{RC_j\}_{\bar{\pi}} = DMUp$ only if $p \in TU$. The details are as follows.

Let the total number of units be n and let set $N = \{1, ..., n\}$. As always, GEN is the set of currently known generators and U the set of status unidentified units. Upon solving the relevant GBA LP (GBA LP-5 or GBA LP-7), suppose we had arrived at the optimal set of weights $\pi^* = (v^*, u^*, \beta^*)$. Assume that the efficiency analysis of DMUt is not complete as $U^+ \neq \{\phi\}$. In this case, we apply the FindNewGen procedure using the reduced costs of the units given by $RC_j = u^* Y_j - v^* X_j + \beta^*$ to identify a new generator.

[30] $\underset{j \in N}{ArgMax}\{RC_j\}_{\pi^*}$ denotes that the ArgMax of the $RC_j, j \in N$ values is computed at π^*.

Let $\underset{j\in N}{Max}\{RC_j\}_{\pi^*} = \alpha_1$. Since, $\underset{j\in N/U}{Max}\{RC_j\}_{\pi^*} = 0$, $\underset{j\in U}{Max}\{RC_j\}_{\pi^*} = \alpha_1$. Also let $\underset{j\in N}{ArgMax}\{RC_j\}_{\pi^*} = \{t_1, t_2, ..., t_k\}$; i.e., $k \geq 2$ units are tied for the maximum reduced cost value α_1. Let the set of tied units be denoted by $TU = \{t_1, t_2, ..., t_k\}$. The optimal weights are such that $\pi_f^* \geq 0$ but $\pi_f^* \not> 0$; i.e., some of the factors' weights are 0 and we need to identify one P-K efficient unit among the k tied units.

Let, $\underset{j\in N/TU}{Max}\{RC_j\}_{\pi^*} = \alpha_2$. In other words, α_2 is the second highest reduced cost value at π^*. By definition, $0 \leq \alpha_2 < \alpha_1$. Also, let $\alpha_3 = (\alpha_1 - \alpha_2) > 0$.

Now, consider the unary set of weights $\pi^1 = (1,1,0)$. π^1 is composed of input and output weight vectors of 1s of appropriate dimensions with the dual value of the convexity constraint β^1 fixed at 0. Let $\underset{j\in N}{Max}\{RC_j\}_{\pi^1} = \omega_1$, $\underset{j\in N}{Min}\{RC_j\}_{\pi^1} = \omega_2$ and $\omega_3 = (\omega_1 - \omega_2)$. Assume for now that $\omega_1 \neq \omega_2$ and hence $\omega_3 > 0$. Let, $\gamma_1 = \dfrac{\omega_3}{\alpha_3}$ and note that by definition $\gamma_1 > 0$. Finally let $\gamma_2 = \gamma_1 + \varepsilon, \varepsilon > 0$[31].

Consider the scaled set of weights given by $\pi^S = (\gamma_2 \times \pi^*)$. Since we are simply multiplying the original LP weights π^* by γ_2, $\underset{j\in N}{Max}\{RC_j\}_{\pi^S} = (\gamma_2 \times \alpha_1)$ and $\underset{j\in N}{ArgMax}\{RC_j\}_{\pi^S} = \{t_1, t_2, ..., t_k\}$. We scale the original LP weights π^* by γ_2 to counterbalance the reduced cost values at the unary weights π^1 upon their amalgamation. Consider now the synthesised set of weights $\bar{\pi} = \pi^S + \pi^1$. Note that $\bar{\pi}_f > 0$ because $\pi_f^S \geq 0$ and $\pi_f^1 > 0$. Also, as $\bar{\pi}_f > 0$, the units achieving the global maximum of the reduced cost value at $\bar{\pi}_f$ must be P-K efficient.

We prove in lemma 7.1 below that the global maximum of the reduced cost values at $\bar{\pi}$ will be achieved only by one or more of the units in set TU.

[31] We assume a value of 1 for ε in our illustrations and computational experiments.

Lemma 7.1: Assuming $\omega_1 \neq \omega_2$, at $\bar{\pi}$, $\underset{j \in N}{ArgMax}\{RC_j\}_{\bar{\pi}} = \underset{j \in TU}{ArgMax}\{RC_j\}_{\bar{\pi}}$.

In words, lemma 7.1 states that the global maximum of the reduced cost value at $\bar{\pi}$ will only be achieved by one or more units in set TU, provided $\omega_1 \neq \omega_2$.

Proof: First consider the effect of choosing the scaling factor γ_1 on π^* instead of γ_2, i.e., define $\pi'^S = (\gamma_1 \times \pi^*)$ and $\bar{\pi}' = \pi'^S + \pi^1$. Now, given that $\bar{\pi}' = \pi'^S + \pi^1$, $\{RC_j\}_{\bar{\pi}'} = \{RC_j\}_{\pi'^S} + \{RC_j\}_{\pi^1}$.

It follows that for $j \in TU$, $\{RC_j\}_{\pi'^S} = (\gamma_1 \times \alpha_1)$ and we know that $\underset{j \in N}{Min}\{RC_j\}_{\pi^1} = \omega_2$. Therefore, over the tied units, $\underset{j \in TU}{Min}\{RC_j\}_{\bar{\pi}'} \geq (\gamma_1 \times \alpha_1) + \omega_2$.

On the other hand, among the remaining units, $\underset{j \in N/TU}{Max}\{RC_j\}_{\pi'^S} = (\gamma_1 \times \alpha_2)$ and we know that $\underset{j \in N}{Max}\{RC_j\}_{\pi^1} = \omega_1$. Therefore, $\underset{j \in N/TU}{Max}\{RC_j\}_{\bar{\pi}'} \leq (\gamma_1 \times \alpha_2) + \omega_1$. So, $\underset{j \in TU}{Min}\{RC_j\}_{\bar{\pi}'} - \underset{j \in N/TU}{Max}\{RC_j\}_{\bar{\pi}'} \geq [(\gamma_1 \times \alpha_1) + \omega_2] - [(\gamma_1 \times \alpha_2) + \omega_1]$. Rearranging the terms, $\underset{j \in TU}{Min}\{RC_j\}_{\bar{\pi}'} - \underset{j \in N/TU}{Max}\{RC_j\}_{\bar{\pi}'} \geq [\gamma_1 \times (\alpha_1 - \alpha_2)] - (\omega_1 - \omega_2)$.

Recall that $\gamma_1 = \dfrac{\omega_3}{\alpha_3} = \dfrac{(\omega_1 - \omega_2)}{(\alpha_1 - \alpha_2)}$. So, $\gamma_1 \times (\alpha_1 - \alpha_2) - (\omega_1 - \omega_2) = 0$. Hence, at a scaling factor of γ_1, $\underset{j \in TU}{Min}\{RC_j\}_{\bar{\pi}'} \geq \underset{j \in N/TU}{Max}\{RC_j\}_{\bar{\pi}'}$. It follows that if $\pi^S = \gamma_2 \times \pi^* = (\gamma_1 + \varepsilon) \times \pi^*$ and $\bar{\pi} = (\gamma_2 \times \pi^*) + \pi^1$, $\underset{j \in TU}{Min}\{RC_j\}_{\bar{\pi}} > \underset{j \in N/TU}{Max}\{RC_j\}_{\bar{\pi}}$. This dictates that, provided $\omega_1 \neq \omega_2$, the global maximum of the reduced cost values at $\bar{\pi}$ will only be achieved by the units in set TU, i.e., $\underset{j \in N}{ArgMax}\{RC_j\}_{\bar{\pi}} = \underset{j \in TU}{ArgMax}\{RC_j\}_{\bar{\pi}}$. **Q.E.D.**

Now consider the highly restrictive possibility of $\omega_1 = \omega_2$, i.e., all the n units achieve the same reduced value at $\pi^1 = (1,1,0)$. So $\omega_1 - \omega_2 = 0$ and hence $\underset{j \in TU}{Min}\{RC_j\}_{\bar{\pi}} - \underset{j \in N/TU}{Max}\{RC_j\}_{\bar{\pi}} = (g \times (\alpha_1 - \alpha_2)) - (\omega_1 - \omega_2) = g \times (\alpha_1 - \alpha_2) > 0$;
$\underset{j \in TU}{Min}\{RC_j\}_{\bar{\pi}} - \underset{j \in N/TU}{Max}\{RC_j\}_{\bar{\pi}} = (g \times (\alpha_1 - \alpha_2)) - (\omega_1 - \omega_2) = g \times (\alpha_1 - \alpha_2) > 0$;eights

π^*. It is easy to see that any positive value of g is enough to ensure that $\underset{j \in N}{ArgMax}\{RC_j\}_{\bar{\pi}} = \underset{j \in TU}{ArgMax}\{RC_j\}_{\bar{\pi}}$. The simplest way to implement this would be to let $g = 1$ leading to $\bar{\pi} = \pi^* + \pi^1$.

7.2.2.1.1 Illustration of the closed-form solution to break ties under VRS

Let us consider the following 10 DMU, 2 inputs and 2 outputs DEA problem presented in table 7-3 below to illustrate the above closed-form solution.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j
A	1	0	3	4	12
B	1	9	40	7	12
C	6	0	60	2	6
D	6	9	10	5	6
E	0	40	22	1	-37
F	0	15	8	1	-12
G	1	18	5	5	-3
H	4	0	8	4	12
I	4	30	150	10	0
J	5	16	7	2	-10
π_f^*	*0*	*1*	*0*	*3*	

Table 7-3: Illustration of closed-form solution to break ties 1

Assume that GEN = {I}, DMUt = {A} and upon solving the relevant GBA LP, we arrived at the optimal weights defined by $\pi_f^* = (0, 1, 0, 3)$ with β^*=0[32]; at π^*, $\underset{j \in N}{Max}\{RC_j\}_{\pi^*} = 12 > 0$ and $\underset{j \in N}{ArgMax}\{RC_j\}_{\pi^*} = \{A, B, H\}$. Unit I is in the optimal basis and consequently has $RC_I = 0$; units E, F, G and J are strictly inside the current partial PPS indicated by their non-positive reduced costs. Using the notations introduced earlier, $\alpha_1 = 12$, $\alpha_2 = 6$ and $\alpha_3 = (\alpha_1 - \alpha_2) = 12 - 6 = 6$.

[32] Clearly the closed-form solution is unaffected if $\beta^* \neq 0$ as it is a constant and its value (positive or negative) affects the RC_j value of all the units uniformly. Here, for illustrative purposes we have let $\beta^* = 0$.

Now consider table 7-4 below where for the same dataset, the RC_j values of the units are listed at the unary weights $\pi_f^1 = (1, 1, 1, 1)$ with $\beta^1 = 0$.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j
A	1	0	3	4	6
B	1	9	40	7	37
C	6	0	60	2	56
D	6	9	10	5	0
E	0	40	22	1	-17
F	0	15	8	1	-6
G	1	18	5	5	-9
H	4	0	8	4	8
I	4	30	150	10	126
J	5	16	7	2	-12
π_f^1	*1*	*1*	*1*	*1*	

Table 7-4: Illustration of closed-form solution to break ties 2

Using the notions introduced earlier, $\omega_1 = 126$, $\omega_2 = -17$, $\omega_3 = (\omega_1 - \omega_2) = 126 - (-17) = 143$; also, $\gamma_1 = \frac{\omega_3}{\alpha_3} = \frac{143}{6} = 23.83$ and $\gamma_2 = \gamma_1 + 1 = 24.83$.

Now, the synthesised set of weights are given by $\bar{\pi} = \pi^S + \pi^1$ where $\pi^S = \gamma_2 \times \pi^*$. Hence, $\bar{\pi} = (1,\ 24.83,\ 1,\ 74.5)$ with $\beta^* = 0$. At $\bar{\pi}$, the reduced costs are listed in table 7-5 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j
A	1	0	3	4	300
B	1	9	40	7	337
C	6	0	60	2	203
D	6	9	10	5	153
E	0	40	22	1	-896.83
F	0	15	8	1	-290
G	1	18	5	5	-70.49
H	4	0	8	4	302
I	4	30	150	10	146.00
J	5	16	7	2	-246.33
$\bar{\pi}_f$	*1*	*24.83*	*1*	*74.5*	

Table 7-5: Illustration of closed-form solution to break ties 3

We know that the RC_j values at $\bar{\pi}$ are the sum total of the reduced cost values at π^S and π^1. The RC_j values at π^* are magnified by the factor $\gamma_2 = 24.83$ such that it counterbalances the effect of the reduced cost values at the unary weights π^1. The only purpose of the unary weights π^1 is to ensure that the final set of weights is strictly positive. By magnifying the reduced cost values at π^* by the factor γ_2 we ensure that at the synthesised set of weights $\bar{\pi}$, the global maximum of the reduced cost value is achieved only by units in set $TU = \{A, B, H\}$. Here, unit B in set TU achieves the unique maximum reduced cost value at $\bar{\pi}$.

7.2.2.2 Closed-form solution to resolve ties under CRS

Once again ties in the CRS case are an issue only if the optimal LP weights π^* is not strictly positive. In contrast to the VRS case, the closed-form solution approach in the CRS case amalgamates a 'twice-scaled version' of the original LP weights π^{SS} to a 'scaled version' of the unary weights π^{1S}. The additive operation ensures that the synthesised final weights $\bar{\pi}$ are strictly positive.

The closed-form approach under CRS can be broken down into 4 steps. In the first step, the original LP weights π^* are scaled to π^S such that only the tied units are strictly outside the partial PPS. In the second step, the unary weights π^1 are also scaled to π^{1S} so that the reduced cost values of all the n units are non-positive. In the third step, the modified LP weights π^S are scaled again to π^{SS} so as to ensure that upon adding them to the scaled unary weights π^{1S}, only the originally tied units have a strictly positive reduced cost value. In the final step, the weights π^{SS} and π^{1S} are amalgamated. Following the above four steps will ensure that at the final positive weights $\bar{\pi}$, only the tied units have ratio values > 1 and hence the global maximum of the ratio value can only be achieved by the originally tied units. In this section we will derive the analytic form of the closed-form solution. In the subsequent section, we will illustrate its employment using two examples, viz., when the maximum ratio value is finite and secondly, when it is infinite (indeterminate).

Assume that upon solving the relevant GBA LP (GBA LP-1 or GBA LP-3), we had arrived at the optimal set of weights $\pi^* = (v^*, u^*)$. Let the efficiency analysis of DMUt be incomplete, i.e., $U^+ \neq \{\phi\}$. In this case, we apply the FindNewGen procedure using the ratio values, $R_j = \frac{u^* Y_j}{v^* X_j}$, to identify a new generator among the units in U. Let $\underset{j \in N}{Max}\{R_j\}_{\pi^*} = r_1 > 1$. Note that $\underset{j \in N}{Max}\{R_j\}_{\pi^*}$ could be an indeterminate ratio value, i.e., $r_1 = \infty$, and the following closed-form solution approach will work in such a case as well.

Since, $\underset{j \in N/U}{Max}\{R_j\}_{\pi^*} = 1$, $\underset{j \in U}{Max}\{R_j\}_{\pi^*} = r_1$. Let $\underset{j \in N}{ArgMax}\{R_j\}_{\pi^*} = \{t_1, t_2, \ldots, t_k\}$; i.e., $k \geq 2$ units are tied for the maximum ratio value r_1. Once again, let the set of tied units be denoted by $TU = \{t_1, t_2, \ldots, t_k\}$. The weights are such that $\pi^* \geq 0$ but $\pi^* \not> 0$.

Step 1: Scaling π^* to π^S such that only the tied units are strictly outside the partial PPS.

Let, $\underset{j \in N/TU}{Max}\{R_j\}_{\pi^*} = r_2$; i.e., r_2 gives the second highest R_j value among the units in N. By definition, $1 \leq r_2 < r_1$. Consider the modified set of weights $\pi^S = (r_2 \times v^*, u^*)$. As we have simply scaled (multiplied) the input weights by r_2, the ratio values also get scaled down by the same r_2 value; i.e., at π^S, $\underset{j \in N/TU}{Max}\{R_j\}_{\pi^S} = 1$, $\{R_j\}_{\pi^S} = \frac{r_1}{r_2} = r_3 > 1, j \in TU$. Also, $\underset{j \in N}{ArgMax}\{R_j\}_{\pi^S} = \{t_1, t_2, \ldots, t_k\}$, $\underset{j \in N/TU}{Max}\{RC_j\}_{\pi^S} = 0$ and $\underset{j \in TU}{Min}\{RC_j\}_{\pi^S} > 0$, where, RC_j, the reduced costs of the units at π^*, are given by $u^* Y_j - v^* X_j$. Let, $\underset{j \in TU}{Min}\{RC_j\}_{\pi^S} = \alpha_1 > 0$.

Step 2: Scaling unary weights π^1 to π^{1S} so that $RC_j \leq 0, \forall j$.

Now, consider the unary set of weights $\pi^1 = (1,1)$. Let $\underset{j\in N}{Max}\{R_j\}_{\pi^1} = r_4$. Consider the modified set of unary weights $\pi^{1S} = (r_4 \times 1,\ 1)$. As we have simply scaled (multiplied) the input weights of π^1 by r_4, the ratio values also get scaled by the same r_4 factor; i.e., at π^{1S}, $\underset{j\in N}{Max}\{R_j\}_{\pi^{1S}} = 1$ and $\underset{j\in N}{Max}\{RC_j\}_{\pi^{1S}} = 0$. Let, $\underset{j\in N}{Min}\{RC_j\}_{\pi^{1S}} = \alpha_2$ and assume for now that $\alpha_2 \neq 0$; hence, $\alpha_2 < 0$. Let, $\gamma_1 = \left|\frac{\alpha_2}{\alpha_1}\right|$ and note that by definition $\gamma_1 > 0$. Finally, let $\gamma_2 = \gamma_1 + \varepsilon, \varepsilon > 0$.

<u>Steps 3 & 4:</u> Scaling π^S by γ_2 to obtain π^{SS} and amalgamating π^{SS} with π^{1S} to synthesise $\bar{\pi}$.

Consider the twice-scaled set of weights given by $\pi^{SS} = (\gamma_2 \times \pi^S)$. As we are scaling both the input and output weights at π^S by γ_2 the R_j values remain the same at π^{SS}, i.e., $\underset{j\in N}{Max}\{R_j\}_{\pi^{SS}} = \underset{j\in TU}{Max}\{R_j\}_{\pi^{SS}} = r_3$ and so $\underset{j\in N}{ArgMax}\{R_j\}_{\pi^{SS}} = \{t_1, t_2, \ldots, t_k\}$. However, the RC_j values get scaled by γ_2 and so $\underset{j\in TU}{Min}\{RC_j\}_{\pi^{SS}} = (\gamma_2 \times \alpha_1) > 0$ and $\underset{j\in N/TU}{Max}\{RC_j\}_{\pi^{SS}} = \gamma_2 \times 0 = 0$.

Consider now the synthesised set of weights $\bar{\pi}$ obtained using the additive operation $\bar{\pi} = \pi^{SS} + \pi^{1S}$. Note that $\bar{\pi} > 0$ as $\pi^{SS} \geq 0$ and $\pi^{1S} > 0$.

We prove in lemma 7.2 below that, assuming $\alpha_2 \neq 0$, at $\bar{\pi}$ the global maximum of the ratio values will only be achieved by the units in set TU.

Lemma 7.2: Assuming $\alpha_2 \neq 0$, at $\bar{\pi}$, $\underset{j\in N/TU}{Max}\{R_j\}_{\bar{\pi}} < \underset{j\in TU}{Min}\{R_j\}_{\bar{\pi}}$.

In other words, lemma 7.2 states that the maximum of the ratio values at $\bar{\pi}$ will only be achieved by the units in set TU, provided that $\alpha_2 \neq 0$.

Proof: First consider the effect of choosing the scaling factor of γ_1 on π^S instead of γ_2, i.e., define $\pi'^{SS} = (\gamma_1 \times \pi^S)$ and $\bar{\pi}' = \pi'^{SS} + \pi^{1S}$. Now, given that $\bar{\pi}' = \pi'^{SS} + \pi^{1S}$, $\{RC_j\}_{\bar{\pi}'} = \{RC_j\}_{\pi'^{SS}} + \{RC_j\}_{\pi^{1S}}$.

It follows that among the tied units, $\underset{j\in TU}{Min}\{RC_j\}_{\pi'^{SS}} = (\gamma_1 \times \alpha_1)$ and we know that $\underset{j\in N}{Min}\{RC_j\}_{\pi^{1S}} = \alpha_2$. Therefore, $\underset{j\in TU}{Min}\{RC_j\}_{\overline{\pi}'} \geq (\gamma_1 \times \alpha_1) + \alpha_2$.

On the other hand, among the remaining units, $\underset{j\in N/TU}{Max}\{RC_j\}_{\pi'^{SS}} = \gamma_1 \times 0 = 0$ and we know that $\underset{j\in N}{Max}\{RC_j\}_{\pi^{1S}} = 0$. Therefore, $\underset{j\in N/TU}{Max}\{RC_j\}_{\overline{\pi}'} \leq 0$.

So, $\underset{j\in TU}{Min}\{RC_j\}_{\overline{\pi}'} \geq \gamma_1 \times \alpha_1 + \alpha_2 = \left|\frac{\alpha_2}{\alpha_1}\right| \times \alpha_1 + \alpha_2 \geq 0$ as $\alpha_2 < 0$.

Consequently, $\underset{j\in TU}{Min}\{R_j\}_{\overline{\pi}'} \geq 1$. Hence, at a scaling factor of γ_1, $\underset{j\in TU}{Min}\{RC_j\}_{\overline{\pi}'} \geq 0$ and $\underset{j\in TU}{Min}\{R_j\}_{\overline{\pi}'} \geq 1$. It follows that if $\pi^{SS} = \gamma_2 \times \pi^S = (\gamma_1 + \varepsilon) \times \pi^S$ and $\overline{\pi} = (\gamma_2 \times \pi^S) + \pi^{1S}$, $\underset{j\in TU}{Min}\{RC_j\}_{\overline{\pi}} > 0$ and $\underset{j\in TU}{Min}\{R_j\}_{\overline{\pi}} > 1$. Given that $\underset{j\in N/TU}{Max}\{RC_j\}_{\overline{\pi}'} \leq 0$ and similarly, $\underset{j\in N/TU}{Max}\{RC_j\}_{\overline{\pi}} \leq 0$ and consequently, $\underset{j\in N/TU}{Max}\{R_j\}_{\overline{\pi}} \leq 1$, $\underset{j\in TU}{Min}\{R_j\}_{\overline{\pi}} > \underset{j\in N/TU}{Max}\{R_j\}_{\overline{\pi}}$. Hence, provided $\alpha_2 \neq 0$, the global maximum of the ratio values at $\overline{\pi}$ will only be achieved by the units in set TU as all the tied units have a ratio value > 1 at $\overline{\pi}$. **Q.E.D.**

Now consider the highly restrictive possibility of $\alpha_2 = 0$, i.e., all the n units have the same ratio value at π^1. So $\underset{j\in TU}{Min}\{RC_j\}_{\overline{\pi}} = (g \times \alpha_1) > 0$; where, $g > 0$ is the scaling factor on π^{SS}. It is easy to see that any positive value of g is enough to ensure that $\underset{j\in N/TU}{Max}\{R_j\}_{\overline{\pi}} < \underset{j\in TU}{Min}\{R_j\}_{\overline{\pi}}$. The simplest way to implement this would be to let $g = 1$ leading to $\overline{\pi} = \pi^{SS} + \pi^{1S}$.

7.2.2.2.1 Illustration of the closed-form solution to break ties under CRS

We will consider two examples to illustrate the above closed-form solution approach, viz., one in which the data is strictly positive and the maximum ratio is finite and another in which the data has some zeroes and the maximum ratio value is indeterminate.

Consider first the following 10 DMUs, 2 inputs and 2 outputs positive data example shown in table 7-6 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	3.5	6	7
B	5	3	40	10.5	18	7
C	6	2	60	2	2	2
D	6	9	10	5	1	1.11
E	2	40	22	1	-38	0.05
F	3	15	8	1	-13	0.13
G	1	3.5	5	12.25	21	7
H	4	3	8	4	5	2.67
I	4	20	150	10	0	1
J	5	16	7	2	-12	0.25
π^*	*0*	*1*	*0*	*2*		

Table 7-6: Illustration of closed-form solution to break ties 1

Assume that $GEN = \{I\}$, DMUt = A and upon solving the relevant GBA LP, we had arrived at the optimal weights $\pi^* = (0, 1, 0, 2)$. At π^*, $\underset{j\in N}{Max}\{R_j\}_{\pi^*} = 7 > 1$ and $\underset{j\in N}{ArgMax}\{R_j\}_{\pi^*} = \{A, B, G\}$. Using the notations introduced earlier, $r_2 = 2.67$ and the modified optimal weights are $\pi^S = (0, 2.67, 0, 2)$. The reduced cost and ratio values at π^S can be seen in table 7-7 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	3.5	4.33	2.625
B	5	3	40	10.5	12.99	2.625
C	6	2	60	2	-1.33	0.75
D	6	9	10	5	-14	0.416
E	2	40	22	1	-104.66	0.018
F	3	15	8	1	-38.00	0.05
G	1	3.5	5	12.25	15.16	2.625
H	4	3	8	4	0	1
I	4	20	150	10	-33.34	0.375
J	5	16	7	2	-38.67	0.0937
π^S	*0*	*2.67*	*0*	*2*		

Table 7-7: Illustration of closed-form solution to break ties 2

As discussed earlier, at π^S, only the tied units in *TU* have reduced cost values > 0 and consequently, ratio values > 1; now, $\underset{j\in TU}{Min}\{RC_j\}_{\pi^S} = \alpha_1 = 4.33 > 0$.

Consider the following table where for the same dataset, the ratio and reduced cost values are shown at the unary weights $\pi^1 = (1,1,1,1)$.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	3.5	4.5	3.25
B	5	3	40	10.5	42.5	6.31
C	6	2	60	2	54	7.75
D	6	9	10	5	0	1
E	2	40	22	1	-19	0.54
F	3	15	8	1	-9	0.5
G	1	3.5	5	12.25	12.75	3.83
H	4	3	8	4	5	1.71
I	4	20	150	10	136	6.67
J	5	16	7	2	-12	0.428
π^1	*1*	*1*	*1*	*1*		

Table 7-8: Illustration of closed-form solution to break ties 3

By scaling the input weights by the maximum ratio value of 7.75, at the modified unary weights $\pi^{1S} = (7.75, 7.75, 1, 1)$, we get the following reduced cost and ratio values.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	3.5	-9	0.41
B	5	3	40	10.5	-11.5	0.81
C	6	2	60	2	0	1
D	6	9	10	5	-101.25	0.129
E	2	40	22	1	-302.5	0.07
F	3	15	8	1	-130.5	0.064
G	1	3.5	5	12.25	-17.625	0.49
H	4	3	8	4	-42.25	0.22
I	4	20	150	10	-26	0.86
J	5	16	7	2	-153.75	0.055
π^{1S}	*7.75*	*7.75*	*1*	*1*		

Table 7-9: Illustration of closed-form solution to break ties 4

At π^{1S}, $\underset{j \in N}{Min}\{RC_j\}_{\pi^{1S}} = \alpha_2 = -302.5$, $\gamma_1 = \left|\frac{\alpha_2}{\alpha_1}\right| = \left|\frac{-302.5}{4.33}\right| = 69.807$ and $\gamma_2 = \gamma_1 + 1 = 70.807$. Now the synthesised set of weights is given by $\bar{\pi} = \pi^{SS} + \pi^{1S} = \gamma_2 \times \pi^S + \pi^{1S} = (7.75,\ 196.57,\ 1,\ 142.61)$. At $\bar{\pi}$, the reduced cost and ratio values are shown in table 7-10 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	3.5	297.83	2.457
B	5	3	40	10.5	909.0	2.446
C	6	2	60	2	-94.41	0.785
D	6	9	10	5	-1092.56	0.398
E	2	40	22	1	-7713.71	0.020
F	3	15	8	1	-2821.19	0.050
G	1	3.5	5	12.25	1056.29	2.518
H	4	3	8	4	-42.25	0.931
I	4	20	150	10	-2386.26	0.397
J	5	16	7	2	-2891.65	0.0917
$\bar{\pi}$	*7.75*	*196.57*	*1*	*142.61*		

Table 7-10: Illustration of closed-form solution to break ties 5

The reduced costs of the n units at $\bar{\pi}$ are the sum total of the reduced cost values at π^{SS} and π^{1S}. The reduced cost values at the modified optimal weights π^{S} at which some of the units in U, viz., $\{A,B,G\}$, were tied for the maximum ratio value are magnified by the factor γ_2 such that it counterbalances the effect of the RC_j and the corresponding R_j values at π^{1S}. Hence, by magnifying the RC_j values at π^{S} by γ_2 we ensure that at $\bar{\pi}$, the RC_j values are strictly positive only for the units in $TU = \{A,B,G\}$. This would dictate that $\{R_j\}_{\bar{\pi}} > 1$ only if $j \in TU$. The maximum of the ratio values at $\bar{\pi}$ is achieved by units among the k tied units and since the weights are strictly positive, they must be P-K efficient. In the above example, unit G, which was one of the originally tied units, achieves the unique maximum ratio value at $\bar{\pi}$.

Consider now the following example with 10 DMUs, 2 inputs and 2 outputs shown in table 7-11 below. The data has some zero values in them and at a particular iteration, the maximum ratio value is indeterminate (infinite) and tied for some units in U.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	0	3	3.5	7.00	∞
B	5	0	40	10.5	21.00	∞
C	6	2	60	2	2.00	2.00
D	6	9	10	5	1.00	1.11
E	2	40	22	1	-38.00	0.05
F	3	15	8	1	-13.00	0.13
G	1	0	5	12.25	24.50	∞
H	4	3	8	4	5.00	2.67
I	4	20	150	10	0.00	1.00
J	5	16	7	2	-12.00	0.25
π^*	*0*	*1*	*0*	*2*		

Table 7-11: Illustration of closed-form solution to break ties 1

Assume that $GEN = \{I\}$, DMUt = A and upon solving the relevant GBA LP, we arrived at the optimal weights $\pi^* = (0, 1, 0, 2)$. At π^*, $\underset{j\in N}{Max}\{R_j\}_{\pi^*} = \infty > 1$ and $\underset{j\in N}{ArgMax}\{R_j\}_{\pi^*} = \{A, B, G\}$. Unit I is in the optimal basis and has a ratio value of 1 and units E, F and J are strictly inside the partial PPS with ratio values < 1. Using the notations introduced earlier, $r_2 = 2.67$ and the modified optimal weights are $\pi^S = (0, 2.67, 0, 2)$. The reduced cost and ratio values at π^S can be seen in table 7-12 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	0	3	3.5	7	∞
B	5	0	40	10.5	21	∞
C	6	2	60	2	-1.33	0.75
D	6	9	10	5	-14	0.416
E	2	40	22	1	-104.67	0.018
F	3	15	8	1	-38	0.05
G	1	0	5	12.25	24.5	∞
H	4	3	8	4	0	1
I	4	20	150	10	-33.34	0.375
J	5	16	7	2	-38.67	0.093
π^S	*0*	*2.67*	*0*	*2*		

Table 7-12: Illustration of closed-form solution to break ties 2

At π^S only the tied units in TU have strictly positive reduced cost values and hence ratio value = ∞ > 1; $\underset{j\in TU}{Min}\{RC_j\}_{\pi^S} = \alpha_1 = 7$.

Now consider the following table where for the same dataset, the ratio and reduced cost values are listed at the unary weights $\pi^1 = (1,1,1,1)$.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	0	3	3.5	5.5	6.5
B	5	0	40	10.5	45.5	10.1
C	6	2	60	2	54	7.75
D	6	9	10	5	0	1
E	2	40	22	1	-19	0.54
F	3	15	8	1	-9	0.5
G	1	0	5	12.25	16.25	17.25
H	4	3	8	4	5	1.71
I	4	20	150	10	136	6.66
J	5	16	7	2	-12	0.42
π^1	1	1	1	1		

Table 7-13: Illustration of closed-form solution to break ties 3

By scaling the input weights by the maximum ratio value of 17.25, at the modified unary weights $\pi^{1S} = (17.25, 17.25, 1, 1)$, we get the following reduced cost and ratio values shown in table 7-14 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	0	3	3.5	-10.75	0.37
B	5	3	40	10.5	-35.75	0.585
C	6	2	60	2	-76	0.449
D	6	9	10	5	-243.75	0.0579
E	2	40	22	1	-701.5	0.0317
F	3	15	8	1	-301.5	0.0289
G	1	3.5	5	12.25	0	1
H	4	3	8	4	-108.75	0.099
I	4	20	150	10	-254	0.386
J	5	16	7	2	-353.25	0.0248
π^{1S}	*17.25*	*17.25*	*1*	*1*		

Table 7-14: Illustration of closed-form solution to break ties 4

At $\pi^{1S}, \underset{j \in N}{Min}\{RC_j\}_{\pi^{1S}} = \alpha_2 = -701.5$, $\gamma_1 = \left|\frac{\alpha_2}{\alpha_1}\right| = \left|\frac{-701.5}{7}\right| = 100.21$ and $\gamma_2 = \gamma_1 + 1 = 101.21$. Now the synthesised set of weights is given by $\bar{\pi} = \pi^{SS} + \pi^{1S} = \gamma_2 \times \pi^S + \pi^{1S} = (17.25,\ 287.14,\ 1,\ 203.42)$. At $\bar{\pi}$, the reduced cost and ratio values are shown in the table below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	0	3	3.5	697.72	41.44
B	5	3	40	10.5	2089.66	25.22
C	6	2	60	2	-210.957	0.688
D	6	9	10	5	-1660.69	0.382
E	2	40	22	1	-11294.8	0.0195
F	3	15	8	1	-4147.49	0.048
G	1	3.5	5	12.25	2479.65	144.74
H	4	3	8	4	-108.75	0.883
I	4	20	150	10	-3627.68	0.375
J	5	16	7	2	-4266.72	0.088
$\bar{\pi}$	*17.25*	*287.14*	*1*	*203.42*		

Table 7-15: Illustration of closed-form solution to break ties 5

The reduced cost values at the modified optimal weights π^S at which some of the units in U, $\{A,B,G\}$, are tied for the maximum indeterminate ratio value are magnified by the factor γ_2 such that it counterbalances the effect of the reduced cost values at π^{1S}. Hence, by magnifying the reduced cost values at π^S by γ_2 we ensure that at $\bar{\pi}$, the reduced cost values are strictly positive only for $TU = \{A,B,G\}$ and hence $\{R_j\}_{\bar{\pi}} > 1$ only for $j \in TU$. Also, by amalgamating the 'twice-scaled' LP weights π^{SS} with π^{1S}, we ensure that $\bar{\pi}$ is strictly positive and hence the global maximum ratio value at $\bar{\pi}$ will be finite. In the above example, unit G, one of the originally tied units at an indeterminate ratio, achieves the unique maximum *finite* ratio value at $\bar{\pi}$.

7.3 Strictly Positive multiplier values for generators

7.3.1 Literature review

An important topic in the DEA literature that is closely connected with the closed-form solutions developed in the previous section is obtaining strictly positive multiplier values for the generators without explicit weight restrictions. A recent article by Cooper et al (2007) proposed a Mixed Integer Linear Program (MILP) based approach under CRS assumption for obtaining strictly positive multiplier values for a generator that currently has some zero weights. The approach necessitates solving two MILPs for a generator to obtain positive multiplier values for it. In the first step, they solve a MILP for a generator wherein they try to identify a set of multiplier values that has the maximal support in the sense of the number of generators (including the generator under consideration) sitting on the corresponding hyperplane. In the second step, they solve another MILP, wherein they try to identify a set of multiplier values that are not only strictly positive, but has the maximal support achieved in the first step. See Cooper et al (2007) for a review of early works on this important topic, a detailed discussion of their approach, and application of their method to a real world example considered in Dyson and Thanassoulis (1988). See also Argyris (2008) for a discussion on why the approach presented in Cooper et al (2007) could fail in a special case to arrive at strictly positive multiplier values although Argyris' discussion was presented for the VRS case while Cooper et al (2007) approach was only presented for the CRS case[33]. To show that Argyris' claim is also valid for the CRS case, we present an example in table 7-16 below in which the method developed in Cooper et al (2007) fails.

Consider the following 5 DMU, 2 inputs, 2 outputs dataset presented below.

DMU	*X1*	*X2*	*Y1*	*Y2*
A	2	2	2	4
B	2	2.5	2	4.8
C	2	3.5	2	5.2
D	4	3	4	6
E	4	1	3	5

Table 7-16 : Example in which Cooper et. al.'s (2007) approach fails

All the 5 units are extreme-efficient and the extreme rays of the PPS described by the five DMUs can be seen in table 7-17 below.

Extreme Rays	*v*1	*v*2	*u*1	*u*2	Extreme-efficient units supported
w1	77	92	0	80	B, E
w2	19	4	0	10	B, C
w3	11	8	9	5	A, B, E
w4	13	0	0	5	C
w5	0	5	0	1	E
w6	0	3	1	0	E
w7	5	4	8	0	D, E
w8	3	2	3	1	A, D, E
w9	1	0	1	0	A, B, C, D

Table 7-17 : Extreme rays for the 5 DMU, 4 factor example

[33] Incidentally, Argyris' (2008) discussion on fully-dimensional facets and efficient facets needs a careful inspection as he seems to have missed the fact that under VRS assumption, there can be efficient facets supporting fewer than m generators.

Now among the 9 extreme-rays, if we consider ray *w9*, units A, B, C and D are supported at the corresponding hyperplane. For these extreme-efficient units, ray *w9* provides the facet with maximal support of 4 extreme-efficient units. However, this is a weak-efficient facet[34] as it has $v2 = 0$ and $u2 = 0$ and there are no other extreme-rays that has a support of 4 extreme-efficient units but with strictly positive multiplier values. The only efficient facets for the 5 extreme-efficient units are given by rays *w3* and *w8* with maximal support of 3 $(= m-1)$ units. In other words, only if to their first MILP, they add a constraint restricting the maximal support to be $\leq m-1$, the above issue could be avoided[35].

A current working paper by Bougnol et al (2010) also looks at obtaining positive set of multiplier values for extreme-efficient units using Interior Point Methods (IPM) under VRS assumption. Their method necessitates transforming the original dataset and solving an LP on the transformed data for each extreme-efficient unit using IPM. A desirable feature of their method is that, at least in theory, the multiplier values arrived at using the IPM to solve their LP are not only strictly positive but also the solution lies in the analytic centre of the optimal face. In other words, apart from being positive, their multiplier values satisfy a well-defined optimization criterion by lying in the analytic centre of optimal face. This optimization criterion, according to them, is desirable as it imparts balance and uniformity of weights and slack values (see, Gonzalez-Lima et al, 1996). Although this work is still at a nascent stage, we can identify 2 limitations with it. The first limitation is that they need to transform the original dataset and solve an additional LP for each extreme-efficient unit using IPM to obtain positive multiplier values for it. The second limitation is that, as they indicate, different implementations of IPM to solve their LP arrived at different solutions; although the solutions from the different implementations are all strictly positive, they do not necessarily lie in the analytic centre of the optimal face as desired.

[34] A facet is weak-efficient if some of the multiplier values it defines are 0. Else, it is strong-efficient or efficient.

[35] Professor Jose Dula in a private communication on 12/06/2010 acknowledged this – *"Srini, I have been thinking about the <=m-1 constraint you propose for the MILP in (4)...it is an interesting way to resolve the obvious problems with the current formulation by Cooper Ruiz and Sirvent (and Olesen and Petersen)...One needs to be tactful, however especially in deference to Cooper".*

Given that their work is still a working paper, we will overlook it for comparison purposes.

In this section, unlike Cooper et al (2007), whose method was presented only for the CRS case, we describe closed-form solutions to achieve positive multiplier values under CRS and VRS assumptions thereby circumventing the need to solve two additional MILPs. Our closed-form solution approach differs from that of Cooper et al (2007) in that the final multiplier values for a generator obtained using our approach has the least support in the data. In other words, the corresponding generator sits alone on the new hyperplane described by strictly positive weights. This situation is also encountered in the approach presented in Charnes et al (1991) which required solving multiple LPs for a generator[36] to obtain positive multiplier values for it.

7.3.2 A closed-form solution for achieving positive weights under CRS

The closed-form solution approach to break ties under CRS assumption can be applied with some modification to obtain positive weights for the generators. The procedure is essentially the same as in breaking ties in that it involves amalgamating a 'twice-scaled version' of original LP weights π^{SS} to a 'scaled version' of the unary weights π^{1S}. The *only* difference is that instead of more than one unit achieving the maximum R_j value at the original optimal weights π^*, the maximum ratio value is uniquely achieved by a single unit which is confirmed to be a generator[37]; however, $\pi^* \not> 0$ and we are interested in providing this generator with a set of strictly positive multiplier values. We will describe the closed-form solution approach below before presenting an example to illustrate its working.

Assume that upon solving the relevant GBA LP (GBA LP-1 or GBA LP-3), we had arrived at the optimal set of weights $\pi^* = (v^*, u^*)$. The efficiency analysis of DMUt is not resolved as $U^+ \neq \{\phi\}$. We apply the FindNewGen

[36] Their approach requires solving as many additional LPs for a generator as there are 0 valued input and output weights at the optimal solution.

[37] Note that a necessary condition for our closed-form solution to guarantee a positive set of multiplier values for any generator is that it achieves the *unique maximum* ratio value at π^*.

procedure using the ratio values $R_j = \dfrac{u^*Y_j}{v^*X_j}$ and realize that $\underset{j\in N}{Max}\{R_j\}_{\pi^*} = r_1 > 1$ and $\underset{j\in N}{ArgMax}\{R_j\}_{\pi^*} = \{DMUg\}$. The maximum of the ratio values could be achieved at an indeterminate ratio, i.e., $r_1 = \infty$, and the following closed-form solution will work in this case as well. Some of the weights in π^* are 0 and although DMUg is confirmed to be a generator, we are interested in providing it with non-zero multiplier values.

Let, $\underset{j\in N/g}{Max}\{R_j\}_{\pi^*} = r_2$. By definition, $1 \le r_2 < r_1$. Consider the modified set of weights $\pi^S = (r_2 \times v^*, u^*)$. At π^S, $\underset{j\in N/g}{Max}\{R_j\}_{\pi^S} = 1$, $\{R_g\}_{\pi^S} = \dfrac{r_1}{r_2} = r_3 > 1$, $\underset{j\in N}{ArgMax}\{R_j\}_{\pi^S} = \{DMUg\}$; also, $\underset{j\in N/g}{Max}\{RC_j\}_{\pi^S} = 0$ and $\{RC_g\}_{\pi^S} = \alpha_1 > 0$, where, RC_j are the reduced costs of the units.

Now, consider the unary set of weights $\pi^1 = (1,1)$. Let $\underset{j\in N}{Max}\{R_j\}_{\pi^1} = r_4$. Consider the modified set of unary weights $\pi^{1S} = (r_4 \times 1, 1)$. At π^{1S}, $\underset{j\in N}{Max}\{R_j\}_{\pi^{1S}} = 1$ and $\underset{j\in N}{Max}\{RC_j\}_{\pi^{1S}} = 0$. Let, $\underset{j\in N}{Min}\{RC_j\}_{\pi^{1S}} = \alpha_2$ and assume for now that $\alpha_2 \ne 0$ and hence, $\alpha_2 < 0$. Let, $\gamma_1 = \left|\dfrac{\alpha_2}{\alpha_1}\right|$ and $\gamma_2 = \gamma_1 + \varepsilon$, $\varepsilon > 0$. Consider, the scaled set of weights given by $\pi^{SS} = (\gamma_2 \times \pi^S)$. At π^{SS}, as we have scaled both the input and output weights by γ_2, $\underset{j\in N}{Max}\{R_j\}_{\pi^{SS}} = \{R_g\}_{\pi^{SS}} = r_3$; also, $\underset{j\in N/g}{Max}\{RC_j\}_{\pi^{SS}} = \gamma_2 \times 0 = 0$ and $\{RC_g\}_{\pi^{SS}} = (\gamma_2 \times \alpha_1) > 0$. Consider now the synthesised set of weights $\bar{\pi} = \pi^{SS} + \pi^{1S}$. Note that $\bar{\pi} > 0$ as $\pi^{1S} > 0$ and $\pi^{SS} \ge 0$.

We prove in lemma 7.3 that, provided $\alpha_2 \ne 0$, at the synthesised set of weights $\bar{\pi}$, the global maximum of the ratio values will only be achieved by DMUg.

Lemma 7.3: At $\bar{\pi}$, $\underset{j \in N}{ArgMax}\{R_j\}_{\bar{\pi}} = DMUg$, provided $\alpha_2 \neq 0$.

In words, lemma 7.3 states that the global maximum of ratio values at $\bar{\pi}$ will only be achieved by DMUg, assuming $\alpha_2 \neq 0$.

Proof: First consider the effect of choosing the scaling factor of γ_1 on π^S instead of γ_2, i.e., define $\pi'^{SS} = (\gamma_1 \times \pi^S)$ and $\bar{\pi}' = \pi'^{SS} + \pi^{1S}$. Now, given that $\bar{\pi}' = \pi'^{SS} + \pi^{1S}$, $\{RC_j\}_{\bar{\pi}'} = \{RC_j\}_{\pi'^{SS}} + \{RC_j\}_{\pi^{1S}}$.

It follows that for DMUg, $\{RC_g\}_{\pi'^{SS}} = (\gamma_1 \times \alpha_1)$ and we know that $\underset{j \in N}{Min}\{RC_j\}_{\pi^{1S}} = \alpha_2$. Therefore, $\{RC_g\}_{\bar{\pi}'} \geq (\gamma_1 \times \alpha_1) + \alpha_2$.

On the other hand, among the remaining units, $\underset{j \in N/g}{Max}\{RC_j\}_{\pi'^{SS}} = \gamma_1 \times 0 = 0$ and we know that $\underset{j \in N}{Max}\{RC_j\}_{\pi^{1S}} = 0$. Therefore, $\underset{j \in N/g}{Max}\{RC_j\}_{\bar{\pi}'} \leq 0$.

So, $\{RC_g\}_{\bar{\pi}'} \geq \gamma_1 \times \alpha_1 + \alpha_2 = \left|\frac{\alpha_2}{\alpha_1}\right| \times \alpha_1 + \alpha_2 \geq 0$ as $\alpha_2 < 0$. Consequently, $Min\{R_g\}_{\bar{\pi}'} \geq 1$. Hence, at a scaling factor of γ_1, $Min\{RC_g\}_{\bar{\pi}'} \geq 0$ and $Min\{R_g\}_{\bar{\pi}'} \geq 1$. It follows that if $\pi^{SS} = \gamma_2 \times \pi^S = (\gamma_1 + \varepsilon) \times \pi^S$ and $\bar{\pi} = (\gamma_2 \times \pi^S) + \pi^{1S}$, $\{RC_g\}_{\bar{\pi}} > 0$ and $\{R_g\}_{\bar{\pi}} > 1$. Given that $\underset{j \in N/g}{Max}\{RC_j\}_{\bar{\pi}'} \leq 0$ and similarly, $\underset{j \in N/g}{Max}\{RC_j\}_{\bar{\pi}} \leq 0$ and consequently, $\underset{j \in N/g}{Max}\{R_j\}_{\bar{\pi}} \leq 1$, $Min\{R_g\}_{\bar{\pi}} > \underset{j \in N/g}{Max}\{R_j\}_{\bar{\pi}}$. Hence, the global maximum of the ratio values at $\bar{\pi}$ will only be achieved unit g, i.e., $\underset{j \in N}{ArgMax}\{R_j\}_{\bar{\pi}} = DMUg$, provided $\alpha_2 \neq 0$.

Q.E.D.

Now consider the highly restrictive possibility of $\alpha_2 = 0$, i.e., all the n units have the same ratio value at π^1. So $Min\{RC_g\}_{\bar{\pi}} = (h \times \alpha_1) > 0$; where, $h > 0$ is the scaling factor on π^{SS}. It is easy to see that any positive value of h

is enough to ensure that $\underset{j \in N/g}{Max}\{R_j\}_{\bar{\pi}} < Min\{R_g\}_{\bar{\pi}}$. The simplest way to implement this would be to let $h = 1$ leading to $\bar{\pi} = \pi^{SS} + \pi^{1S}$.

7.3.2.1 Illustration of the closed-form solution to ensure positive weights under CRS

We will consider the following example in which the data is positive and the maximum ratio is at a finite value to illustrate the above closed-form solution.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	4	7	8
B	5	3	40	10.5	18	7
C	6	2	60	2	2	2
D	6	9	10	5	1	1.11
E	2	40	22	1	-38	0.05
F	3	15	8	1	-13	0.13
G	1	3.5	5	12.25	21	7
H	4	3	8	4	5	2.67
I	4	20	150	10	0	1
J	5	16	7	2	-12	0.25
π^*	*0*	*1*	*0*	*2*		

Table 7-18 : Closed-form solution to obtain positive multiplier values 1

Assume that $GEN = \{I\}$, DMUt = A and upon solving the relevant GBA LP, we arrived at the optimal weights $\pi^* = (0,1,0,2)$. At π^*, $\underset{j \in N}{Max}\{R_j\}_{\pi^*} > 1$ and $\underset{j \in N}{ArgMax}\{R_j\}_{\pi^*} = \{A\}$. Unit A is confirmed to be a generator but it has one 0 input weight and one 0 output weight. Suppose we are interested in providing it with strictly positive weights. Using the notations introduced earlier, $r_2 = 7$ and the modified optimal weights are $\pi^S = (0, 7, 0, 2)$. The reduced cost and ratio values at π^S can be seen in table 7-19 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	4	1	1.143
B	5	3	40	10.5	0	1
C	6	2	60	2	-10	0.286
D	6	9	10	5	-53	0.159
E	2	40	22	1	-278	0.007
F	3	15	8	1	-103	0.019
G	1	3.5	5	12.25	0	1
H	4	3	8	4	-13	0.381
I	4	20	150	10	-120	0.143
J	5	16	7	2	-108	0.036
π^S	*0*	*7*	*0*	*2*		

Table 7-19 : Closed-form solution to obtain positive multiplier values 2

As discussed earlier, at π^S only unit A has a reduced cost value > 0 and ratio value > 1; $RC_A = \alpha_1 = 1$.

Now, consider the following table where for the same dataset, the ratio and reduced cost values are shown at the unary weights $\pi^1 = (1,1,1,1)$.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	4	5	3.5
B	5	3	40	10.5	42.5	6.31
C	6	2	60	2	54	7.75
D	6	9	10	5	0	1
E	2	40	22	1	-19	0.54
F	3	15	8	1	-9	0.5
G	1	3.5	5	12.25	12.75	3.83
H	4	3	8	4	5	1.71
I	4	20	150	10	136	6.67
J	5	16	7	2	-12	0.428
π^1	*1*	*1*	*1*	*1*		

Table 7-20 : Closed-form solution to obtain positive multiplier values 3

By scaling the input weights by the maximum ratio value of 7.75, at the modified unary weights $\pi^{1S} = (7.75, 7.75, 1, 1)$, we get the following reduced cost and ratio values.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	4	-8.5	0.451
B	5	3	40	10.5	-11.5	0.81
C	6	2	60	2	0	1
D	6	9	10	5	-101.25	0.129
E	2	40	22	1	-302.5	0.07
F	3	15	8	1	-130.5	0.064
G	1	3.5	5	12.25	-17.625	0.49
H	4	3	8	4	-42.25	0.22
I	4	20	150	10	-26	0.86
J	5	16	7	2	-153.75	0.055
π^{1S}	*7.75*	*7.75*	*1*	*1*		

Table 7-21 : Closed-form solution to obtain positive multiplier values 4

At $\pi^{1S}, \underset{j\in N}{Min}\{RC_j\}_{\pi^{1S}} = \alpha_2 = -302.5$, $\gamma_1 = \left|\frac{\alpha_2}{\alpha_1}\right| = \left|\frac{-302.5}{1}\right| = 302.5$ and $\gamma_2 = \gamma_1 + 1 = 303.5$. Now the synthesised set of weights is given by $\bar{\pi} = \pi^{SS} + \pi^{1S} = \gamma_2 \times \pi^S + \pi^{1S} = (7.75,\ 2132.25, 1,\ 608)$. At $\bar{\pi}$, the reduced cost and ratio values are shown in table 7-22 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j	R_j
A	1	1	3	4	295	1.137
B	5	3	40	10.5	-11.5	0.998
C	6	2	60	2	-3035	0.295
D	6	9	10	5	-16186.75	0.158
E	2	40	22	1	-84675.5	0.0073
F	3	15	8	1	-31391	0.0192
G	1	3.5	5	12.25	-17.625	0.997
H	4	3	8	4	-3987.75	0.379
I	4	20	150	10	-36446	0.146
J	5	16	7	2	-32931.75	0.0358
$\bar{\pi}$	*7.75*	*2132.25*	*1*	*608*		

Table 7-22 : Closed-form solution to obtain positive multiplier values 5

The reduced costs of the n units at $\bar{\pi}$ are the sum total of the reduced costs at π^{SS} and π^{1S}. The reduced cost values at the modified optimal weights π^S at which unit A achieved the unique maximum ratio value is magnified by the factor γ_2 such that it counterbalances the effect of the reduced cost values and the corresponding ratio values of the remaining units at π^{1S}. Hence, by magnifying the reduced cost values at π^S by γ_2 we ensure that at $\bar{\pi}$, the reduced cost value is strictly positive only for unit A and hence $\{R_A\}_{\bar{\pi}} > 1$. From table 7-22, we can see that the maximum of the ratio values at $\bar{\pi}$ is once again achieved by unit A but now at a set of strictly positive weights.

7.3.3 Closed-form solution for achieving positive multiplier values under VRS

Once again, the closed-form solution approach to break ties under the VRS assumption can be applied with some modification to obtain positive weights for any generator. The procedure is essentially the same as in breaking ties in that it involves amalgamating a 'scaled version' of original LP weights π^S to the unary weights π^1. The only difference is that instead of more than one unit achieving the maximum reduced cost value at the original optimal weights π^*, the maximum reduced value is uniquely achieved by a single unit which is confirmed to be a generator[38]; however, $\pi_f^* \not> 0$ and we are interested in providing this generator with a set of strictly positive multiplier values.

We will describe the closed-form solution approach below before presenting an example to illustrate its working.

Upon solving the relevant GBA LP (GBA LP-5 or GBA LP-7), we had arrived at the optimal set of weights $\pi^* = (v^*, u^*, \beta^*)$. The efficiency analysis of DMUt is not resolved as $U^+ \neq \{\phi\}$. Let $\underset{j\in N}{Max}\{RC_j\}_{\pi^*} = \alpha_1 > 0$. Since, $\underset{j\in N/U}{Max}\{RC_j\}_{\pi^*} = 0$, $\underset{j\in U}{Max}\{RC_j\}_{\pi^*} = \alpha_1$. Let $\underset{j\in N}{ArgMax}\{RC_j\}_{\pi^*} = \{DMUg\}$. Although DMUg is confirmed to be a generator, $\pi_f^* \not> 0$; where, π_f^* represent the weights of the input and output factors alone.

Suppose we are interested in providing DMUg with strictly positive multiplier values for its input and output factors. Let, $\underset{j\in N/g}{Max}\{RC_j\}_{\pi^*} = \alpha_2$. By definition, $0 \leq \alpha_2 < \alpha_1$. Let $\alpha_3 = (\alpha_1 - \alpha_2) > 0$.

[38] Note that a necessary condition for our closed-form solution to guarantee a positive set of multiplier values for any generator is that it achieves the *unique maximum* reduced cost value at π^*.

Now, consider the unary set of weights $\pi^1 = (1,1,0)$. Let $\underset{j\in N}{Max}\{RC_j\}_{\pi^1} = \omega_1$, $\underset{j\in N}{Min}\{RC_j\}_{\pi^1} = \omega_2$ and $\omega_3 = (\omega_1 - \omega_2)$. Assume for now that $\omega_1 \neq \omega_2$ and hence $\omega_3 > 0$. Let, $\gamma_1 = \frac{\omega_3}{\alpha_3}$ and note that by definition $\gamma_1 > 0$; finally, let $\gamma_2 = \gamma_1 + \varepsilon, \varepsilon > 0$. Consider, the scaled set of weights given by $\pi^S = (\gamma_2 \times \pi^*)$. At π^S, $\underset{j\in N}{Max}\{RC_j\}_{\pi^S} = RC_g = (\gamma_2 \times \alpha_1)$ and $\underset{j\in N}{ArgMax}\{RC_j\}_{\pi^S} = \{DMUg\}$.

Consider now the synthesised set of weights $\overline{\pi} = \pi^S + \pi^1$. Note that $\overline{\pi}_f > 0$ as $\pi^1_f > 0$ and $\pi^S_f \geq 0$. We prove in lemma 7.4 that, assuming $\omega_1 \neq \omega_2$, at the synthesised set of weights $\overline{\pi}$, $\underset{j\in N}{ArgMax}\{RC_j\}_{\overline{\pi}} = DMUg$; i.e., the global maximum of the reduced cost values will only be achieved by DMUg.

Lemma 7.4: Assuming $\omega_1 \neq \omega_2$, at $\overline{\pi}$, $\underset{j\in N}{ArgMax}\{RC_j\}_{\overline{\pi}} = DMUg$.

In words, lemma 7.4 states that the global maximum of the reduced cost value at $\overline{\pi}$ will only be achieved by DMUg, provided $\omega_1 \neq \omega_2$.

Proof: First consider the effect of choosing the scaling factor γ_1 on π^* instead of γ_2, i.e., define $\pi'^S = (\gamma_1 \times \pi^*)$ and $\overline{\pi}' = \pi'^S + \pi^1$. Now, given that $\overline{\pi}' = \pi'^S + \pi^1$, $\{RC_j\}_{\overline{\pi}'} = \{RC_j\}_{\pi'^S} + \{RC_j\}_{\pi^1}$.

It follows that for DMUg, $\{RC_g\}_{\pi'^S} = (\gamma_1 \times \alpha_1)$ and we know that $\underset{j\in N}{Min}\{RC_j\}_{\pi^1} = \omega_2$. Therefore, $Min\{RC_g\}_{\overline{\pi}'} \geq (\gamma_1 \times \alpha_1) + \omega_2$.

On the other hand, among the remaining units, $\underset{j\in N/g}{Max}\{RC_j\}_{\pi'^S} = (\gamma_1 \times \alpha_2)$ and we know that $\underset{j\in N}{Max}\{RC_j\}_{\pi^1} = \omega_1$. Therefore, $\underset{j\in N/g}{Max}\{RC_j\}_{\overline{\pi}'} \leq (\gamma_1 \times \alpha_2) + \omega_1$.

So, $Min\{RC_g\}_{\bar{\pi}'} - \underset{j\in N/g}{Max}\{RC_j\}_{\bar{\pi}'} \geq [(\gamma_1 \times \alpha_1) + \omega_2] - [(\gamma_1 \times \alpha_2) + \omega_1]$.

Rearranging the terms, $Min\{RC_g\}_{\bar{\pi}'} - \underset{j\in N/g}{Max}\{RC_j\}_{\bar{\pi}'} \geq [\gamma_1 \times (\alpha_1 - \alpha_2)] - (\omega_1 - \omega_2)$.

Recall that $\gamma_1 = \dfrac{\omega_3}{\alpha_3} = \dfrac{(\omega_1 - \omega_2)}{(\alpha_1 - \alpha_2)}$. So, $\gamma_1 \times (\alpha_1 - \alpha_2) - (\omega_1 - \omega_2) = 0$.

Hence, at a scaling factor of γ_1, $Min\{RC_g\}_{\bar{\pi}'} \geq \underset{j\in N/g}{Max}\{RC_j\}_{\bar{\pi}'}$. It follows that if $\pi^S = \gamma_2 \times \pi^* = (\gamma_1 + \varepsilon) \times \pi^*$ and $\bar{\pi} = (\gamma_2 \times \pi^*) + \pi^1$, $Min\{RC_g\}_{\bar{\pi}} > \underset{j\in N/g}{Max}\{RC_j\}_{\bar{\pi}}$. This dictates that, provided $\omega_1 \neq \omega_2$, the global maximum of the reduced cost values at $\bar{\pi}$ will only be achieved by DMUg, i.e., $\underset{j\in N}{ArgMax}\{RC_j\}_{\bar{\pi}} = DMUg$.

Q.E.D.

Now consider the highly restrictive possibility of $\omega_1 = \omega_2$, i.e., all the n units achieve the same reduced value at $\pi^1 = (1,1,0)$. So $\omega_1 - \omega_2 = 0$ and hence $Min\{RC_g\}_{\bar{\pi}} - \underset{j\in N/g}{Max}\{RC_j\}_{\bar{\pi}} = (h \times (\alpha_1 - \alpha_2)) - (\omega_1 - \omega_2) = h \times (\alpha_1 - \alpha_2) > 0$; where, $h > 0$ is the scaling factor to be applied on the original optimal weights π^*. It is easy to see that any positive value of h is enough to ensure that $\underset{j\in N}{ArgMax}\{RC_j\}_{\bar{\pi}} = DMUg$. The simplest way to implement this would be to let $h = 1$ leading to $\bar{\pi} = \pi^* + \pi^1$.

7.3.3.1 Illustration of the closed-form solution to ensure positive weights under VRS

We will illustrate the above closed-form solution using the following 10 DMUs, 2 inputs, 2 outputs DEA problem presented in table 7-23 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j
A	1	0	3	3	9
B	1	12	40	7	9
C	6	0	60	2	6
D	6	9	10	5	6
E	0	40	22	1	-37
F	0	15	8	1	-12
G	1	18	5	5	-3
H	4	0	8	4	12
I	4	30	150	10	0
J	5	16	7	2	-10
π_f^*	*0*	*1*	*0*	*3*	

Table 7-23 : Closed-form solution to obtain positive multipliers 1

Assume that $GEN = \{I\}$, DMUt = A and upon solving the relevant LP we arrived at the optimal weights defined by $\pi^* = (0,1,0,3)$ with $\beta^* = 0$; at π^*, $\underset{j \in N}{Max}\{RC_j\}_{\pi^*} > 0$ and $\underset{j \in N}{ArgMax}\{RC_j\}_{\pi^*} = \{H\}$. Although unit H is guaranteed to be a generator, it has one zero input weight and one zero output weight and we are interested in providing it with strictly positive set of weights. Using the notations introduced earlier, $\alpha_1 = 12$, $\alpha_2 = 9$ and $\alpha_3 = (\alpha_1 - \alpha_2) = 12 - 9 = 3$.

Now, consider the following table where for the same dataset, the reduced cost values of the units are shown at the unary weights $\pi^1 = (1,1,1,1)$ with $\beta^1 = 0$.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j
A	1	0	3	3	5
B	1	9	40	7	34
C	6	0	60	2	56
D	6	9	10	5	0
E	0	40	22	1	-17
F	0	15	8	1	-6
G	1	18	5	5	-9
H	4	0	8	4	8
I	4	30	150	10	126
J	5	16	7	2	-12
π_f^1	*1*	*1*	*1*	*1*	

Table 7-24 : Closed-form solution to obtain positive multipliers 2

Now, $\omega_1 = 126$, $\omega_2 = -17$, $\omega_3 = (\omega_1 - \omega_2) = 126 - (-17) = 143$; also, $\gamma_1 = \frac{\omega_3}{\alpha_3} = \frac{143}{3} = 47.67$ and $\gamma_2 = \gamma_1 + 1 = 48.67$. The synthesised set of weights are given by $\bar{\pi} = \pi^S + \pi^1$ where $\pi^S = \gamma_2 \times \pi^*$. Hence, $\bar{\pi} = (1, 49.67, 1, 147.01)$ with $\bar{\beta} = 0$. At the modified hybrid weights, the reduced costs are shown in table 7-25 below.

DMU	*X1*	*X2*	*Y1*	*Y2*	RC_j
A	1	0	3	3	443.03
B	1	9	40	7	472.03
C	6	0	60	2	348.02
D	6	9	10	5	292.02
E	0	40	22	1	-1817.79
F	0	15	8	1	-590.04
G	1	18	5	5	-155.01
H	4	0	8	4	592.04
I	4	30	150	10	126
J	5	16	7	2	-498.7
$\bar{\pi}_f$	*1*	*49.67*	*1*	*147.01*	

Table 7-25 : Closed-form solution to obtain positive multipliers 3

The reduced cost values at $\bar{\pi}$ are the sum total of the reduced cost values at the modified original optimal weights π^S and the unary weights π^1. The reduced cost values at π^* at which unit H achieved the unique maximum reduced cost value are magnified by the factor $\gamma_2 = 48.67$ such that it counterbalances the effect of the reduced cost values at π^1. Hence, by magnifying the reduced cost values at π^* by the factor γ_2, we ensure that at the synthesised set of weights $\bar{\pi}$, the maximum of the reduced cost values is once again uniquely achieved by unit H but now at a set of strictly positive multiplier values.

7.4 Conclusion

In this chapter, we presented closed-form solutions to deal with the technical challenges of ties and indeterminate ratios while employing GBA under different returns to scale assumptions. We also presented closed-form solutions to arrive at a strictly positive set of multiplier values for the generators under different returns to scale assumptions. Note that by employing GBA, it is easy to provide strictly positive multiplier values for every generator. To see this, recall that generators are identified in GBA in two different ways. Either ArgMax is unique, leading to a generator, possibly with some zero multipliers - but the closed-form solution can lead to strictly positive multipliers; else, a generator is found by breaking ties using a closed-form method which also guarantees positive multipliers.

In the subsequent chapter, we will discuss the computational experiments carried out in solving oriented and non-oriented CRS and VRS models using GBA to compare its computational performance against Dula's BuildHull and the conventional two-phase algorithm.

8 COMPUTATIONAL RESULTS FOR GBA

This chapter presents the computational results of solving oriented and non-oriented DEA models under different returns to scale assumption and data characteristics. In solving the various models, we compare the computational performance of GBA against Dula's BuildHull algorithm and the conventional solution procedure. Using extensive computational experiments, Dula (1998, 2008) has shown that his BuildHull algorithm is computationally superior to the hierarchical decomposition procedure of Barr and Durchholz (1997) and the conventional solution procedure in solving the additive VRS model. We compare the computational performance of GBA against BuildHull in solving oriented and non-oriented VRS models using the same datasets that Dula has employed in his studies. Additionally, we show how GBA and BuildHull compare with the conventional solution procedure in solving the VRS models. Lastly, we compare the computational performance of GBA against BuildHull and the conventional solution procedure in solving the input-oriented CRS model using a problem suite developed for this purpose.

This chapter is organized as follows. In section 8.1, we present the problem suite and algorithmic descriptions of the three algorithms, viz., GBA, BuildHull, and the standard two-phase procedure to solve the output-oriented VRS model. We discuss the LP solver employed, the programming language and the software environment in which the experiments were carried out. We then examine the limitations of our computational experiments. Subsequently, we present the computational results and study the individual impact of each of three characteristics of a DEA dataset, namely, cardinality, dimension and density, on computational time. The above structure is followed in section 8.2 in studying the computational performance of the three algorithms in solving the additive VRS model and in 8.3 in solving the input-oriented CRS model.

Results indicate that GBA is consistently faster than BuildHull in solving the oriented CRS and VRS models. While solving the additive VRS model, BuildHull outperforms GBA although inconsistently at higher dimensions. We provide two alternative ways of modifying GBA to solve the additive VRS model and show using preliminary experiments that both the alternative approaches have distinctive advantages over BuildHull with one of them consistently outperforming BuildHull at higher dimensions.

8.1 Competitive algorithms to solve the output-oriented VRS model

Dula (1998, 2008) has shown the computational advantage of employing the BuildHull algorithm over the standard solution procedure and the hierarchical decomposition procedure using the problem suite developed in Dula (1998). The datasets developed in Dula (1998) were generated for the specific case where the returns to scale is variable. Results, as in Dula (1998, 2010), show that under any characteristic of the dataset, BuildHull is consistently faster than the other two. In order to prove the computational superiority of GBA over BuildHull, we will use the same problem suite that Dula has developed and employed in all his papers (see, Dula, 1998; Dula & Thrall, 2001; Dula & Lopez, 2002; Dula & Lopez, 2006; Dula, 2008; Dula & Lopez, 2009). We discuss this problem suite in section 8.1.1. Using this problem suite, we study the computational performance of GBA, BuildHull and the conventional two-phase algorithm in solving the output-oriented VRS model. The primary reason for choosing the output-oriented VRS model[39] is that the sponsor organisation of this research, the DfE, is interested in using this model to evaluate the primary schools in England. In addition, it is the most commonly used model in the parametric, semi-parametric and non-parametric literature on productivity theory (see, Fried et al, 2008). Towards the end of this section, we discuss the computational performance of GBA, BuildHull, and PIM DEA Soft v2, the efficiency analysis software currently employed by the DfE, in solving a real dataset provided by the them.

8.1.1 Description of the problem suite

The problem suite used in our experiments is obtained from Dula's website (http://www.people.vcu.edu/~jdula/LargeScaleDEAdata/). There are 64

[39] Note that the characteristics of a dataset (density, cardinality and dimension) are unaffected by the orientation of the analysis. Also, there is no immediate reason to expect that the orientation of the DEA model will have a differential impact on the computational performance of the three algorithms.

datasets with varying characteristics of dimension, density and cardinality. In this section, we present the configuration of the 64 datasets and refer to Dula (2008) for a detailed discussion of the Data Generating Process employed in generating these datasets and the advantages of generating datasets in such fashion.

In the 64 datasets, the cardinality varies between 2500 and 10000 DMUs, the dimension between 5 and 20 factors, and the density between 1% and 50%. Each of the three characteristics of the dataset can take 4 different values within their ranges. In particular, the cardinality can be 2500, 5000, 7500 or 10000, the dimension can be 5, 10, 15 or 20, and the density 1%, 13%, 25% or 50%, in total creating $4 \times 4 \times 4 = 64$ datasets. In the first 16 datasets, the dimension of the problem is fixed at 5 which includes 2 inputs and 3 outputs. In the first 4 of these 16 datasets, the cardinality is fixed at 2500 and the density is varied between 1% and 50%[40]. In the second 4 of the 16 datasets, the cardinality is fixed at 5000, in the third 4, cardinality is fixed at 7500 and in the final 4, it is fixed at 10000 with the density varied between 1% and 50% in each 4 of the 16 datasets. In the second 16 datasets, the dimension is fixed at 10 which includes 4 inputs and 6 outputs; in the third 16 datasets, the dimension is fixed at 15 which includes 7 inputs and 8 outputs, and in the last 16 datasets, the dimension is fixed at 20 which includes 9 inputs and 11 outputs[41]. Within each 16 datasets, the cardinality and density values are distributed in the manner described above.

8.1.2 Technology and Implementations

The three algorithms were implemented in R 2.8.1 which is a software environment primarily for statistical computation and software development. Its source code is freely available under the GNU General Public License. R 2.8.1 is the programming language in which the algorithms were written. To solve the LP problems, R 2.8.1 is interfaced with the linear programming solver lp_solve 5.5.0.14. lp_solve 5.5.0.14 is a non-commercial linear and integer programming

[40] Note that in real datasets, the density is typically never more than 10% as was confirmed in our experience with the school's data provided by the DfE and in earlier researches by Barr and Durchholz (1997) and Dula (1998).
[41] In real datasets that we and others have come across (see, Barr & Durchholz, 1997; Dula, 1998), the dimension and cardinality values do not typically go beyond 10 factors and 15000 DMUs respectively.

solver based on the revised simplex method and branch and bound procedure for solving linear and integer problems. The form of lp_solve 5.5.0.14 used was the callable library; i.e., a collection of functions were called from inside a R 2.8.1 program. Although, it has been well documented that non-commercial LP solvers (like lp_solve) are inferior in computational performance when compared to the commercial solver CPLEX[42], lp_solve 5.5.0.14 was selected as the solver of choice for two reasons. The principal reason is that, similar to R 2.8.1, lp_solve 5.5.0.14 is freely available under the GNU General Public License and hence there are no licensing issues involved. This enables our programs to be readily tested and used by others upon installing R 2.8.1 and lp_solve 5.5.0.14, which are both open source. Second, the interfacing of lp_solve 5.5.0.14 with R 2.8.1 is uncomplicated and the interfacing package is freely available (see, http://cran.r-project.org/web/packages/lpSolve/index.html). Unfortunately, this is not the case with interfacing the CPLEX solver with R 2.8.1. Neither is CPLEX freely available to use within R 2.8.1 and nor is the interfacing straightforward under Windows XP operating system. Although, CPLEX could have been our solver of choice, as we are employing the same (software environment, programming language and) LP solver to compare the performance of the three algorithms, the differential impact of the solver on any one of the algorithm's performance, if any, is expected to be marginal. The programs were executed on a dedicated DELL personal computer with INTEL Pentium E8200 CPU at 2.66 GHz and 1.95 GB of RAM running a Windows XP operating system.

Now that we have determined the software environment, the LP solver and the programming language to write and execute the algorithms, we can decide on the structure of the three algorithms. Since, GBA and BuildHull are similar in their philosophy, they are written with similar structures. As the three algorithms are already described in enough detail in the earlier chapters, we will only present their procedures here. The programmed versions of GBA, BuildHull and the conventional two-phase procedure to solve the output-oriented VRS model can be seen in Appendix 5.

[42] See, http://lpsolve.sourceforge.net/5.0/LinearProgrammingFAQ.htm#Q2, http://lionhrtpub.com/orms/surveys/LP/LP-survey.html, http://plato.asu.edu/ftp/lpfree.html, http://plato.asu.edu/ftp/milpf.html and http://scip.zib.de/ for discussions on benchmark results of LP and MILP solvers.

Before describing the procedures, we would like to mention three important points. First, the LP acceleration technique of reoptimization or hot-start is not employed in any of the three algorithms. Second, DEA specific computational enhancements like early identification of efficient units or reduced basis entry are not employed in the experiments carried out. Only a naïve implementation of the three algorithms is considered. Third, among the list of scaling algorithms provided as part of the solver options, the 'numerical range based scaling' algorithm was consistent and faster; hence, the particular scaling algorithm was invoked within the solver while solving the LPs. We elaborate on the first two points in section 8.1.3.

8.1.2.1 Description of the GBA procedure

As we are solving the output-oriented VRS model using GBA, we need to take care of the infeasibility problem. To ensure feasibility of any LP solved using GBA, we introduce DMUt into the coefficient matrix alongside the set of generators at a penalty of -2. The technical challenge of ties was resolved using the closed-form solution approach presented in chapter 7.

There are two parts to the implementation of GBA. The first part is the initialisation and the second part is the actual run of the algorithm. The initialisation part of GBA is accomplished by evaluating the reduced costs RC_j of all the units at unary weights $(1,1,0)$[43] and choosing the unit that achieved the maximum RC_j value as our starting generator[44]. All other units are placed in the set of status unresolved units U. The second part of the algorithm is the actual run of the GBA. The DMUs are evaluated in the order present in the dataset.

GBA, described as above, requires solving $(n-1)$ LP problems and the size of the LP increases from $(m+1)\times 3$ to $(m+1)\times(k+2)$, where k is the number of generators in the dataset. The algorithmic procedure of GBA is presented below.

[43] Since the weights for the input and output factors are strictly positive, the unit(s) achieving the maximum RC_j value must be P-K efficient which were confirmed to be generators in our experiments.

[44] No tie was observed for the maximum RC_j value at $(1,1,0)$.

PROCEDURE GBA

```
Step 0: Initialisation
set $GEN = \{\phi\}$, $U = \{1 \ldots n\}$, $TU = \{\phi\}$;
where,
GEN is the set of generators;
$U$ is the set of status unresolved DMUs;
$TU$ is the set of tied units for $ArgMax$ in $RC_j, j \in U$ at $\pi^*$;
    0.1 Evaluate $RC_j = 1Y_j - 1X_j, \forall j \in U$ and let $\underset{j \in U}{ArgMax}\{RC_j\} = DMUf$;
    0.2 Move DMUf to set GEN.
End Initialisation.
Step 1: Iteration. While $U \neq \{\phi\}$, do:
    1.1 Select the first DMU from $U$, DMUt, and solve MGBA LP-4 for it; let the
        optimal weights be $\pi^* = (v^*, u^*, v_0^*)$;
    1.2 Evaluate $RC_j$ at $\pi^*$ for $j \in U$;
    1.3 If $\underset{j \in U}{Max}\{RC_j\} \leq 0$, do:
        1.3.1 Record the optimal weights, peers and slacks for DMUt;
        1.3.2 Remove DMUt from $U$ and go to Step 1.1;
    1.4 If $\underset{j \in U}{Max}\{RC_j\} > 0$, do:
        1.4.1 Compute $\underset{j \in U}{ArgMax}\{RC_j\}$ and $TU$;
            1.4.1.1 If $|TU| > 1$, go to Step 1.4.2;
            1.4.1.2 If $|TU| = 1$ and $\underset{j \in U}{ArgMax}\{RC_j\} = DMUf$, record the optimal
                weights for DMUf; Move DMUf to set GEN; Go to Step 1.1;
        1.4.2 Resolve the tie in $ArgMax$ using the closed-form solution. Identify
              one P-K efficient unit, $DMUq \in TU$ and record the weights for it;
            1.4.2.1 Move DMUq to set GEN; Go to Step 1.1;
End Procedure.
```

8.1.2.2 Description of BuildHull and the Standard two-phase algorithm

The procedure for solving the output-oriented VRS model using BuildHull is similar to what has been described in Dula (1998) except for the initialisation part and the subroutine to resolve ties. Similar to GBA, BuildHull was initialised by evaluating the reduced costs RC_j of all the units at unary weights $(1,1,0)$. It is important to note that although Dula (1998) described an LP based procedure to resolve ties, the procedure was never implemented to our knowledge[45]. Ties in our implementation of BuildHull were resolved using the closed-form solution described in chapter 7.

Both BuildHull and the standard DEA algorithm are two-phase procedures and BuildHull requires solving $2n-k$ LPs while the standard DEA algorithm requires solving $2n$ LPs. The maximum number of columns in any LP solved in BuildHull is $k+1$, where k is the number of extreme-efficient units in the dataset. The number of columns in the standard procedure is fixed at n in phase-1 and $(n+m)$ in phase-2. For the sake of completeness, we describe our implementation of the BuildHull algorithm and the standard two-phase algorithm below.

PROCEDURE BUILDHULL

Step 0: Initialisation

set $E=\{\phi\}$, $U=\{1...n\}$, $TU=\{\phi\}$;

where,

[45] In a recent version of his BuildHull algorithm, Dula (2010) suggests resorting to a sorting algorithm to identify one extreme-efficient unit among the tied units for the VRS case. This proposal, although valid, was not implemented in his algorithm to our knowledge.

E is the set of generators;

U is the set of status unresolved DMUs;

TU is the set of tied units for $ArgMax$ in $RC_j, j \in U^+$ at π^*;

```
    0.1  Evaluate $RC_j = 1Y_j - 1X_j, \forall j \in U$ and let $\underset{j \in U}{ArgMax}\{RC_j\} = DMUf$;
    0.2  Move DMUf to set E.
End Initialisation.
Step 1: Phase-1 Iteration. While $U \neq \{\phi\}$, do:
    1.1  Select the first DMU from set $U$, DMUt, and solve the phase-1 VRS LP of
         BuildHull for it;
    1.2  Let the optimal weights be $\pi^* = (v^*, u^*, \beta^*)$;
    1.3  If the optimal objective function value is = 0, do:
        1.3.1 Remove DMUt from $U$ and move it to set I; Go to Step 1.1;
    1.4  If the optimal objective function value is > 0, do:
        1.4.1 Compute $RC_j$ at $\pi^*$ for $j \in U^+$ and $TU$;
            1.4.1.1 If $|TU| > 1$, go to Step 1.4.2;
            1.4.1.2 If $|TU| = 1$ and $\underset{j \in U^+}{ArgMax}\{RC_j\}_{\pi^*} = DMUp$, record the
                optimal weights for DMUp; Move DMUp to set E; Go to Step
                1.1;
        1.4.2 Resolve the tie in $ArgMax$ using the closed-form solution. Identify
            one P-K efficient unit, $DMUq \in TU$ and record the weights for it;
        1.4.2.1 Move DMUq to set E; Go to Step 1.1;
  End Phase-1 Procedure.
Step 2: Phase-2 Iteration. While $I \neq \{\phi\}$, do:
    2.1  Select the first DMU from set $U$, DMUt, and solve LP-8 for it against units
         in E; Record the efficiency score, weights, slacks and peers for DMUt;
    2.2  Remove DMUt from I; Go to Step 2.1;
```

```
End Phase-2 Procedure.

End BuildHull Procedure.
```

PROCEDURE STANDARD TWO-PHASE ALGORITHM (VRS)

```
Step 0: Initialisation
set $U = \{1,...,n\}$; $R = \{\phi\}$;
where,
$U$ is the set of status unresolved DMUs;
$R$ is the set of status resolved DMUs;
Step 1: Phase-1 of the standard DEA procedure. While $U \neq \{\phi\}$, do:
    1.1  Select the first DMU in $U$, DMUt, and solve LP-8 for it. Record the
         efficiency score and weights for DMUt;
    1.2  Remove DMUt from $U$ and move it to set $R$; Go to Step 1.1;
End Phase-1 procedure.
Step 2: Phase 2 of the standard DEA procedure. While $R \neq \{\phi\}$, do:
    2.1  Select the first DMU, DMUt, from set $R$ and solve the max-slack VRS
         model for it. Record the input and output slacks and peers for DMUt;
    2.2  Remove DMUt from $R$ ; Go to Step 2.1;
End Phase-2 procedure.
End Standard two-phase algorithm.
```

8.1.3 Limitations of the computational experiments

Before we present the results of our computational experiments, it is important to discuss its limitations. The first two limitations were briefly discussed in section 8.1.2. The first limitation is that the LP solver lp_solve 5.5.0.14 and the programming language R 2.8.1 employed in our study may not be the best of the options to carry out our computational experiments. A better choice could be to employ CPLEX solver's callable library within a Fortran program as in Dula's experiments. As mentioned earlier, the principal advantage of our solver and programming language is that both are open source. Also, as we have programmed the algorithms in the same language and employed the same solver to solve the LPs, the differential impact on any one of the algorithms, if any, is expected to be marginal.

The second limitation is that we have restricted ourselves to only a naïve implementation of the three algorithms. Specifically, the LP acceleration technique of hot starts (reoptimization) and DEA specific enhancement techniques like earlier identification of efficient units (EIE) and restricted basis entry (RBE) were not employed in any of the algorithms. Contrasting Barr and Durchholz (1997), Dula in his studies (see, Dula, 2008; Dula, 2010) has observed that hot starts and RBE can effect substantial computational savings and could be readily applied to the conventional DEA algorithm, the hierarchical decomposition procedure, and BuildHull.

As mentioned in section 3.1, Ali (1993) reports a reduction of 84% for a particular dataset while in another experimental study by Barr and Durchholz (1997) and Dula (2008), it has been shown that RBE can reduce the computational time by 50%.

We did not apply such enhancements in our study for two reasons. First, by employing these enhancements, the computational performance of the three algorithms could be affected differentially and hence, our understanding of the algorithms' performance could be distorted. Second, employing hot starts was not an option provided with the version of our LP solver that can be interfaced with R 2.8.1. Hence, implementing the technique in the algorithms was not straight forward.

The third limitation is that we evaluate the units in the order present in the dataset and did not explore other options. For example, when the evaluation of DMUt is complete, one can choose the unit in U that achieved the second highest RC_j value as the next one because, it is likely to be a new generator (see, Sueyoshi, 1990). Evaluating units in this manner could help us identify all the generators in the dataset earlier. This could have a positive impact on the computational performance of GBA.

Another alternative, on proving that DMUt is a non-generator, would be to identify those units g in U for which the optimal basis for DMUt provides a feasible solution for the envelopment form model.

8.1.4 Computational results and comparison of algorithmic performances

The computational results in solving the 64 datasets using the three algorithms can be seen in Appendix 6. No ties were encountered in solving the 64 datasets using GBA and BuildHull.

The first obvious and encouraging result is that regardless of the cardinality or dimension or density characteristic of the dataset, GBA solves any dataset faster than BuildHull. GBA is also consistently faster and by a considerable margin than the standard two-phase algorithm. BuildHull is also consistently faster than the standard two-phase DEA algorithm. As expected, there is a gradual decrease in the computation advantage of GBA and BuildHull over the standard two-phase procedure as density increases from 1% to 50%. In addition, if one is only interested in solving the first phase of the standard algorithm to obtain the efficiency scores of the units, then BuildHull was slightly outperformed by the conventional algorithm in two instances. When the dimension of the dataset is 5 and density 50%, and when the cardinality of the problem is 2500 or 5000, the 1st phase of the standard procedure is slightly faster than BuildHull. Given that the 1st phase of the standard procedure requires solving n LPs of fixed size $(m+1)\times n$ while BuildHull requires solving $2n-k$ LPs of maximum size $(m+1)\times k$, the two instances wherein the former slightly out-performed the latter is not an altogether surprising result.

To get a better perspective of the individual impact of the characteristics of a dataset on the performance of the three algorithms, dimension, density and cardinality were fixed two at a time and for the four different values of the third factor, the computational times required by the three algorithms are plotted. In the first instance, the dimension and cardinality are fixed at the lower end of their ranges, i.e., at 5 and 2500 respectively, and the impact of density on the performance of GBA, BuildHull, and the standard algorithm (1st Phase and 1&2

Phases) are graphically illustrated in chart 8-1[46] below. Note that for each case, apart from charting the performance of the three algorithms, we also provide a chart comparing GBA with BuildHull alone to discriminate their relative performances better.

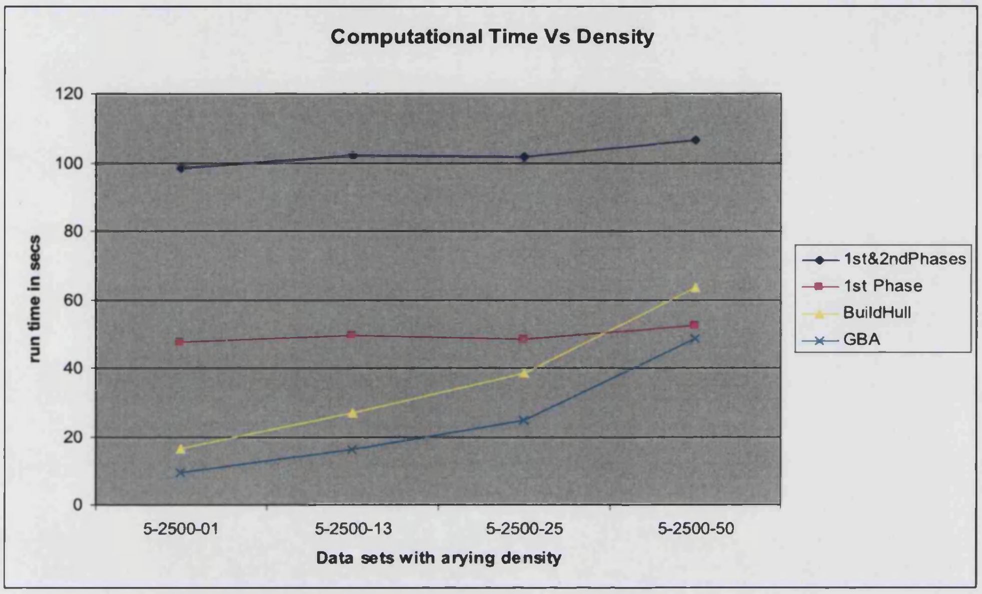


Chart 8-1 : Computational Time versus Density

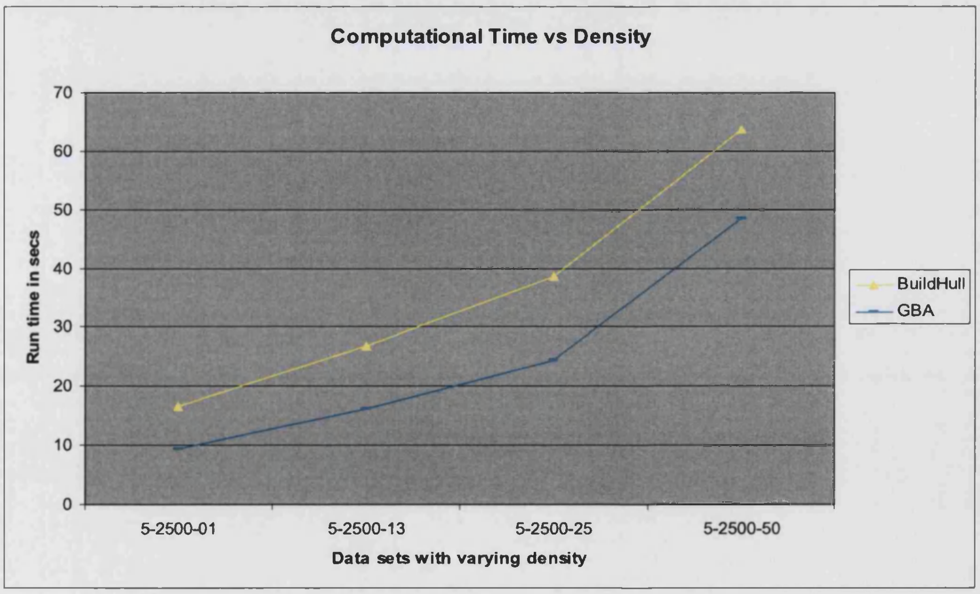


Chart 8-2 : Computational Time versus Density (GBA vs. BuildHull)

[46] On the x-axis, the datasets are varied keeping the dimension and cardinality fixed at 5 and 2500 respectively, and changing the density from 1% to 50%; so 5-2500-01 denotes the dataset with dimension fixed at 5, cardinality at 2500 and density at 1%.

As seen in charts 8-1 and 8-2, GBA solves the problem consistently faster regardless of the density of the dataset, and as density increases, the computational advantage of GBA and BuildHull over the standard procedure reduces gradually. BuildHull performs better than the standard algorithm and the 1st phase of the standard algorithm except when the density is at 50%. The graph that shows the trend when the dimension is fixed at 20 and cardinality at 10000 and the density is varied from 1% to 50% can be seen in charts 8-3 and 8-4 below.

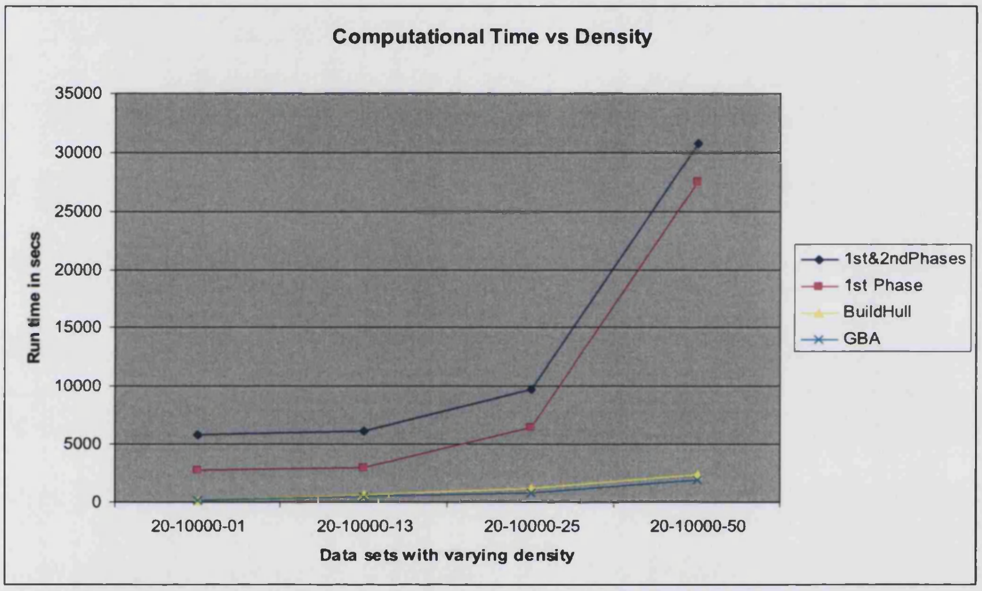


Chart 8-3 : Computational Time versus Density

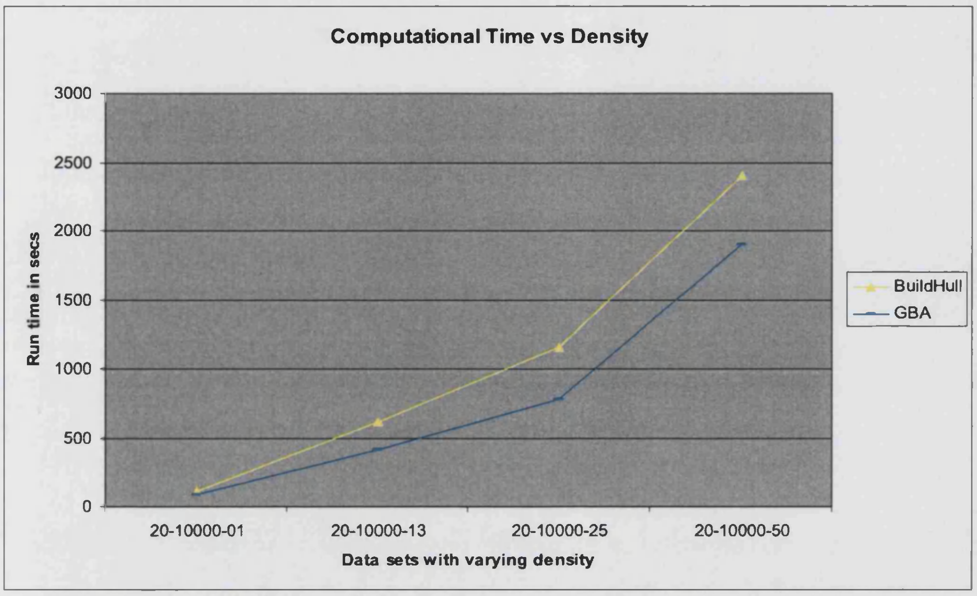


Chart 8-4 : Computational Time versus Density (GBA vs. BuildHull)

In the second scenario, the density and cardinality values are fixed at one end of their ranges, i.e., at 1% and 2500 respectively, and the impact of dimension at values 5, 10, 15 and 20 on the performance of GBA, BuildHull, and the standard algorithm are graphically illustrated in charts 8-5 and 8-6 below.

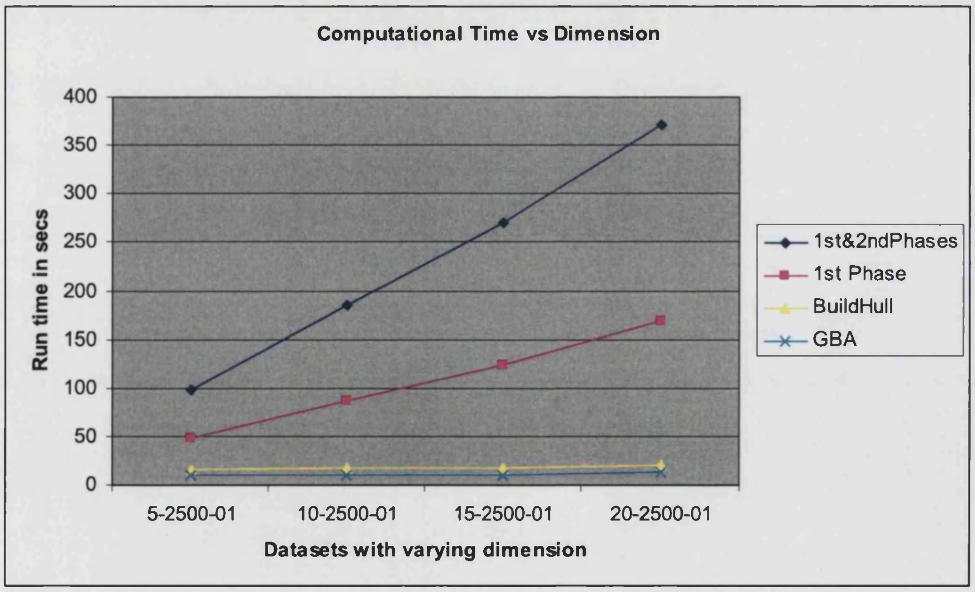


Chart 8-5 : Computational Time versus Dimension

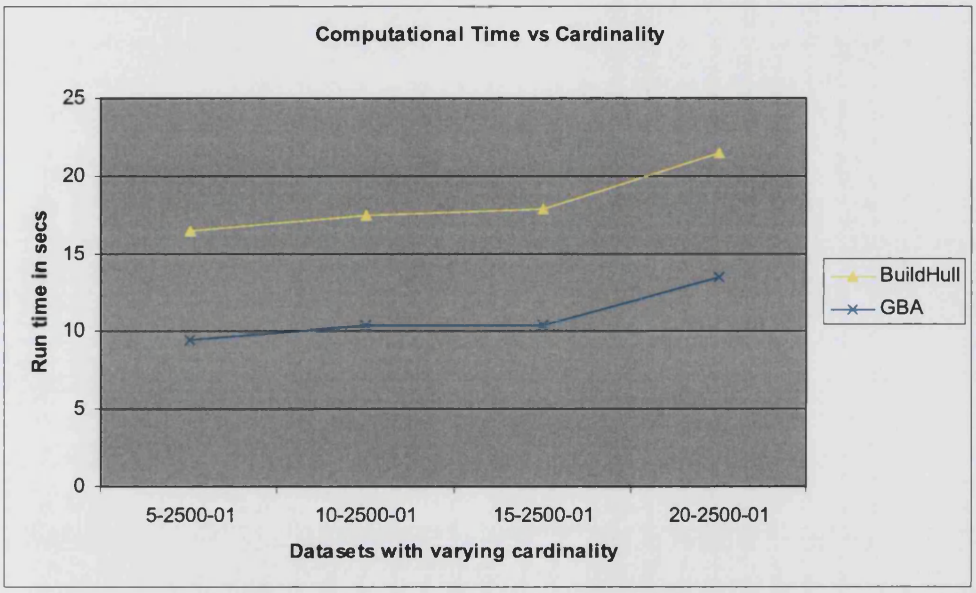


Chart 8-6 : Computational Time versus Dimension (GBA vs. BuildHull)

As seen in charts 8-5 and 8-6, GBA solves the problem consistently faster regardless of the dimension of the dataset and its computational advantage over BuildHull is almost unaffected by the dimension of the dataset. BuildHull also performs better than the standard algorithm in all cases. The graph that shows the trend at the other extreme when the density is fixed at 50% and cardinality at 10000 and the dimension is varied from 5 to 20 can be seen below.

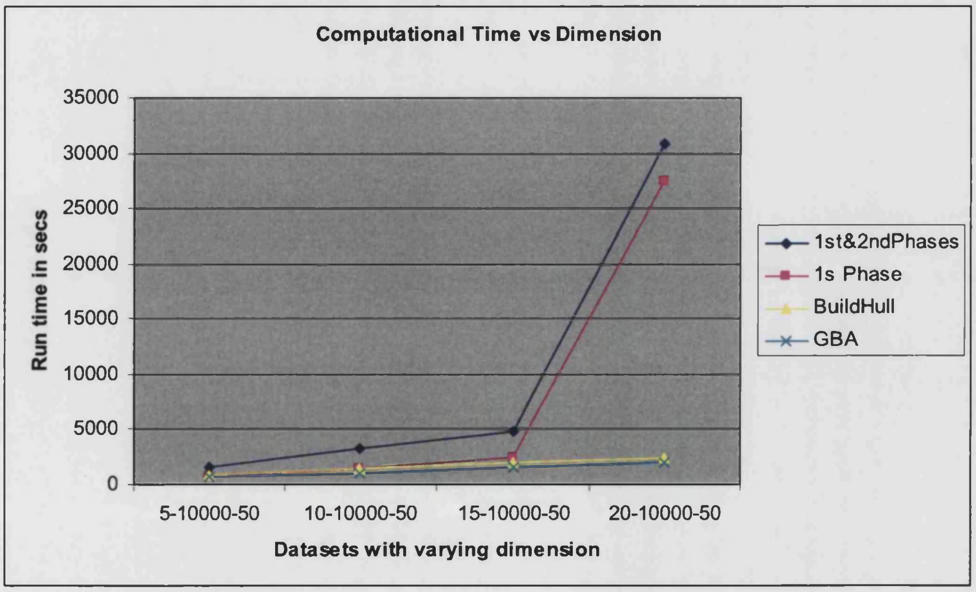


Chart 8-7 : Computational Time versus Dimension

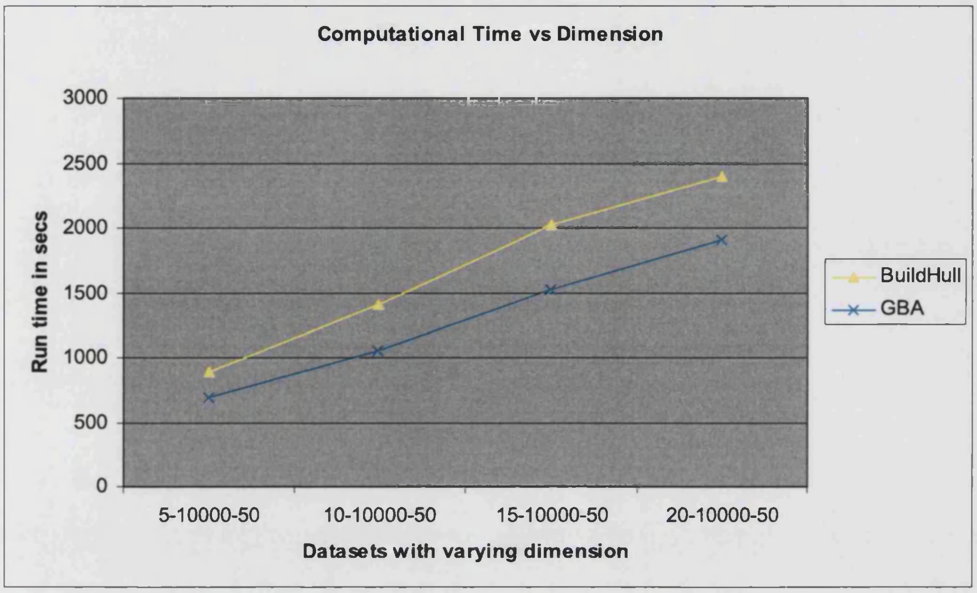


Chart 8-8 : Computational Time versus Dimension (GBA vs. BuildHull)

In the third scenario, the density and dimension values are fixed at one end of their ranges, i.e., at 1% and 5 respectively, and the impact of cardinality at values 2500, 5000, 7500 and 10000 on the performance of GBA, BuildHull, and the standard algorithm are graphically illustrated in the charts below.

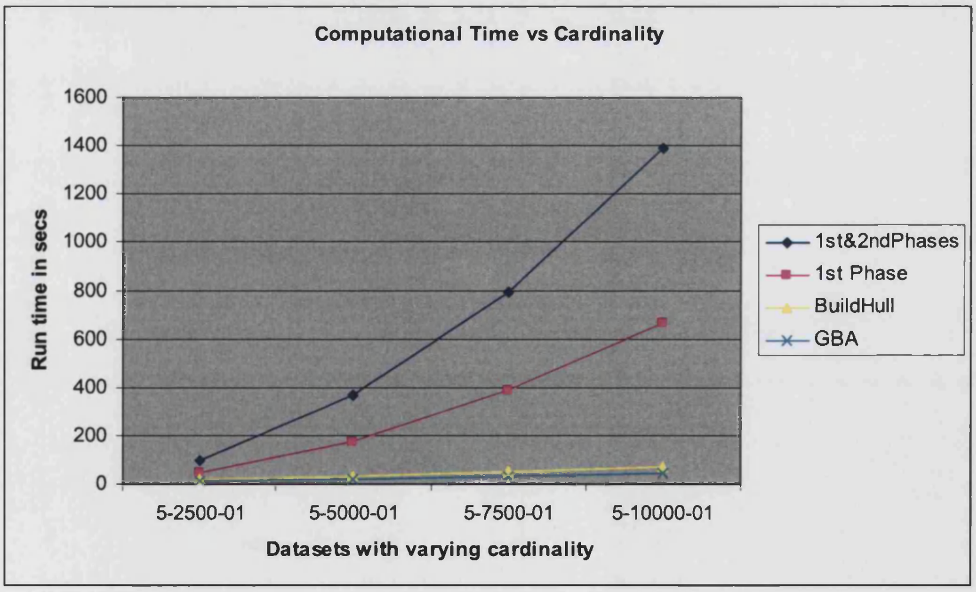


Chart 8-9 : Computational Time versus Cardinality

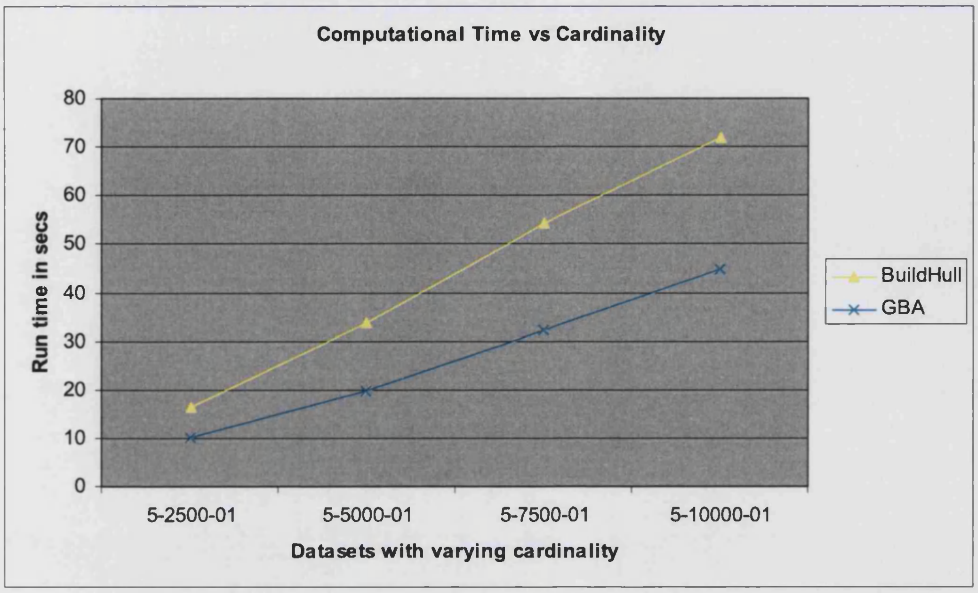


Chart 8-10 : Computational Time versus Cardinality (GBA vs. BuildHull)

As seen from charts 8-9 and 8-10, GBA solves the problem consistently faster regardless of the cardinality of the dataset. BuildHull also performs better than the standard algorithm in all cases. To discriminate better the time taken by the three algorithms, the following graph in chart 8-11 shows the computational performance for the same scenario in log seconds[47].

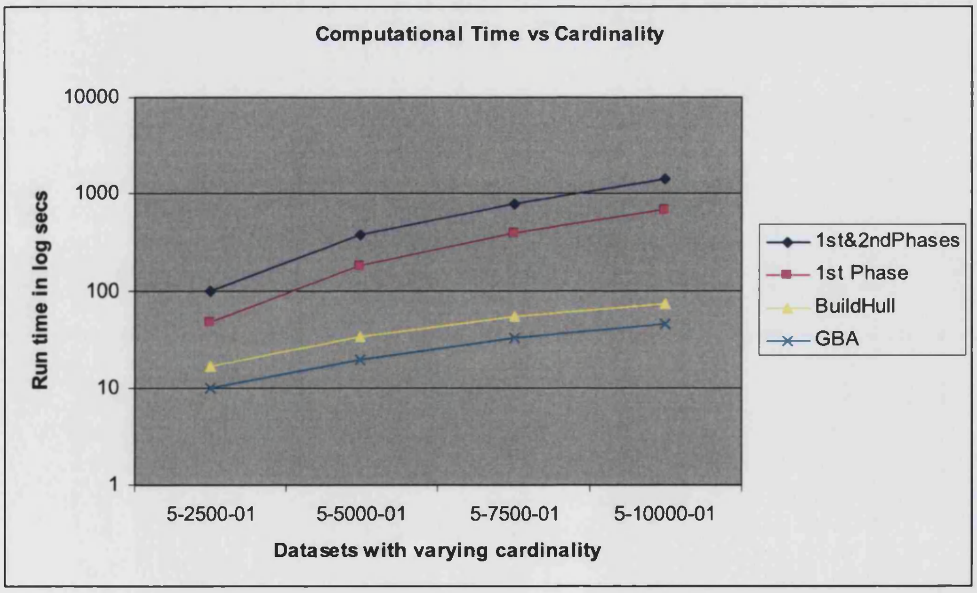


Chart 8-11 : Computational Time (log seconds) versus Cardinality

[47] Note that the shape of the log-curve depicts the proportional increase in the time taken by the algorithms to solve the datasets.

The same trend is also observed at the other end of the extreme when the density is fixed at 50%, dimension at 20 and the cardinality is varied from 2500 through 10000. The corresponding graph can be seen in charts 8-12 and 8-13 below.

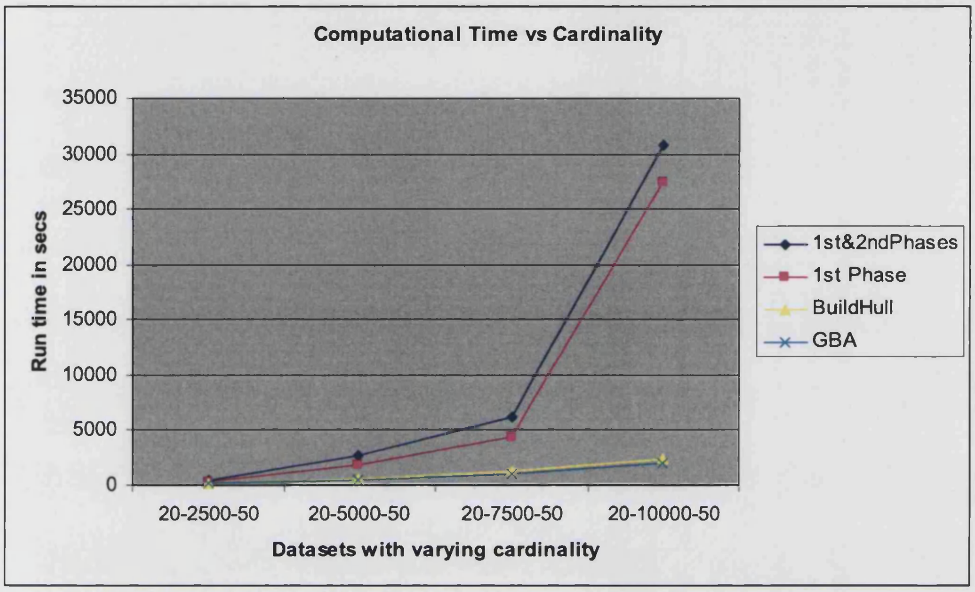


Chart 8-12 : Computational Time versus Cardinality

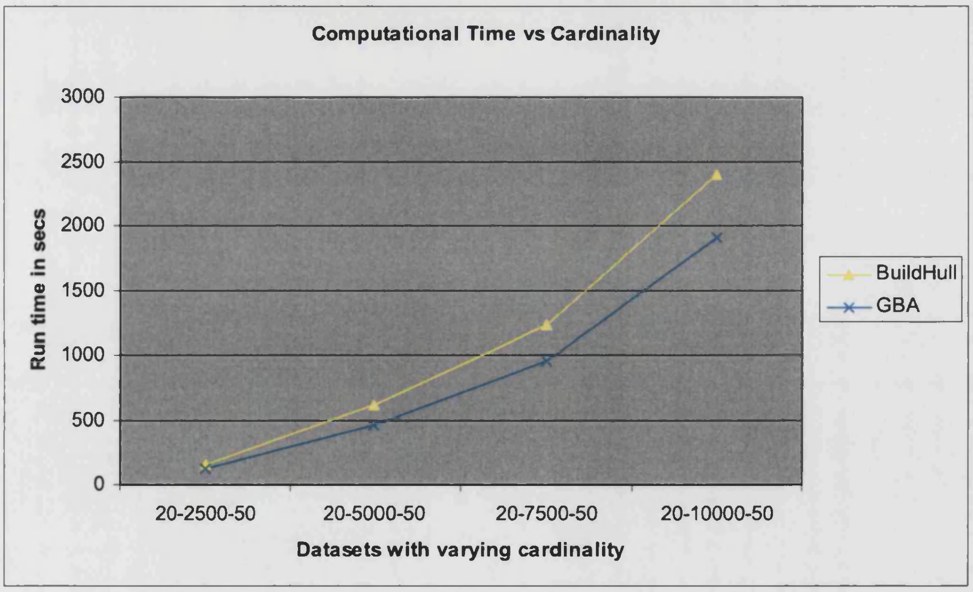


Chart 8-13 : Computational Time versus Cardinality (GBA vs. BuildHull)

Once again, to discriminate the time taken by the three algorithms, the following graph shows the computational performance for the same scenario in log seconds.

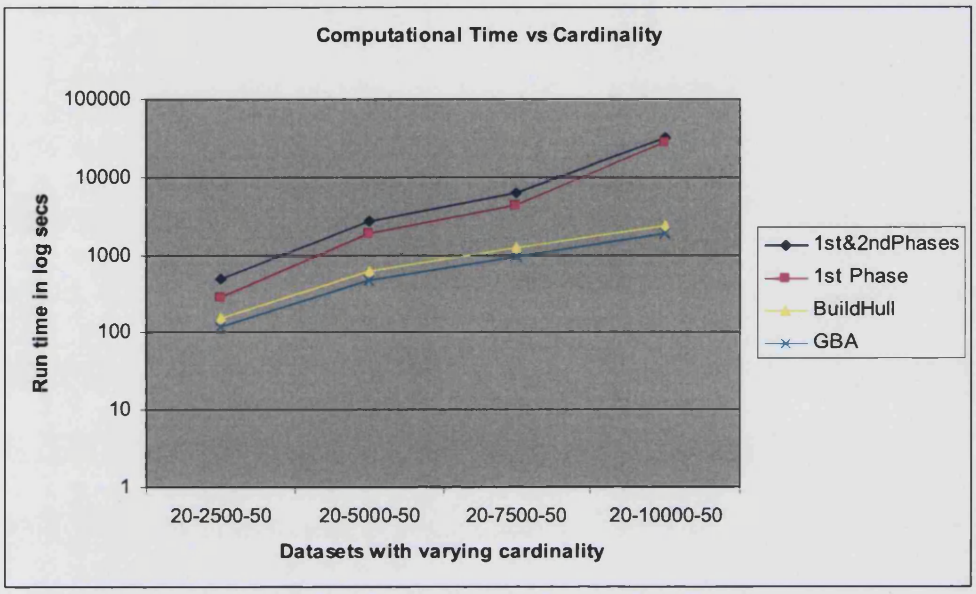


Chart 8-14 : Computational Time (log seconds) versus Cardinality

We also recorded the time taken by GBA, BuildHull and PIM DEA SOFT v2 to solve a real dataset received from the DfE. The real dataset involved 13216 DMUs (primary schools in England) with 6 inputs and 3 outputs. Zeroes existed in both inputs and outputs. We were asked to provide the efficiency score of the 13216 schools using the output-oriented VRS model.

BuildHull took 156.77 seconds to solve the dataset. Unlike in the simulated datasets, 8 ties were encountered with an average of about 111 units per tie. GBA took 105.27 seconds, a 33% reduction in time compared to BuildHull. Again, unlike in simulated datasets, 3 ties were encountered with an average of about 39 units per tie. The commercial DEA software, PIM DEA SOFT v2, took over 1 hour to solve the same dataset[48].

[48] Both BuildHull and GBA identified 188 extreme-efficient units among the 13216 units. Hence, the density of the real dataset is 1.42%.

8.2 Competitive algorithms to solve the additive VRS model

The same problem suite (of 64 datasets) has been employed to compare the performance of the three algorithms to solve the additive VRS model. The algorithmic procedures of GBA and BuildHull remain the same as in the oriented case, except for the LP model solved and the non-issue of ties[49]. To avoid duplication of material, their procedures are not repeated here. Unlike in the oriented models, the standard additive model has a single phase. Once again, its procedure is not presented here as it remains essentially the same as in the oriented case except for the LP solved and the absence of a second phase. The programmed version of GBA, BuildHull and the standard algorithm to solve the additive VRS model can be seen in Appendix 6.

The three limitations of our computational experiments observed in our design to solve the output-oriented VRS model using the algorithms are also applicable here.

8.2.1 Comparison of algorithmic performances

The computational results from solving the 64 datasets using the three algorithms can be seen in Appendix 7. Once again, the 'numerical range based scaling' algorithm was invoked in the solver while solving the LPs.

The first obvious result is that GBA and BuildHull consistently outperformed the standard algorithm. While employing GBA to solve the additive model, each LP is of size $(m+1)\times(g+m+1)$, where $g \leq k$ is the number of generators identified at a particular iteration and we solve $(n-1)$ such LPs. This became a handicap at combinations of higher dimension and density values. In contrast, while employing BuildHull to solve the additive model, the first phase LPs are of size $(m+1)\times(h+1)$, where $h \leq k$ is the number of generators identified at a particular iteration, and $(n-1)$ such LPs were solved. In the second phase, $(n-k)$ LPs of fixed size $(m+1)\times(k+m)$ were solved, where k is the number of generators in the dataset. In our experiments, we

[49] Recall that ties are a non-issue here as all the tied units are guaranteed to be P-K efficient and can be moved to set GEN.

observed that when the dimension of the dataset is 20, regardless of the density and cardinality values, BuildHull was able to solve the dataset slightly faster than GBA although not consistently. Also, when the dimension is 15 and cardinality is 7500 or 10000, regardless of the density of the dataset, BuildHull was able to solve the dataset slightly faster than GBA. At all other combinations, GBA solved the dataset consistently faster than BuildHull.

Based on preliminary experiments, we can provide two promising alternative approaches to GBA while solving datasets with extreme dimension characteristics using the additive model. First, it could be prudent to adopt a two-phase approach to GBA wherein one solves the corresponding oriented model in phase-1, score all the non-generators and identify the generators, and in phase-2, solve the additive model to score the non-generators using only the set of generators identified in phase-1. In this modified two-phase version of GBA, $(n-1)$ LPs of maximum size $(m+1)\times(k+2)$ are solved in the first phase and $n-k$ LPs of fixed size $(m+1)\times(k+m)$ are solved in the second phase. This can be seen as analogous to the BuildHull procedure in identifying all the generators in phase-1 and scoring the remaining units in phase-2. However, at a possibly higher computational cost, if we use GBA to solve additive models in two phases as described above, we could provide the efficiency score, peers and slacks for all the non-generators in the phase-1 using the (input or output) oriented model, and also the peers and slacks of the non-generators in the second phase using the additive model.

The second alternative approach is to solve the multiplier form (dual of the envelopment form) of the additive model using GBA. In this version of GBA, we will be solving $(n-1)$ LPs of maximum size $(k+m+1)\times(m+1)$, where k is the number of generators in the dataset. It is expected that at higher combinations of dimension and density values, the performance of GBA applied to the multiplier form of the additive VRS model can be competitive compared to BuildHull and GBA. To evaluate this procedure, we also programmed a modified version of BuildHull wherein in the second phase, the multiplier form of the additive model is solved for each non-generator using only the set of generators identified in phase-1; i.e., in the second phase of BuildHull, $(n-k)$ LPs of fixed size $(k+m)\times(m+1)$ are solved. Some preliminary results can be seen in table 8-1 below. The programmed version of the two alternative approaches to GBA along with the modified BuildHull procedure to solve the additive VRS model can be seen in Appendix 8.

Datasets	BuildHull	BuildHull_m	GBA_org	GBA_2ph	GBA_mult
20-2500-01	24.11	30.59	17.03	27.22	16.83
20-2500-13	73.49	67.56	131.05	74.77	42.19
20-5000-01	59.45	54.22	39.47	62.08	39
20-5000-13	241.28	263.76	1533.81	267.89	173.1
20-7500-01	120.09	122.81	97.73	133.33	96.86
20-7500-13	524.92	599.43	2337.53	574.64	385.25
20-10000-01	165.54	170.71	139.72	211.36	132.77
20-10000-13	921.61	1043.04	3022.21	1032.19	665.67
05-2500-01	17.11	47.66	8.83	29.84	15.8
05-2500-13	26.82	40.54	15.96	54.86	31.79
05-5000-01	33.96	59.27	19.6	65.93	34.38
05-5000-13	78.81	128.95	48.08	136.22	52.72
09-13216-01	161.97	178.72	104.32	749.41	129.74

Table 8-1 : Alternative approaches to solve the additive model

In the above table, we consider 12 of Dula's datasets wherein in the first 8, the dimension is 20, cardinality is 2500, 5000, 7500 or 10000, and density is 1% or 13%; in the next 4, the dimension is 5, cardinality is 2500 or 5000, and density is 1% or 13%. We also solved the real dataset (with 13216 DMUs, 9 factors and 1.42% density) from the DfE using the 5 algorithms. The table shows the time taken in seconds to solve the 13 datasets using Dula's BuildHull, the modified version of BuildHull (BuildHull_m), original GBA (GBA_org), the two-phase approach to GBA (GBA_2ph) and the multiplier version of GBA (GBA_mult).

From the above table, the only clear trends are that when the dimension is 20 (extremely high dimension), GBA_2ph takes more time than BuildHull to solve the datasets. Also, among the 5 algorithms, GBA_mult solves any dataset consistently faster. GBA_2ph is expected to take more time than BuildHull but still this two-phase approach to GBA requires only a moderate amount of additional time over BuildHull while providing information from solving the oriented and additive models. And lastly, when the dimension is 5 or 9 (small to medium dimension), GBA_org is the fastest among the 5 options.

It must be clear that given our limited experiments, the above results cannot be regarded as being conclusive. The two alternative GBA procedures are promising avenues for future research although additional work is required to make GBA consistently faster over BuildHull while solving datasets of any characteristics using additive models.

8.3 Competitive algorithms to solve the input-oriented CRS model

In this section, we study the computational performance of GBA, BuildHull, and the standard two-phase procedure in solving different datasets under CRS assumption. In addition, we evaluate the individual impact of dimension, density and cardinality characteristics on the performance of the three algorithms. So far, Dula (1998, 2008) has carried out computational studies only for the VRS models and the characteristics of the datasets he has generated under VRS assumption do not hold true under CRS. We will describe our data generation process (DGP) under CRS assumption before we present the limitations of our experimental design and the results.

8.3.1 Problem suite under CRS

For comparison purposes, it was decided that the characteristics of the datasets generated using the data generation process under CRS assumption must be similar to that used under VRS. Hence, the number of datasets was fixed at 64 and the density was allowed to take values 1%, 13%, 25% or 50%, dimension 5, 10, 15 or 20, and cardinality 2500, 5000, 7500 or 10000. However, unlike in the VRS datasets, the number of outputs in any m dimensional dataset was fixed at 1. This is because the datasets were generated using the traditional Cobb-Douglas functional form[50] (Cobb & Douglas, 1928) given by $y = x_1^{\alpha_1} \times x_2^{\alpha_2} \times ... \times x_{m1}^{\alpha_{m1}}$; where y is the single output and x_i are the input values for $i = 1$ to m_1. As we assume that the production technology exhibits CRS, the exponents on the input

[50] Many simulation studies in the DEA literature typically use some variant of the Cobb-Douglas form to generate the datasets (see, Banker et al, 1993; Banker et al, 2004; Banker & Natarajan, 2008; Banker & Parthasarathy, 2009; Banker et al, 2010).

values must sum to 1, i.e., $\sum_{i=1}^{m_1} \alpha_i = 1$. Since adapting the Cobb-Douglas functional form for multiple outputs would mean loss of control of the final density of the dataset, the number of outputs was fixed at 1. To our knowledge, there is no other uncomplicated way of generating datasets under CRS wherein the density is decided beforehand. This implies that in all the 64 datasets that were generated, the number inputs is given by $(m-1)$[51].

The datasets were generated in the following fashion[52]. First, the cardinality and dimension of the dataset are fixed to a certain value within their ranges. The $(m-1)$ inputs are then generated randomly and independently from a uniform distribution in the range (chosen arbitrarily) between 1.5 and 25. The exponents α_i's, for the $(m-1)$ inputs are generated using another uniform distribution in the range between 0 and 1[53]. The random exponents are then scaled such that they sum to 1. The density of the dataset is then decided to take any value among 1%, 13%, 25% and 50%. The density along with the cardinality of the dataset will determine the number of generators and non-generators. For the units that are generators, the output value is determined by the Cobb-Douglas functional form $y_i = \prod_{i=1}^{m_1} x_i^{\alpha_i}$ ensuring that they are on the frontier of the CRS production technology. For the units that are non-generators, the output value as determined by the Cobb-Douglas functional form is scaled down by a factor of 30 to make the corresponding units strictly interior (and hence, inefficient) w.r.t the CRS production frontier. By varying the cardinality, dimension and density of the DEA problem, 64 datasets were generated. Before evaluating the performance of the three algorithms, the 64 datasets were verified to ensure that each of the dataset has the number of generators defined by the density of the problem.

[51] Cobb-Douglas functional form imposes equality between isoquants and efficient subsets thereby eliminating any slacks (see, Fried et al, 2008). Hence, all the efficient units are P-K efficient and for all practical purposes, generators.

[52] Unlike previous studies involving simulation of DEA datasets, the parameter values to generate the datasets were chosen arbitrarily here. The parameter values are not expected to have a differential impact on the performance of the three algorithms.

[53] The uniform distribution function in R 2.8.1 does not take the extreme values of the range provided.

Having explained the data generating process, we now turn our attention to the structure of the three algorithms. Since, GBA and BuildHull are similar in their philosophy, they are written with similar structure. Also, as the algorithms were already described in enough detail in chapters 3 and 4, we will only present their procedures here. The programmed version of the data generating process along with the GBA, BuildHull and the standard two-phase procedure to solve the input-oriented CRS model can be seen in Appendix 9.

8.3.2 Description of GBA

As all the 64 datasets are strictly positive, any GBA LP we solve will be feasible. Hence, there is no need to introduce DMUt into the coefficient matrix at some penalty value. The only other issues to be taken care of are ties and indeterminate ratios. As we are dealing with strictly positive datasets, indeterminate ratios cannot occur. Ties can still happen and are resolved using the closed-form solution described in chapter 7.

Once again, there are two parts to the implementation of GBA. The first part is the initialisation and the second part is the actual run of the algorithm. The initialisation part of GBA is carried out by evaluating the R_j values of all the units at unary weights $(1,1)$[54] and choosing the unit that achieved the maximum of the R_j values as our starting generator[55]. At the end of the initialisation part of the algorithm, we are guaranteed to find one generator to be included in set GEN. All other units are placed in the set of status unresolved units U.

The second part of the algorithm is the actual run of the GBA procedure. The DMUs are evaluated in the order present in the dataset. As in the VRS case, the LP acceleration technique of reoptimization is not employed in any of the three algorithms. Also, DEA specific enhancements like EIE or RBE are not employed in the experiments carried out. As in the earlier experiments, the

[54] Since the weights are strictly positive, the unit(s) achieving the maximum R_j value must be P-K efficient (confirmed to be generators in our experiments).

[55] There was no tie for the maximum R_j value at $(1,1)$.

'numerical range based scaling' algorithm was invoked within the solver while solving the LPs. The algorithmic procedure of GBA is presented below.

```
PROCEDURE GBA
Step 0: Initialisation
set $GEN = \{\phi\}$, $U = \{1...n\}$, $TU = \{\phi\}$;
where,
GEN is the set of generators;
$U$ is the set of status unresolved DMUs;
$TU$ is the set of tied units for $ArgMax$ in $R_j, j \in U$ at $\pi^*$;
    0.1 Evaluate $R_j = \frac{1Y_j}{1X_j}, j \in U$, and let $\underset{j \in U}{ArgMax}\{R_j\} = DMUf$;
    0.2 Move DMUf to set GEN.
End Initialisation.
Step 1: Iteration. While $U \neq \{\phi\}$, do:
    1.1   Select the first DMU from $U$, DMUt, and solve MGBA LP-1 for it; let the
          optimal weights be $\pi^* = (v'^*, u'^*)$;
    1.2   Evaluate $RC_j$ at $\pi^*$ for $j \in U$;
    1.3   If $\underset{j \in U}{Max}\{RC_j\} \leq 0$, do:
          1.3.1 Record the optimal weights, peers and slacks for DMUt;
          1.3.2 Remove DMUt from $U$ and go to Step 1.1;
    1.4   If $\underset{j \in U}{Max}\{RC_j\} > 0$, do:
          1.4.1 Compute $R_j, j \in U$ and $TU$;
              1.4.1.1 If $|TU| > 1$, go to Step 1.4.2;
              1.4.1.2 If $|TU| = 1$ and $\underset{j \in U}{ArgMax}\{R_j\} = DMUf$, record the optimal
                      weights for DMUf; Move DMUf to set GEN; Go to Step 1.1;
```

```
        1.4.2 Resolve the tie in $ArgMax$ using the closed-form solution. Identify
              one P-K efficient unit, $DMUq \in TU$ and record the weights for it;
            1.4.2.1 Move DMUq to set GEN; Go to Step 1.1;
End Procedure.
```

8.3.3 Description of BuildHull and standard algorithm

The procedure to solve the input-oriented CRS model using BuildHull is similar to the version in Dula (2010) except for the initialisation part and the subroutine to resolve ties. Note that Dula (1998, 2010) uses an alternative ratio R_j^D to identify extreme-efficient units in phase-1. For the sake of completeness, we describe our implementation of the BuildHull algorithm and the standard two-phase algorithm below.

PROCEDURE BUILDHULL

```
Step 0: Initialisation
set $GEN = \{\phi\}$, $U = \{1...n\}$, $TU = \{\phi\}$;
where,
GEN is the set of generators;
$U$ is the set of status unresolved DMUs;
$TU$ is the set of tied units for $ArgMax$ in $R_j^D, j \in U^+$ at $\pi^*$;
    0.1 Evaluate $R_j = \frac{1Y_j}{1X_j}, j \in U$, and let $\underset{j \in U}{ArgMax}\{R_j\} = DMUf$;
    0.2 Move DMUf to set GEN.
End Initialisation.
Step 1: Phase-1 Iteration. While $U \neq \{\phi\}$, do:
    1.1 Select the first DMU from set $U$, DMUt, and solve the phase-1 CRS LP of
        BuildHull for it;
```

```
1.2 Let the optimal weights be $\pi^* = (v^*, u^*)$;
1.3 If the optimal objective function value is = 0, do:
    1.3.1 Remove DMUt from $U$ and move it to set I; Go to Step 1.1;
1.4 If the optimal objective function value is > 0, do:
    1.4.1 Compute $R_j^D = \dfrac{u^*Y_j - v^*X_j}{1X_j}$ at $\pi^*$ for $j \in U^+$ and $TU$;
        1.4.1.1 If $|TU| > 1$, go to Step 1.4.2;
        1.4.1.2 If $|TU| = 1$ and $\underset{j \in U^+}{ArgMax}\{R_j^D\}_{\pi^*} = DMUp$, record the
            optimal weights for DMUp; Move DMUp to set E;
            Go to Step 1.1;
    1.4.2 Resolve the tie in $ArgMax$ using the closed-form solution. Identify
        one P- K efficient unit, $DMUq \in TU$ and record the weights for it;
    1.4.2.1 Move DMUq to set E; Go to Step 1.1;
  End Phase-1 Procedure.
Step 2: Phase-2 Iteration. While $I \neq \{\phi\}$, do:
  2.1 Select the first DMU from set $U$, DMUt, and solve LP-1 for it against units
      in E; Record the efficiency score, weights, slacks and peers;
  2.2 Remove DMUt from I; Go to Step 2.1;
End Phase-2 Procedure.
End BuildHull Procedure.
```

PROCEDURE STANDARD TWO-PHASE ALGORITHM

Step 0: Initialisation

set $U = \{1,\dots,n\}$; $R = \{\phi\}$;

where,

U is the set of status unresolved DMUs;

R is the set of status resolved DMUs;

```
Step 1: Phase-1 of the standard DEA procedure. While $U \neq \{\phi\}$, do:
    1.1   Select the first DMU in $U$, DMUt, and solve LP-1 for it. Record the
          efficiency score and weights for DMUt;
    1.2   Remove DMUt from $U$ and move it to set $R$; Go to Step 1.1;
End Phase-1 procedure.
Step 2: Phase 2 of the standard DEA procedure. While $R \neq \{\phi\}$, do:
    2.1   Select the first DMU, DMUt, from set $R$ and solve the max-slack CRS
          model for it. Record the input and output slacks and peers for DMUt;
    2.2   Remove DMUt from $R$; Go to Step 2.1;
End Phase-2 procedure.
End Standard two-phase algorithm.
```

8.3.4 Limitations of the computational experiments

The three limitations that were discussed for the VRS cases, namely, our choice of the LP solver and programming language, non-implementation of enhancement techniques like EIE, RBE and reoptimization, and the order in which the units are evaluated, also hold true under CRS. In addition, our experimental design under CRS has two further limitations. First, the datasets generated are strictly positive, a feature also observed in Dula's datasets. Second, in all our datasets, the number of outputs was fixed at 1. Considering the various limitations of our experimental design, although GBA solves fewer LP problems than BuildHull and is expected to be computationally superior, our computational results cannot be regarded as providing a categorical view of their relative performances.

8.3.5 Computational results and comparison of algorithmic performances

The computational results of solving the 64 datasets can be seen in Appendix 10. No ties were encountered in solving the 64 synthetic datasets using GBA or BuildHull.

The first obvious and gratifying result is that regardless of the characteristic of the DEA dataset, GBA solves the CRS model faster than BuildHull. As expected, both GBA and BuildHull are consistently faster than the standard two-phase procedure. As in the VRS cases, there is a gradual decrease in the computation advantage of GBA and BuildHull over phase-1 of the standard algorithm as density increases.

In addition it was noted that the computational saving margin of GBA over BuildHull is less pronounced while solving CRS models when compared to the VRS models. An obvious reason for this could be that although GBA solves *n* -1 LPs as against BuildHull that requires solving 2*n* - *k* LPs, unlike BuildHull, GBA requires computation of RC_j and R_j values for the units in set U upon solving an LP. Note that under CRS when the dataset is strictly positive, GBA requires solving $(n-1)$ LPs, each of size $m\times(g+1)$ where, g is the number of generators identified at a particular iteration. If the dataset is not strictly positive, the number of columns in the GBA LP increases by one to include the input-output vector of DMUt. Upon solving an LP, *at worst,* GBA requires computing $(n-g)$ RC_j values, and if the duality condition is not satisfied, $(n-g)$ R_j values. Assuming that the total number of generators in the dataset is k, the number of times the RC_j values have to be computed is $(n-1)$ and the number of times the R_j values have to be computed is k. The number of times *ArgMax* has to be computed is $n+k-1$. Assuming that all the mathematical operations involve numbers with d digits, the time complexity of computing RC_j and R_j values is $O(nm^2d^2)$ and $O(nm^2d^4)$ respectively, while the time complexity of the *ArgMax* operation is $O(n)$. Under VRS, since the $(n-g)$ R_j values and the associated *ArgMax* need not be computed k times, the computational saving of GBA over BuildHull is more pronounced than CRS although GBA outperforms BuildHull consistently in both cases.

As in the VRS case, we fixed dimension, density and cardinality two at a time and varied the third factor to note its impact on the computational time taken by the three algorithms. Also, for each case, we provide a chart comparing GBA with BuildHull alone to discriminate their relative performances better.

In the first scenario, the dimension and cardinality are fixed at one end of their ranges, i.e., at 5 and 2500 respectively. The impact of density on the performance of GBA, BuildHull, the standard algorithm are graphically illustrated in charts 8-15 and 8-16 below.

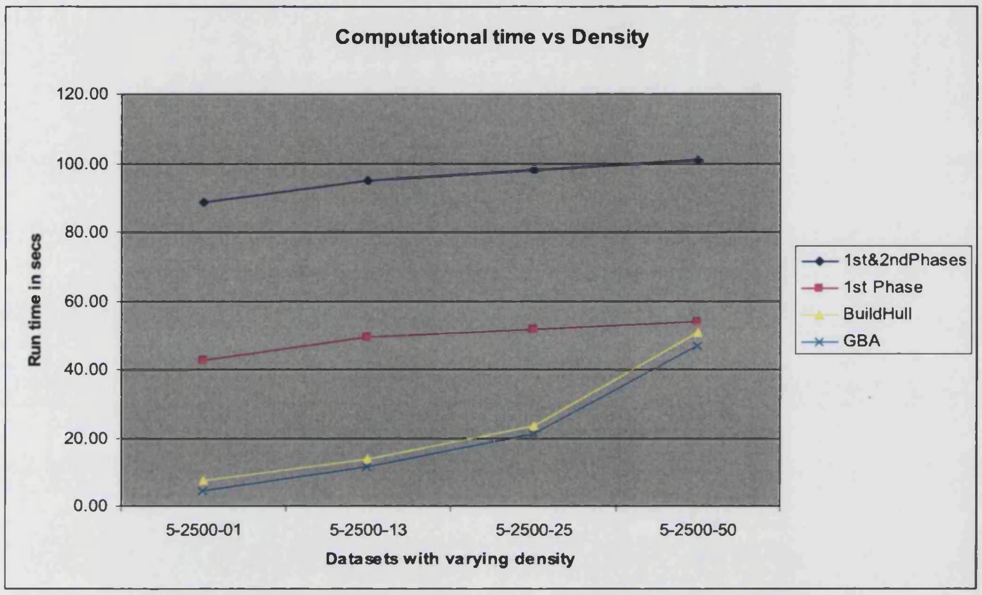


Chart 8-15 : Computational Time versus Density

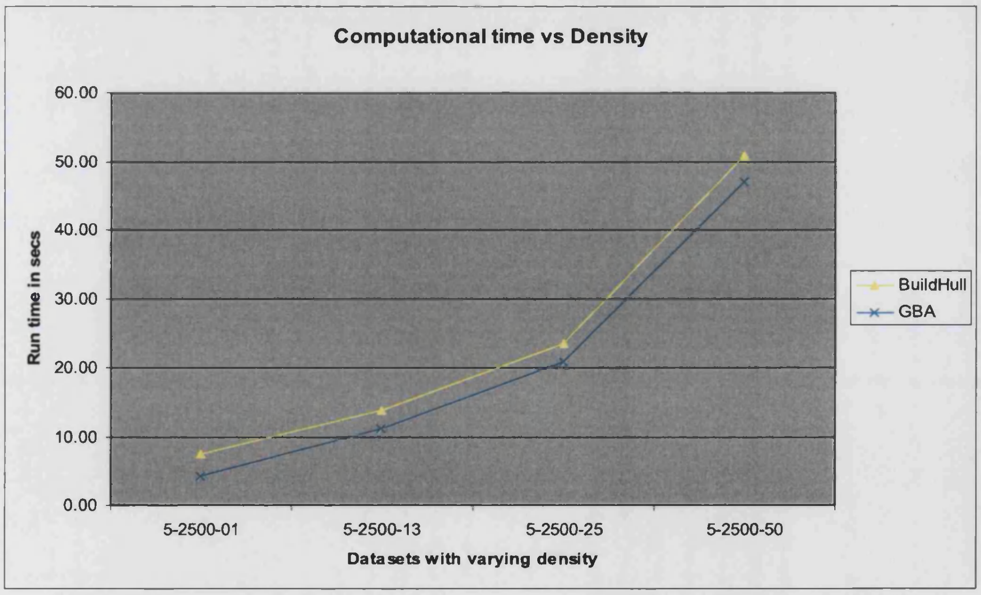


Chart 8-16 : Computational Time versus Density (GBA vs. BuildHull)

As seen in charts 8-15 and 8-16, GBA solves the problem consistently faster regardless of the density of the dataset. BuildHull also performs better than the standard algorithm regardless of the density of the dataset. The same trend is also observed at the other extreme where the dimension is fixed at 20 and cardinality at 10000, and the density is varied from 1% to 50%. The corresponding charts can be seen below.

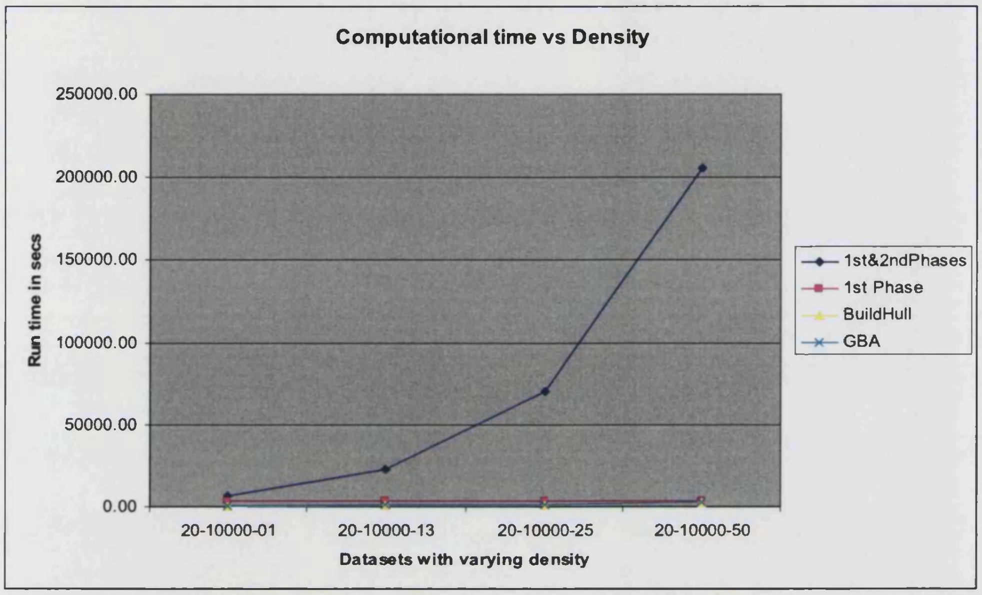


Chart 8-17 : Computational Time versus density

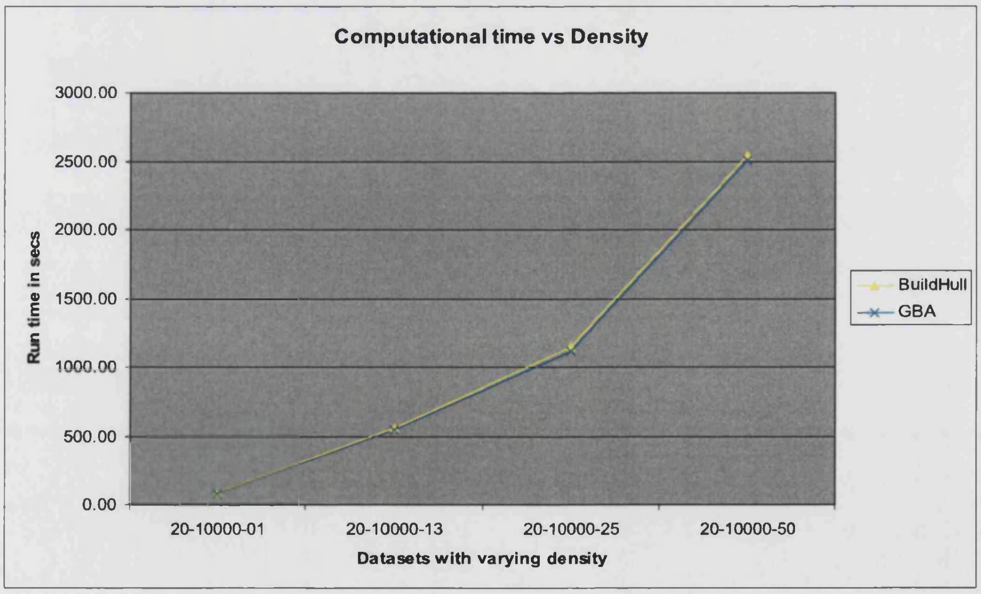


Chart 8-18 : Computational Time versus density (GBA vs. BuildHull)

To discriminate the time taken by the three algorithms better, the following chart shows the computational performance in log seconds for the same scenario.

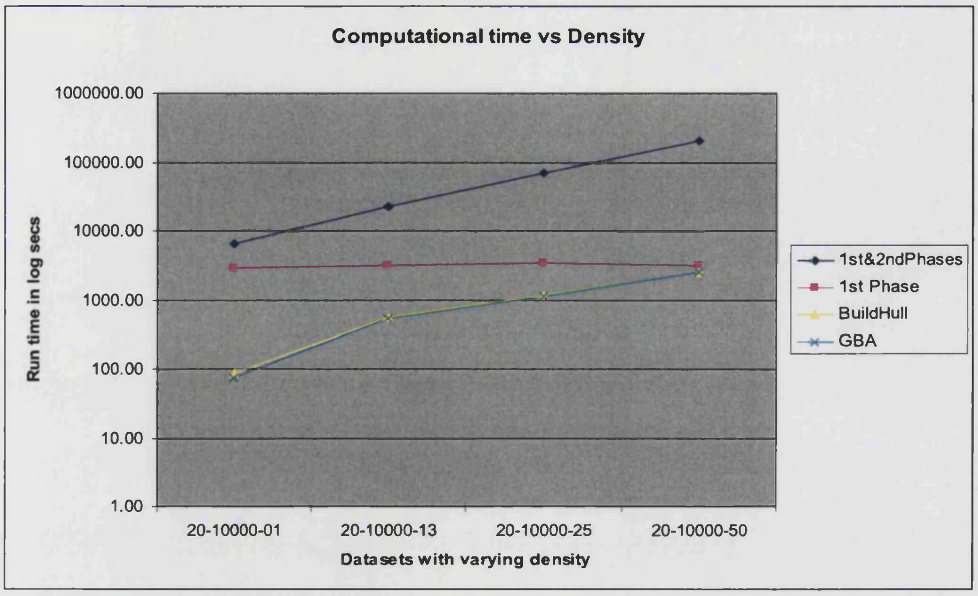


Chart 8-19 : Computational Time (in log seconds) versus density

In the second scenario, the density and cardinality values are fixed at one end of their ranges, i.e., at 1% and 2500 respectively, and the impact of dimension at values 5, 10, 15 and 20 on the performance of GBA, BuildHull, and the standard algorithm are graphically illustrated in the below charts.

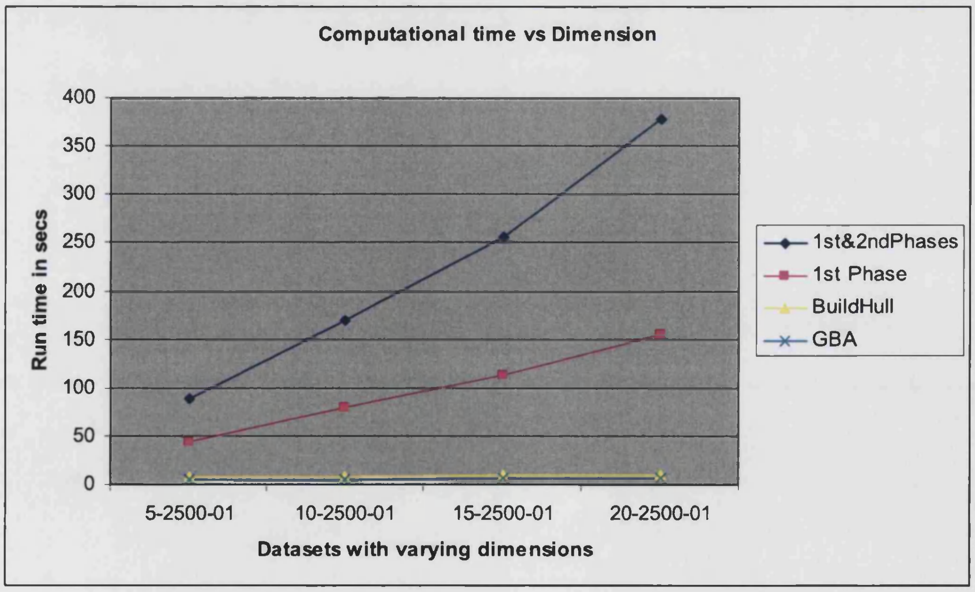


Chart 8-20 : Computational Time versus dimension

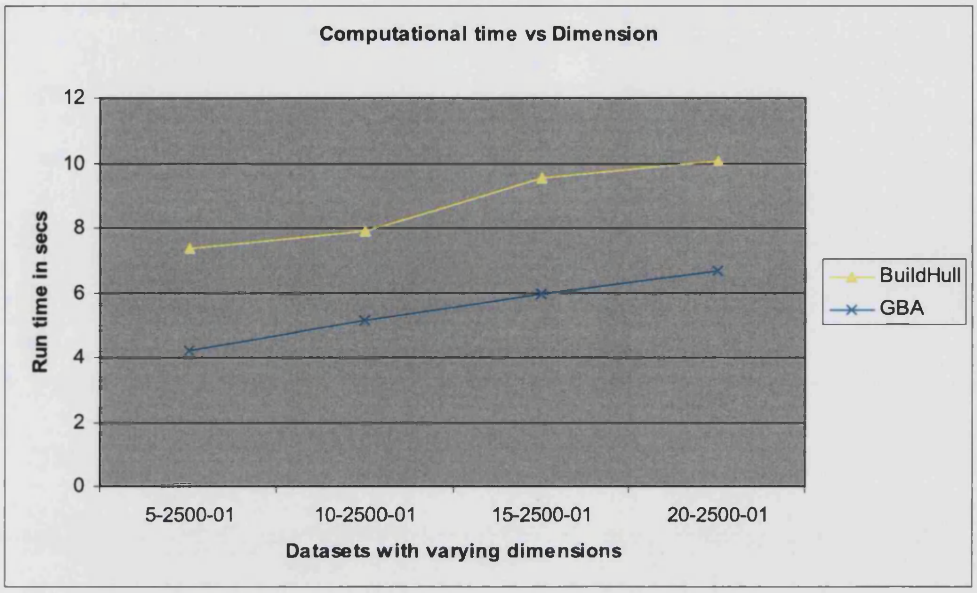


Chart 8-21 : Computational Time versus dimension (GBA vs. BuildHull)

As seen from charts 8-20 and 8-21, GBA solves the dataset consistently faster than the other two regardless of the dimension value. BuildHull also performs better than the standard algorithm in all cases. The same trend is also observed at the other extreme of the ranges wherein the density is fixed at 50% and cardinality at 10000 and the dimension is varied from 5 through 20. The corresponding graph can be seen in charts 8-22 and 8-23 below.

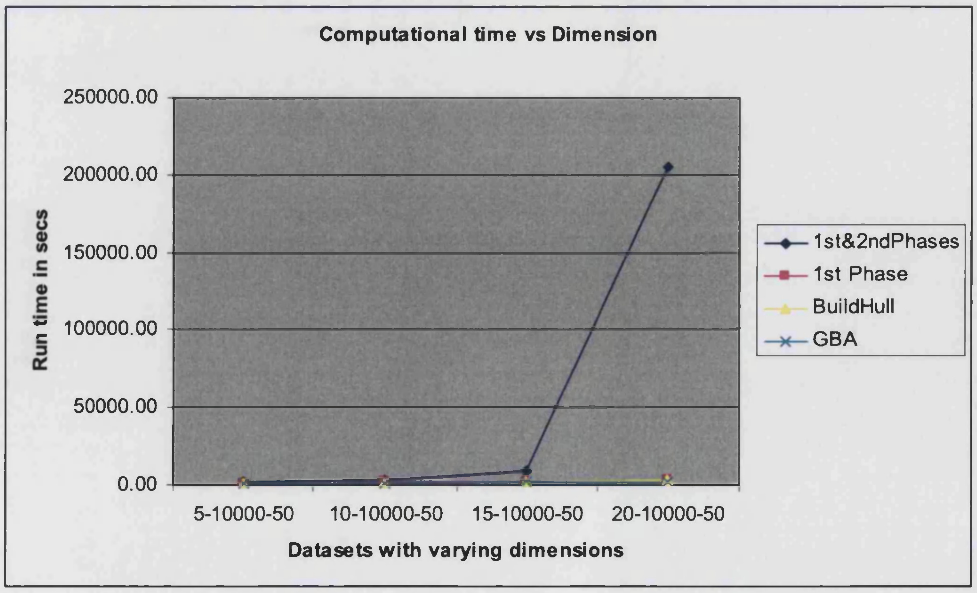


Chart 8-22 : Computational Time versus dimension

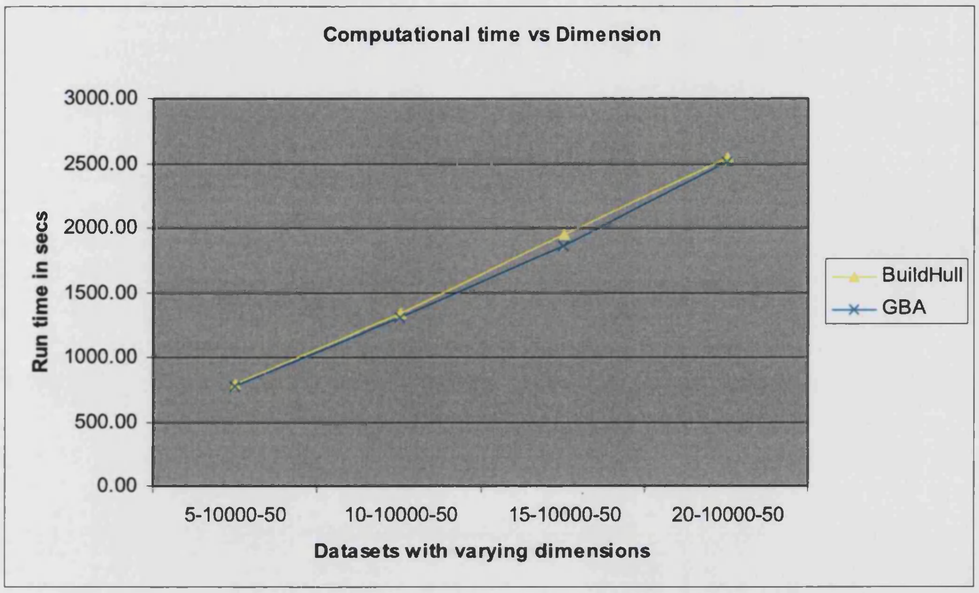


Chart 8-23 : Computational Time versus dimension (GBA vs. BuildHull)

To discriminate the time taken by the three algorithms better, the following chart shows the computational performance in log seconds for the same scenario.

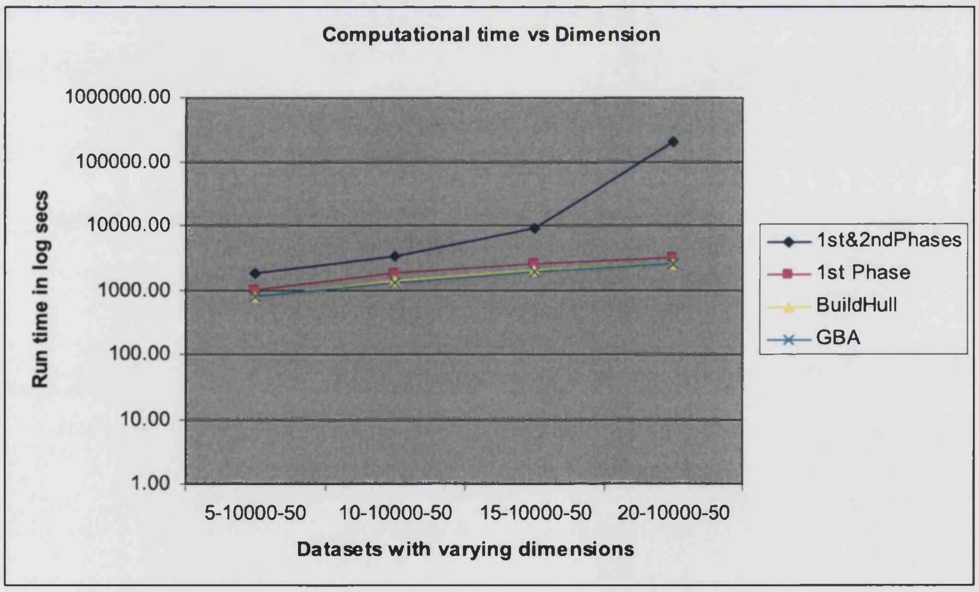


Chart 8-24 : Computational Time (in log seconds) versus dimension

In the third scenario, the density and dimension values are fixed at one end of their ranges, i.e., at 1% and 5 respectively, and the impact of cardinality at values 2500, 5000, 7500 and 10000 on the performance of GBA, BuildHull, the standard algorithm are graphically illustrated in the below charts.

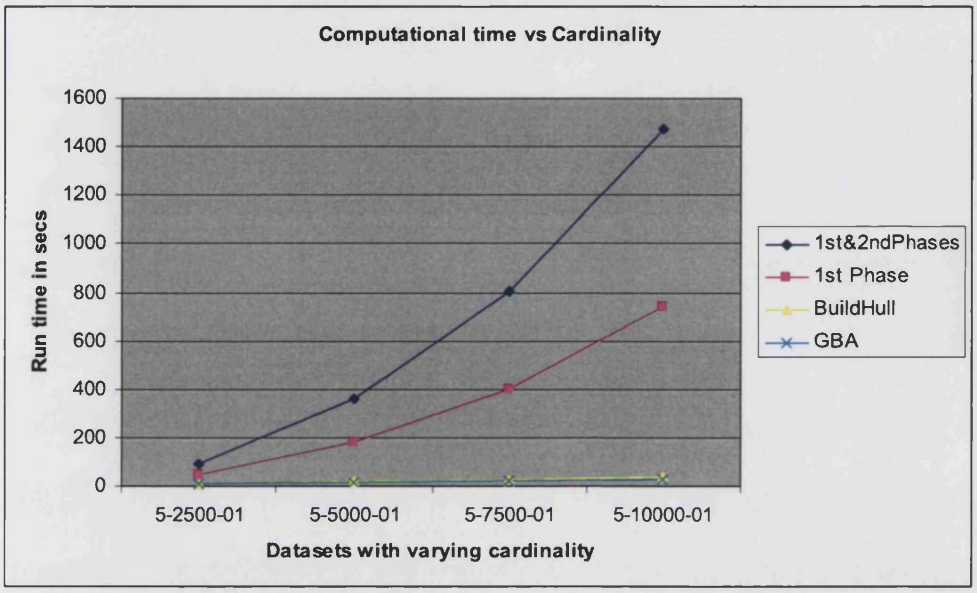


Chart 8-25 : Computational Time versus cardinality

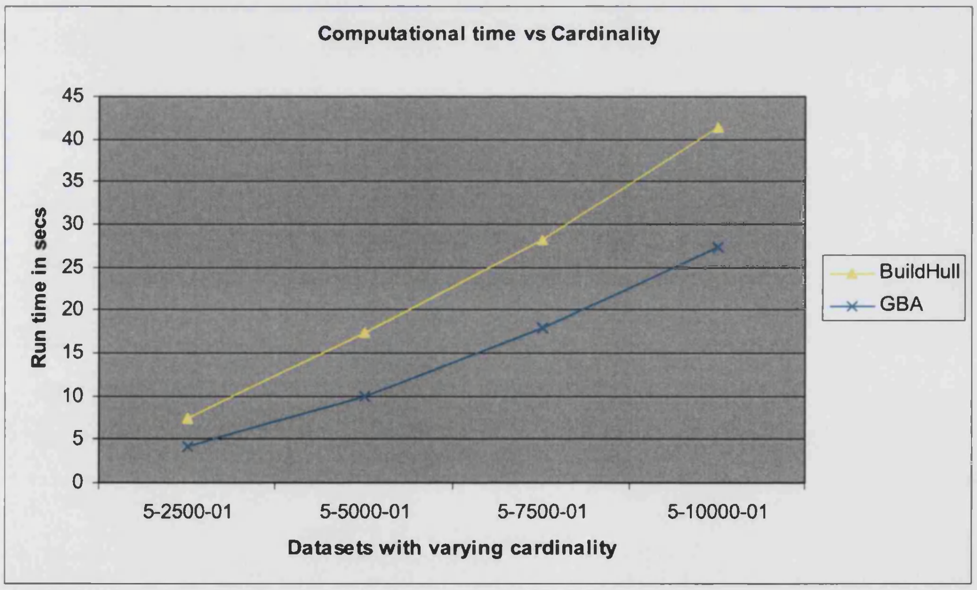


Chart 8-26 : Computational Time versus cardinality (GBA vs. BuildHull)

As seen from charts 8-25 and 8-26, GBA solves the dataset consistently faster than the other two regardless of the cardinality value. BuildHull also performs better than the standard algorithm in all cases. To discriminate the time taken by the three algorithms better, the following chart shows the computational performance in log seconds for the above scenario.

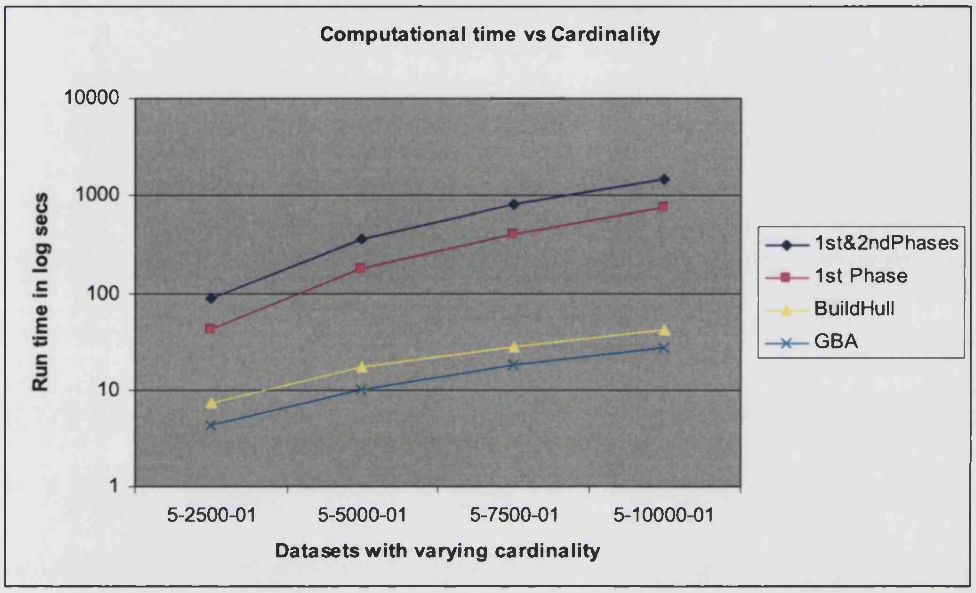


Chart 8-27 :Computational Time (in log seconds) versus cardinality

The same trend is also observed at the other extreme when the density is fixed at 50% and dimension at 20 and the cardinality is varied from 2500 through 10000. Charts 8-28 and 8-29 illustrate this.

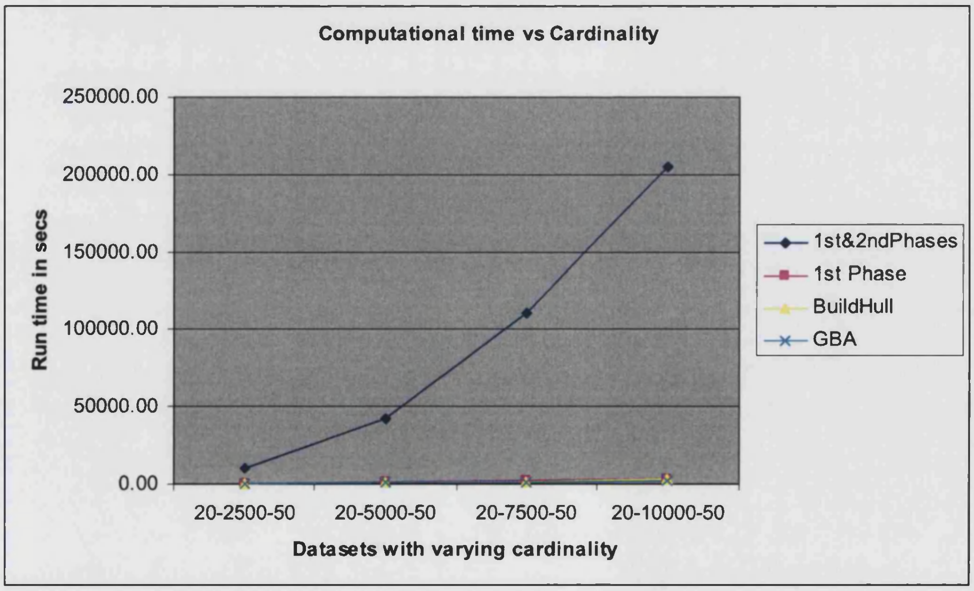


Chart 8-28 : Computational Time versus cardinality

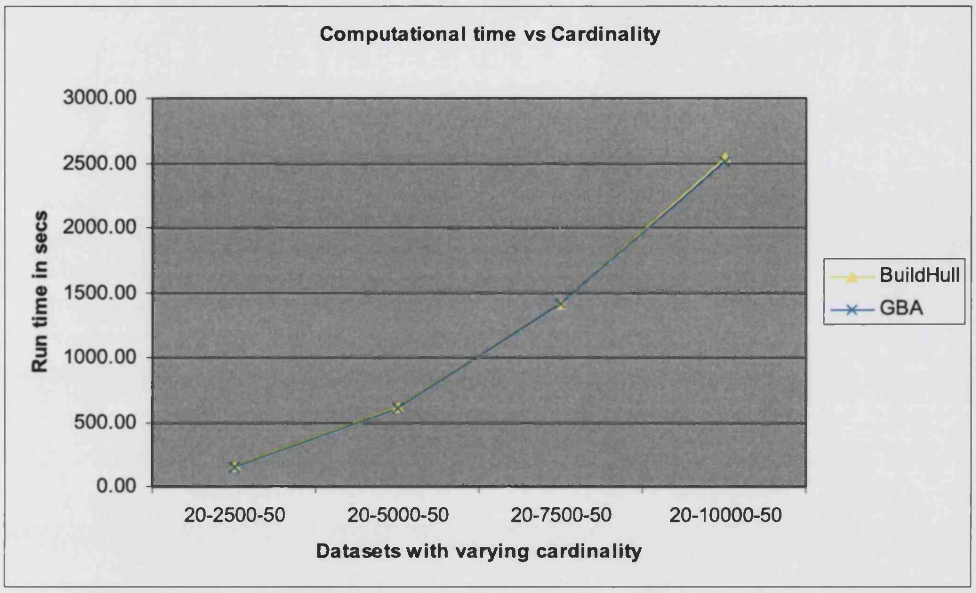


Chart 8-29 : Computational Time versus cardinality (GBA vs. BuildHull)

Once again to discriminate better the computational time taken by the three algorithms, the following graph shows the computational performance in log seconds for the above scenario.

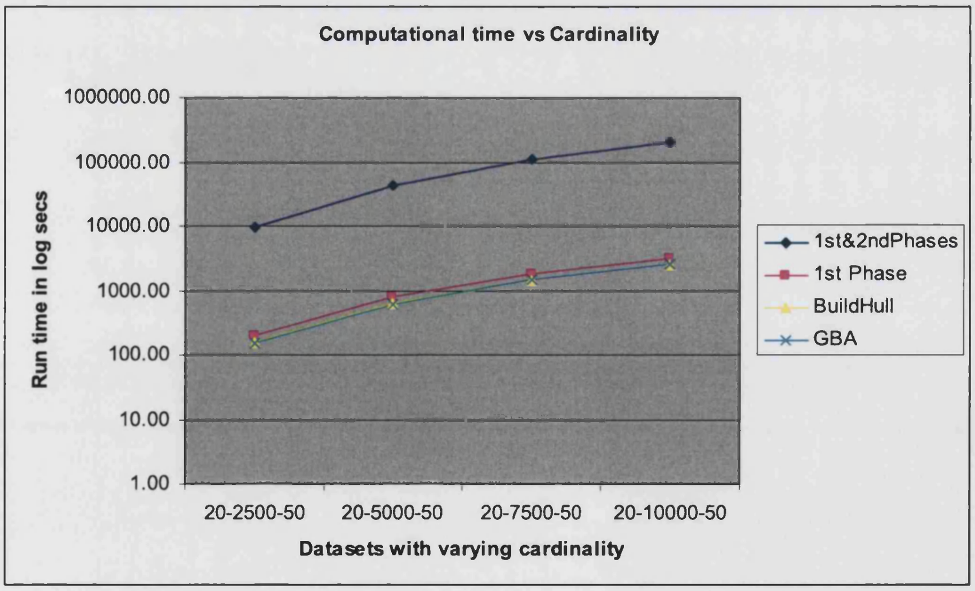


Chart 8-30 : Computational Time (in log seconds) versus cardinality

8.4 Conclusion

In this chapter, we presented the computational results of solving the output-oriented VRS model, additive VRS model and input-oriented CRS model under different characteristics of a dataset to compare the performance of GBA against Dula's BuildHull algorithm and the conventional solution procedure. We also examined the individual impact of each of the characteristic of a DEA dataset, namely, the cardinality, dimension and density, on the performance of the three algorithms under CRS and VRS assumptions. In conclusion, bearing in mind the limitations of our experimental design, we confirmed using our experiments with real and simulated datasets that GBA is consistently faster over BuildHull and the standard two-phase algorithm for solving oriented VRS and CRS models with the computational saving more pronounced in the VRS case.

In the case of additive models, the BuildHull algorithm is slightly faster, although not consistently, than GBA while processing datasets with dimension values of over 15, although such extreme values for dimension are not common in real datasets. While solving additive models, we provided two promising alternative ways of improving the performance of GBA. Using preliminary experiments we showed that both the alternative approaches have distinctive advantages over BuildHull with one of them consistently outperforming BuildHull at higher dimensions. However, both these alternative approaches to GBA are at a developing stage and additional research is required to enable GBA to solve datasets using the additive models consistently faster than BuildHull for any characteristic of the dataset.

In chapter 9, we discuss the avenues for future research.

9 DIRECTIONS FOR FUTURE RESEARCH

Before we discuss directions for future research, we summarize the contributions of the thesis. The two major contributions are theoretical in nature and are:

1. The thesis develops an alternative algorithm for processing datasets using standard DEA models and important extensions of the basic models such as weight restricted models;
2. The thesis develops closed-form solutions to construct strictly positive weights for the extreme-efficient units without explicit weight restrictions;

Extensive computational testing verifies and validates the new algorithm and demonstrates that upon a naïve implementation of the competitive algorithms, GBA consistently outperforms the others.

We now present directions for future research. These are listed under three themes:

1. enhancing the computational experiments,
2. extension of GBA to handle negative data and the Free Disposal Hull (FDH) models, and
3. extension of the closed-form solution approaches.

9.1 Enhancing the computational experiments

The most obvious question to ask here is what are the limitations of GBA? If we set some arbitrary time limit to computation time – say one hour – what size problems can be solved in that time? Size could be taken as made up of three parameters, viz., density, dimension and cardinality and various combinations of the three can be tried. Obviously the answer would vary according to the dataset, the hardware and the optimizer used, but it would be interesting to see if one could solve problems with even low density and dimensionality, but 100,000 DMUs for example. The DfES wanted at one stage to solve a DEA model for each pupil. However impractical such a DEA model may be, would one be able to solve it in real time?

Unlike Dula we carried out our experiments without programming DEA specific enhancements and re-optimization techniques. Also we did not use CPLEX which is the industry standard. For comparative purposes it does not seem crucial but it would be interesting to see what difference it makes to the GBA run times if these are used.

Two further lines of research were mentioned in section 8.1.3 in connection with choosing the next unit to be evaluated. In particular, we mention that when the evaluation of DMUt is complete, one can choose the unit in U that achieved the second highest RC_j or R_j value as the one to be evaluated in the subsequent iteration. Also, if DMUt is identified as a non-generator, one could use the optimal basis to check if the same basic variables give a feasible solution in the envelopment form model for any other unit in U. We have developed some preliminary theoretical results for this problem, building on dominance rules. It will be useful to expand on these results and also test the two approaches to select the next unit to be evaluated.

In the Data Generating Process used for our computational experiments under CRS, two further limitations in terms of data positivity and unary output were highlighted. These two limitations could be relaxed in future works.

Finally while solving the additive models, it was noted that BuildHull outperforms GBA for datasets with extreme dimension values. Two alternative approaches of modifying GBA to solve the additive VRS model were presented. Using preliminary results we showed that both of them have distinctive advantages over BuildHull with one of them consistently outperforming BuildHull at higher dimensions. However, it was also noted that at smaller dimensions, the original GBA approach is consistently faster than all the options. Hence, additional work is required to develop an unifying approach to GBA to solve the additive models that can consistently outperform BuildHull for any data characteristics.

9.2 Extension of GBA to handle negative data and the FDH models

First, given the swell of literature devoted to dealing with negative data (see, Emrouznejad et al, 2010 for a recent review), it will be useful to extend the application of GBA to handle negative data as well. This can be pursued in two ways and are listed below:

i. examining the translation invariance property of the various DEA models and identifying cases where GBA can or cannot be applied;
ii. examining whether GBA can handle the alternative DEA models developed for this purpose.

There are cases where negative data does not cause a problem as some DEA models are invariant to data translation, i.e., the solution is unaffected upon adding a large enough constant to the data thereby translating it to the positive orthant. This data translation technique is applied to all the models whose solution are invariant to this translation. In particular, we know that the additive VRS model is translation invariant w.r.t inputs and outputs (Ali & Seiford, 1990), and the oriented VRS models are partially translation invariant if only inputs (outputs) have negative data while solving the output (input) oriented model. Hence it will be possible to apply GBA upon translation to these models.

While addressing the issue of negative data that arises in many applications (such as net profit/loss or negative returns if treated as outputs), various alternative DEA models have been developed in the literature. A partial list could include the modified slacks-based model of Sharp et al (2006), range directional model of Portela et al (2004), and semi-oriented radial measure model of Emrouznejad et al (2010). It will be useful to investigate whether GBA can handle such models.

Second, it will be useful to extend the application of GBA to solve the Free Disposal Hull (FDH) models introduced in Deprins et al (1984). FDH models are similar to the oriented VRS models but with the additional requirement that the efficiency evaluations are effected from only the actually observed units. These models are typically solved using sorting algorithms or MILPs (see, Cooper et al, 2000). GBA cannot be readily applied to solve the FDH models partly because of the 'Principal Theorem' of Thrall (1999) which shows that there are generators which do not satisfy Characteristic 3 in 4.1. However, Agrell and Tind (2001) have developed a linear programming equivalent of the corresponding MILP models. It will be useful to examine whether GBA can be applied to solve it efficiently.

9.3 Extension of the closed-form solutions

Regarding the closed-form solutions to break ties and construct strictly positive multiplier values, there are four avenues for future work. First, it will be satisfying to extend the closed-form solutions such that only the zero valued multipliers are changed to some positive value. Preliminary analysis indicate that for the VRS case, such an amendment is straight-forward and the current version of the closed-form solution can be applied without any modification wherein (in place of the unary weights) only the zero valued multipliers are changed to some positive value. However, such an extension is complicated in the CRS case.

The second extension would be to restrict the range of the multiplier values obtained from the closed-form solutions. The closed-form solutions in the current form produce a set of strictly positive multiplier values without any regard to the range of the values it ends up with.

A closely connected point is to enable the closed-form solutions such that they satisfy some pre-specified restrictions on the set of multipliers. Preliminary analysis indicates that some commonly applied restrictions can be accommodated by the closed-form solutions under the VRS case. This is because π^* will automatically satisfy the restrictions which would have been incorporated in the LP. All we need is some positive weight vector π^r which also satisfies the restrictions and synthesise π^* with π^r instead of the unary weights. However, yet again, this extension is not straight-forward in the closed-form solutions employed in the CRS case.

Lastly, it was mentioned in chapter 7 that the necessary condition for the closed-form solutions to produce strictly positive multipliers is that the generator in question achieves the unique maximum R_j (for CRS) or RC_j value (for VRS) at π^*. It will be useful to relax this condition to generalise the application of our closed-form solutions.

Appendix 1 – Shephard's (1970) output distance function and related Debreu-Farrell measure

Similar to representing the production technology $T=\{(X,Y)|\, Y\geq 0 \textit{ can be produced from } X\geq 0, X\neq 0\}$ by the input sets, one can also represent it using the output sets $P(X)$. $P(X)$ can be represented as $P(X)=\{Y:(X,Y)\in T\}$, which for every X have output isoquants $I(X)=\{Y:Y\in P(X), \lambda Y\notin P(X), \lambda>1\}$ and output efficient subsets given by $E(X)=\{Y:Y\in P(X), Y'\notin P(X), Y'\geq Y\}$ and the three sets satisfy $E(X)\subseteq I(X)\subseteq P(X)$.

Shephard's (1970) output distance function provides another functional representation of the production technology under CRS. The output distance function is $D_O=\min\{\lambda:(Y/\lambda)\in P(X)\}$. For $Y\in P(X), D_O(X,Y)\leq 1$ and for $Y\in I(X), D_O(X,Y)=1$. Given standard assumptions on T_C described in chapter 1, the output distance function $D_O(X,Y)$ is non-increasing in X and is non-decreasing, homogeneous of degree +1, and convex in Y.

The Debreu-Farrell output-oriented measure of technical efficiency TE_O is simply the value of the function $TE_O=\max\{\phi:\phi Y\in P(X)\}$ and it again follows that $TE_O(X,Y)=\dfrac{1}{D_O(X,Y)}$[56]. For $Y\in P(X)$, $TE_O(X,Y)\geq 1$ and for $Y\in I(X)$, $TE_O(X,Y)=1$. Similar to the properties of the Shephard's (1953, 1970) distance functions, we can state three properties of the Debreu-Farrell measures following Russell (1988, 1990), namely,

1. $TE_I(X,Y)$ is homogeneous of degree -1 in inputs and $TE_O(X,Y)$ is homogeneous of degree -1 in outputs.
2. $TE_I(X,Y)$ is weakly monotonically decreasing in inputs and $TE_O(X,Y)$ is weakly monotonically decreasing in outputs.

[56] Given that $TE_O(X,Y)\geq 1$, $TE_O(X,Y)$ is sometimes referred to as the radial measure of technical inefficiency of activity (X,Y). Hence, technical efficiency in the output-oriented case is given by $\dfrac{1}{TE_O(X,Y)}$ and is equal to $D_O(X,Y)$.

3. $TE_I(X,Y)$ and $TE_O(X,Y)$ measures are invariant to the unit of measurements of the input and output factors.

Appendix 2 – July 2006 report to the DfE on Dula's work

Report 2 under contract 2 due on 1 July 2006

The best method for computing DEA efficiency scores for large data sets – Dula's work

Introduction:

The first report entitled 'A Review of literature on methods to speed up the solution of large scale data sets in DEA (LSDEA)' outlined three major approaches, viz.,

i) Pre-processing and LP accelerator methods developed since 1993 by Ali [1, 3]

ii) Hierarchical Decomposition by Barr and Durchholz [2] published in 1997 and

iii) Dula's work ([4] to [8]) on finding all efficient DMUs first published as a Mississippi University report in 1998 and in various journals from 2001.

Among all these, Dula's work is by far the best. For really large data sets it is the only viable approach developed so far. For that reason this report concentrates on Dula's work. We give a non-technical overview; an introduction to the technical side of Dula's work with diagrams and our view of the relevance of Dula's work for the DfES in the light of our attempts at implementing Dula's method.

Outline of the report:

a) **An overview of Dula's approach**
b) **Technical details of Dula's algorithm BuildHull:**
   1. **Main concepts and definitions**
   2. **Starting with at least one extreme efficient DMU**
   3. **A typical iteration – Finding an extreme efficient DMU or discarding a DMU**
   4. **Output from Dula's work**
c) **Relevance of Dula's work for the DfES**

a) An overview of Dula's approach

We begin with a bird's eye view of what Dula's method is and explain why we think it is the best. This section is deliberately kept non-technical.

Dula's method works in two stages. Irrespective of the DEA model used (oriented or additive model under variable, constant, increasing or decreasing returns to scale), he sets out to find all the extreme efficient DMUs in stage 1. Then in the second stage the inefficient ones are scored through normal LPs in which only the extreme efficient DMUs are included. The main advantage is that the size of the LP remains relatively small.

Adapting Dula's terminology we shall call the extreme efficient DMUs in a DEA model the generators. See figure 1 where the generators are A, B, C and D. Note that not all the DMUs on the DEA frontier are generators, e.g., DMUs E, I, and J in figure 1. In a constant returns to scale DEA model in which input minimisation is the goal, E, I and J will have efficiency score of 1 but none of them is a generator. E and I have positive slacks while J is on the line segment joining C and D. Roughly speaking the generators consist of all those DMUs on the efficient frontier which can appear as peers in explaining the inefficiency of others and cannot be expressed by a convex linear combination of other DMUs. So, excluding a generator alters the efficient frontier and can lead to a change in the efficiency status of some DMU while excluding a DMU which is not a generator leaves the frontier and the efficiency status of all other DMUs unchanged.

The number of generators in any data set is relatively small. In an application to the state of Texas' southwest district banks containing 8748 banks and nine factors (6 inputs + 3 outputs) Barr and Durchholz [2] report that no more than 1% were efficient. The DfES data set with 9 factors (8 inputs + 1 output) had only 111 generators out of 1258 schools; while in the current data set with 10 factors, out of 1200 non-sixth form (NSF) and 1653 sixth form (SF) schools, only 188 NSF and 232 SF schools are extreme efficient – or generators.

The main problem in solving LSDEA is that if the number of schools is n, we need to solve n LPs of size mx(n+1) where m is the number of factors (inputs + outputs). Restricted basis entry (which eliminates any school identified as inefficient from further computations) and early identification of efficient schools can reduce the size of the largest LP solved as we come closer and closer to n. For example, for the last school analysed, the LP will have only the generators and the last school, the columns of all the inefficient ones having been eliminated on the way. So the largest LP solved is mx(n+1) and the average size is greater than mx(.5n). The beauty of Dula's method is that the largest LP it has to solve is mx(k+1) where k is the number of generators. As k seems not to exceed 15%, the saving in total time is significant and increases rapidly as n, the number of schools, goes up. Moreover, for really large data sets (over 10, 000 schools and if pupil-level model is required even 50, 000 DMUs?) this seems to be the only viable alternative. We discuss this aspect in section c), taking in to account the extra work required by Dula's method to score inefficient DMUs in the second stage.

b) Technical Details of Dula's algorithm BuildHull

1. Main concepts and definitions

Dula's algorithm is called BuildHull. We describe it here for an input minimisation DEA model under the constant returns to scale assumption. BuildHull is an algorithm for finding all the generators. It starts with one generator, easily found by known heuristics, and at each subsequent iteration either finds a new generator or decides that the DMU under consideration is not a generator and can be discarded. Suppose that at a typical iteration there are l generators indexed by 1 to l and collectively denoted by set J_l. The space enveloped by the efficient frontier generated by these l DMUs is called the partial hull, denoted P_l. (See diagram 3 where C is the only DMU in set J_l.) What we are after is a set J_l that generates the *full hull*, i.e., where all n DMUs are within P_l.

So the job is to decide for each DMU whether it is a generator or not. Given l generators we consider a DMUt not belonging to J_l at the next LP iteration. If the input/output vector a^t of DMUt belongs to the partial hull P_l, then t cannot be a generator. We can discard it and consider another DMUt from the ones not classified so far.

When $a^t \notin P_l$, the dual LP solution delivers a hyperplane h that separates a^t from P_l. A theorem proves that among all those DMUs which are outside P_l and on the same side of h as DMUt, there is at least one generator. Modifying optimal LP dual values π^* the algorithm then creates a set of input/output weights (designated π^{bar}) which finds a generator. The process is then repeated with the augmented set J_l and another DMUt from the ones not classified so far.

The most innovative part of the algorithm is the way it decides if DMUt is in the partial hull or not. To achieve this a new DMU is synthesised from set J_l.

Its inputs and outputs relate to the average of inputs and outputs of the DMUs in set J_l. We shall call it the average DMU and designate its input/output vector as av^l.

Definitions:

Let the input/output vector for DMUj be written as:

$a^j = \begin{bmatrix} -X_j \\ Y_j \end{bmatrix}$ where X is the vector of m_1 inputs and Y the vector of m_2 outputs.

The average DMU used for each LP run in BuildHull is defined by the vector:

$av^l = -\frac{1}{|J_l|}\sum_{j \in J_l}(a^j - e)$ where e is an (mx1) vector of 1's.

Vector av^l helps to identify any vector a^t outside P^l. Barring minor technical considerations one could say that the LP solved at each iteration tries to find the minimum multiple α required to express the input/output vector a^t (of DMUt) as a non-negative linear combination of the input/output vectors in J_l and av^l. The Primal and Dual LP solved at each iteration are:

Primal LP (P)	Dual LP (D)
$Min\,\alpha$ $av^l\,\alpha + \sum_{j \in J_l} a^j \lambda_j \geq a^t$ $\alpha \geq 0, \lambda_j \geq 0$	$Max\ \pi a^t$ $s.t.\ \pi a^j \leq 0, j \in J_l$ $\pi(av^l) \leq 1$ $\pi \geq 0$

(Note that row and column vector multiplications such as πa^t are assumed to be for conformable vectors.)

Expressing the optimal primal and dual values by superscript *, we have the following important results.

Lemma 1: $a^t \varepsilon P^l$ if and only if $\alpha^* = 0$. (In words, DMUt is in the partial hull and therefore not a generator if and only if $\alpha^* = 0$.)

Lemma 2: If $\alpha^* > 0$, the hyperplane h defined by $\pi^* a^j = 0$ separates P^l from a^t, ensuring that $\pi^* a^t > 0$.

It is instructive to note why Lemma 2 holds. The strong duality theorem of LP states that the objective function value of the primal and dual LP are equal, leading to $\alpha^* = \pi^* a^t$. As $\alpha^* > 0$, so is πa^t. But any point in P^l has to satisfy the dual LP conditions $\pi^* a^j \leq 0$. Hence π^* delivers a separating hyperplane h.

Before introducing Dula's algorithm we need a few more definitions.

1. $\hat{\pi}$ = a constant vector with value 1's for the m_1 input factors and 0's for the m_2 output factors. For example, in a 2-inputs, 1-output DEA problem, $\hat{\pi} = [1, 1, 0]$
2. Set J^+ consists of all j such that $\pi^* a^j > 0$
3. $\eta^* = Min \dfrac{\pi^* a^j}{\hat{\pi} a^j}, \ s.t. \ j \in J^+$
4. $\pi^{bar} = \hat{\pi} - \left(\dfrac{\pi^*}{\eta^*}\right)$
5. argmin = the index of the DMU that gives minimum η^*

To illustrate the algorithm we have created a 2-input, 1-output example with 8 DMUs (denoted A to H). The model chosen is an input minimisation model under constant returns to scale. See figure 2 for the data of the model and the production frontier in the 2-input space with unit output level.

2. Starting with at least one efficient DMU

This is the first initialisation step of BuildHull. Dula's algorithm can be started with any number of generators or even without any. However, it is easy to find some generators by well known heuristics, obviating the need to use an LP for initialisation. Essentially any set of non-negative weights π such that the ratio of the weighted sum of outputs to weighted sum of inputs is one for only one DMU (the generator) and less than one for all the others, will do. In part d) we briefly describe our work on finding several generators by heuristics.

The data for our example is shown in figure 2 which also shows a plot of the production possibility set enveloped by the production frontier for this data set. In figure 3, C has been identified as a generator by using weights 1 and 2 respectively for the two inputs and weight 7 for the output. This provides a starting point for applying BuildHull for the example in figure 2. The partial hull determined by C is illustrated in figure 3.

3. A typical iteration – Finding a generator or discarding a non-generator DMU

We describe all the iterations needed for the small example in order to illustrate various possibilities. The reader can skip this section when s/he feels comfortable with how the method works.

Iteration 1: Step 2 of BuildHull

We start with J_l = {C}; DMUt = D.

From the formula: $av^l = -\frac{1}{|J_l|}\sum_{j \in J_l}(a^j - e)$ we get :

$$av^l = \begin{bmatrix} 5 \\ 2.5 \\ 0 \end{bmatrix}; d^t = \begin{bmatrix} -8 \\ -1 \\ 1 \end{bmatrix};$$

Figure 4 shows these two vectors. (Strictly speaking what is shown in all these diagrams is the $-X_1$, $-X_2$ space which conforms with the normal diagrammatic representations in which the input vector is not multiplied by -1.) Note that av^l is plotted as vector Z.

The LP solved is:

Min α

$$\text{s.t. } \alpha \begin{bmatrix} 5 \\ 2.5 \\ 0 \end{bmatrix} + \lambda^C \begin{bmatrix} -4 \\ -1.5 \\ 1 \end{bmatrix} \geq \begin{bmatrix} -8 \\ -1 \\ 1 \end{bmatrix}$$

$\alpha \geq 0, \lambda^C \geq 0$

The optimal solution gives $\alpha^* = .2$ and $\pi^* = (0, .4, .6)$

D is outside the partial hull. The separating hyperplane h given by $\pi^* d^j = 0$ is the horizontal line shown in figure 4. For D and E, $\pi^* d^j > 0$; for all other points (i.e., A, B, C, F, G and H) $\pi^* d^j \leq 0$. The scaling factor is $1/\eta^* = 1 / -.0222 = -45$ and $\pi^{bar} = [1, 19, 27]$.

As we can see in figure 5, the separating hyperplane is swivelled while keeping it hinged at C. This creates a new hyperplane with coefficients π^{bar}. The half space indicated by $\pi^* d^j > 0$ is now swept with the new hyperplane to find the point furthest away. This is point D in figure 5. So the new DMU we were investigating is a generator. (Note that π^{bar} also provides a set of optimal weights for D.)

Iteration 2

We repeat Step 2 with J_l = {C, D}; DMUt = G.

$$av^l = \begin{bmatrix} 7 \\ 2.25 \\ 0 \end{bmatrix}; \; d^t = \begin{bmatrix} -7 \\ -4 \\ 1 \end{bmatrix}$$

The LP solved is:

Min α

$$\text{s.t. } \alpha \begin{bmatrix} 7 \\ 2.25 \\ 0 \end{bmatrix} + \lambda^C \begin{bmatrix} -4 \\ -1.5 \\ 1 \end{bmatrix} + \lambda^D \begin{bmatrix} -8 \\ -1 \\ 1 \end{bmatrix} \geq \begin{bmatrix} -7 \\ -4 \\ 1 \end{bmatrix}$$

$$\alpha \geq 0, \lambda^C, \lambda^D \geq 0$$

The optimal solution gives: $\alpha^* = 0$ which implies that DMU G is within the partial hull of units C and D and hence not a generator. Since G is contained within the partial hull, it will also be contained in the full hull and can be discarded from subsequent operations. (Note that this doesn't mean C and D are the peers for G in the final analysis.)

Iteration 3

We repeat Step 2 with J_l = {C, D}; DMUt = F.

$$av^l = \begin{bmatrix} 7 \\ 2.25 \\ 0 \end{bmatrix}; \; d^t = \begin{bmatrix} -3 \\ -7 \\ 1 \end{bmatrix}$$

The LP solved is:

Min α

$$\text{s.t. } \alpha \begin{bmatrix} 7 \\ 2.25 \\ 0 \end{bmatrix} + \lambda^C \begin{bmatrix} -4 \\ -1.5 \\ 1 \end{bmatrix} + \lambda^D \begin{bmatrix} -8 \\ -1 \\ 1 \end{bmatrix} \geq \begin{bmatrix} -3 \\ -7 \\ 1 \end{bmatrix}$$

$$\alpha \geq 0, \lambda^C, \lambda^D \geq 0$$

The optimal solution gives $\alpha^* = 0.143$ and $\pi^* = (0.143, 0, 0.571)$

Figure 6 shows the new partial hull. DMU F is outside P_l , leading to the separating hyperplane h given by $\pi^* d^j = 0$ which is the vertical line passing through C. For F, A, and B, $\pi^* d^j > 0$; for all other points $\pi^* d^j \leq 0$.

The scaling factor $1/\eta^* = 1/-0.0475 = -21.05$ and $\pi^{bar} = [4.01, 1, 12.02]$. The swivel and sweep operation using π^{bar} leading to point A is illustrated in figure 7. Instead of fixing the status of F, we have identified A as a generator. So

F would be considered again in another iteration. In this case it will turn out to be a non-generator, but we cannot be sure of that unless we find a partial hull that contains it. However, the status of A is fixed at this iteration and no LP is wasted.

Iteration 4

We repeat Step 2 with J_l = {C, D, A}; DMUt = B.

$$av^l = \begin{bmatrix} 5.67 \\ 3.17 \\ 0 \end{bmatrix}; d^t = \begin{bmatrix} -2.5 \\ -2.5 \\ 1 \end{bmatrix}$$

The LP solved is:

Min α

$$\text{s.t. } \alpha \begin{bmatrix} 5.67 \\ 3.17 \\ 0 \end{bmatrix} + \lambda^C \begin{bmatrix} -4 \\ -1.5 \\ 1 \end{bmatrix} + \lambda^D \begin{bmatrix} -8 \\ -1 \\ 1 \end{bmatrix} + \lambda^A \begin{bmatrix} -2 \\ -4 \\ 1 \end{bmatrix} \geq \begin{bmatrix} -2.5 \\ -2.5 \\ 1 \end{bmatrix}$$

$$\alpha \geq 0, \lambda^C, \lambda^D, \lambda^A \geq 0$$

The optimal solution gives α^* =.0854 and π^* = (0.122, 0.098, 0.634)

Figure 8 shows the new partial hull. DMU B is outside P_l, leading to the separating hyperplane h given by $\pi^* d^j = 0$ which is a line passing through C and A. For B, $\pi^* d^j > 0$; for all other points $\pi^* d^j \leq 0$.

The scaling factor $1/\eta^*$ = - 59.52 and π^{bar} = [8.26, 6.83, 37.74]

Iterations 5, 6, and 7

Since we have all the generators of the production possibility set, running LPs with J_l = {C, D, A, B}; DMUt = F, H, and E (separately) will return an objective function value of 0 as in iteration 2, implying that none of them are generators and can be discarded.

4. Output from Dula's work

In this section we outline what the current version of Dula's algorithm delivers and what we think can be obtained from it after further work.

What BuildHull delivers for generators

At the end of phase 1 BuildHull delivers a complete set of generators. The efficiency values of each one of these is 1 and the weights π^{bar} used to find the generator gives a set of optimal weights at which the generator in question is efficient. These weights are not the extreme weights normally delivered by the usual LP solved for an input oriented CRS model. They are more like the weights produced by PIMSOFT which are also non-extreme, except that PIMSOFT generated weights are biased towards producing as many non-zero weights as possible. In fact it becomes obvious from the proof of the main result (Result 3 in [4]) that the weights π^{bar} are such that the all the input weights are guaranteed to be non-zero.

Can BuildHull deliver non-zero weights for generators?

By combining π^{bar} with the weight vector produced for the same generator by PIMSOFT we can produce new weights for any generator which have more non-zeros in them. For example if PIMSOFT produced some zero weights for r inputs, a straight forward average of PIMSOFT's and BuildHull's weights will provide a valid weight which will have at least r more non-zeros. It seems straight forward to modify the definition of $\hat{\pi}$ so that all the output weights in π^{bar} are positive. If this hunch proves to be correct, we can produce all non-zero weight vectors for each generator.

There is a strong possibility that with more research work a method can be developed from this to produce non-zero weights for all the generators!

What BuildHull delivers for non-generators:

At first sight it may seem that Phase1 of BuildHull delivers nothing for non-generators except for their classification. But there is more. Suppose in a 1000 schools exercise with 100 generators we have found all the generators after analysing 800 schools. We do not know this until the end, but the information obtained for each of the remaining 200 schools was gleaned with the help of an mx101 LP which is very closely related to the mx101 LP we would solve in phase 2. Consider a typical school t from these 200. The peers obtained for this school by solving the normal DEA model will be closely connected to the peers chosen in phase 1. **Again there is a strong possibility that with more research a method can be developed to score the last 200 schools in this example without solving a separate LP in phase 2.**

c) Relevance of Dula's work for DfES

Currently DfES uses PIMSOFT to analyse DEA models for comparing schools. This seems to work satisfactorily when the number of schools in the model is in hundreds. When thousands of schools are to be analysed there seems to be a significant increase in computation time. To improve performance Hierarchical Decomposition methods are being looked in to. Using Dula's work instead can lead to significant improvements.

To implement Dula's method for scoring all schools and finding a set of optimal weights we need to solve nearly 2n LPs of maximum size mx(k+1). (Nearly 2n because in phase 2 we need to solve an LP for each one of the non-generator school, i.e., n-k LPs, of size mxk+1). Solving nearly twice the number of LPs compared to the normal methods is relevant only if 2n smaller LPs can be done faster than n larger LPs. In a sense this is a practical question needing empirical evidence. Dula [7, 8] reports on tests he has carried out. As presented by him, the evidence is overwhelmingly in favour of his method. While one can be forgiven for being a bit sceptical of this, common sense dictates that he must be right. After all if there are 100,000 schools of which only 1500 are generators, we are comparing the solving of 100,000 LPs with between 100,000 and 55,000 variables as opposed to nearly 200,000 LPs with only 1501 variables at most. Each LP in the first case can take several minutes as opposed to several seconds in the latter case. So while the number of LPs doubles for Dula, the time per LP can be a small fraction of that required by standard approaches.

How can DfES use this? I think it is premature to ask PIMSOFT team to implement Dula in their software without further experimentation with BuildHull. What is needed is to run the largest available data set (say 3000+ schools in the current data set) by Dula's method and also by PIMSOFT. It may be that one could find the generators outside of PIMSOFT and then use it to score all the schools. On the other hand a separate optimisation sub-routine within PIMSOFT may be able to handle the large data set and deliver all the generators to the normal DEA part of it.

Bear in mind though that the working of Dula's algorithm can be significantly improved. In part b) we outlined how all positive weights for generators and actual efficiency scores for some of the non-generators (the ones analysed after all the generators were found) can be achieved. Even more can be done for finding generators without doing LPs. For example, Ali had developed elementary heuristic methods to find some generators. We have developed more sophisticated methods as well as some even simpler ones and are able to produce at lease twice as many generators as Ali's methods do. In fact our simplest approach, requiring only a random number generator, is proving very promising indeed. In all the data sets tried so far (including the DfES data) we are able to find 10% of generators within seconds.

The best way to take this forward is perhaps to give us access to a state of the art solver such as CPLEX and let Nikos and Srini work with this to get the best out of Dula's method.

References

1. Ali, A. I. (1993), Streamlined computation for data envelopment analysis, European Journal of Operational Research 64, 61 –67
2. Barr, R. S., and Durchholz M. L. (1997), Parallel and hierarchical decomposition approaches for solving large-scale DEA models, Annals of Operational Research 73, 339 - 372
3. Chen, Y and Ali, A. I. (2002), Output-input ratio analysis and DEA frontier, European Journal of Operational Research 142, 476 – 479
4. Dula, J. H. *et. al.* (1998), An algorithm for identifying the frame of a pointed finite conical hull, INFORMS journal on computing, 10, No. 3
5. Dula J. H. (1998), An algorithm for the DEA, School of Business, University of Mississippi, MS 38677
6. Dula, J. H., and Thrall, R. M. (2001), A computational framework for accelerating DEA, Journal of Productivity Analysis, 16, 63 – 78
7. Dula, J. H. (2002), Computations in DEA. Pesquisa Operacional, 22, 165-182.
8. Dula, J. H. (2006), A Computational study of DEA with massive data sets, Computers and Operations Research (forthcoming) .

Generators in DEA

In this figure there are 10 DMUs with A, B, C, and D being the generators of the production possibility set (PPS)

A, B, C, D, E, I, and J are the boundary points but not all of them are generators of the PPS.

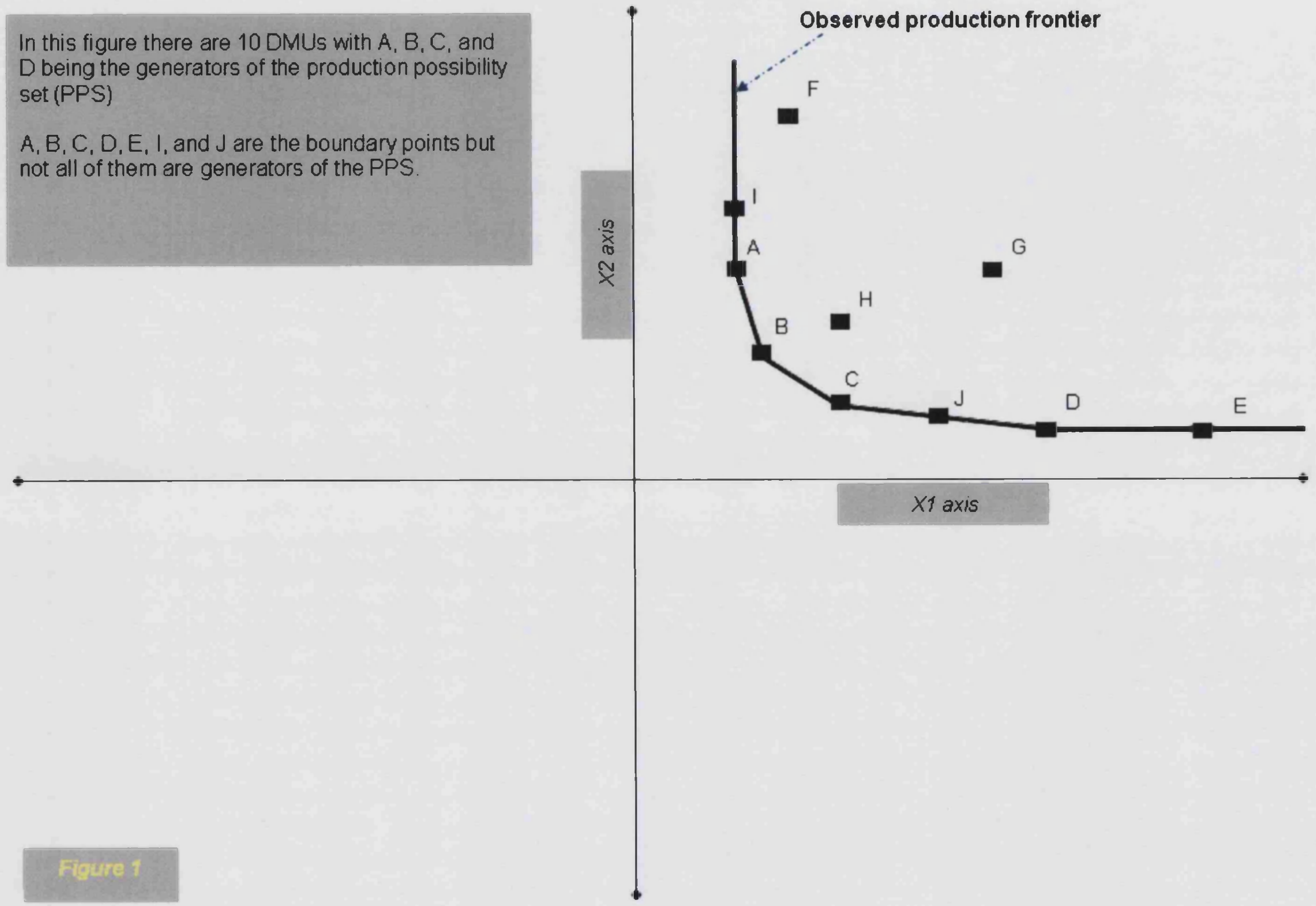
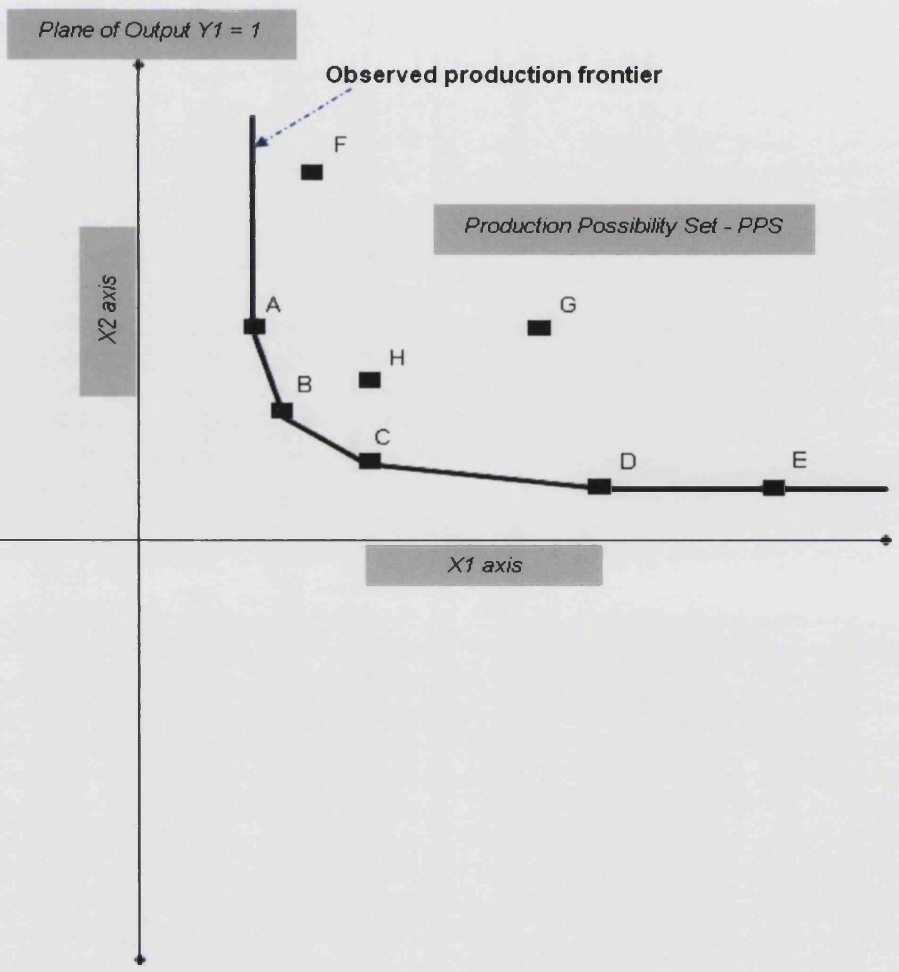


Figure 1

DMU	X1	X2	Y1
A	2	4	1
B	2.5	2.5	1
C	4	1.5	1
D	8	1	1
E	10	1	1
F	3	7	1
G	7	4	1
H	4	3	1



An 8 DMU, 3 factor DEA problem to graphically illustrate Dula's algorithm

Figure 2

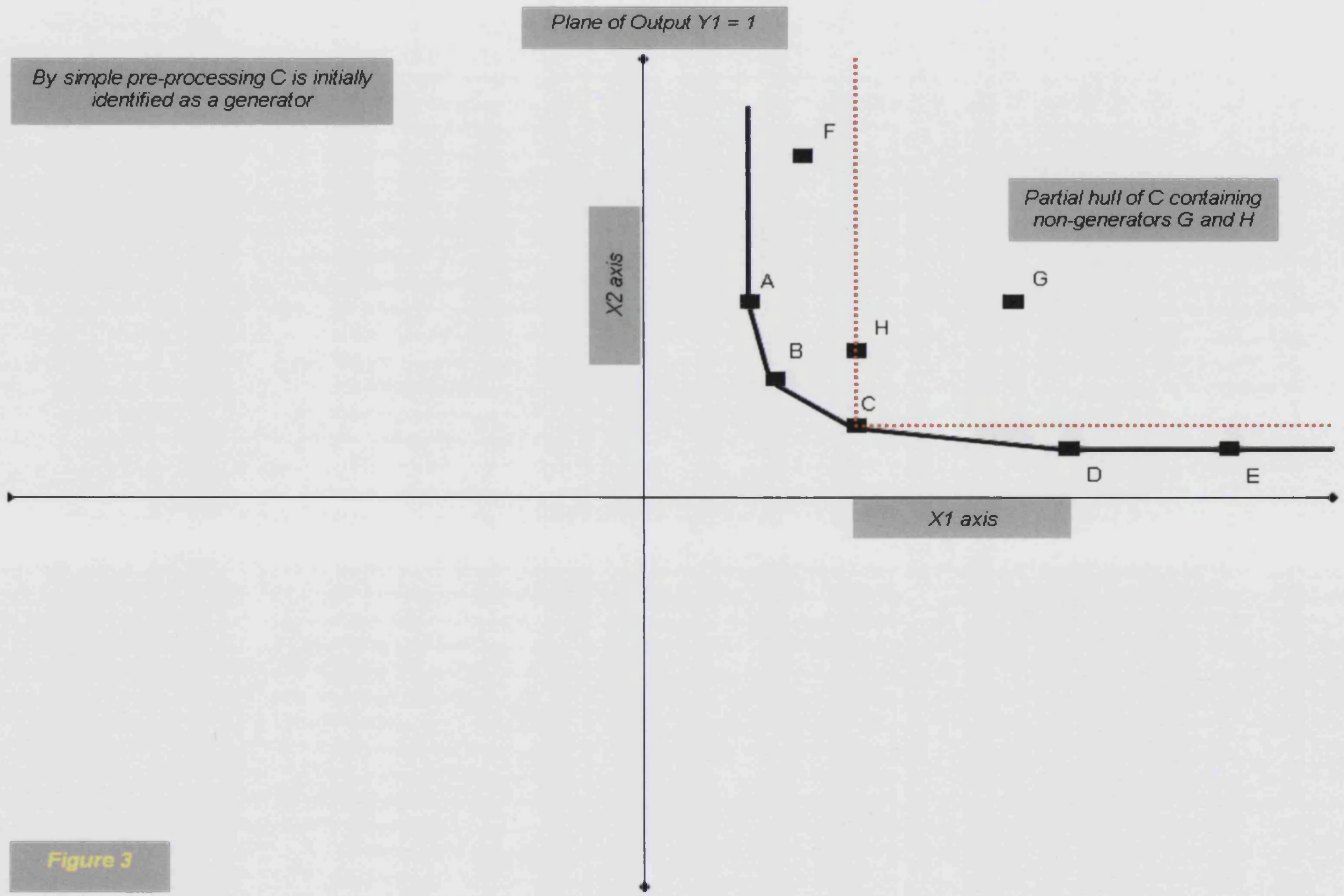
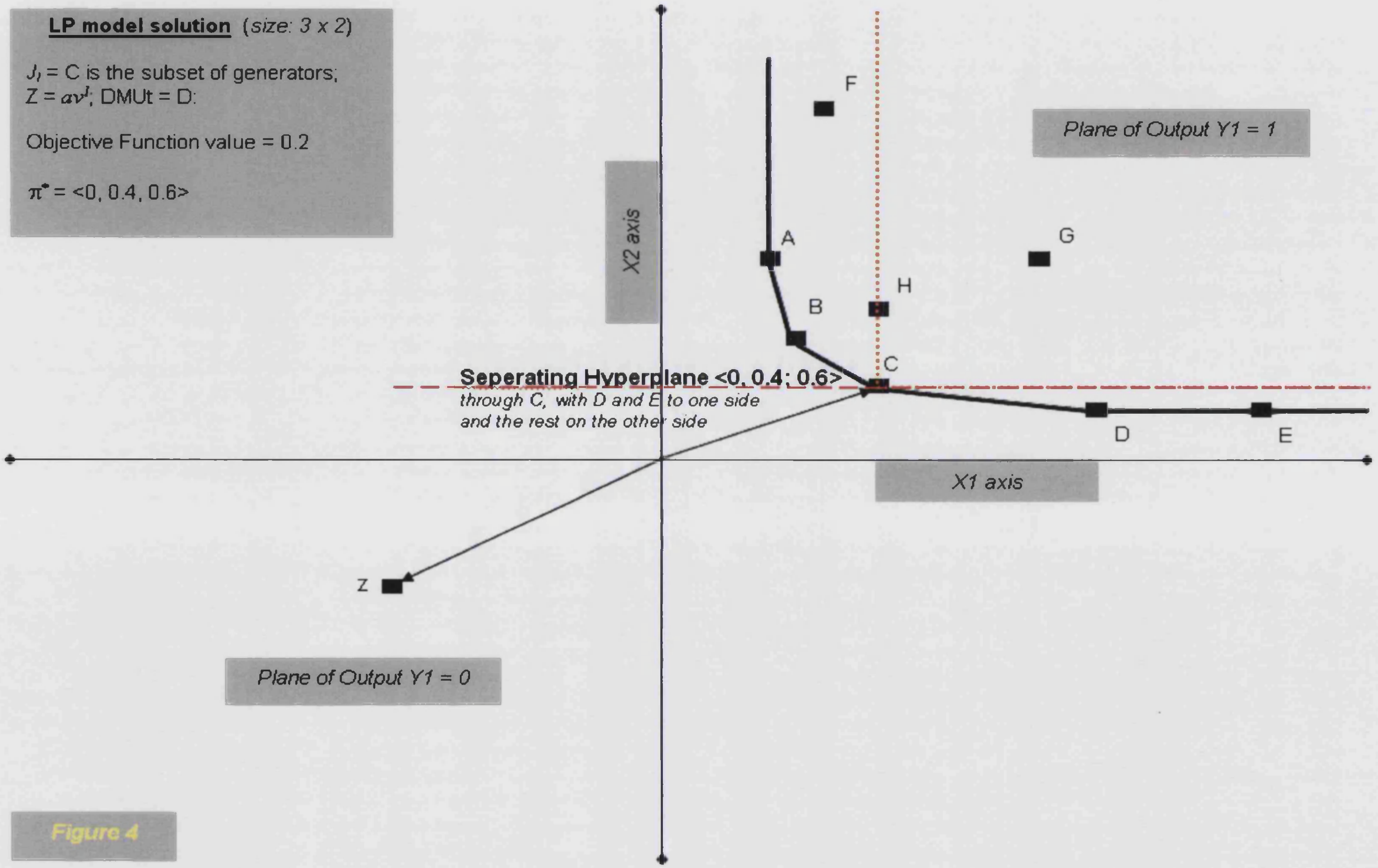

Plane of Output Y1 = 1
By simple pre-processing C is initially identified as a generator
Partial hull of C containing non-generators G and H
X2 axis
X1 axis
A
B
C
D
E
F
G
H

Figure 3



LP model solution (*size: 3 x 2*)

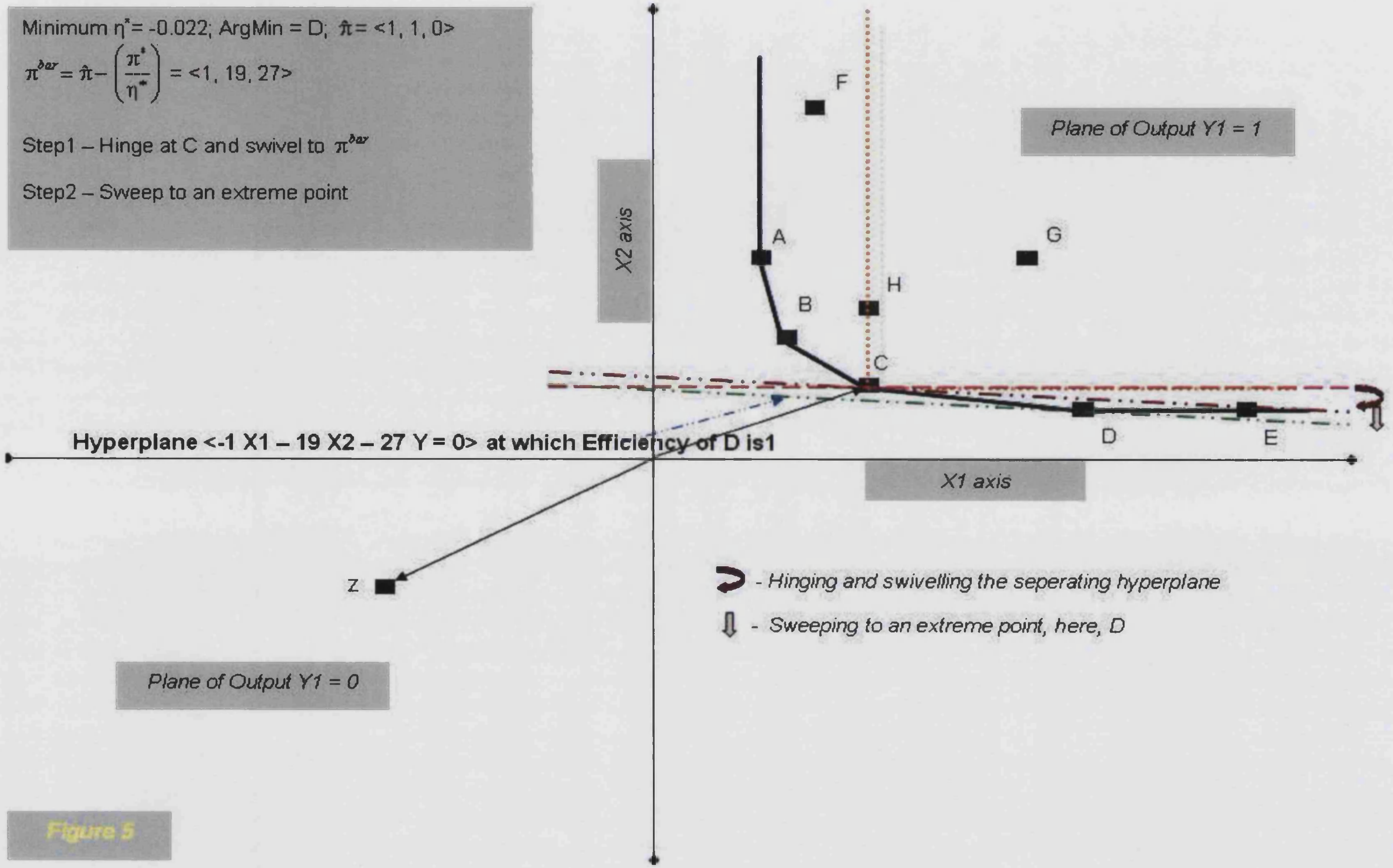
J_I = C is the subset of generators;
Z = av^J; DMUt = D:

Objective Function value = 0.2

π^* = <0, 0.4, 0.6>

Seperating Hyperplane <0, 0.4; 0.6>
through C, with D and E to one side and the rest on the other side

Figure 4



Minimum η^* = -0.022; ArgMin = D; $\hat{\pi}$ = <1, 1, 0>
$\pi^{bar} = \hat{\pi} - \left(\frac{\pi^*}{\eta^*}\right)$ = <1, 19, 27>
Step1 – Hinge at C and swivel to π^{bar}
Step2 – Sweep to an extreme point
F
Plane of Output Y1 = 1
X2 axis
A
G
H
B
C
Hyperplane <-1 X1 – 19 X2 – 27 Y = 0> at which Efficiency of D is1
D
E
X1 axis
Z
- Hinging and swivelling the seperating hyperplane
- Sweeping to an extreme point, here, D
Plane of Output Y1 = 0

Figure 5

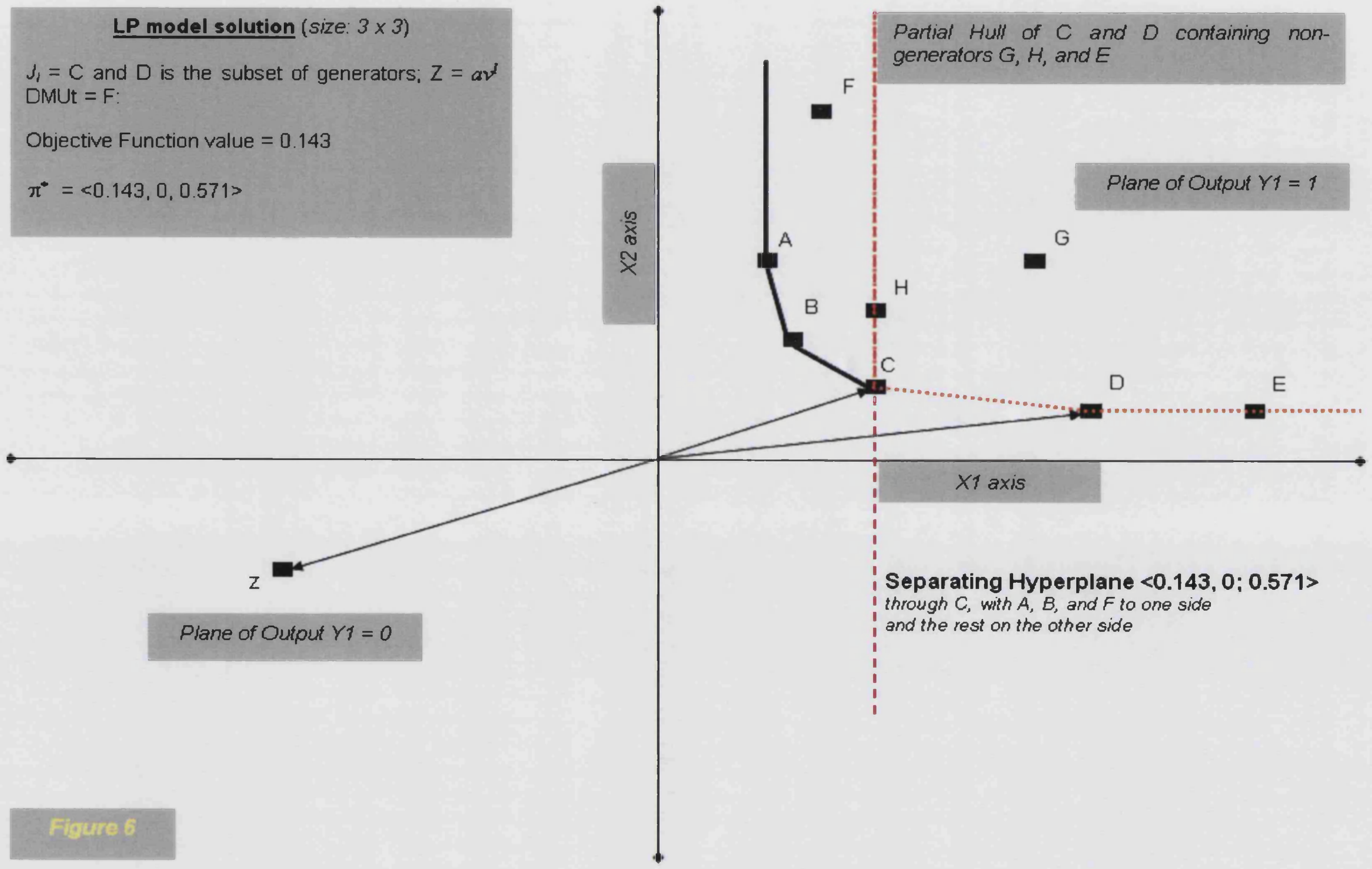

LP model solution (*size: 3 x 3*)
J_i = C and D is the subset of generators; Z = av^j
DMUt = F;
Objective Function value = 0.143
π^* = <0.143, 0, 0.571>
Partial Hull of C and D containing non-generators G, H, and E
Plane of Output Y1 = 1
X2 axis
X1 axis
F
A
B
C
H
G
D
E
Z
Plane of Output Y1 = 0
Separating Hyperplane <0.143, 0; 0.571>
through C, with A, B, and F to one side
and the rest on the other side

Figure 6

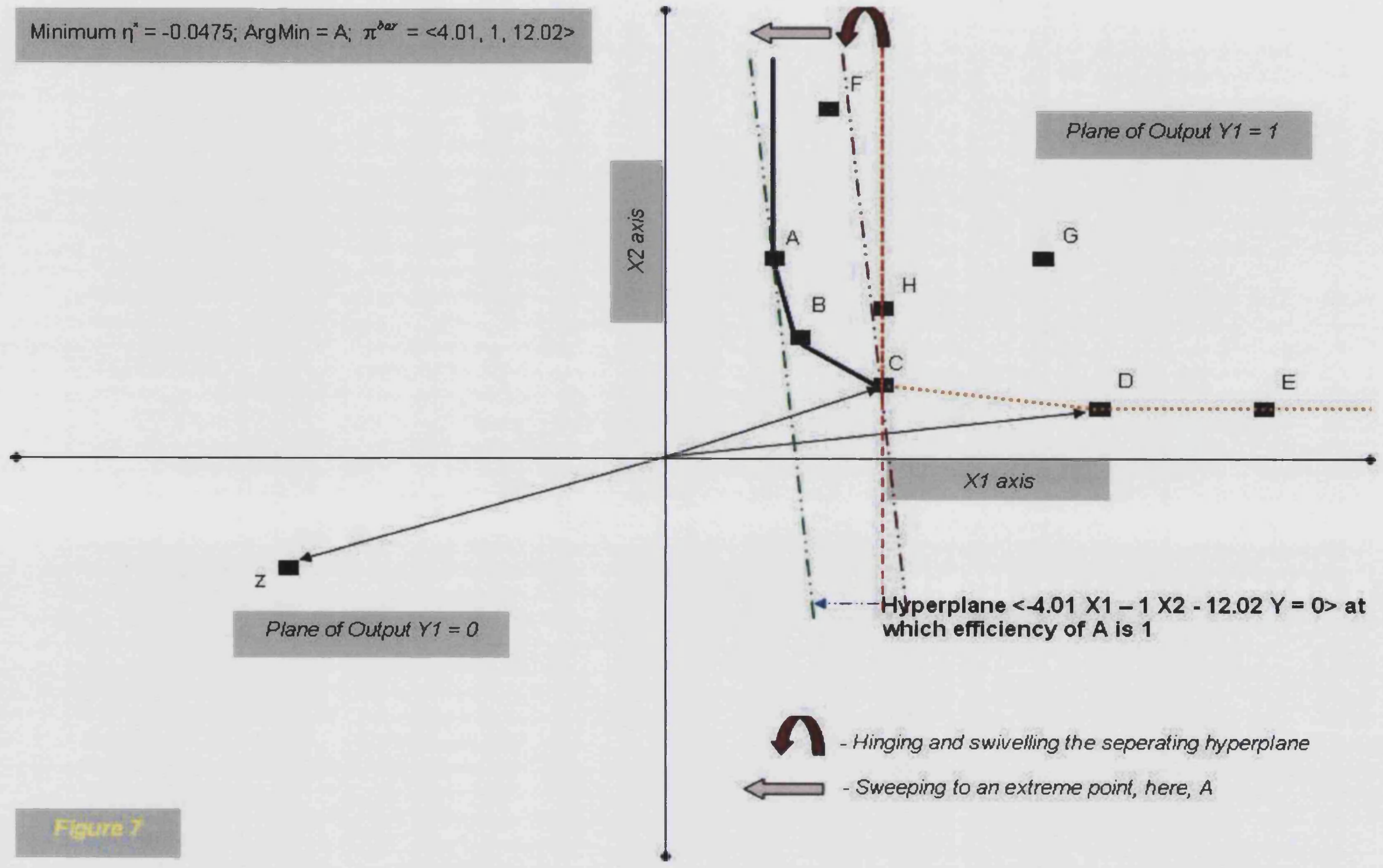

Minimum η^{*} = -0.0475; ArgMin = A; π^{bar} = <4.01, 1, 12.02>
X2 axis
X1 axis
Plane of Output Y1 = 1
Plane of Output Y1 = 0
F
A
B
C
H
G
D
E
Z
Hyperplane <-4.01 X1 – 1 X2 - 12.02 Y = 0> at which efficiency of A is 1
- Hinging and swivelling the seperating hyperplane
- Sweeping to an extreme point, here, A

Figure 7

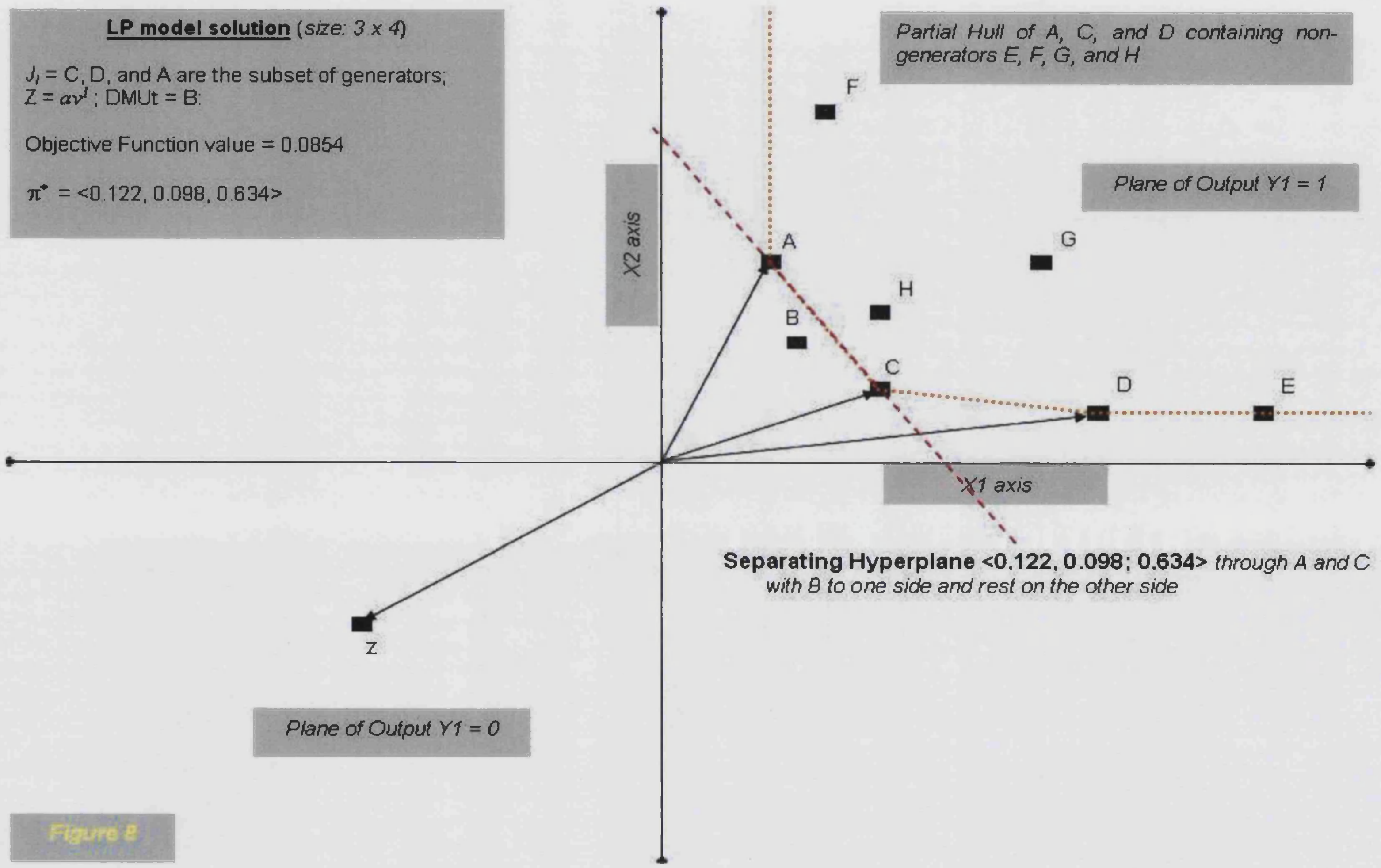

LP model solution (*size: 3 x 4*)
J_l = C, D, and A are the subset of generators;
Z = αv^l ; DMUt = B:
Objective Function value = 0.0854
π^* = <0.122, 0.098, 0.634>
Partial Hull of A, C, and D containing non-generators E, F, G, and H
Plane of Output Y1 = 1
X2 axis
F
A
G
H
B
C
D
E
X1 axis
Separating Hyperplane <0.122, 0.098; 0.634> *through A and C with B to one side and rest on the other side*
Z
Plane of Output Y1 = 0

Figure 8

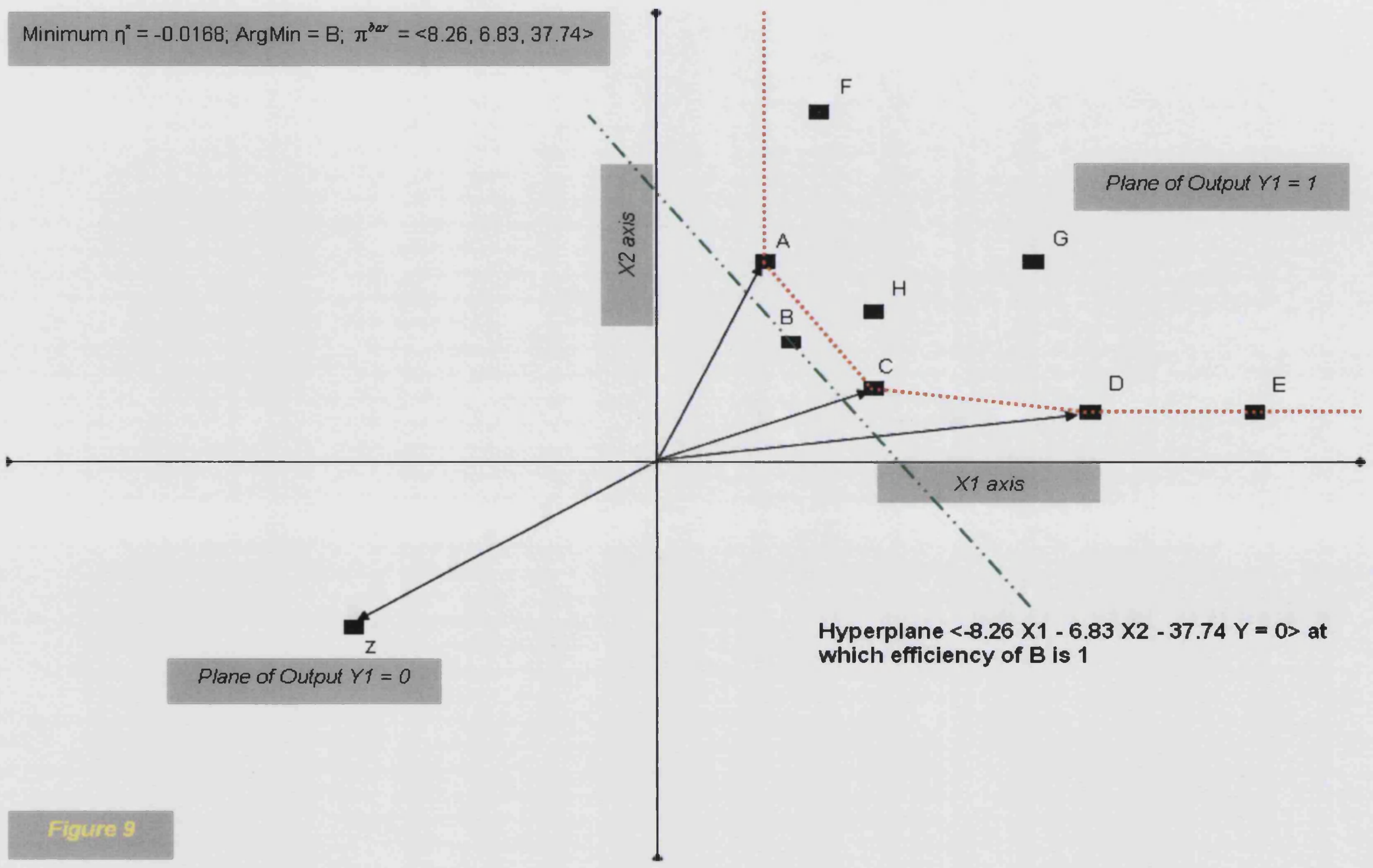
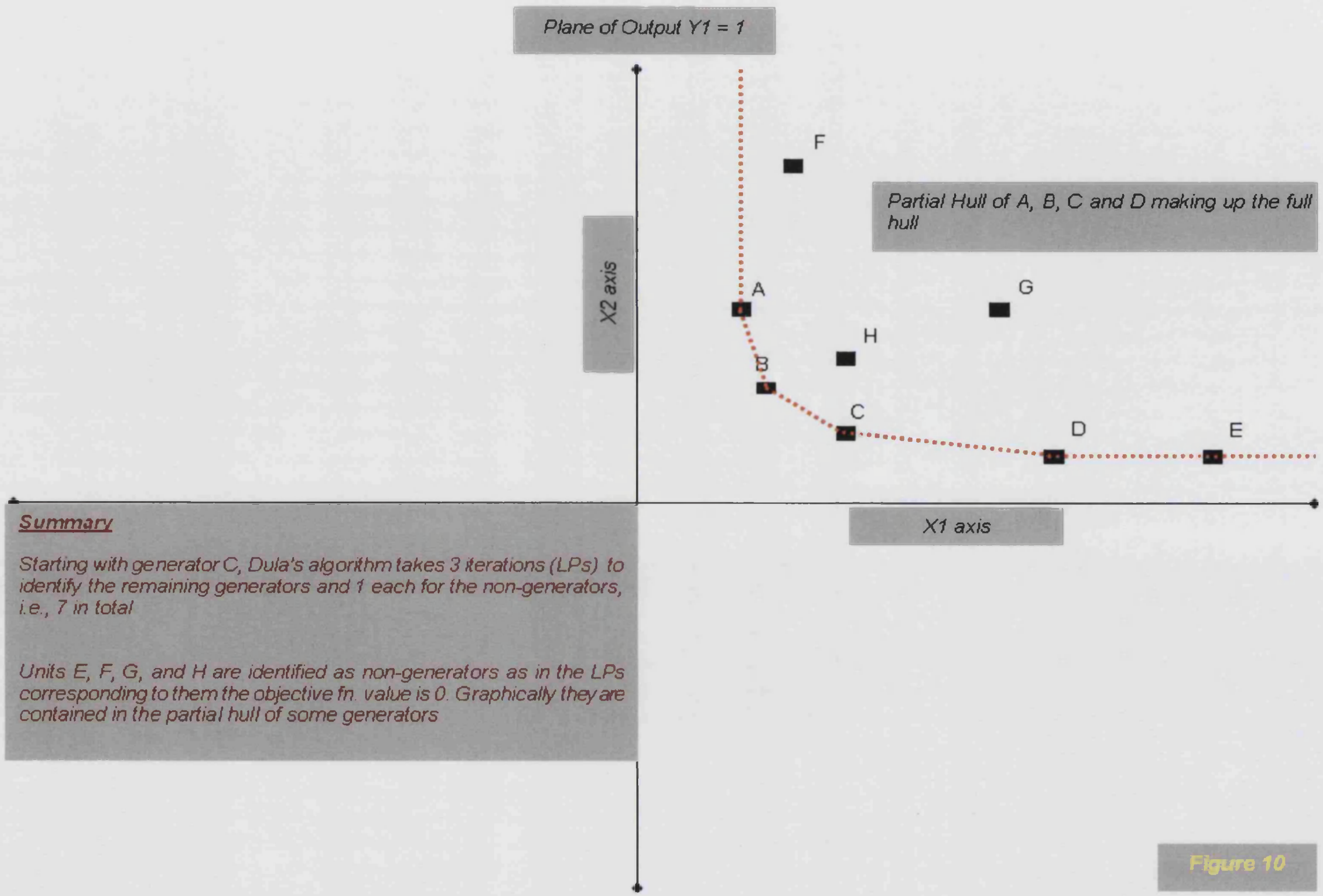

Minimum η* = -0.0168; ArgMin = B; π^bar = <8.26, 6.83, 37.74>
F
Plane of Output Y1 = 1
X2 axis
A
G
H
B
C
D
E
X1 axis
Z
Plane of Output Y1 = 0
Hyperplane <-8.26 X1 - 6.83 X2 - 37.74 Y = 0> at which efficiency of B is 1

Figure 9



Summary

Starting with generator C, Dula's algorithm takes 3 iterations (LPs) to identify the remaining generators and 1 each for the non-generators, i.e., 7 in total

Units E, F, G, and H are identified as non-generators as in the LPs corresponding to them the objective fn. value is 0. Graphically they are contained in the partial hull of some generators

Figure 10

Appendix 3 – Proof that λ_t^* is either 0 or 1 in the penalty enabled GBA models

First, it is obvious that λ_t is bounded between 0 and 1, i.e., $0 \le \lambda_t \le 1$. We show below that there are only two possibilities for λ_t^* in the penalty enabled GBA models regardless of the penalty value, viz., either $\lambda_t^* = 0$ or $\lambda_t^* = 1$.

To show this, we take the particular case of solving DMUt using MGBA LP-1. The logic applied here can be extended with little modification to the other models for any valid penalty value.

Now, DMUt is either inside or on the partial PPS; else, it is strictly outside the partial PPS. We investigate the two cases below.

Case 1: DMUt is inside or on the partial PPS:

In this case a linear combination of some of the generators in GEN dominate DMUt leading to $0 < \theta_t'^* \le 1$. Let the optimal dual values be $\pi_t'^* = (v'^*, u'^*)$. This means that the reduced cost of DMUt, $RC_t = u'^* Y_t - v'^* X_t \le 0$. If $\lambda_t^* > 0$, then the corresponding dual constraint will be binding, i.e., $u'^* Y_t - v'^* X_t = M$. Given that $M > 0$, $RC_t > 0$ but this contradicts with the stipulation that $0 < \theta_t'^* \le 1$. Hence, $\lambda_t^* = 0$.

Note that for any value of $M > 0$, the above reasoning is valid.

Case 2: DMUt is strictly outside the partial PPS:

Consider any generator DMUg in GEN. For $\lambda_g^* > 0$ along with $\lambda_t^* > 0$, the relevant hyperplane h defined by $\pi_t'^* = (v'^*, u'^*)$ must support both DMUt and DMUg. However, this requirement is incompatible. If $\lambda_t^* > 0$, then $RC_t = u'^* Y_t - v'^* X_t = M > 0$. In this case, the hyperplane h is defined by

$u'^{*}Y_j - v'^{*}X_j = M$. If $\lambda_g^{*} > 0$, then $RC_g = 0$ and the hyperplane *h* in this case is given by $RC_j = u'^{*}Y_j - v'^{*}X_j = 0$.

Given that $\underset{j \in GEN}{Max}\{RC_j\} = 0$, and $\lambda_t^{*} > 0$ dictates $RC_t = M$, the hyperplane *h* cannot be supporting both units for a given $\pi_t'^{*}$. Either, *h* supports at DMUt in which case the units in GEN lie strictly inside the half-space $u'^{*}Y_j - v'^{*}X_j < M, \forall j \in GEN$. Else, *h* supports DMUg in GEN in which case the hyperplane separates the units in GEN lying in the half-space $u'^{*}Y_j - v'^{*}X_j \le 0, \forall j \in GEN$ from DMUt lying in the half-space $u'^{*}Y_t - v'^{*}X_t > 0$.

Note again that for any value of $M > 0$, the above reasoning is valid.

Hence, $\lambda_t^{*} \times \lambda_g^{*} = 0$ in the penalty enabled GBA models. This dictates that either $\lambda_t^{*} = 0$ or $\lambda_t^{*} = 1$ to achieve a feasible solution to MGBA LP-1.

Appendix 4 - R codes to solve the output-oriented VRS model using GBA, BuildHull and the standard DEA algorithm

R code to solve the output-oriented VRS model using GBA

```
dudat<-read.xls("d.xls",type="double") /* Reads a dataset from the current directory */
n<-nrow(dudat) /* Beginning of initialisation */
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
m1<-6
m2<-3
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s5<-matrix(nrow=n,ncol=2)
ni<-m1
no<-m2
notie<-0
tieunits<-0
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=-dudat[1:n,m3:m]
x<- -dudat[1:n,1:m1]
y<- dudat[1:n,m3:m]
```

```
dm<-cbind(x,y)
rc<-rowSums(dm)
rc1<-max(rc)
rc2<-min(rc)
par1<-(rc1-rc2)
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<-abs(rowSums(a[,m5:m7]))-rowSums(a[,2:m3])
maxs2<-max(s2)
q1<-which.max(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
d4<-ncol(a4) /* End of initialisation */
g<-1
tim1<-proc.time()
while(sq>=1) /* Beginning of GBA procedure */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m1]=a3[1,2:m3]
b2[,m3:m]=0
b3<-rbind(t(b2),1)
b21<-matrix(nrow=1,ncol=m)
b21[,1:m1]=a3[1,2:m3]
b21[,m3:m]=a3[1,m5:m7]
```

```
c2<-matrix(nrow=1,ncol=m)
c2[,1:m1]=0
c2[,m3:m]=-a3[1,m5:m7]
c3<-rbind(t(c2),0)
asd<-cbind(t(b21),a4[-1,])
asd2<-rbind(asd,1)
abc2<-cbind(c3,asd2)
lt2<-(c(1,pen,rep(0,d4)))
f.dir <-c(rep("<=",m),"==")
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abc2,const.dir=f.dir,const.rhs=b3,compute.sens=1,scale=3)
s3<--(a3[,m5:m7]%*%vtr2$duals[m3:m])-vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1])
sasz<-max(s3)
if(sasz<=0)
{
sq<-nrow(a3)
s5[g,]<-c(a3[1,1],vtr2$solution[1])
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
if(sasz>0)
{
ms3<-max(s3)
smx3<-sum(s3==ms3)
if(smx3>=2) /* Tie breaking routine to identify one P-K efficient unit among the tied units */
{
notie<-notie+1
tieunits<-tieunits+smx3
s3tie<-sort(s3,decreasing=TRUE)
par5<-s3tie[smx3+1]
par6<-ms3-par5
par7<-par1/par6
duals3<-vtr2$duals[1:m7]*par7
duals3[1:m6]<-duals3[1:m6]+1
s3<--(a3[,m5:m7]%*%duals3[m3:m])-(a3[,2:m3]%*%duals3[1:m1])-duals3[m7]
}
sq<-nrow(a3)
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
} /* End of GBA procedure */
```

```
s6<-s5[complete.cases(s5),]
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by GBA to solve the dataset */
notie /* Number of ties encountered */
tieunits /* Average number of tied units per tie */
```

<u>R code to solve the output-oriented VRS model using BuildHull</u>

```
dudat<-read.xls("d.xls",type="double") /* Reads a dataset from the current directory */
n<-nrow(dudat) /* Beginning of initialisation */
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
m1<-6
m2<-3
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s5<-matrix(nrow=n,ncol=2)
ni<-m1
no<-m2
notie<-0
tieunits<-0
```

```
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=-dudat[1:n,m3:m]
x<- -dudat[1:n,1:m1]
y<- dudat[1:n,m3:m]
dm<-cbind(x,y)
rc<-rowSums(dm)
rc1<-max(rc)
rc2<-min(rc)
par1<-(rc1-rc2)
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<-abs(rowSums(a[,m5:m7]))-rowSums(a[,2:m3])
maxs2<-max(s2)
q1<-which.max(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
d4<-ncol(a4) /* End of initialisation */
g<-1
tim1<-proc.time()
while(sq>=1) /* Beginning of Phase-1 of BuildHull procedure */
{
b2<-matrix(nrow=1,ncol=m)
b2[1,1:m]<-1
```

```
c2<-matrix(nrow=1,ncol=m)
c2[1,]=-a3[1,-1]
c3<-rbind(t(c2),1)
abc2<-cbind(t(b2),-a4[-1,])
fp<-c(0,rep(1,d4))
abc3<-rbind(abc2,fp)
lt2<-(c(1,rep(0,d4)))
f.dir <-c(rep(">=",m),"==")
vtr2<-lp(direction="min",objective.in=lt2,const.mat=abc3,const.dir=f.dir,const.rhs=c3,compute.sens=1,scale=3)
if(vtr2$solution[1]==0)
{
sq<-nrow(a3)
s5[g,]<-a3[1,]
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
else
if(vtr2$solution[1]>0)
{
sq<-nrow(a3)
if(sq==1)
{
s3<--(a3[,m5:m7]%*%vtr2$duals[m3:m])+vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1])
}
else
if(sq>1)
```

```
{
s3<--(a3[,m5:m7]%*%vtr2$duals[m3:m])+vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1])
}
ms3<-max(s3)
smx3<-sum(s3==ms3)
if(smx3>=2) /* Tie breaking routine to identify one P-K efficient unit among the ties */
{
s3tie<-sort(s3,decreasing=TRUE)
par5<-s3tie[smx3+1]
par6<-ms3-par5
par7<-par1/par6
duals3<-vtr2$duals[1:m7]*par7
duals3[1:m6]<-duals3[1:m6]+1
s3<--(a3[,m5:m7]%*%duals3[m3:m])-(a3[,2:m3]%*%duals3[1:m1])+duals3[m7]
}
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
```

```
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
} /* End of Phase-1 of BuildHull procedure */
s6<-s5[complete.cases(s5),]
sx<-ncol(a4)
zx<-(n-sx)
if(zx==1)
{s6<-t(s6)}
s11<-matrix(nrow=zx,ncol=2)
a5<-matrix(nrow=sx,ncol=m)
a5=a4[-1,]
for(i in 1:zx) /* Beginning of Phase-2 of BuildHull procedure */
{
b<-matrix(nrow=1,ncol=m)
b[,1:m1]=s6[i,2:m3]
b[,m3:m]=0
b2<-rbind(t(b),1)
c<-matrix(nrow=1,ncol=m)
c[,1:m1]=0
c[,m3:m]=-s6[i,m5:m7]
c1<-t(c)
fp2<-c(0,rep(1,sx))
abc<-cbind(c1,a5)
abc3<-rbind(abc,fp2)
lt3<-c(1,rep(0,sx))
f.dir <-c(rep("<=",m),"==")
```

```
vtr3<-lp(direction="max",objective.in=lt3,const.mat=abc3,const.dir=f.dir,const.rhs=b2,compute.sens=1,scale=3)
s11[i,]<-cbind(s6[i,1],vtr3$solution[1])
} /* End of Phase-2 of BuildHull procedure */
tim /* Time taken by BuildHull to solve the dataset */
notie /* Number of ties encountered */
tieunits /* Average number of tied units per tie */
```

<u>R code to solve the output-oriented VRS model using the conventional DEA algorithm</u>

```
dudat<-read.xls("DEA.xls",type="double") /* Reads the dataset from the current directory */
n<-nrow(dudat) /* Beginning of initialisation */
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
m1<-7
m2<-8
m3<-m1+1
m4<-m2+1
m<-m1+m2
nm<-n+1
mn<-m+1
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=-dudat[1:n,m3:m]
x1<-t(dudat[1:n,1:m1])
dim(x1)<-c(m1,n)
y2<--dudat[1:n,m3:m]
tim1<-proc.time()
```

```
for(i in 1:n) /* Beginning of Phase-1 of the conventional algorithm */
{
b<-matrix(nrow=1,ncol=m)
b[,1:m1]=x1[,i]
b[,m3:m]=rep(0,m2)
b1<-rbind(t(b),1)
c<-matrix(nrow=1,ncol=m)
c[,1:m1]=rep(0,m1)
c[,m3:m]=-y2[i,]
fp2<-c(0,rep(1,n))
abc<-cbind(t(c),t(a))
abc3<-rbind(abc,fp2)
dim(abc3)<-c(mn,nm)
lt<-(c(1,rep(0,n)))
f.dir <-c(rep("<=",m),"==")
vtr<-lp(direction="max",objective.in=lt,const.mat=abc3,const.dir=f.dir,const.rhs=b1,compute.sens=1,scale=3)
s1[i]<-1/vtr$solution[1]
} /* End of Phase-1 of the conventional algorithm */
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by Phase-1 of the conventional algorithm */
for(j in 1:n) /* Beginning of Phase-2 of the conventional algorithm */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m1]=t(x1[,j])
b2[,m3:m]=-y2[j,]*1/s1[j]
b3<-rbind(t(b2),1)
a2<-matrix(nrow=n,ncol=m)
a2[,1:m1]=t(x1)
a2[,m3:m]=-y2
sl1<-matrix(nrow=m,ncol=m1)
sl1[1:m1,1:m1]=diag(m1)
sl1[m3:m,]=rep(0,m2)
sl2<-matrix(nrow=m,ncol=m2)
sl2[1:m1,]=rep(0,m1)
sl2[m3:m,1:m2]=-diag(m2)
abcd<-cbind(sl1,sl2,t(a2))
fp3<-c(rep(0,m),rep(1,n))
abcd2<-rbind(abcd,fp3)
f.dir2<-c(rep("==",m),"==")
lt2<-c(rep(1,m),rep(0,n))
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abcd2,const.dir=f.dir2,const.rhs=b3,compute.sens=1,scale=3)
s2[j]<-sum(vtr2$solution[1:m])
}/* End of Phase-2 of the conventional algorithm */
tim3<-proc.time()
tim<-tim3-tim1
tim /* Total time taken by the conventional algorithm */
```

Appendix 5 - Computational performance of the competitive algorithms in solving the output-oriented VRS model

Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)	Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)
5-2500-01	16.5	9.47	98.19	47.47	**10-2500-01**	17.44	10.36	185.36	87.05
5-2500-13	26.86	16.19	102.14	49.44	**10-2500-13**	33.88	20.92	191.67	89.47
5-2500-25	38.61	24.5	101.67	48.27	**10-2500-25**	51.73	33.35	201.84	90.65
5-2500-50	63.58	48.42	106.51	52.37	**10-2500-50**	91.16	68.75	205.25	91.19
5-5000-01	33.78	19.64	366.02	176.64	**10-5000-01**	37.69	23.19	716.5	336.05
5-5000-13	76.43	44.5	370.24	180.65	**10-5000-13**	107.85	64.3	766.92	353.44
5-5000-25	121.78	79.24	376.36	178.88	**10-5000-25**	176	114.7	783.56	364.57
5-5000-50	232.92	178	392.16	188.81	**10-5000-50**	348.03	262.22	825.6	372.13
5-7500-01	54.19	32.19	792.95	383.43	**10-7500-01**	57.3	38.25	1631.55	764
5-7500-13	145.82	88.98	826.67	395.26	**10-7500-13**	212.04	130.81	1770.97	811.61
5-7500-25	254.02	166.95	837.72	403.7	**10-7500-25**	382.99	249.78	1738.06	790.84
5-7500-50	512.82	389.38	866.02	426.19	**10-7500-50**	786.64	595.25	1786.41	788.26
5-10000-01	71.84	44.7	1388.48	662.89	**10-10000-01**	87.41	60.08	2758.63	1290.45
5-10000-13	243.91	146.23	1511.32	714.36	**10-10000-13**	372.1	228	2952.36	1366.01
5-10000-25	437.79	283.38	1480.84	708.81	**10-10000-25**	686.69	441.5	3015.35	1370.14
5-10000-50	889.87	683.58	1518.36	741.34	**10-10000-50**	1415.07	1053.96	3184.28	1411.23

Computational performance of the competitive algorithms in solving the output-oriented VRS model (Contd.)

Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)	Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)
15-2500-01	17.83	10.39	269.79	123.15	**20-2500-01**	21.53	13.45	371.58	168.69
15-2500-13	40.64	25.5	339.08	118.67	**20-2500-13**	48.61	31.97	361.67	165.36
15-2500-25	63.68	42.64	290.64	136.06	**20-2500-25**	82.97	58.04	381.71	184.6
15-2500-50	122.33	92.29	312.23	149.7	**20-2500-50**	155.94	118.73	479.85	276.52
15-5000-01	39.72	26.03	1057.53	486.2	**20-5000-01**	52.72	32.82	1808.2	708.67
15-5000-13	135.27	86.05	1109.06	518.94	**20-5000-13**	175.77	116.23	1480.17	712.69
15-5000-25	242.77	157.15	1138.03	531.41	**20-5000-25**	302.52	204.94	1820.28	1039.33
15-5000-50	478.11	357.44	1172.45	556	**20-5000-50**	611.22	460.53	2675.05	1839.89
15-7500-01	68.66	51.77	2447.45	1172.11	**20-7500-01**	77.48	57.85	3164.79	1430.81
15-7500-13	291.44	197.63	2444.99	1142.78	**20-7500-13**	364.46	243.13	3403.11	1632.72
15-7500-25	562.25	365.36	2509.72	1187.72	**20-7500-25**	673.91	448.67	4603.53	2803.26
15-7500-50	1143.89	867.57	2633.24	1301.57	**20-7500-50**	1233.39	947.53	6159.97	4306.56
15-10000-01	101.01	72.33	4145.72	1992.64	**20-10000-01**	112.46	86.18	5786.72	2722.22
15-10000-13	500.78	378.4	4363.72	2083.6	**20-10000-13**	617.73	411.04	6065.78	2895.7
15-10000-25	994.25	654.31	4468.03	2148.45	**20-10000-25**	1155.51	777.41	9736.8	6486.5
15-10000-50	2022.14	1520.06	4713.23	2319.67	**20-10000-50**	2401.78	1899.21	30788.23	27471.23

Appendix 6 - R codes to solve the additive VRS model using GBA, BuildHull and the standard DEA algorithm

R code to solve the additive VRS model using GBA

```
dudat<-read.xls("DEA.xls",type="double") /* Reads a dataset from the current directory */
n<-nrow(dudat) /* Beginning of the initialisation */
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
m1<-9
m2<-11
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s5<-matrix(nrow=n,ncol=2)
ni<-m1
no<-m2
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=dudat[1:n,m3:m]
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<--rowSums(a[,m5:m7])+rowSums(a[,2:m3])
```

```
mins2<-min(s2)
q1<-which.min(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
d4<-ncol(a4)
g<-1
pen<--1 /* End of initialization */
tim1<-proc.time()
while(sq>=1) /* Beginning of the GBA procedure */
{
c2<-matrix(nrow=1,ncol=m)
c2[1,]=abs(a3[1,-1])
c3<-rbind(t(c2),1)
sl1<-matrix(nrow=m,ncol=m1)
sl1[1:m1,1:m1]=diag(m1)
sl1[m3:m,]=rep(0,m2)
sl2<-matrix(nrow=m,ncol=m2)
sl2[1:m1,]=rep(0,m1)
sl2[m3:m,1:m2]=-diag(m2)
abcd<-cbind(sl1,sl2,a4[-1,])
fp3<-c(rep(0,m),rep(1,d4))
abcd2<-rbind(abcd,fp3)
abcd3<-cbind(abcd2,c3)
f.dir2<-c(rep("==",m),"==")
lt2<-c(rep(1,m),rep(0,d4),pen)
```

```
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abcd3,const.dir=f.dir2,const.rhs=c3, compute.sens=1,scale=3)
s3<-(a3[,-1:-m3]%*%vtr2$duals[m3:m])+vtr2$duals[m7]+(a3[,2:m3]%*%vtr2$duals[1:m1])
sasz<-min(s3)
if(sasz>=0)
{
sq<-nrow(a3)
s5[g,]<-c(a3[1,1],sum(vtr2$solution[1:m]))
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
else
if(sasz<0)
{
sq<-nrow(a3)
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-min(sx3[,m8])
q4<-which.min(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
} /* End of GBA procedure */
s6<-s5[complete.cases(s5),]
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by the GBA to solve the dataset */
```

R code to solve the additive VRS model using BuildHull

```
dudat<-read.xls("DEA.xls",type="double") /* Reads a dataset from the current directory */
n<-nrow(dudat) /* Beginning of initialisation */
m1<-9
m2<-11
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s1<-matrix(nrow=n,ncol=1)
s5<-matrix(nrow=n,ncol=m+1)
ni<-m1
```

```
no<-m2
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=-dudat[1:n,m3:m]
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<-rowSums(a[,m5:m7])+rowSums(a[,2:m3])
mins2<-min(s2)
q1<-which.min(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
d4<-ncol(a4)
g<-1 /* End of initialization */
tim1<-proc.time()
while(sq>=1) /* Beginning of Phase-1 of BuildHull */
{
b2<-matrix(nrow=1,ncol=m)
b2[1,1:m]<-1
c2<-matrix(nrow=1,ncol=m)
c2[1,]=-a3[1,-1]
c3<-rbind(t(c2),1)
abc2<-cbind(t(b2),-a4[-1,])
fp<-c(0,rep(1,d4))
abc3<-rbind(abc2,fp)
```

```
lt2<-(c(1,rep(0,d4)))
f.dir <-c(rep(">=",m),"==")
vtr2<-lp(direction="min",objective.in=lt2,const.mat=abc3,const.dir=f.dir,const.rhs=c3,compute.sens=1,scale=3)
if(vtr2$solution[1]==0)
{
sq<-nrow(a3)
s5[g,]<-a3[1,]
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
else
if(vtr2$solution[1]>0)
{
sq<-nrow(a3)
if(sq==1)
{
s3<--(a3[,m5:m7]%*%vtr2$duals[m3:m])+vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1])
}
else
if(sq>1)
{
s3<--(a3[,m5:m7]%*%vtr2$duals[m3:m])+vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1])
}
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
} /* End of Phase-1 of BuildHull */
s6<-s5[complete.cases(s5),]
sx<-ncol(a4)
zx<-(n-sx)
s2<-matrix(nrow=zx,ncol=1)
if(zx==1)
{s6<-t(s6)}
s11<-matrix(nrow=zx,ncol=2)
a5<-matrix(nrow=sx,ncol=m)
a5=a4[-1,]
for(j in 1:zx) /* Beginning of Phase-2 of BuildHull */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m]=abs(t(s6[j,-1]))
```

```
b3<-rbind(t(b2),1)
sl1<-matrix(nrow=m,ncol=m1)
sl1[1:m1,1:m1]=diag(m1)
sl1[m3:m,]=rep(0,m2)
sl2<-matrix(nrow=m,ncol=m2)
sl2[1:m1,]=rep(0,m1)
sl2[m3:m,1:m2]=-diag(m2)
abcd<-cbind(sl1,sl2,abs(a5))
fp3<-c(rep(0,m),rep(1,sx))
abcd2<-rbind(abcd,fp3)
f.dir2<-c(rep("==",m),"==")
lt2<-c(rep(1,m),rep(0,sx))
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abcd2,const.dir=f.dir2,const.rhs=b3,compute.sens=1,scale=3)
s2[j]<-sum(vtr2$solution[1:m])
} /* End of Phase-2 of BuildHull */
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by BuildHull to solve the dataset */
```

R code to solve the output-oriented VRS model using the conventional DEA algorithm

```
dudat<-read.xls("DEA.xls",type="double") /* Reads the dataset from the current directory */
n<-nrow(dudat) /* Beginning of initialisation */
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
m1<-9
m2<-11
```

```
m3<-m1+1
m4<-m2+1
m<-m1+m2
nm<-n+1
mn<-m+1
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=-dudat[1:n,m3:m]
x1<-t(dudat[1:n,1:m1])
dim(x1)<-c(m1,n)
y2<--dudat[1:n,m3:m]
tim1<-proc.time()
for(j in 1:n) /* Beginning of conventional algorithm */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m1]=t(x1[,j])
b2[,m3:m]=-y2[j,]
b3<-rbind(t(b2),1)
a2<-matrix(nrow=n,ncol=m)
a2[,1:m1]=t(x1)
a2[,m3:m]=-y2
sl1<-matrix(nrow=m,ncol=m1)
sl1[1:m1,1:m1]=diag(m1)
sl1[m3:m,]=rep(0,m2)
sl2<-matrix(nrow=m,ncol=m2)
sl2[1:m1,]=rep(0,m1)
sl2[m3:m,1:m2]=-diag(m2)
abcd<-cbind(sl1,sl2,t(a2))
fp3<-c(rep(0,m),rep(1,n))
abcd2<-rbind(abcd,fp3)
f.dir2<-c(rep("==",m),"==")
lt2<-c(rep(1,m),rep(0,n))
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abcd2,const.dir=f.dir2,const.rhs=b3,scale=3)
s2[j]<-sum(vtr2$solution[1:m])
} /* End of conventional algorithm */
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by the conventional algorithm to solve the dataset */
```

Appendix 7 - Computational performance of the competitive algorithms in solving the additive VRS model

Dataset	BuildHull (secs)	GBA (secs)	Std Additive model (secs)	Dataset	BuildHull (secs)	GBA (secs)	Std Additive model (secs)
5-2500-01	17.11	8.83	68.6	**10-2500-01**	18.85	10.05	130.99
5-2500-13	26.82	15.96	71.3	**10-2500-13**	36.79	22.64	133.54
5-2500-25	38.86	24.82	71.36	**10-2500-25**	59	37.9	138.45
5-2500-50	64.95	50.09	71.23	**10-2500-50**	98.89	83.98	132.98
5-5000-01	33.96	19.6	267.69	**10-5000-01**	38.57	24.06	525.39
5-5000-13	78.81	48.08	267.83	**10-5000-13**	120.48	76.06	526.75
5-5000-25	128.33	83.23	280.63	**10-5000-25**	198	135.19	547.53
5-5000-50	240.47	185.84	271.06	**10-5000-50**	375.42	299.95	538.61
5-7500-01	54.81	32.3	593.35	**10-7500-01**	64.53	40.57	1174
5-7500-13	154.82	94.13	600.66	**10-7500-13**	250.69	155.83	1219.92
5-7500-25	270.29	175.31	609.17	**10-7500-25**	431.53	297.49	1212.29
5-7500-50	523.89	410.67	602.96	**10-7500-50**	836.61	674.18	1179.36
5-10000-01	76.7	46.53	1037.85	**10-10000-01**	91.12	61.19	2071.57
5-10000-13	260.55	159.36	1046.44	**10-10000-13**	415.5	269.14	2190.15
5-10000-25	456.5	301.31	1059.72	**10-10000-25**	762.12	520.5	2097.48
5-10000-50	914.45	714.23	1071.3	**10-10000-50**	1501.83	1203.5	2169.21

Computational performance of the competitive algorithms in solving the additive VRS model (contd.)

Dataset	BuildHull (secs)	GBA (secs)	Std Additive model (secs)	Dataset	BuildHull (secs)	GBA (secs)	Std Additive model (secs)
15-2500-01	21.31	11.6	183.45	**20-2500-01**	24.11	17.03	269.05
15-2500-13	49.25	39.6	178.14	**20-2500-13**	73.49	131.05	272.8
15-2500-25	77.35	63.78	186.88	**20-2500-25**	128.52	1434.9	268.11
15-2500-50	143.45	128.51	197.28	**20-2500-50**	200.92	1944.67	312.53
15-5000-01	44.52	31.35	766.56	**20-5000-01**	59.45	39.47	1824.39
15-5000-13	166.09	126.09	783.68	**20-5000-13**	241.28	1533.81	1846.67
15-5000-25	362.16	298.42	799.2	**20-5000-25**	424.42	1653.58	2332.08
15-5000-50	654.19	620.21	802.2	**20-5000-50**	789	1778.97	2413.56
15-7500-01	89.29	59.12	1839.44	**20-7500-01**	120.09	97.73	3221.3
15-7500-13	394.6	392.91	1832.7	**20-7500-13**	524.92	2337.53	3301.68
15-7500-25	715.29	1258.17	1945.3	**20-7500-25**	905.67	3266.24	3611.11
15-7500-50	1402.67	1940.27	1962.56	**20-7500-50**	1832.5	4745.32	6264.22
15-10000-01	140.06	121.66	3447.93	**20-10000-01**	165.54	139.72	5570.45
15-10000-13	673.69	972.3	3380.14	**20-10000-13**	921.61	3022.21	5810.16
15-10000-25	1255.26	2911.31	3656.89	**20-10000-25**	1824.92	5500.06	6866.34
15-10000-50	2530.95	3418.15	3848.12	**20-10000-50**	3242.82	7257.86	11763.13

Appendix 8 - R codes of the two alternative GBA approaches and an alternative BuildHull approach to solve the additive VRS model

R code of the two-phase version of GBA to solve the additive VRS model

```
dudat<-read.xls("d.xls",type="double") /* Reads a dataset from the current directory */
n<-nrow(dudat) /* Beginning of initialisation */
m1<-6
m2<-3
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s1<-matrix(nrow=n,ncol=1)
s5<-matrix(nrow=n,ncol=m+1)
ni<-m1
no<-m2
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=-dudat[1:n,m3:m]
x=dudat[1:n,1:m1]
y=-dudat[1:n,m3:m]
e<-c(1:n)
a<-c(e,a)
```

```
dim(a)<-c(n,m+1)
s2<-rowSums(a[,m5:m7])+rowSums(a[,2:m3])
mins2<-min(s2)
dm<-cbind(-x,-y)
rc<-rowSums(dm)
rc1<-max(rc)
rc2<-min(rc)
par1<-(rc1-rc2)
q1<-which.min(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
d4<-ncol(a4)
g<-1
pen<--2 /* End of initilisation */
tim1<-proc.time() /* Beginning of phase-1 of GBA */
while(sq>=1)
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m1]=a3[1,2:m3]
b2[,m3:m]=0
b3<-rbind(t(b2),1)
b21<-matrix(nrow=1,ncol=m)
b21[,1:m1]=a3[1,2:m3]
b21[,m3:m]=a3[1,m5:m7]
c2<-matrix(nrow=1,ncol=m)
```

```
c2[,1:m1]=0
c2[,m3:m]=-a3[1,m5:m7]
c3<-rbind(t(c2),0)
asd<-cbind(t(b21),a4[-1,])
asd2<-rbind(asd,1)
abc2<-cbind(c3,asd2)
lt2<-(c(1,pen,rep(0,d4)))
f.dir <-c(rep("<=",m),"==")
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abc2,const.dir=f.dir,const.rhs=b3,compute.sens=1,scale=3)
s3<-round(-(a3[,-1:-m3]%*%vtr2$duals[m3:m])-vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1]),digits=10)
sasz<-max(s3)
if(sasz<=0)
{
sq<-nrow(a3)
s5[g,]<-a3[1,]
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
else
if(sasz>0)
{
ms3<-max(s3)
smx3<-sum(s3==ms3)
if(smx3>=2) /* Tie breaking routine to identify one P-K efficient unit among the tied units */
{
notie<-notie+1
tieunits<-tieunits+smx3
s3tie<-sort(s3,decreasing=TRUE)
par5<-max(s3tie[smx3+1],0)
par6<-ms3-par5
par7<-abs(par1/par6)+1
duals3<-vtr2$duals[1:m7]*par7
duals3[1:m]<-duals3[1:m]+1
s3<-(-(a3[,m5:m7]%*%duals3[m3:m]))-(a3[,2:m3]%*%duals3[1:m1])-duals3[m7]
}
sq<-nrow(a3)
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
} /* End of Phase-1 of GBA */
```

```
s6<-s5[complete.cases(s5),]
sx<-ncol(a4)
zx<-(n-sx)
s2<-matrix(nrow=zx,ncol=1)
if(zx==1)
{s6<-t(s6)}
s11<-matrix(nrow=zx,ncol=2)
a5<-matrix(nrow=sx,ncol=m)
a5=a4[-1,]
for(j in 1:zx) /* Beginning of Phase-2 of GBA */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m]=abs(t(s6[j,-1]))
b3<-rbind(t(b2),1)
sl1<-matrix(nrow=m,ncol=m1)
sl1[1:m1,1:m1]=diag(m1)
sl1[m3:m,]=rep(0,m2)
sl2<-matrix(nrow=m,ncol=m2)
sl2[1:m1,]=rep(0,m1)
sl2[m3:m,1:m2]=-diag(m2)
abcd<-cbind(sl1,sl2,abs(a5))
fp3<-c(rep(0,m),rep(1,sx))
abcd2<-rbind(abcd,fp3)
f.dir2<-c(rep("==",m),"==")
lt2<-c(rep(1,m),rep(0,sx))
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abcd2,const.dir=f.dir2,const.rhs=b3,compute.sens=0,scale=3)
s2[j]<-sum(vtr2$objval[1])
} /* End of Phase-2 of GBA */
```

```
tim /* Time taken by GBA to solve the dataset */
notie /* Number of ties encountered */
tieunits /* Average number of tied units per tie */
```

R code of the multiplier GBA approach to solve the additive VRS model

```
dudat<-read.xls("DEA.xls",type="double",colNames=FALSE) /* Reads the dataset from the current directory */
n<-dudat[1,1] /* Beginning of initialisation */
n2<-n+1
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
m1<-dudat[1,2]
m2<-dudat[1,3]
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s5<-matrix(nrow=n,ncol=2)
ni<-m1
no<-m2
notie<-0
tieunits<-0
pen<-0.1
x<- dudat[2:n2,1:m1]
```

```
y<- -dudat[2:n2,m3:m]
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=x
a[1:n,m3:m]=y
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<--rowSums(a[,m5:m7])-rowSums(a[,2:m3])
maxs2<-max(s2)
q1<-which.max(s2)
a4<-a[q1,]
dim(a4)<-c(1,m+1)
a3<-a[-q1,]
sq<-nrow(a3)
d4<-nrow(a4)
g<-1 /* End of Initialisation */
tim1<-proc.time()
while(sq>=1) /* Beginning of the GBA procedure */
{
abcd<-cbind(a4,1,-1)
abcd1<-abcd[,-1]
abcd2<-c(a3[1,-1],1,-1)
adc<-diag(m+2)
adc3<-rbind(abcd1,abcd2,adc)
c2<-c(rep(0,d4),-1,rep(1,m),0,0)
c3<-t(t(c2))
m15<-d4+m+3
f.dir2<-c(rep(">=",m15))
```

```
lt2<-c(a3[1,-1],1,-1)
dim(lt2)<-c(1,m8)
vtr2<-lp(direction="min",objective.in=lt2,const.mat=adc3,const.dir=f.dir2,const.rhs=c3, compute.sens=1,scale=3)
s3<-round(-(a3[,-1:-m3]%*%vtr2$solution[m3:m])-vtr2$solution[m7]+vtr2$solution[m8]-(a3[,2:m3]%*%vtr2$solution[1:m1]),digits=10)
sasz<-max(s3)
if(sasz<=0)
{
ssls<-abs(-(a3[1,-1:-m3]%*%vtr2$solution[m3:m])-vtr2$solution[m7]+vtr2$solution[m8]-(a3[1,2:m3]%*%vtr2$solution[1:m1]))
sq<-nrow(a3)
s5[g,]<-c(a3[1,1],ssls)
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
else
if(sasz>0)
{
sq<-nrow(a3)
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-min(sx3[,m8])
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-rbind(a4,sk1)
a4<-unique(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-nrow(a4)
}
} /* End of GBA procedure */
s6<-s5[complete.cases(s5),]
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by GBA to solve the dataset */
```

R code of the multiplier BuildHull approach to solve the additive VRS model

```
dudat<-read.xls("d57.xls",type="double") /* Reads a dataset from the Current directory */
n<-nrow(dudat) /* Beginning of initialisation */
m1<-9
m2<-11
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
```

```
s1<-matrix(nrow=n,ncol=1)
s5<-matrix(nrow=n,ncol=m+1)
ni<-m1
no<-m2
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=dudat[1:n,1:m1]
a[1:n,m3:m]=-dudat[1:n,m3:m]
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<-rowSums(a[,m5:m7])+rowSums(a[,2:m3])
mins2<-min(s2)
q1<-which.min(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
d4<-ncol(a4)
g<-1 /* End of initialization */
tim1<-proc.time()
while(sq>=1) /* Beginning of Phase-1 of BuildHull */
{
b2<-matrix(nrow=1,ncol=m)
b2[1,1:m]<-1
c2<-matrix(nrow=1,ncol=m)
c2[1,]=-a3[1,-1]
c3<-rbind(t(c2),1)
```

```
abc2<-cbind(t(b2),-a4[-1,])
fp<-c(0,rep(1,d4))
abc3<-rbind(abc2,fp)
lt2<-(c(1,rep(0,d4)))
f.dir <-c(rep(">=",m),"==")
vtr2<-lp(direction="min",objective.in=lt2,const.mat=abc3,const.dir=f.dir,const.rhs=c3,compute.sens=1,scale=3)
if(vtr2$solution[1]==0)
{
sq<-nrow(a3)
s5[g,]<-a3[1,]
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
else
if(vtr2$solution[1]>0)
{
sq<-nrow(a3)
if(sq==1)
{
s3<--(a3[,m5:m7]%*%vtr2$duals[m3:m])+vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1])
}
else
if(sq>1)
{
s3<--(a3[,m5:m7]%*%vtr2$duals[m3:m])+vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1])
}
```

```
ms3<-max(s3)
smx3<-sum(s3==ms3)
if(smx3>=2) /* Tie breaking routine to identify one extreme-efficient unit among the tied units */
{
s3tie<-sort(s3,decreasing=TRUE)
par5<-s3tie[smx3+1]
par6<-ms3-par5
par7<-par1/par6
duals3<-vtr2$duals[1:m7]*par7
duals3[1:m6]<-duals3[1:m6]+1
s3<--(a3[,m5:m7]%*%duals3[m3:m])-(a3[,2:m3]%*%duals3[1:m1])+duals3[m7]
}
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
```

```
}
} /* End of Phase-1 of BuildHull */
s6<-s5[complete.cases(s5),]
sx<-ncol(a4)
zx<-(n-sx)
s2<-matrix(nrow=zx,ncol=1)
if(zx==1)
{s6<-t(s6)}
s11<-matrix(nrow=zx,ncol=2)
a4<-t(a4)
for(j in 1:zx) /* Beginning of Phase-2 of BuildHull */
{
abcd<-cbind(a4,1,-1)
abcd1<-(abcd[,-1])
adc<-diag(m+2)
adc3<-rbind(abcd1,adc)
c2<-c(rep(0,d4),rep(1,m),0,0)
c3<-t(t(c2))
m15<-d4+m+2
f.dir2<-c(rep(">=",m15))
lt2<-c(s6[j,-1],1,-1)
dim(lt2)<-c(1,m8)
vtr2<-lp(direction="min",objective.in=lt2,const.mat=adc3,const.dir=f.dir2,const.rhs=c3, compute.sens=1,scale=3)
sslc<-abs(-(s6[j,-1:-m3]%*%vtr2$solution[m3:m])-vtr2$solution[m7]+vtr2$solution[m8]-(s6[j,2:m3]%*%vtr2$solution[1:m1]))
s11[j,]<-c(s6[j],sslc)
} /* End of Phase-2 of BuildHull */
tim /* Total time taken by BuildHull to solve the dataset */
notie /* Number of ties encountered */
```

tieunits /* Average number of tied units per tie */

Appendix 9 – R codes of the DGP, GBA, BuildHull and the standard DEA algorithm to solve the input-oriented CRS model

CRS Data Generating Process using the Cobb-Douglas production function

```
n<-5000 /* Assume the number of units in the dataset to be 5000 */
m1<-4 /* Assume the number of inputs to be 4 */
m2<-1 /* The number of outputs in any dataset is 1 */
den<-1 /* Assume the density of the dataset to be 1% */
n1<-den*n/100
n2<-n1+1
n3<-n-n1
n4<-n3+1
x<-matrix(nrow=n,ncol=m1)
for(j in 1:m1) /* Generating m1 random inputs using the uniform distribution function */
{
x[,j]<-runif(n,1.5,25)
}
y<-matrix(nrow=n,ncol=m2)
r1<-runif(m1) /* Generating m1 random coefficients using the uniform distribution function */
r2<-sum(r1)
r3<-r1/r2 /* Scaling the exponents so that they sum to 1 */
for(i in 1:n3) /* Generating inefficient points */
{
y[i]=prod((x[i,]^r3))/30
}
for(j in n4:n) /* Generating efficient points */
{
y[j]=prod((x[j,]^r3))
}
```

R code to solve the input-oriented CRS model using GBA

```
s1<-matrix(nrow=n,ncol=1) /* Beginning of initialisation */
s2<-matrix(nrow=n,ncol=1)
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s5<-matrix(nrow=n,ncol=2)
ni<-m1
no<-m2
notie<-0
tieunits<-0
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=x
a[1:n,m3:m]=-y
dm<-cbind(x,y)
rc<-rowSums(dm)
rc1<-max(rc)
rc2<-min(rc)
par1<-(rc1-rc2)
e<-c(1:n)
a<-c(e,a)
```

```
dim(a)<-c(n,m+1)
s2<--(a[,m5:m7])/rowSums(a[,2:m3])
maxs2<-max(s2)
q1<-which.max(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
d4<-ncol(a4)
g<-1 /* End of initialisation */
tim1<-proc.time()
while(sq>=1) /* Beginning of the GBA procedure */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m1]=0
b2[,m3:m]=- a3[1,m5:m7]
c2<-matrix(nrow=1,ncol=m)
c2[,1:m1]= a3[1,2:m3]
c2[,m3:m]=0
abc2<-cbind(t(c2),-a4[-1,])
lt2<-(c(1,rep(0,d4)))
f.dir <-c(rep(">=",m))
vtr2<-lp(direction="min",objective.in=lt2,const.mat=abc2,const.dir=f.dir,const.rhs=t(b2),compute.sens=1,scale=3)
s3<--(a3[,m5:m7]*vtr2$duals[m3:m])/(a3[,2:m3]%*%vtr2$duals[1:m1])
sasz<-max(s3)
if(sasz<=1)
{
```

```
sq<-nrow(a3)
s5[g,]<-c(a3[1,1],vtr2$solution[1])
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
if(sasz>1)
{
ms3<-max(s3)
smx3<-sum(s3==ms3)
if(smx3>=2) /* Tie breaking routine to identify one P-K efficient unit among the tied units */
{
notie<-notie+1
tieunits<-tieunits+smx3
s4<--(a3[,m5:m7]*vtr2$duals[m3:m])-(a3[,2:m3]%*%vtr2$duals[1:m1])
s3tie<-sort(s4,decreasing=TRUE)
par5<-s3tie[smx3+1]
par6<-ms3-par5
par7<-par1/par6
duals3<-(vtr2$duals[1:m]*par7)+1
s3<--(a3[,m5:m7] *duals3[m3:m])/(a3[,2:m3]%*%duals3[1:m1])
}
sq<-nrow(a3)
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
q4<-which.max(sx3[,m8])
```

```
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
} /* End of GBA procedure */
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by GBA to solve the dataset */
notie /* Number of ties encountered */
tieunits /* Average number of tied units per tie */
```

<u>R code to solve the input-oriented CRS model using BuildHull</u>

```
s1<-matrix(nrow=n,ncol=1) /* Beginning of Initialisation */
s2<-matrix(nrow=n,ncol=1)
m3<-m1+1
m4<-m2+1
m5<-m3+1
```

```
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
s5<-matrix(nrow=n,ncol=m+1)
ni<-m1
no<-m2
notie<-0
tieunits<-0
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=x
a[1:n,m3:m]=-y
dm<-cbind(-x,y)
rc<-rowSums(dm)
rc1<-max(rc)
rc2<-min(rc)
par1<-rc1-rc2
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<--(a[,m5:m7])/rowSums(a[,2:m3])
maxs2<-max(s2)
q1<-which.max(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
sq<-nrow(a3)
```

```
d4<-ncol(a4)
g<-1 /* End of initialisation */
tim1<-proc.time()
while(sq>=1) /* Beginning of Phase-1 of BuildHull */
{
b2<-matrix(nrow=1,ncol=m)
b2[1,1:m]<-1
c2<-matrix(nrow=1,ncol=m)
c2[1,]=-a3[1,-1]
c3<-rbind(t(c2))
abc2<-cbind(t(b2),-a4[-1,])
abc3<-rbind(abc2)
lt2<-(c(1,rep(0,d4)))
f.dir <-c(rep(">=",m))
vtr2<-lp(direction="min",objective.in=lt2,const.mat=abc3,const.dir=f.dir,const.rhs=c3,compute.sens=1,scale=3)
s3<--(a3[,m5:m7]*vtr2$duals[m3:m])/(a3[,2:m3]%*%vtr2$duals[1:m1])
if(vtr2$solution[1]==0)
{
sq<-nrow(a3)
s5[g,]<-a3[1,]
a3<-a3[-1,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
else
if(vtr2$solution[1]>0)
{
```

```
sq<-nrow(a3)
if(sq==1)
{
s3<--(a3[,m5:m7]*vtr2$duals[m3:m])/(a3[,2:m3]%*%vtr2$duals[1:m1])
}
else
if(sq>1)
{
s3<--(a3[,m5:m7]*vtr2$duals[m3:m])/(a3[,2:m3]%*%vtr2$duals[1:m1])
}
ms3<-max(s3)
smx3<-sum(s3==ms3)
if(smx3>=2) /* Tie breaking routine to identify one extreme-efficient units among the tied units */
{
notie<-notie+1
tieunits<- tieunits+smx3
s4<--(a3[,m5:m7]*vtr2$duals[m3:m])-(a3[,2:m3]%*%vtr2$duals[1:m1])
s3tie<-sort(s4,decreasing=TRUE)
par5<-s3tie[smx3+1]
par6<-ms3-par5
par7<-par1/par6
duals3<-vtr2$duals[1:m7]*par7
duals3[1:m]<-duals3[1:m]+1
s3<--(a3[,m5:m7]*duals3[m3:m])/(a3[,2:m3]%*%duals3[1:m1])
}
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
```

```
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
} /* End of Phase-1 of BuildHull */
s6<-s5[complete.cases(s5),]
sx<-ncol(a4)
zx<-(n-sx)
if(zx==1)
{s6<-t(s6)}
s11<-matrix(nrow=zx,ncol=2)
a5<-matrix(nrow=sx,ncol=m)
a5=a4[-1,]
for(i in 1:zx) /* Beginning of Phase-2 of BuildHull */
{
b<-matrix(nrow=1,ncol=m)
b[,1:m1]=0
b[,m3:m]= -s6[i,m5:m7]
```

```
b2<-rbind(t(b))
c<-matrix(nrow=1,ncol=m)
c[,1:m1]=s6[i,2:m3]
c[,m3:m]=0
c1<-t(c)
abc<-cbind(c1,-a5)
abc3<-rbind(abc)
lt3<-c(1,rep(0,sx))
f.dir <-c(rep(">=",m))
vtr3<-lp(direction="min",objective.in=lt3,const.mat=abc3,const.dir=f.dir,const.rhs=b2,compute.sens=1,scale=3)
s11[i,]<-cbind(s6[i,1],vtr3$solution[1])
} /* End of Phase-2 of BuildHull */
tim2<-proc.time()
tim<-tim2-tim1
tim /* Total time taken by BuildHull to solve the dataset */
notie /* Number of ties encountered */
tieunits /* Average number of tied units per tie */
```

<u>R code to solve the input-oriented CRS model using the standard DEA algorithm</u>

```
s1<-matrix(nrow=n,ncol=1) /* Beginning of initialisation */
s2<-matrix(nrow=n,ncol=1)
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
```

```
m8<-m7+1
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=-x
a[1:n,m3:m]=y
tim1<-proc.time()
for(i in 1:n) /* Beginning of Phase-1 of the conventional algorithm */
{
b<-matrix(nrow=1,ncol=m)
b[,1:m1]=t(x[i,])
b[,m3:m]=rep(0,m2)
c<-matrix(nrow=1,ncol=m)
c[,1:m1]=rep(0,m1)
c[,m3:m]=y[i,]
c1<-t(c)
abc<-cbind(t(b),t(a))
lt<-(c(1,rep(0,n)))
f.dir <-c(rep(">=",m))
vtr<-lp(direction="min",objective.in=lt,const.mat=abc,const.dir=f.dir,const.rhs=c1, scale=3)
s1[i,]<-vtr$solution[1]
} /* End of Phase-1 of the conventional algorithm */
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by Phase-1 of the conventional algorithm */
for(j in 1:n) /* Beginning of Phase-2 of the conventional algorithm */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m1]=t(x[j,])*s1[j]
b2[,m3:m]=y[j]
b3<-t(b2)
sl1<-matrix(nrow=m,ncol=m1)
sl1[1:m1,1:m1]=diag(m1)
sl1[m3:m,]=rep(0,m2)
sl2<-matrix(nrow=m,ncol=m2)
sl2[1:m1,]=rep(0,m1)
sl2[m3:m,1:m2]=-diag(m2)
abcd<-cbind(sl1,sl2,abs(t(a)))
f.dir2<-c(rep("==",m))
lt2<-c(rep(1,m),rep(0,n))
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abcd,const.dir=f.dir2,const.rhs=b3, scale=3)
s2[j,]<-sum(vtr2$solution[1:m])
} /* End of Phase-2 of the conventional algorithm */
tim3<-proc.time()
tim<-tim3-tim1
tim /* Total time taken by the conventional algorithm to solve the dataset */
```

Appendix 10 - Computational performance of the competitive algorithms in solving the input-oriented CRS model

Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)	Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)
5-2500-01	7.39	4.17	88.47	42.65	**10-2500-01**	7.92	5.11	169.12	79.51
5-2500-13	13.91	11.03	95.14	49.28	**10-2500-13**	20.44	17.74	179.75	88.04
5-2500-25	23.61	20.75	98.02	51.50	**10-2500-25**	37.24	34.49	186.94	93.58
5-2500-50	50.90	46.89	101.11	53.67	**10-2500-50**	81.78	78.95	196.55	100.08
5-5000-01	17.30	9.91	360.03	177.92	**10-5000-01**	21.05	13.77	703.29	335.75
5-5000-13	44.78	39.39	417.00	224.88	**10-5000-13**	73.38	68.57	760.19	378.89
5-5000-25	84.30	77.74	424.32	230.90	**10-5000-25**	142.10	133.70	787.36	400.57
5-5000-50	198.39	186.05	416.85	226.68	**10-5000-50**	334.96	316.42	809.40	409.60
5-7500-01	28.11	17.90	801.10	401.21	**10-7500-01**	36.89	26.14	1608.44	783.13
5-7500-13	99.48	89.30	887.40	473.79	**10-7500-13**	154.78	149.53	1705.75	861.55
5-7500-25	186.54	172.58	893.61	482.91	**10-7500-25**	316.92	310.07	1774.93	901.87
5-7500-50	445.16	421.05	926.98	507.05	**10-7500-50**	749.86	734.86	1826.36	924.44
5-10000-01	41.45	27.20	1469.97	741.75	**10-10000-01**	57.59	43.28	2843.33	1393.08
5-10000-13	164.47	151.49	1590.69	854.00	**10-10000-13**	273.25	263.96	3120.25	1614.14
5-10000-25	334.11	314.89	1693.55	957.99	**10-10000-25**	549.92	529.57	3181.53	1651.67
5-10000-50	797.75	766.20	1751.50	989.38	**10-10000-50**	1340.25	1296.34	3341.36	1760.05

Computational performance of GBA, BuildHull and Standard two-phase procedure in solving the input-oriented CRS model (contd.)

Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)	Dataset	BuildHull (secs)	GBA (secs)	1st&2ndPhases (secs)	1stPhase (secs)
15-2500-01	9.54	5.92	255.34	112.98	**20-2500-01**	10.06	6.64	377.23	154.98
15-2500-13	27.86	25.33	303.04	137.95	**20-2500-13**	34.79	32.33	1254.18	178.41
15-2500-25	52.18	48.86	336.07	143.33	**20-2500-25**	69.00	65.39	3683.23	190.03
15-2500-50	116.28	111.16	429.17	151.19	**20-2500-50**	153.29	148.33	9658.47	193.37
15-5000-01	23.41	17.45	1064.31	476.75	**20-5000-01**	27.60	21.49	1572.37	653.98
15-5000-13	105.83	101.59	1312.29	603.76	**20-5000-13**	136.25	130.40	5431.12	780.84
15-5000-25	202.96	200.42	1491.36	614.42	**20-5000-25**	279.06	268.50	16344.87	814.61
15-5000-50	468.60	458.04	1879.12	615.37	**20-5000-50**	619.78	600.58	42289.59	787.05
15-7500-01	44.71	34.31	2466.61	1132.08	**20-7500-01**	53.89	43.92	3644.08	1552.93
15-7500-13	228.50	223.60	3011.96	1313.95	**20-7500-13**	301.69	295.06	12465.48	1699.32
15-7500-25	454.78	448.17	3474.16	1376.60	**20-7500-25**	663.53	630.34	38075.79	1825.13
15-7500-50	1085.03	1034.95	4906.78	1437.17	**20-7500-50**	1414.70	1409.30	110427.07	1838.12
15-10000-01	70.66	56.33	4432.20	2047.99	**20-10000-01**	88.21	74.92	6547.97	2809.32
15-10000-13	395.05	387.67	5451.03	2447.23	**20-10000-13**	565.17	540.42	22559.97	3164.99
15-10000-25	812.51	809.00	6427.69	2546.34	**20-10000-25**	1147.36	1110.31	70445.62	3375.99
15-10000-50	1946.52	1858.16	9120.21	2483.81	**20-10000-50**	2542.44	2500.10	205250.31	3176.76

Appendix 11 - R code to solve the output-oriented VRS model with built-in subroutine to construct strictly positive multiplier values for the generators

```
dudat<-read.xls("H:/d5.xls",type="double",colNames=FALSE) /*Reads a dataset from the current directory */
n<-dudat[1,1] /* Beginning of initialisation */
n2<-n+1
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
m1<-dudat[1,2]
m2<-dudat[1,3]
m3<-m1+1
m4<-m2+1
m5<-m3+1
m6<-m4+1
m<-m1+m2
m7<-m+1
m8<-m7+1
notie<-0
tieunits<-0
slack<-matrix(nrow=n,ncol=m)
weight1<-matrix(nrow=n,ncol=m+1)
gens<-matrix(nrow=n,ncol=m8)
inprsav<-matrix(nrow=n,ncol=m1)
peerf<-matrix(nrow=n,ncol=m7)
s5<-matrix(nrow=n,ncol=m+1)
ni<-m1
no<-m2
```

```
x<- dudat[2:n2,1:m1]
y<- -dudat[2:n2,m3:m]
a<-matrix(nrow=n,ncol=m)
a[1:n,1:m1]=x
a[1:n,m3:m]=y
pen<--2
s1<-matrix(nrow=n,ncol=1)
s2<-matrix(nrow=n,ncol=1)
s5<-matrix(nrow=n,ncol=2)
ni<-m1
no<-m2
notie<-0
tieunits<-0
dm<-cbind(-x,-y)
rc<-rowSums(dm)
rc1<-max(rc)
rc2<-min(rc)
par1<-(rc1-rc2)
e<-c(1:n)
a<-c(e,a)
dim(a)<-c(n,m+1)
s2<-abs(rowSums(a[,m5:m7]))-rowSums(a[,2:m3])
maxs2<-max(s2)
q1<-which.max(s2)
a4<-t(a[q1,])
dim(a4)<-c(m+1,1)
a3<-a[-q1,]
a11<-a3
```

```
sq<-nrow(a3)
d4<-ncol(a4)
gn<-1
gens[gn,]<-c(a4[gn,1],rep(1,m),0)
g<-1
gn<-2 /* End initialisation */
tim1<-proc.time()
while(sq>=1) /* Beginning of GBA procedure */
{
b2<-matrix(nrow=1,ncol=m)
b2[,1:m1]=a3[1,2:m3]
b2[,m3:m]=0
b3<-rbind(t(b2),1)
b21<-matrix(nrow=1,ncol=m)
b21[,1:m1]=a3[1,2:m3]
b21[,m3:m]=a3[1,m5:m7]
c2<-matrix(nrow=1,ncol=m)
c2[,1:m1]=0
c2[,m3:m]=-a3[1,m5:m7]
c3<-rbind(t(c2),0)
asd<-cbind(t(b21),a4[-1,])
asd2<-rbind(asd,1)
abc2<-cbind(c3,asd2)
lt2<-(c(1,pen,rep(0,d4)))
f.dir <-c(rep("<=",m),"==")
vtr2<-lp(direction="max",objective.in=lt2,const.mat=abc2,const.dir=f.dir,const.rhs=b3,compute.sens=1, scale=3)
s3<-round(-(a3[,m5:m7]%*%vtr2$duals[m3:m])-vtr2$duals[m7]-(a3[,2:m3]%*%vtr2$duals[1:m1]),digits=10)
duals3<-vtr2$duals[1:m7]
```

```
duals3m<-vtr2$duals[1:m]
dim(duals3)<-c(1,m7)
dim(duals3m)<-c(1,m)
sasz<-max(s3)
if(sasz<=0)
{
sq<-nrow(a3)
s5[g,]<-c(a3[1,1],1/vtr2$solution[1])
peer<-matrix(nrow=1,ncol=d4)
for(j in 1:d4){if(vtr2$solution[j+2]>0){peer[j]<-a4[1,j]}}
peer2<-t(peer)
peer3<-peer2[complete.cases(peer2),]
dim(peer3)<-c(1,length(peer3))
peer4<-matrix(nrow=1,ncol=m)
peer4<-c(peer3,rep(0,m-length(peer3)))
peerf[g,]<-c(a3[1,1],peer4)
sol<-vtr2$solution[3:length(vtr2$solution)]
sol<-replace(sol,sol=="0",NA)
sol<-na.exclude(sol)
if(ncol(peer3)==1){compu<-(a[peer3,-1]*sol[1:length(peer3)])}
if(ncol(peer3)>1){compu<-colSums(a[peer3,-1]*sol[1:length(peer3)])}
projp<-matrix(nrow=1,ncol=m)
projp[1:m1]<-(abs(a3[1,2:m3])*vtr2$solution[1])
projp[m3:m]<-abs(a3[1,m5:m7])
inprsav[g,]<-abs(a3[1,2:m3])-projp[1:m1]
slack[g,]<-abs(projp-abs(compu))
weight1[g,]<-vtr2$duals[1:m7]
a3<-a3[-1,]
```

```
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
g<-g+1
}
if(sasz>0)
{
ms3<-max(s3)
smx3<-sum(s3==ms3)
if(smx3>=2) /* Tie breaking routine to identify one P-K efficient unit among tied units */
{
notie<-notie+1
tieunits<-tieunits+smx3
s3tie<-sort(s3,decreasing=TRUE)
par5<-max(s3tie[smx3+1],0)
par6<-ms3-par5
par7<-abs(par1/par6)+1
duals3<-vtr2$duals[1:m7]*par7
duals3[1:m]<-duals3[1:m]+1
s3<-(-(a3[,m5:m7]%*%duals3[m3:m]))-(a3[,2:m3]%*%duals3[1:m1])-duals3[m7]
duals3m<-duals3[1:m]
dim(duals3m)<-c(1,m)
}
if(smx3==1)
{
if(apply(duals3m,1,prod)==0) /* Subroutine to generator positive multiplier values for the generators */
{
s3tie<-sort(s3,decreasing=TRUE)
if(length(s3tie>1)){ par5<-max(s3tie[smx3+1],0)}
if(length(s3tie==1)){par5<-0}
par6<-ms3-par5
par7<-abs(par1/par6)+1
duals3<-vtr2$duals[1:m7]*par7
duals3[1:m]<-duals3[1:m]+1
s3<-(-(a3[,m5:m7]%*%duals3[m3:m]))-(a3[,2:m3]%*%duals3[1:m1])-duals3[m7]
}
}
sq<-nrow(a3)
sx3<-cbind(a3,s3)
dim(sx3)<-c(sq,m8)
sd<-max(sx3[,m8])
q4<-which.max(sx3[,m8])
sk<-sx3[q4,]
dim(sk)<-c(1,m8)
sk1<-sk[,-m8]
dim(sk1)<-c(1,m7)
a4<-cbind(a4,t(sk1))
a4<-unique(t(a4))
a4<-t(a4)
gens[gn,]<-c(a4[1,gn],duals3)
gn<-gn+1
sdxc<-sk1[1]
a3<-a3[-q4,]
dim(a3)<-c(sq-1,m+1)
sq<-sq-1
d4<-ncol(a4)
}
```

```
} /* End of GBA procedure */
s6<-s5[complete.cases(s5),]
tim2<-proc.time()
tim<-tim2-tim1
tim /* Time taken by the GBA to solve the dataset */
notie /* Number of ties encountered by GBA */
tieunits /* Average number of tied units per tie */
```

BIBLIOGRAPHY

Afriat, S. (1972). Efficiency estimation of production functions. *International Economic Review* 13(3), 568-598.

Agrell, P. J., and Tind, J. (2001). A dual approach to nonconvex frontier models. *Journal of Productivity Analysis* 16 (2), 129 - 147.

Ali, A.I. and Seiford, L.M. (1990). Translation invariance in data envelopment analysis. *Operations Research Letters* 9, 403-405.

Ali, A. I. (1993). Streamlined computation for data envelopment analysis. *European Journal of Operational Research* 64, 61 –67.

Ali, A. I. (1994). Computational Aspects of DEA. In: Charnes A., Cooper, W.W., Lewin A., Seiford, L.M., editors. Data envelopment analysis, theory, methodology and applications, 63-88.

Andersen, P. and Petersen, N.C. (1993). A procedure for ranking efficient units in data envelopment analysis. *Management Science* 39, 1261-12264.

Appa, G. and Parthasarathy, S. (2006a). 2nd report submitted to the DfES.

Appa, G. and Parthasarathy, S. (2007). A faster algorithm for solving DEA models. *Proceedings of the 5th International Symposium on DEA*.

Appa, G., Argyris, N., and Parthasarathy, S. (2006b). A faster and trimmer algorithm based on generators for solving DEA problems. *LSE Working Paper LSEOR 06.84*, ISBN: 07530 2076 9.

Aparicio, J., Ruiz, J. L., and Sirvent, I. (2007). Closest targets and minimum distance to the Pareto-efficient frontier in DEA. *Journal of Productivity Analysis* 28 (3), 209 -218.

Argyris, N. (2008). Polyhedral attributes of Production Possibility Sets in Data Envelopment Analysis, with applications to Sensitivity Analysis and Cross-Evaluation Methodologies. PhD thesis, London School of Economics.

Andersen, P., and Petersen, N. C. (1993). A procedure for ranking efficient units in Data Envelopment Analysis. *Management Science* 39 (10), 1261 – 1264.

Banker, R. D., Charnes, A., and Cooper, W. W. (1984). Some models for the estimation of technical and scale inefficiencies in data envelopment analysis. *Management Science* 30, 1078-1092.

Banker, R. D. (1984). Estimating most productive scale size using data envelopment analysis. *European Journal of Operational Research* 17, 35 – 44.

Banker, R.D. and Gifford, J.L. (1988). A relative efficiency model for the evaluation of public health nurse productivity. Mellon University Mimeo, Carnegie.

Banker, R. D. and Thrall, R. M. (1992). Estimation of returns to scale using data envelopment analysis. *European Journal of Operational Research* 62, 74–84.

Banker, R. D., Cooper, W. W., Seiford, L., Thrall, M., and Zhu, J. (2004). Returns to Scale in different DEA models. *European Journal of Operational Research* 154, 345-362.

Banker, R. D. and Chang, H. (2006). The super-efficiency procedure for outlier identification, not for ranking efficient units. *European Journal of Operational Research* 175, 1311-1320.

Banker, R. D. and Parthasarathy, S (2009). Estimation and inference in two-stage, semi-parametric models of production processes: a comment. In preparation.

Banker, R. D., Natarajan, R, and Parthasarathy, S (2010). Nonparametric Estimation and Statistical Tests of Components of Productivity Change. In preparation.

Barr, R. S. and Durchholz, M. L. (1997). Parallel and hierarchical decomposition approaches for solving large-scale DEA models. *Annals of Operational Research* 73, 339 – 372.

Boles, J. N. (1971). The 1130 Farrell Efficiency System — Multiple Products, Multiple Factors. *Giannini Foundation of Agricultural Economics*, February 1971.

Bougnol, M. L., Dula, J.H., and Rouse, P. (2010). Interior point methods in DEA to determine non-zero multiplier weights. In preparation.

Charnes, A., Cooper, W. W., and Rhodes, E. (1978). Measuring the efficiency of decision making units. *European Journal of Operational Research* 2, 429-441.

Charnes, A, Cooper, W.W, and Rhodes, E. (1979). Short communications: measuring the efficiency of decision-making units. *European Journal of Operational Research* 3 (4), 339.

Charnes, A., Cooper, W.W, and Rhodes, E. (1981). Evaluating program and managerial efficiency: an application of data envelopment analysis to program follow through. *Management Science* 27 (6), 668 - 697.

Charnes, A., and Cooper, W. W. (1984). The Non-Archimedean CCR Ratio for Efficiency Analysis: A Rejoinder to Boyd and Fare. *European Journal of Operational Research* 15, 333-334.

Charnes, A., Cooper, W. W., Golany, B. and Stutz, J. (1985). Foundations of Data Envelopment Analysis for Pareto-Koopmans efficient empirical production functions. *Journal of Econometrics* 30, 91 – 107.

Charnes, A., and Cooper, W. W. (1984). Preface to topics in Data Envelopment Analysis. *Annals of Operational Research* 2, 59-94.

Charnes, A., Cooper, W. W., and Thrall, R. M. (1991). A structure for classifying and characterizing efficiency and inefficiency in Data Envelopment Analysis. *Journal of Productivity Analysis* 2, 197 - 237

Charnes, A., S. Haag, P. Jaska and J. Semple (1992). Sensitivity of efficiency classifications in the additive model of data envelopment analysis. *International Journal Systems Science* 23, 789-798.

Charnes, A., Rousseau, J. J., and Semple, J. H. (1996). Sensitivity and stability of efficiency classifications in data envelopment analysis. *Journal of Productivity Analysis* 7, 5-18.

Chen, Y. (2005). Measuring super-efficiency in DEA in the presence of infeasibility. *European Journal of Operational Research* 161, 545-551.

Chen, Y. and Ali, A. I. (2002). Output-input ratio analysis and DEA frontier. *European Journal of Operational Research* 142, 476 – 479.

Chen, W.C. and Cho, W.J. (2009). A procedure for large-scale DEA computations. *Computers and Operations Research* 36 (6), 1813 – 1824.

Cobb, C. W. and Douglas, P. H. (1928). A Theory of Production. *American Economic Review* 18, 139–165.

Coelli, T. (1998). A multi-stage methodology for the solution of orientated DEA models. *Operations Research Letters* 23 (3-5), 143 – 149.

Cook, W.D., Liang, L., Zha, Y., and Zhu, J. (2008). A modified super-efficiency DEA model for infeasibility. *Journal of the Operational Research Society* 60 (2), 276-281.

Cooper, W. W, Seiford, L. M., and Tone, K (2000). Data Envelopment Analysis. Kluwer Academic publications.

Cooper, W. W., Park, S. K., and Ciurana, P. J. T. (2000). Marginal rates and Elasticities of substitution with additive models in DEA. *Journal of Productivity Analysis* 13, 105–123.

Cooper, W. W., Ruiz, J. L., Sirvent, I (2007). Choosing weights from alternative optimal solutions of dual multiplier models in DEA. *European Journal of Operational Research* 180 (1), 443-458.

Debreu, G. (1951). The Coefficient of Resource Utilization. *Econometrica 19,* 14–22.

Deprins, D., Simar, L., and Tulkens, H (1984). Measuring Labor Efficiency in Post Offices. In Marchand, M., Pestieau, P., and Tulkens, H. (eds.), The Performance of Public Enterprises: Concepts and Measurements. Amsterdam: North Holland.

Dula, J. H., and Helgason, R. V. (1996). A new procedure for identifying the frame of the convex hull of a finite collection of points in multidimensional space. *European Journal of Operational Research* 92, 352-367.

Dula, J. H. and Hickman, B. L. (1997). Effects of excluding the column from being scored from the DEA envelopment LP technology matrix. *Journal of the Operational Research Society* 48, 1001 – 1012.

Dula, J. H. (1998, 2007, 2010). An algorithm for the DEA. Working paper, School of Business, University of Mississippi, MS 38677.

Dula, J. H., Helgason, R. V., and Venugopal, N. (1998). An algorithm for finding the frame of a pointed finite conical hull. *INFORMS Journal of Computing* 10, 323 – 300.

Dula, J. H. and Thrall, R. M. (2001). A Computational Framework for Accelerating DEA. *Journal of Productivity Analysis* 16, 63–78.

Dula, J.H. and Lopez, F.J. (2006). Algorithms for the frame of a finitely generated unbounded polyhedron. *INFORMS Journal on Computing* 18 (1), 97 - 110.

Dula, J. H. (2008). A Computational study of DEA with massive data sets. *Computers and Operations Research* 35 (4), 1191 – 1203.

Dula, J.H. and Lopez, F.J. (2009). Preprocessing DEA. *Computers and Operations Research* 36 (4), 1204 – 1220.

Emrouznejad, A., Parker, B. R., and Tavares, G. (2008). Evaluation of research in efficiency and productivity: A survey and analysis of the first 30 years of scholarly literature in DEA. *Socio-Economic Planning Sciences* 42 (3), 151 – 157.

Emrouznejad, A., Anouze, A. L., and Thanassoulis, E. (2010). A semi-oriented radial measure for measuring the efficiency of decision making units with negative data, using DEA. *European Journal of Operational Research* 200 (1), 297 – 304.

Färe, R., Grosskopf, S., and Logan, J. (1983). The relative efficiency of Illinois electric utilities. *Resources and Energy* 5, 349-367.

Färe, R. and Grosskopf, S. (1994). Comment On: Estimation of returns to scale using data envelopment analysis. *European Journal of Operational Research* 79, 379-382.

Fare, R., Grosskopf, S., Lindgren, B., and Roos, P. (1994). Productivity Developments in Swedish Hospitals: A Malmquist Output Index Approach. In: Charnes, A., Cooper, W., Lewin, A. Seiford, L., (Editors), Data Envelopment Analysis: Theory, Methodology and Applications. Kluwer Academic Publishers, Boston: 253-172.

Farrell, M. J. (1957). The Measurement of Productive Efficiency of Production. *Journal of the Royal Statistical Society Series* A, 120(III), 253–281.

Farrell, M. J. and Fieldhouse, M. (1962). Estimating Efficient Productions Functions under Increasing Returns to Scale. *Journal of the Royal Statistical Society* 125, 252–267.

Forsund, F. R. and Sarafoglou, N. (2002). On the origins of data envelopment analysis. *Journal of Productivity Analysis* 17 (1-2), 23 – 40.

Fried, H. O., Lovell, C. A. K., and Schmidt, S. S. (2008). The Measurement of Productive Efficiency and Productivity Growth. Oxford University Press.

Gattoufi, S., Oral, M., and Reisman, A. (2004). Data Envelopment Analysis literature: a bibliography update (1996–2001). *Socio-Economic Planning Sciences* 38 (2–3), 122–159.

Gonzales-Lima, M.D., Tapia, R.A., and Thrall, R.A., 1996. On the construction of strong complementarity slackness solutions for DEA linear programming problems using a primal-dual interior-point method. *Annals of Operations Research* 66, 139 -162.

Koopmans, T. C. (1951). Activity Analysis of Production and Allocation. New York: Wiley.

Korhonen, P. J. and Siitari, P. A. (2009). A dimensional decomposition approach to identifying efficient units in large-scale DEA models. *Computers and Operations Research* 36 (1), 234 – 244.

Kuosmanen, T. and Post, G. T. (1999). Robust Efficiency Measurement: Dealing with Outliers in Data Envelopment Analysis. *Rotterdam Institute for Business Economic Studies Report* 9911.

Lovell, C. A. K. and Rouse, A. P. B. (2003). Equivalent standard DEA models to provide super-efficiency scores. *Journal of the Operational Research Society* 54, 101-108.

Olesen, O. B., and Petersen, N. C. (1996). Indicators of Ill-Conditioned Data Sets and Model Misspecification in Data Envelopment Analysis: An Extended Facet Approach. *Management Science* 42 (2), 205-219.

Pareto, V. (1906, 1971). Manual of Political Economy. Societa' Editrice Libraria, Milan. Reprint. Augustus Kelley, New York.

Pastor, T. (1996). Translational invariance in DEA : A generalization. *Annals of Operations Research* 66 (2), 93 - 102.

Portela, M.C.A.S., Thanassoulis, E., and Simpson, G. (2004). A directional distance approach to deal with negative data in DEA: An application to bank branches. *Journal of Operational Research Society* 55 (10), 1111–1121.

Ray, S., and Desli, E. (1997). Productivity Growth, Technical Progress, and Efficiency Change in Industrialized Countries: Comment. *American Economic Review* 87, 1033-1039.

Rousseau, J.J. and J.H. Semple (1995). Two-person ratio efficiency games. *Management Science* 41, 435-441.

Russell, R. R. (1988). On the Axiomatic Approach to the measurement of technical efficiency. In: W. Eichorn, ed., Measurement in Economics: Theory and Applications of Economic Indices. Heidelberg: Physica – Verlag.

Russell, R. R. (1990). Continuity of Measures of Technical Efficiency. *Journal of Economics Theory* 51 (2), 255 – 267.

Seiford, L. M. and Zhu, J. (1998a). Stability regions for maintaining efficiency in data envelopment analysis. *European Journal of Operational Research* 108 (1), 127 – 139.

Seiford, L. M. and Zhu, J. (1998b). Sensitivity analysis of DEA models for simultaneous changes in all the data. *Journal of the Operational Research Society* 49 (10), 1060 – 1071.

Seiford, L. M. and Zhu, J. (1999). Infeasibility of super-efficiency data envelopment analysis models. *INFOR Information Systems and Operational Research* 37 (2), 174 – 187.

Sharp, J.A., Liu, W.B., and Meng, W. (2006). A modified slacks-based measure model for data envelopment analysis with 'natural' negative outputs and inputs. *Journal of Operational Research Society* 57 (11), 1–6.

Shephard, R. W. (1953). Cost and Production Functions. Princeton University Press, Princeton, N.J.

Shephard, R. W. (1970). The Theory of Cost and Production Functions, Princeton University Press, Princeton, N.J..

Simar, L. and Wilson, P. (1998). Sensitivity analysis of efficiency scores: How to bootstrap in nonparametric frontier models. *Management Science*, 44, 49-61.

Simar, L. and Wilson, P. (1999). Estimating and bootstrapping Malmquist Indices. *European Journal of Operational Research*, 115 (3), 459-471.

Simar, L. and Wilson, P. (2000). A general methodology for bootstrapping in non-parametric frontier models. *Journal of Applied Statistics*, 27, 779-802.

Simar, L. (2003). Detecting Outliers in Frontier Models : A Simple Approach. *Journal of Productivity Analysis*, 20 (3), 391-424.

Sueyoshi, T. and Chang, Y. L. (1989). Efficient algorithm for additive and multiplicative model in data envelopment analysis. *Operations Research Letters* 8, 205 - 213.

Sueyoshi, T. (1990). A special algorithm for an additive model in data envelopment analysis. *Journal of the Operational Research Society* 41, 249-257.

Thompson, R. G., Dharmapala, P. S. and Thrall, R. M. (1993). Importance for DEA of zeros in data, multipliers, and solutions. *Journal of Productivity Analysis* 4 (4), 379 - 390.

Thrall, R. M. (1996a). The lack of invariance of optimal dual solutions under translation. *Annals of Operations Research* 66 (2), 103-108.

Thrall, R. M. (1996b). Duality, classification and slacks in DEA. *Annals of Operations Research* 66 (2), 109-138.

Thrall, R. M. (1999). What is the Economic Meaning of FDH?. *Journal of Productivity Analysis* 11, 243–250.

Wilson, P. (1995). Detecting influential observations in data envelopment analysis. *Journal of Productivity Analysis* 6, 27-45.

Xue, M. and Harker, P. T. (2002). Note: Ranking DMUs with Infeasible Super-Efficiency DEA Models. *Management Science* 48 (5), 705–710.

Zhu, J. (1996). Robustness of the efficient DMUs in data envelopment analysis. *European Journal of Operational Research* 90, 451-460.