AD-A283 214

:NTATION PAGE

Form Approved
OMB No. 0704-0188

| | | |
|---|---|---|
| 1. AGENCY USE ONLY (*Leave blank*) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |

4. TITLE AND SUBTITLE
LARGE-SCALE DATA ENVELOPMENT ANALYSIS MODELS AND RELATED APPLICATIONS

5. FUNDING NUMBERS

6. AUTHOR(S)
MATTHEW L. Durchholz

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Southern Methodist University

8. PERFORMING ORGANIZATION REPORT NUMBER
94-026D

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DEPARTMENT OF THE AIR FORCE
AFIT/CI
2950 P STREET
WRIGHT-PATTERSON AFB OH 45433-7765

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release IAW 190-1
Distribution Unlimited
MICHEAL M. BRICKER, SMSgt, USAF
Cheif Administration

12b. DISTRIBUTION CODE
F

13. ABSTRACT (*Maximum 200 words*)

14. SUBJECT TERMS

15. NUMBER OF PAGES
131

16. PRICE CODE

| | | | |
|---|---|---|---|
| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |

NSN 7540-01-280-5500

## GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements.***

**Block 1.** Agency Use Only *(Leave blank)*.

**Block 2.** Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

**Block 3.** Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

**Block 4.** Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

**Block 5.** Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

| | | | |
|---|---|---|---|
| C | - Contract | PR | - Project |
| G | - Grant | TA | - Task |
| PE | - Program Element | WU | - Work Unit Accession No. |

**Block 6.** Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

**Block 7.** Performing Organization Name(s) and Address(es). Self-explanatory.

**Block 8.** Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

**Block 9.** Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

**Block 10.** Sponsoring/Monitoring Agency Report Number. *(If known)*

**Block 11.** Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

**Block 12a.** Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

| | |
|---|---|
| DOD - | See DoDD 5230.24, "Distribution Statements on Technical Documents." |
| DOE - | See authorities. |
| NASA - | See Handbook NHB 2200.2. |
| NTIS - | Leave blank. |

**Block 12b.** Distribution Code.

| | |
|---|---|
| DOD - | Leave blank. |
| DOE - | Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports. |
| NASA - | Leave blank. |
| NTIS - | Leave blank. |

**Block 13.** Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

**Block 14.** Subject Terms. Keywords or phrases identifying major subjects in the report.

**Block 15.** Number of Pages. Enter the total number of pages.

**Block 16.** Price Code. Enter appropriate price code *(NTIS only)*.

**Blocks 17. - 19.** Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

**Block 20.** Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

94-0260D

# LARGE-SCALE DATA ENVELOPMENT ANALYSIS MODELS AND RELATED APPLICATIONS

A Dissertation Presented to the Graduate Faculty of the

School of Engineering and Applied Science

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

*Doctor of Philosophy*

with a

Major in Operations Research

by

Matthew L. Durchholz

(B.S., United States Air Force Academy, 1981)
(M.A., Southern Methodist University, 1987)
(M.S., Southern Methodist University, 1991)

May 21, 1994

LARGE-SCALE DATA ENVELOPMENT ANALYSIS

MODELS AND RELATED APPLICATIONS

Approved by:

Richard S. Barr

DR. RICHARD S. BARR

[illegible]

~~DR.~~ JOSE H. DULA

Richard V. Helgason

DR. RICHARD V. HELGASON

[illegible]

DR. JOSEPH G. HIRSCHBERG

[illegible]

DR. JEFFERY L. KENNINGTON

Marion G Sobol

DR. MARION G. SOBOL

Durchholz, Matthew L. B.S., United States Air Force Academy, 1981
M.A., Southern Methodist University, 1987
M.S., Southern Methodist University, 1991

<u>Large-Scale Data Envelopment Analysis
Models and Related Applications</u>

Advisor: Associate Professor Richard S. Barr

Doctor of Philosophy degree conferred May 21, 1994

Dissertation completed October 18, 1993

This dissertation presents several advances in data envelopment analysis (DEA), a method for assessing the efficiency of decision units through the identification of empirical best-practice frontiers. First, a new hierarchical decomposition approach for solving large-scale problems is described with results of computational testing in both serial and parallel environments, that dramatically reduces the solution time for realistic DEA applications. Second, a new set of models for stratifying and ranking decision units provides important newer insights into the relationships among the units than what was possible with traditional frontier analysis. Because of the intensive computational requirements of these models, their practicality builds on the effectiveness of the hierarchical process. Finally, a new means of assessing the robustness of a decision-unit's efficiency is given which spans all current models and assists managers in their evaluation of process and organizational improvement options. It is expected that these advances will permit practitioners and researchers to be more expansive and ambitious in their use of this important class of models, and will hopefully encourage new and even more exciting applications of DEA.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

## LIST OF TABLES

## ACKNOWLEDGMENTS

The completion of this effort marks the attainment of a goal that I set while I was an undergraduate student at the United States Air Force Academy. The process that allowed me to complete this work came about because of the influence and guidance of many people to whom I am greatly indebted.

My sincere thanks goes to the U.S. Air Force which sponsored all my graduate education. I am particularly grateful to Colonel Daniel Litwhiler and Colonel Raymond Franck for making the opportunity to pursue the Ph.D. possible.

I owe a great debt to my dissertation committee, especially my advisor and friend, Dr. Richard Barr. The committee's guidance and recommendations made the seemingly impossible task of completing a Ph.D. program in three years a reality. I also acknowledge Dr. Lawrence Seiford who offered support and provided valuable recommendations which kept my work on schedule.

Without the encouragement of my family and friends, I would not have had the endurance or drive to complete this work. I appreciate the members of my community group from Fellowship Bible Church who welcomed me as a family member. Their prayers and emotional support gave me the strength to see this effort through.

Finally, I'm thankful to my love, Leila McLendon (soon to be Durchholz). God has blessed me with such a wonderful friend during the most intense periods of my research. I look forward to having her as my wife.

# CHAPTER I

# MEASURING EFFICIENCY WITH DATA ENVELOPMENT ANALYSIS

## 1.1. Overview

In both competitive and regulated markets, measuring the productive efficiency of firms and industries is important to economic theorists, policy makers, and production managers. For example, the measurement of productive efficiency before and after policy changes are enacted on a firm's operations will reveal the effects of those changes. These measures of efficiency can also assist managers in identifying those portions of the production process that, if made more efficient, would increase output, without absorbing more input resources. For private industry, this can allow the competitive firm to increase market share, effectively translate raw resources to consumable outputs, and pass the savings on to the customer. For regulated industries, inefficiencies can be revealed and policy effectiveness measured.

### 1.1.1. Firm Productivity: Early Efficiency Measures

One early approach to efficiency measurement had been to provide a mathematical representation of a firm's economic activity, based on a set of behavioral assumptions. The actual behavior of a firm is compared to the model to determine the effectiveness of the firm at achieving productive efficiency.

Unfortunately, establishing a mathematical representation for even a simple production process can be a difficult task. Since the measured results of the model depend directly on the behavioral assumptions, the validity of the assumptions must be scrutinized carefully. Moreover, in practice, policy makers are seldom able to agree on which factors to include in the model formulation and as different models are employed, results can vary greatly. Consequently the results of the mathematical model may not reflect true economic activity as much as the degree to which they reflect the assumptions made in designing the model.

Another limitation of this modeling approach is that there are few common behavioral assumptions that exist across a variety of industries. Therefore, a new complex mathematical model must be developed for each industrial application. In addition, once a model is established for an industry, the model may be applicable only to a specific time period. This lack of a unifying methodology, based on a limited number of assumptions, for all industries and time frames, proved to be unsatisfactory. Therefore, this approach to measuring efficiency has not achieved wide acceptance.

Because of the above limitations, other approaches were developed which did not involve building mathematical models, but centered directly on measuring the efficiency of a firm by observing the quantity of inputs consumed and the quantity of outputs produced. Those firms producing more output per unit of input, compared to other firms, would naturally be considered more efficient. Such engineering-style ratios of output to input could also be used to rank the firms by order of efficiency.

For the case of a single input and a single output this ratio analysis is straightforward. But for the more common case of a firm with multiple inputs and multiple outputs, the measurement task becomes more complex. One approach has been to form a variety of simple ratios and rank the firms using a prioritization of the ratios. Determining the relevant ratios and assigning the appropriate priority to each proved to be tenuous. Attempts to aggregate the outputs into a single weighted measure and form a ratio with aggregated inputs has also been offered. However, no consensus emerged regarding the appropriate weighted index value to apply to each output and input.

## 1.1.2. Modern Efficiency Measurement

In 1957, M. J. Farrell [51] proposed a solution to the problem of identifying a single measure of productive efficiency for processes which transform multiple inputs into multiple outputs. Based on extensions of ratio analysis, he was able to avoid the index number problems that characterized previous approaches. Because of the computational resources available at the time, Farrell restricted his attention to applications involving a single output but with multiple inputs.

As computer technology advanced, Farrell's methodology for determining productive efficiency drew increased attention. In 1978, Charnes, Cooper, and Rhodes [37] used fractional programming techniques to demonstrate the equivalence of Farrell's non-linear measure to a well-defined linear-programming formulation. Whereas Farrell's metric had been limited to the case of a single output because of the complexity of solving the more generalized approach, the Charnes, Cooper, and Rhodes (CCR) method offered an elegant reformulation and solution procedure for the multiple-output multiple-input case. Theorists, practitioners, policy makers, and managers now had a means of measuring the relative efficiency for all firms of interest. This technique is called *data envelopment analysis* (DEA). Since the original CCR paper, over 400 articles have been published regarding usage and extensions of DEA [80]. This growing acceptance among theoreticians and practitioners reflects an emerging interest in obtaining a single efficiency measure to evaluate a firm's productivity.

As DEA has gained in popularity, the need to examine the computational characteristics of the DEA formulations has become evident. Some general-purpose linear programming codes, when applied to DEA, have proven to be unreliable, computationally expensive (in terms of time to solution), and limited in the size of problems that can be solved; as a result, they limit the full usefulness of the advantages that the DEA approach offers. Surprisingly, with the explosive growth of DEA over the last decade, little has been written to specifically address the issues concerning the computational aspects of DEA models.

### 1.1.3. Objectives of the Research

This dissertation contributes to the development of the DEA methodology by examining the computational issues of the foundational models. One purpose of this work is to identify the special structural characteristics associated with the DEA formulations which, when exploited, can result in significant computational improvements in solution times.

As newer computer technology is used, these improvements become more relevant. Experience indicates that advances in mathematical programming have been closely linked to advances in computer technology. Consequently, the breadth of applications for data envelopment analysis will grow wider as the computational advantages brought by newer technologies are identified and employed. Also, since DEA spans the fields of Operations Research and Economics, improved computational tools will magnify the synergism existing in the interdisciplinary nature of DEA, yielding new insights and expanded applications.

Specifically, this dissertation introduces a parallel programming approach to solve large-scale DEA problems. Because of their nature, DEA models are ideal candidates for exploiting the relatively new parallel computing machinery. During the process of adapting DEA to this new technology, new insights into the DEA problems were realized. This dissertation will identify improvements that can be made to all DEA codes (both serial and parallel), answer problematic concerns that have been revealed in DEA, and present new approaches for DEA analysis that are possible with the new computationally efficient codes.

In addition, a new DEA model is presented for grouping and rank ordering the firm's operations and accounting for possible outliers in the data. Also, a new approach to DEA sensitivity analysis helps paint a picture of the robustness of the operations of a particular evaluation unit. Both of these advances build upon the gains realized in the computational research.

## 1.2. Notation and Conventions

Because the DEA measure of efficiency is derived from the solution of a mathematical programming model, a description of the notation and conventions which will be used throughout this dissertation is necessary. As much as possible, these notations will conform to the standard emerging in the DEA literature. Matrices are denote bold upper case Latin letters. The $j^{th}$ column of the matrix $\mathbf{X}$ is denoted by $X_j$ anc element in the $i^{th}$ row and $j^{th}$ column is described as $x_{ij}$. Small Greek and Latin bold letters represent vectors; the $i^{th}$ element of vector $\boldsymbol{\lambda}$ is given by $\lambda_i$. The Greek letter $\varepsilon$ is an arbitrarily small positive non-Archimedean scalar value, i.e., the value is not explicitly defined. The vector of all zeroes is denoted by $\mathbf{0}$. The vector of all ones is denoted as $\mathbf{1}$. Surplus and slack vectors of the linear programming problems will be denoted by $\boldsymbol{s}$. Superscripts will denote the constraints to which the surplus or slack variables are to be associated. Scalars are denoted by small italicized Latin and Greek letters. Subscripts on scalars are used to distinguish two or more variables that share common characteristics in the problem but assume different values. Any variable superscripted by "*"(e.g., $z^*$), represents the value the variable takes on as part of the optimal solution for the stated problem. All vectors and matrices are treated in a manner such that the dimensions are conformable for multiplication. A vector is treated as a row when it appears on the left of a matrix and as a column when it appears on the right; thus, alleviating the need to distinguish between row and column vectors. The inner product of two vectors $\boldsymbol{\mu}$ and $Y_o$ is denoted by $\boldsymbol{\mu} Y_o$. Point sets are denoted by capital Latin letters. Both subscripts and superscripts may uniquely identify a set described within a given context. Elements of those sets are denoted by small Latin letters with appropriate superscripts and subscripts. The cardinality of set E will be denoted by |E|.

## 1.3. Characteristics of DEA Efficiency Measures

It is important to understand what is meant by *efficiency* with reference to DEA, since economists have a multitude of definitions and measures. In simplest terms, a measure of how well a firm succeeds in producing as much of its outputs as possible for a given amount of its inputs is called the efficiency of operations for that firm. But what is the basis of comparison? One approach is to establish an absolute standard, usually based on a mathematical model, by which all firms will be evaluated. The shortcomings of this approach have already been discussed. An alternate approach would be to measure a firm's performance relative to other firms which are producing like outputs using the same type of inputs. Of course, the quantity of outputs produced as well as the amount of inputs consumed in the production process will vary from firm to firm. It is precisely this variation that will indicate a firm's relative efficiency compared to its contemporaries.

Data envelopment analysis identifies: those firms producing the most output for a given amount of input and those firms consuming the least input to produce a given level of output. Figure 1.1 depicts the performance of a group of firms (each shown as a point in the scatter diagram) that produce one type of output and consume one type of input. The efficient firms are termed *technically efficient* and form an upper boundary by which the remaining inefficient firms are compared.[1] This empirically derived boundary,

[1] Farrell [51] gives a complete description of technical efficiency which measures a firm's success in producing maximum output for a given amount of inputs. This differs from other measures of efficiency, such as price efficiency, which measures the firm's success of choosing an optimal set of inputs to produce the output. The efficient firms identified by the Farrell measure are a subset of the technically efficient firms.

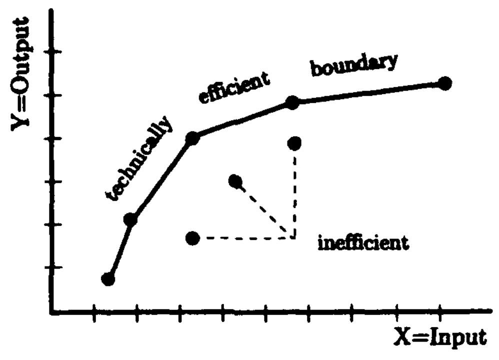

Figure 1.1. Total product curve

together with the connecting line segments, is called the *efficient frontier* or *efficient production surface.*

Because DEA identifies technical efficiency in general terms of quantities of outputs and inputs, the methodology can be applied to competitive firms, regulated firms, and nonprofit organizations. This broad applicability contributes to the growing use of DEA. However, the approach also has shortcomings. *First of all, the technically* efficient subset of firms under analysis may not be very *effective* in translating inputs into outputs. Most would agree that a firm engaged in a poor production process, even though performing better than contemporaries, should still not be considered "effective." Farrell [51], as well as Charnes and Cooper [31], recognized this shortcoming and state clearly that the resulting DEA measure concerns relative efficiency while the question of effectiveness, or absolute efficiency, remains with the policy makers and analysts.

However, the attractiveness of DEA is that it compares performance with the best actually achieved rather than some unattainable ideal. If one is concerned with the question of effectiveness, DEA could first be used to identify the technically efficient firms. The analyst could then limit the scope to these firms to address the question of effectiveness. Consequently, DEA continues to be of value even when the effectiveness of firms is drawn into question. Likewise, if a parametric mathematical representation is desired, the analyst may limit the scope only to the best performing firms from which to build the model. Consequently, DEA can be of value in legitimizing a given parametric approach. For the case where prices are not available for either the outputs or the inputs (e.g., improvement in test scores for elementary school students participating in Headstart programs [39]), DEA provides an ideal approach to measuring productive efficiency. However, for cases where prices, costs, measures of quality, etc., are available, DEA can still be useful in developing alternate measures of efficiency which use this additional data.

Because the economic term *firm* offers the connotation of for-profit organizations, a more general definition is used in DEA literature. Since each organization must decide how best to transform available inputs into desired outputs, each organization will be referred to as a *decision-making unit* (DMU).

## 1.4. A History of Frontier Estimation Models

### 1.4.1. The Technical Efficiency Measure

If the set $N = \{1, 2, \ldots, n\}$ is the index set of DMUs being compared, we may define the efficiency of each DMU$_j, j \in N$, as $\hat{E}_j = \frac{y_{1j}}{x_{1j}}$. A DMU$_j$ then is technically efficient if $\hat{E}_j = z = \max_{k \in N}\{\hat{E}_k\}$, thus having the highest output-to-input ratio. For the case of multiple inputs and/or multiple outputs, $\hat{E}_j$ can be defined in a manner similar to the one-input one-output case.

The production process for DMUs with two inputs and one output can be viewed as in Figure 1.2. The definition of efficiency can be extended to the case of two inputs and one output by re-defining $\hat{E}_j = \frac{y_{1j}}{v_1 x_{1j} + v_2 x_{2j}}$, where $v_q$ represents a weighting factor for input $q$. The efficient DMUs are determined as before and result in the greatest ratio of output to weighted inputs.

Naturally, the above metrics can be generalized to incorporate an arbitrary number of inputs and outputs. If $R = \{1, \ldots, s\}$ is an ordered index set of $s$ outputs and

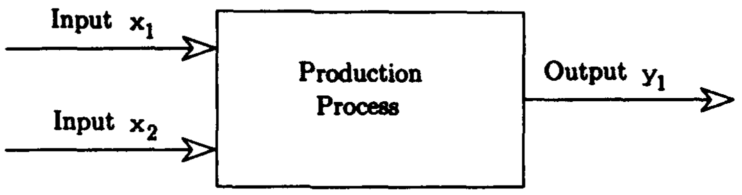

Figure 1.2. Decision-making unit

$Q = \{1, \ldots, m\}$ is an ordered index set of $q$ inputs, then the efficiency for $\text{DMU}_j, j \in N$, can be defined as:

$$E_j = \frac{\sum_{r \in R} u_r y_{rj}}{\sum_{q \in Q} v_q x_{qj}}$$

where $u_r$ and $v_q$ are weights placed on output $r \in R$ and input $q \in Q$, respectively.

A critical concern of this approach is determining the appropriate weights. Theorists and policy makers could assign specific values to the weights, but their choices would be open to criticism and the validity of the resulting efficiency measure could be suspect.

An alternate approach, presented by Farrell, avoids this problem. If the efficient DMUs were known, the weights could be defined so that $E_j$ would equal 1 for all efficient DMUs and would be less than 1 for all inefficient DMUs. Consequently, the problem of finding the relative technical efficiency measure for a given decision making unit, $\text{DMU}_o$, can therefore be described as:

$$\max\ z = E_o \tag{1.1}$$

$$\text{s.t.} \quad E_j \leq 1, \quad j \in N \tag{1.2}$$

$$u_r \geq 0, \quad r \in R \tag{1.3}$$

$$v_q \geq 0, \quad q \in Q \tag{1.4}$$

To find all of the efficient decision-making units in N, the above fractional programming problem must be solved $n$ times, once for each DMU of the set. The computational requirements to solve the fractional programming problem — at least in this form — make solving anything other than simple problems impractical.

### 1.4.2. Data Envelopment Analysis Approach

#### 1.4.2.1. The CCR Multiplier Model

The work of Charnes, Cooper, and Rhodes [37,38] transformed Farrell's nonlinear fractional programming problem for determining E$_j$ into a linear programming problem. The computational advantages that the linear programming formulation offered made it possible to solve for the efficiency of DMUs with many inputs and outputs. The new formulation to find E$_j$ has become known as the CCR Multiplier Model [2] and is written as:

$$(\overline{CCR}_o^i) \qquad \begin{aligned} \max\ z = & \boldsymbol{\mu} Y_o && (1.5)\\ \text{s.t.}\quad \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} & \leq \mathbf{0} && (1.6)\\ \boldsymbol{\nu} X_o & = 1 && (1.7)\\ \boldsymbol{\mu} & \geq \varepsilon\mathbf{1} && (1.8)\\ \boldsymbol{\nu} & \geq \varepsilon\mathbf{1} && (1.9) \end{aligned}$$

In this model, let there be $n$ DMUs with $\mathbf{X} = [X_1, \ldots, X_n]$ and $\mathbf{Y} = [Y_1, \ldots, Y_n]$. Vectors $X_j, Y_j$ denote the observed data where, $X_j = (x_{1j}, \ldots, x_{mj})$ is a column vector of observed inputs and $Y_j = (y_{1j}, \ldots, y_{sj})$ is a column vector of observed outputs for each DMU. In this model, $\boldsymbol{\mu}$ is the vector of multiplier weights applied to the outputs and $\boldsymbol{\nu}$ the multiplier weights applied to the inputs. To insure that the multipliers remain strictly

[2] This formulation is termed input-oriented; a similar output-oriented version will be described in a later discussion.

positive, a non-Archimedean scalar, $\varepsilon$, is used in constraints (1.8) and (1.9). Each DMU$_j$ is assumed to have the same inputs and outputs but in varying amounts. Originally, all observed inputs and outputs were assumed to be strictly positive. Later, it was shown [42] that the CCR multiplier model holds for inputs and outputs that are strictly non-negative.

An attractive feature of this linear programming formulation, consistent with Farrell's development, is that no prior knowledge of the weighted values is necessary to identify the efficiency of a DMU. Here, the maximization problem finds the most favorable weights possible for DMU$_o$ while satisfying all constraints. As a result, the weights, $\boldsymbol{\mu}^*$ and $\boldsymbol{\nu}^*$, are found to make DMU$_o$ look as efficient as possible while insuring that the ratio $\sum_{r=1}^{s} \mu_r^* y_{rj} / \sum_{i=1}^{m} \nu_i^* x_{ij}$ for all other DMUs never exceeds one. The normalization constraint $\boldsymbol{\nu} X_o = 1$ (1.7) linearizes Farrel's objective function and, with (1.6), insures that the optimal objective function value, $z^*$, never exceeds 1. Indeed optimal solutions to the problem will find $0 < z^* \leq 1$ with efficient DMUs indicated by $z^* = 1$.

Note that the non-Archimedean constant, $\varepsilon$, represents an arbitrarily small, strictly positive value that insures positivity of the optimal multiplier values. Without the non-Archimedean term, a particular DMU$_o$ could have an optimal objective value $z^* = 1$ with one or more of the multipliers equal to zero, indicating DMU$_o$ is relatively efficient if one or more of the positive input or output variables are ignored. In this case, the DMU will be on the efficient frontier but may not be Pareto-Koopmans efficient [42]. For example, let the inputs of DMU$_o$ consist of $X_o > \mathbf{0}$ and the outputs result in $Y_o > \mathbf{0}$. Let DMU$_o^*$ be identical to DMU$_o$ in its production of outputs $Y_o^* = Y_o$ but with $X_o^* \leq X_o$ where the equality holds for all $x_{io}^*, i = 1, \ldots, m$, except for $x_{ko}^* < x_{ko}$ where $1 \leq k \leq m$. Assume at optimality, DMU$_o$ is determined to be efficient with $\boldsymbol{\mu}^* > \mathbf{0}$ and $\boldsymbol{\nu}^* \geq \mathbf{0}$ with only $\nu_k^* = 0$. In this case, DMU$_o^*$ would also be efficient. It is clear, however, that

DMU$_o$ is inefficient relative to DMU$_o^*$ because both DMUs produce the identical levels of output but DMU$_o^*$ consumes less of one input. DMU$_o$ cannot be Pareto-Koopmans efficient if, while maintaining the same levels of all outputs, the production possibility set allows for a decrease of an input while holding all other input levels constant. A similar example can be constructed when one of the multipliers of an output variable is equal to zero at optimality. A DMU$_o$ with $z^* = 1$ but with at least one multiplier, $\mu_r^*$ or $\nu_i^*$, equal to zero in all alternate solutions is termed "weakly efficient" by Charnes, Cooper, and Thrall [42].

### 1.4.2.2. CCR Envelopment Model

The linear programming CCR multiplier model has an associated dual formulation which has become known as the (input-oriented) *CCR envelopment model.* This model is written as:

$$(CCR_o^i) \qquad \min\ w = \theta - \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \tag{1.10}$$

$$\text{s.t.} \quad \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_o \tag{1.11}$$

$$\theta X_o - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0} \tag{1.12}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0} \tag{1.13}$$

$$\theta \text{ free} \tag{1.14}$$

where $\boldsymbol{s}^o$ and $\boldsymbol{s}^i$ are the slacks associated with the output and input constraints, respectively. This is the most widely discussed DEA formulation in current literature. Even though the CCR multiplier and envelopment models yield equivalent optimal objective values, the envelopment model is more efficient, computationally, because of the smaller number of constraints.[3]

[3] With the envelopment model, the number of constraints equals the number of outputs and inputs observed for each DMU. The number of variables equals the number

The non-Archimedean term, $\varepsilon$, of the CCR models poses a problem. In theory, this term is denoted by an arbitrarily small positive value; in practice though, it must be explicitly defined. Ali and Seiford [9] show that the choice of $\varepsilon$ can dramatically effect the solution of problems and produce erroneous efficiency scores. This concern can be overcome as described in the next section.

### 1.4.2.3. Ali's IDEAS Formulation

Ali [3] presented a variation of the CCR envelopment model which eliminated the need to explicitly define $\varepsilon$. This new approach for the $CCR_o^i$ formulation, called the integrated data envelopment analysis system (IDEAS), can be written as the following pair of linear programs:

(IDEAS Phase I)

$$\min \theta \tag{1.15}$$

$$\text{s.t.} \quad \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_o \tag{1.16}$$

$$\theta X_o - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0} \tag{1.17}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0} \tag{1.18}$$

$$\theta \text{ free} \tag{1.19}$$

(IDEAS Phase II)

$$\max \boldsymbol{s}^o + \boldsymbol{s}^i \tag{1.20}$$

$$\text{s.t.} \quad \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_o \tag{1.21}$$

$$\theta X_o - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0} \tag{1.22}$$

$$\theta = \theta^* \tag{1.23}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0} \tag{1.24}$$

$$\theta \text{ free} \tag{1.25}$$

---

of DMUs. In data envelopment analysis, the number of DMUs being observed, $n$, is typically much larger than the number of output and input variables $(s + m)$. If this were not the case, all DMUs would likely be efficient.

The $\theta^*$ in Phase II is the optimal solution found in Phase I. It has been shown that a DMU is efficient if and only if $\theta^* = 1, s^{o*} = 0, s^{i*} = 0$ for the problem above [3,9,31,52]. Otherwise, a weakly efficient DMU would have $\theta^* = 1$ and at least one positive slack [33], or a DMU with $\theta^* < 1$ would be inefficient with $\theta^*$ measuring the extent of the inefficiency.

The non-Archimedean constant has been removed from the problem formulation but at a possible computational cost of solving an additional linear program for each DMU during the two-stage approach.[4] Later, this paper will show how the computational costs of this approach can be minimized to achieve the benefit of eliminating the non-Archimedean term.

### 1.4.3. DEA Model Variations

The above efficiency measures all share a common shortcoming: they only account for constant returns to scale. As a result, only a subset of the DMUs on the efficient production surface may be revealed. This can best be seen by means of an illustration. Figure 1.3 depicts the case of a collection of DMUs, each with a single output and single input.

The efficient production surface is outlined by DMUs a, b, c, d, and e. Only $DMU_b$, and $DMU_c$ exhibit *constant returns to scale*. In economics terms, these are the DMUs that result in the greatest average product and represent the *most productive scale size* in terms of the level of output produced.[5] The points which exhibit constant returns to scale on the production surface can be found by drawing a line through the origin which is tangent to the production surface. The other points on the production

[4] Just as the so-called "Big M" value is not explicitly defined in the "Phase I–II" approach to solving linear programming problems, IDEAS uses a two-stage process to circumvent the need to explicitly define $\varepsilon$.

[5] Banker specifically identifies relationships between the most productive scale size and DEA in his 1984 paper [10].

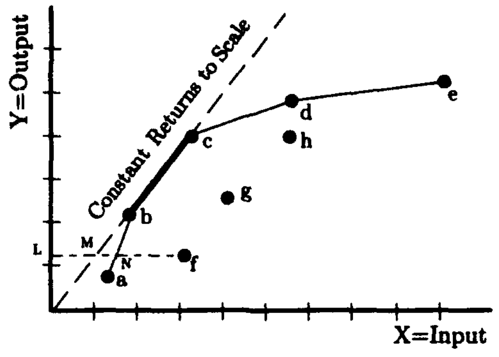

Figure 1.3. Technically efficient DMUs

surface, are technically efficient, but are not *scale efficient* at the most-productive scale size. Technical efficiency implies that for a given level of output, the efficient DMUs consume the least amount of input.

If constant scale efficiency implies a most-productive scale size, how can $DMU_b$ and $DMU_c$ both exhibit constant returns when they produce different levels of outputs? Even though $DMU_b$, and $DMU_c$ produce different levels of output, the amount of the difference is exactly proportional to the difference in the amount of inputs. Consequently, the two DMUs are called *scale equivalent.* $DMU_d$ and $DMU_e$ represent *decreasing returns to scale* compared to the most productive scale-efficient DMUs because they consume more input, but produce less than a proportional increase in output. Moving from $DMU_a$ to the most productive scale efficient DMUs represents *increasing returns to sca*
because the percentage increase in the amount of output produced exceeds the percentage increase in the amount of inputs consumed.

All of the above discussion extends to higher dimensions. By limiting the scope to constant returns to scale, only a portion of the efficient production surface is identified. Consequently, extensions to the CCR data envelopment analysis measure were developed to identify all technically efficient DMUs.

Farrell and Fieldhouse [52] noticed this limitation to the Farrell measure in 1962. They presented a "grouping method" to account for increasing and decreasing returns to scale. In this methodology, DMUs with similar input and output amounts were grouped together and efficiency scores determined with respect to the other DMUs within the particular group. This method never gained wide acceptance, and in the case of multiple outputs and multiple inputs, the process became too complex and unreliable.

1.4.3.1. BCC Variable-Returns-to-Scale Model

In 1984, Banker, Charnes, and Cooper [11] successfully modified the original CCR envelopment formulation to allow for variable returns to scale. The BCC input-oriented envelopment model can be written as:

$$(BCC_o^i) \qquad \min\ \theta - \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \tag{1.26}$$

$$\text{s.t.} \qquad \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_o \tag{1.27}$$

$$\theta X_o - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0} \tag{1.28}$$

$$\mathbf{1}\boldsymbol{\lambda} = 1 \tag{1.29}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0} \tag{1.30}$$

$$\theta \text{ free} \tag{1.31}$$

In their paper, Banker, Charnes, and Cooper, show how this new formulation reveals the entire efficient production surface and all technically efficient DMUs.[6] In a sense, the additional constraint automatically groups DMUs with similar output levels and determines the appropriate efficiency score. For inefficient DMU$_f$ in Figure 1.3, the efficiency score will be less than 1. The overall efficient score found by the CCR model

[6] One assumption of this formulation is that regions of increasing returns to scale are followed by regions of constant or decreasing returns to scale. This is beneficial because it satisfies the normal economic assumption of convexity of the production function.

will be calculated as the ratio of the lengths of the line segments $\frac{LM}{Lf}$. In a like manner the technical efficiency score, found by the BCC model, will be $\frac{LN}{Lf}$. The scale efficient measure, $\frac{LM}{LN}$, can be found by dividing the CCR result by the BCC result. Obviously, if a DMU is both scale and technically efficient, the efficiency measure will be 1 in both cases. Notice that for some DMUs, such as $\text{DMU}_g$, the scale and technical inefficiency scores will be identical. In other cases, such as $\text{DMU}_d$, the DMU will be technically efficient but not scale efficient. Therefore, to determine the technical, scale, and overall efficiency, two linear programs, the CCR and the BCC models, must be solved for each DMU.

### 1.4.4. Compendium of Data Envelopment Models

The formulations of efficiency measure discussed so far are all input-oriented, that is, the efficient DMUs are discovered by observing an output level and identifying the DMU that consumes the least input. Inefficient DMUs are projected onto the efficient surface by proportionally reducing all inputs while holding output constant. The projection distance yields the efficiency score.

However, a different perspective can be accomplished by viewing efficiency from an *output-oriented* perspective. In this case, the efficient DMUs are discovered by observing a given level of input and identifying the DMU that produces the most output. The inefficient DMUs are projected onto the efficient surface by proportionally increasing all outputs while holding inputs constant. The projection distance is the output-oriented inefficiency score. It is important to note that the input- and output-oriented schemes of the BCC model identify the same efficient surface, which is composed of all technically efficient DMUs. However, the scores for the inefficient DMUs may vary between the two formulations. For the CCR model, the input- and output-oriented schemes will identify the same most-productive-scale-size efficient surface and the scores for the inefficient

DMUs of the input-oriented model will be the reciprocals of the scores for the output-oriented model. Seiford and Thrall [81] present a generalization of the BCC and CCR models with input- and output-oriented schemes and define models with non-decreasing returns to scale (NDRS) and non-increasing returns to scale (NIRS). Table 1.1 and Table 1.2 describe the envelopment and multiplier model formulations.

The analysis that follows in this paper will be based on the input- and output-oriented models presented here. To identify all types of efficiency for a given DMU, eight linear programs must be solved. This involves identifying the scale and technical efficiency measures for both the input- and output-oriented models using the two-phase approach of the IDEAS model. Other variations of these models have been developed which further increase the computational complexity of determining the DEA efficiency measures for the DMUs of interest.

Although applications containing tens of thousands of DMUs are now beginning to arise in practice, the computational issues associated with such large-scale models have not been addressed in the literature. This research explores parallel processing as a means of approaching these challenging problems. The next section presents an overview of the salient concepts associated with parallel programming as a background for this discussion.

## 1.5. Parallel Computing Fundamentals

A relatively new and expanding field that offers great promise for solving difficult large-scale problems is application-level parallel processing. Computational gains can be achieved in a multitasking setting where the power of multiple processing elements act concurrently to find the solution for a single problem. It is important to distinguish the multitasking of a single application from multiprogramming wherein an operating system allows a computer to execute multiple unrelated programs.

Table 1.1.-- Envelopment models

| | |
|---|---|
| | (Input-oriented) |
| $CCR^i_j$ : | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \theta_j} \{\theta_j - \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_j, \theta_j X_j - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0}\}$ |
| $BCC^i_j$ : | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \theta_j} \{\theta_j - \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_j, \theta_j X_j - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0}, \mathbf{1}\boldsymbol{\lambda} = 1\}$ |
| $NDRS^i_j$: | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \theta_j} \{\theta_j - \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_j, \theta_j X_j - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0}, \mathbf{1}\boldsymbol{\lambda} \geq 1\}$ |
| $NIRS^i_j$ : | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \theta_j} \{\theta_j - \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_j, \theta_j X_j - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0}, \mathbf{1}\boldsymbol{\lambda} \leq 1\}$ |
| | (Output-oriented) |
| $CCR^o_j$ : | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \phi_j} \{\phi_j + \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{X}\boldsymbol{\lambda} + \boldsymbol{s}^i = X_j, \phi_j Y_j - \mathbf{Y}\boldsymbol{\lambda} + \boldsymbol{s}^o = \mathbf{0}\}$ |
| $BCC^o_j$ : | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \phi_j} \{\phi_j + \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{X}\boldsymbol{\lambda} + \boldsymbol{s}^i = X_j, \phi_j Y_j - \mathbf{Y}\boldsymbol{\lambda} + \boldsymbol{s}^o = \mathbf{0}, \mathbf{1}\boldsymbol{\lambda} = 0\}$ |
| $NDRS^o_j$: | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \phi_j} \{\phi_j + \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{X}\boldsymbol{\lambda} + \boldsymbol{s}^i = X_j, \phi_j Y_j - \mathbf{Y}\boldsymbol{\lambda} + \boldsymbol{s}^o = \mathbf{0}, \mathbf{1}\boldsymbol{\lambda} \geq 0\}$ |
| $NIRS^o_j$ : | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}, \phi_j} \{\phi_j + \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) \mid \mathbf{X}\boldsymbol{\lambda} + \boldsymbol{s}^i = X_j, \phi_j Y_j - \mathbf{Y}\boldsymbol{\lambda} + \boldsymbol{s}^o = \mathbf{0}, \mathbf{1}\boldsymbol{\lambda} \leq 0\}$ |
| | (Non-oriented) |
| $ADD_j$ : | $\min\limits_{\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0}} \{-\mathbf{1}\boldsymbol{s}^i - \mathbf{1}\boldsymbol{s}^o \mid \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = Y_j, -\mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = -X_j, \mathbf{1}\boldsymbol{\lambda} = 1\}$ |

Table 1.2.-- Multiplier models

| | |
|---|---|
| | (Input-oriented) |
| $\overline{CCR^i_j}$ : | $\max\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}} \{z = \boldsymbol{\mu} Y_j \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} \leq \mathbf{0}; \boldsymbol{\nu} X_j = 1; \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}\}$ |
| $\overline{BCC^i_j}$ : | $\max\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}, u_*} \{z = \boldsymbol{\mu} Y_j + u_* \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} + u_*\mathbf{1} \leq \mathbf{0}; \boldsymbol{\nu} X_j = 1; \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}, u_* \text{ free}\}$ |
| $\overline{NDRS^i_j}$: | $\max\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}, u_*} \{z = \boldsymbol{\mu} Y_j + u_* \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} + u_*\mathbf{1} \leq \mathbf{0}; \boldsymbol{\nu} X_j = 1; \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}, u_* \geq 0\}$ |
| $\overline{NIRS^i_j}$ : | $\max\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}, u_*} \{z = \boldsymbol{\mu} Y_j + u_* \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} + u_*\mathbf{1} \leq \mathbf{0}; \boldsymbol{\nu} X_j = 1; \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}, u_* \leq 0\}$ |
| | (Output-oriented) |
| $\overline{CCR^o_j}$ : | $\min\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}} \{z = \boldsymbol{\nu} X_j \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} \leq \mathbf{0}, \boldsymbol{\mu} Y_j = 1, \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}\}$ |
| $\overline{BCC^o_j}$ : | $\min\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}, v_*} \{z = \boldsymbol{\nu} X_j + v_* \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} - v_*\mathbf{1} \leq \mathbf{0}, \boldsymbol{\mu} Y_j = 1, \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}, v_* \text{ free}\}$ |
| $\overline{NDRS^o_j}$: | $\min\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}, v_*} \{z = \boldsymbol{\nu} X_j + v_* \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} - v_*\mathbf{1} \leq \mathbf{0}, \boldsymbol{\mu} Y_j = 1, \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}, v_* \leq 0\}$ |
| $\overline{NIRS^o_j}$ : | $\min\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}, v_*} \{z = \boldsymbol{\nu} X_j + v_* \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} - v_*\mathbf{1} \leq \mathbf{0}, \boldsymbol{\mu} Y_j = 1, \boldsymbol{\mu} \geq \varepsilon\mathbf{1}, \boldsymbol{\nu} \geq \varepsilon\mathbf{1}, v_* \geq 0\}$ |
| | (Non-oriented) |
| $\overline{ADD}_j$ : | $\max\limits_{\boldsymbol{\mu}, \boldsymbol{\nu}, \omega} \{\boldsymbol{\mu} Y_j - \boldsymbol{\nu} X_j + \omega \mid \boldsymbol{\mu}\mathbf{Y} - \boldsymbol{\nu}\mathbf{X} + \mathbf{1}\omega \geq \mathbf{0}, -\boldsymbol{\mu} \geq -\mathbf{1}, -\boldsymbol{\nu} \geq -\mathbf{1}, \omega \text{ free}\}$ |

The basic premise in multitasking in a parallel environment is that, for any given problem, a large number of tasks must be completed to successfully arrive at a correct solution. If these tasks can be scheduled for simultaneous execution on multiple processors, opportunities exist for achieving dramatic reductions in solution times. This fundamental motivation for parallel programming suggests great performance gains can be made using new hardware technology if the work can be decomposed effectively.

Substantial performance gains can be difficult to achieve. While modest gains are possible for many applications in a parallel setting, outstanding results occur only for a much smaller set of problems. To achieve outstanding results, the majority of steps of the algorithm must be done in parallel, and the algorithm must closely match the architecture of the underlying machine. As a result, advances in algorithm performance are tied to how well they can exploit the computing machinery. As advances in technology offer new challenges in developing improved algorithms to match new hardware, insights into computational aspects of the problem can be gained as the new algorithms are developed. These problem insights can be useful both to the parallel and serial environments. Therefore, even though the challenge to achieve outstanding gains through parallel programming may seem formidable, the challenge also has the potential to provide new insights into the solution process that may otherwise go unnoticed.

### 1.5.1. Measuring Parallel Performance

*Parallel processing* consists of coordinating the power of multiple processors to accomplish a given task concerning a given set of data [20]. To measure the impact of parallelization, a metric is needed to serve as a basis of comparison between serial and parallel performance.

Of particular interest is a measure of the ability of parallel processing to improve the real time to solve a particular problem. Such improvement permits existing problems to be solved faster, and may render solvable problems whose dimensions exceed traditional computational capacities. Regardless of the motivation, a measure is needed to reflect the computational improvement offered by the parallel algorithm over a serial algorithm to accomplish the same task.

*Speedup*, of which there are several definitions [20], is the most common measure of the effects of parallelization on a problem. In this research report relative speedup is used. *Relative speedup*, $S(p)$ is the ratio of a problem's solution time using the serial version of the algorithm to the time using the parallel version of the same algorithm with $p$ processors on the same machine [62]. In the development of any parallel algorithm, a design objective is to use all resources efficiently to achieve *linear speedup* where $S(p) = p$. The possibility of achieving *superlinear speedup*, $S(p) > p$, is open for debate [20].

Many factors determine the speedup potential for any parallel algorithm. In 1967, G.M. Amdahl formulated a law describing the maximum possible speedup on a parallel computer with $p$ processors. Amdahl's law can be written as:

$$S(p) \leq \frac{1}{\alpha + (1-\alpha)/p}$$

where $S(p)$ represents speedup using $p$ processors, and $\alpha$ is the fraction of code that must be run in serial (i.e., is not parallelizable). From the law we can see

$$\lim_{p \to \infty} S(p) \leq \frac{1}{\alpha}.$$

Therefore, according to Amdahl's law, linear speedup may only be possible when a small portion of the programming steps is restricted to the serial case.

Figure 1.4 depicts the maximum-attainable speedup, according to Amdahl's law, for various ranges of $p$ and $\alpha$. This illustrates the inherent difficulty of achieving linear

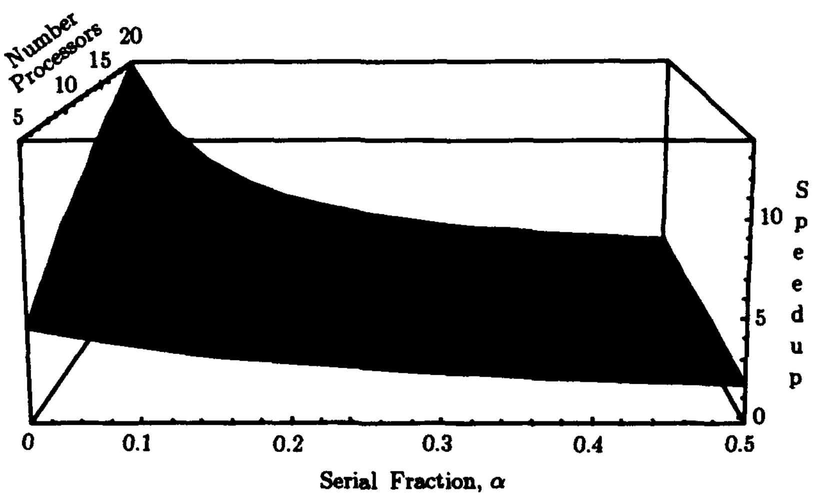

Figure 1.4. Maximum-attainable speedup as a function of $\alpha$ and $p$

speedup. It becomes obvious, that the achievement of linear speedup as the number of processors increases demands almost complete parallelization of the algorithm's steps.

*Efficiency*, defined as $E(p) = \frac{S(p)}{p}$ is another measure commonly used to determine the performance of the parallel algorithm, reflecting speedup normalized by the number of processors. An efficiency of 1 reflects linear speedup; a value less than 1 is the fraction of linear speedup attained.

Table 1.3 reveals the different values of $\alpha$ that are necessary to achieve a given level of efficiency for various number of processors. This table underscores the difficulty of achieving linear speedup, which rises proportionately with the number of processors used. With more processors, the fraction of tasks that require serial implementation must be very small. However, as Amdahl noticed, the fraction, $\alpha$ often depends on the problem size. If $n$ measures problem size, then $\alpha(n) \to 0$ as $n \to \infty$ since, for many problems, the amount of work to be done in serial is at a fixed level regardless of problem size. This fixed amount of work becomes negligible for large problem sizes. As a result, for a fixed

Table 1.3. -- Necessary Amdahl fraction $\alpha$ levels

| Number of | Efficiency | | | |
|---|---|---|---|---|
| processors | .8 | .9 | .95 | .99 |
| 5 | .0625 | .0278 | .0132 | .0025 |
| 10 | .0278 | .0123 | .0058 | .0011 |
| 15 | .0179 | .0079 | .0038 | .0007 |
| 20 | .0132 | .0058 | .0028 | .0005 |

$p$, $S(p) \rightarrow p$ as $n \rightarrow \infty$. Consequently, linear speedup may be possible for large, highly decomposable problems.

### 1.5.2. Parallel Computer Architectures

When approaching a problem for parallel implementation, the designer must insure the algorithm takes into account the advantages of the specific hardware on which the algorithm will be executed. Unlike single-processor machines, where serial algorithms will execute in a more or less common fashion across many different types of computers, the parallel implementation must be tailored to the specific hardware. Parallel architectures take on many different configurations, each offering its own unique set of advantages and disadvantages. Consequently, an algorithm designed for a particular parallel machine may be completely inappropriate for another type of parallel computer. In 1972 Flynn [56] offered a classification scheme for all types of parallel computers. The four classifications are dependent on the type of *instruction stream* and *data stream* the computer is capable of controlling. The instruction stream is the sequence of programming steps carried out on the computer to arrive at a solution to the problem. The objects on which the instructions act are called the data stream. The single-instruction, single-data (SISD) computers are the traditional uniprocessor computers which range from a personal computer to the Cray-1 supercomputer. The single-instruction, multiple-data computers (SIMD) are represented by multi-processor computers where each processor executes the same instruction sequences on different sets of data. The best-known machine in this category is the Connection Machine (models CM-1 and CM-2) that contains up to 64,000 processors. Currently, there are no known general-purpose multiple-instruction, single-data (MISD) computers. The most common parallel architecture is the multiple-instruction, multiple-data (MIMD) computer which can simultaneously operate multiple independently executing processors on different data sets.

Both the SIMD and MIMD architectures require communication between the processors to insure integrity of the solution procedure. The communication can occur either through shared-memory access via a central switch (bus) or by message-passing through an interconnected network in a distributed system. The shared-memory, or tightly coupled, design offers the advantage of minimal communication overhead and simplified programming, but the bus bandwidth limits the number of processors. With tightly coupled shared memory, each processor takes the same amount of time to access a particular memory location. Although distributed processing systems pose few limits on the number of processors, they have higher communication overhead and require more elaborate programming techniques.

All computational testing for this research was conducted on a tightly coupled MIMD Sequent Symmetry S81 (Rev B) at Southern Methodist University. It is configured with 20 Intel 80386 processors and Weitek numeric coprocessors and 32 megabytes of sharable memory. A Unix operating system is used to schedule the parallel tasks.

### 1.5.3. Programming MIMD Multiprocessors

Parallel processing consists of concurrent manipulation of data among several processors. The goal of parallel algorithm design is to minimize $\alpha$ while maintaining a balance of computational load across all processors. The overall goal of the use of parallelism is to increase the speed in solving the problem.

Several approaches exist in parallel programming to distribute the work load among the processors. The two primary methods of performing parallel operations are functional partitioning and data partitioning [75]. The two may be used separately or in combination with any algorithm on a MIMD machine. With *functional partitioning*, the various processors perform different functions on the same data set to complete the problem. An analogy of this process can be made with the construction of a house. Several tasks are needed to complete the structure. The foundation must be poured, framing done, plumbing and electrical work completed, dry wall hung, bricks laid, and the finished work painted. While some tasks depend on the completion of others, many can be done concurrently. In the this case, the electricians play the role of one processor, the plumbers another, etc. each operating differently on different aspects of the data set, the house.

*Data partitioning* or *domain decomposition* is fundamentally different from functional partitioning. In data partitioning, each processor is assigned a similar task to be conducted on different parts of the data. Again, analogies can be drawn with the construction of the house. As with multiple processors, multiple bricklayers can work together to either complete a single wall or multiple walls faster than a single worker. Again, for successful completion, communication must occur between the different processors to insure a correct outcome.

Determining whether to use functional partitioning or data partitioning is critical to the successful implementation of the parallel algorithm. An incorrect choice could limit useful speedup because of the added time for synchronization and communication between processors. Additionally, load imbalances can occur where some processors wait for others to complete their assigned work. In the DEA study reported here, data partitioning was used because of special characteristics of the DEA linear-programming problems; exceptional speedup was accomplished across a large number of processors.

Once a functional-partitioning or data-partitioning approach is chosen, a decision must be made on how to schedule the tasks on the available processors. Two methods of scheduling the jobs are *pre-scheduling* and *self-scheduling* [75]. With pre-scheduling, the distribution of work across the processors is set prior to compilation and requires little overhead. This type of scheduling is ideal when the tasks require the same amount of time. If the tasks require differing amounts of time, the workload could be severely imbalanced across the processors. A second method to schedule the tasks is known as self-scheduling or dynamic scheduling. This method allows each processor to schedule its own tasks at run time by selecting work from a task queue; once the task is completed, the processor returns to the queue to acquire additional work. The overhead of coordinating the tasks and maintaining the work queue is more time consuming than with the pre-scheduled approach, but self-scheduling maintains a more balanced workload across all processors. For the DEA code described herein, the self-scheduling approach was used since tasks required varying amounts of time, and workload balance was a high priority in the algorithm design.

## 1.6. Objectives

This dissertation presents several advances in data envelopment analysis. First, a new computational approach for solving large scale problems is described, with testing presented for both serial and parallel environments. Second, a new DEA modeling and evaluation scheme is presented that gives deeper insights into the relationships among the decision-making units than traditional analysis. Finally, a new means of assessing the robustness of a decision-unit's efficiency is given which spans all current models. All of the new models described can build on the computational advances to make practical their implementation in industrial and governmental applications.

Chapter II describes a state-of-the-art parallel approach to solving DEA problems. By using linear programming theory, a computationally efficient approach is introduced and the solution to previously unsolvable problems will be analyzed. The parallel approach motivates a new hierarchical decomposition methodology for solving DEA by first isolating the efficient DMUs and then determining the DEA scores for the inefficient DMUs. The new approach requires fewer computing resources and improves solution time by one to two orders of magnitude for large-scale DEA applications. The hierarchical approach also reduces the need to duplicate effort when solving variations of the DEA models on the same set of data. This opens the door to a new range of applications for the family of DEA models.

These computational improvements make more practical a new approach to ranking DMUs as discussed in Chapter III. This approach may be useful to managers by providing a different "picture" of the efficient surface and sublayers. This will allow managers to observe more options for improving production techniques by striving not only for improved efficiency scores, but by improving its rank order among competitors. The approach will also demonstrate how outliers, which may distort the DEA efficiency measure, can be accounted for and corrected.

In Chapter IV an improved sensitivity analysis approach is presented which helps develop a picture for the robustness of the operations of a particular DMU. The motivation for the chapter resulted from the need to know how much the use of inputs and production of outputs can change before a DMU changes its efficiency status.

Chapter V closes with observations and conclusions drawn from this research effort. Future research directions are discussed along with a summary of the advances made in this report.

# CHAPTER II

# NEW ADVANCES IN COMPUTATIONAL DEA

Data envelopment analysis (DEA) has become an established nonparametric methodology for assessing the relative efficiency of comparable production and other decision-making units. As such, it has evolved into an approach to evaluating certain types of data sets: namely, those that can be viewed as emanating from a (economic) production function. The range of applicability turns out to be a wide one and, while the approach is new—relative to other multivariate analysis techniques—its usage and variety of applications has grown rapidly since its introduction in 1978.

Surprisingly, little has been published on the computational aspects of DEA [3,4,6,7,9,46,76]. Since DEA typically involves the solution of a large number of linear programs, many practitioners and researchers assume that the repeated use of standard linear programming codes is sufficient for an analysis. Unfortunately this is not the case. Specialized codes are needed to correctly handle the preemptive prioritized multiple objectives (reflecting the models' non-Archimedian infinitesimal) and to coordinate and expedite the solution of the large number of related linear programs.

The state of the art for DEA codes has also been limited in the size of data sets that can be evaluated in a reasonable time: typically in the hundreds of data points. (Codes that handle 1,000 observations have been reported, but run times can be on the order of days [5].) This study was motivated by large applications we have encountered: franchise analysis (e.g., over 8,000 McDonald's restaurants, 6,500 Century 21 real-estate offices, and approximately 5,000 H&R Block tax-preparation service centers), the Federal

Reserve Bank's efficiency study of 8,742 U.S. banks, a V.A. study of over 20,000 hospitals, and U.S. Postal Service evaluations of over 30,000 branches. These problem sizes are clearly beyond the limits of current DEA codes.

This chapter describes a new code for solving large-scale DEA problems in a reasonable amount of time, demonstrate its performance on a real-world application, and report on its ability to exploit parallel processing to further accelerate solution time. We also introduce a new decomposition algorithm that streamlines the solution of problem sets and provide in-depth computational testing of the code on small- and large-scale problems, including the largest DEA problems reported to date.

## 2.1. Computational DEA

Key to the development of software for putting DEA into practice are the mathematical underpinnings and associated means of exploiting domain-specific structure for computational gain. We now briefly summarize relevant DEA concepts, describe efficient implementation techniques, and present the results of computational testing of a new DEA research code.

### 2.1.1. DEA Fundamentals

As described previously, data envelopment analysis is a family of models for assessing and analyzing the relative transformational efficiency of similar decision-making units. A *decision-making unit* (DMU) is any organization, process, or other entity that uses a set of resources (*inputs*) to produce a set of products or services (*outputs*). All DEA models have the same data requirements: for each DMU $j$, a nonegative observed value for each of the $s$ outputs ($Y_j$) and $m$ inputs ($X_j$), with at least one positive input and output.

A DMU's *efficiency* is defined in the tradition of engineering and financial analysis: the ratio of the weighted total output produced to the total input consumed, or

$\boldsymbol{\mu} Y_j / \boldsymbol{\nu} X_j$. For each DMU in the analysis there is associated a separate linear program which determines such an efficiency ratio $\theta$, where $0 < \theta \leq 1$ is typically implied.

The various DEA models separate the set of $n$ DMUs (D) into efficient (E*) and inefficient (I*) sets. A DMU is *efficient* if $\theta = 1, \boldsymbol{s}^o = \mathbf{0}$, and $\boldsymbol{s}^i = \mathbf{0}$. Mathematically, the members of E* are extreme points, extreme rays, or lie on a convex surface which, when taken with their associated facets, form a piecewise-linear empirical production surface, or efficient frontier. Managerially, these units exhibit best practice of transformational efficiency, relative to their inefficient peers.

Otherwise, a DMU is deemed *inefficient* (i.e., when $\theta < 1, \boldsymbol{s}^i \neq \mathbf{0}$, or $\boldsymbol{s}^o \neq \mathbf{0}$), and the observation lies within, not on, the efficient frontier. In this case, the unit uses more input for the same level of output (or produces lower output for the same level of input) as another unit, or some combination of units. For each DMU $d \in$ I*, the corresponding linear program's solution suggests a set of inputs and outputs that would make $d$ efficient. This *virtual DMU* is obtained by a projection (not necessarily orthogonal) onto the efficient frontier, and is expressed as a linear combination of $d$'s *reference set.* The reference set contains those efficient DMUs with which $d$ is being directly compared. The virtual DMU's inputs and outputs are determined as $\hat{Y}_j = \mathbf{Y}\boldsymbol{\lambda}^*$ and $\hat{X}_j = \mathbf{X}\boldsymbol{\lambda}^*$.

Although there are many formulations of DEA models (see Table 1.1 and Table 1.2), the useful results in each case are similar in nature and are to be determined in practice by a DEA code. A complete data envelopment analysis produces, for each DMU: (1) its efficiency score, $\theta$; (2) a set of weights, $\boldsymbol{\mu}$ and $\boldsymbol{\nu}$, for the outputs and inputs, respectively; and, for inefficient units, (3) values for the slack variables $\boldsymbol{s}^o$ and $\boldsymbol{s}^i$, (4) a set of reference DMUs, and (5) values for $\boldsymbol{\lambda}$, from which a virtual DMU can be computed. In the next section, we describe some of the techniques that have been used to expedite the calculation of these items.

## 2.1.2. Survey of Efficiency Techniques

In this discussion, the following notation will be used. We define: $D = \{1, \ldots, n\}$, the index set of DMUs; $B_j^*$ ($\overline{B_j^*}$) is the set of basic (nonbasic) variables associated with an optimal basis for an envelopment-form DEA model applied to DMU $j \in D$; $\theta_j^*$, $\boldsymbol{s}_j^{o*}$, and $\boldsymbol{s}_j^{i*}$ are the values of $\theta$, $\boldsymbol{s}^o$, and $\boldsymbol{s}^i$, respectively, in $B_j^*$; $\Lambda_j = \{i \in D | \lambda_i \in B_j^*\}$ ; $\overline{\lambda}_i$ is the reduced cost of weight $\lambda_i$ in a given optimal solution; $E^* = \{j \in D | \theta_j^* = 1,\ \boldsymbol{s}_j^{o*} = \mathbf{0},$ and $\boldsymbol{s}_j^{i*} = \mathbf{0}\}$; and $I^* = D - E^*$.

### 2.1.2.1. Choice of Model Form

For each DEA model (CCR, BCC, Additive), either the envelopment or the multiplier forms may be used to compute solutions. Since typically $(s + m) \ll n$, there will be fewer constraints and a smaller basis inverse to be maintained with the envelopment form, used by virtually all DEA codes. This results in $(n + s + m)$ variables and $(s + m)$ or $(s + m + 1)$ constraints.

Since the number and difficulty of pivots in a linear program grows more rapidly in the number of constraints than in the number of variables, this choice of the envelopment model becomes an increasingly favorable one as the size of the data set expands. This advantage can be further exploited as the analysis reveals the memberships of $E^*$ and $I^*$.

### 2.1.2.2. Early Identification of Efficient DMUs

It has been observed in [6,33] that if $i \in D$ and $i \in B_j^*$, $j \in D$, then DMU$_i$ is efficient. This is shown formally, as follows.

**Theorem 1.** *For any* $j \in D$, $\Lambda_j \subseteq E^*$ .

**Proof.** (CCR version) Let the optimal solution to the primal (envelopment) problem have a set of basic variables $B^*$. As shown in [41], $B^*$ must contain at least one $\lambda_j > 0$, $j \in D$. If $\lambda_j > 0$, then, by complementary slackness, the corresponding constraint of the dual problem must be strictly satisfied and $\mu Y_j - \nu X_j = 0$. Hence $\mu Y_j / \nu X_j = 1$, and a multiplier set has been found for which the dual LP associated with DMU$_j$ would result in $z = 1$, the maximum attainable. Hence, DMU$_j$ is efficient. ∎

This theorem holds for all standard DEA models and has many consequences for computational efficiency in data envelopment analysis. The early identification of members of E $\subset$ E$^*$ permits: (1) bypassing unsolved subproblems associated with members of E, if values of the weights and slacks are not needed; (2) near-optimal advanced starting solutions for known efficient DMUs; and (3) pricing strategies that give priority to members of E. The use of observation 1 can reduce the number of linear programming subproblems to be solved in the analysis, while the observations 2 and 3 can expedite the solution of subproblems that must be optimized.

In software implementations, the status of each DMU$_j$, and its associated $\lambda_j$, can be easily maintained as: member of E $\subseteq$ E$^*$, member of I $\subseteq$ I$^*$, or unknown (member of U = D − E − I). When the subproblem associated with DMU$_j$ is solved, $j$ can be added to E or I. In addition, from Theorem 1, B$_j^*$ may reveal new members of E—from the reference set, if DMU$_j$ is inefficient, or from degenerate basic variables, if DMU$_j$ is efficient.

Because of the computational value of early status identification, the following corollary is also useful for streamlining the solution of DEA problems.

**Corollary 1.** *If, at optimality,* $\overline{\lambda}_i = 0, i \in$ D, *then* $i \in$ E$^*$.

**Proof.** Let $z^*$ be the optimal solution to the DEA linear programming problem for DMU$_j$. Let $z^* = z^* - \sum_{k \in \mathrm{N}} \overline{\lambda}_k \lambda_k$ where N is the current set of indices of nonbasic variables. Let $i \in$ N with $\overline{\lambda}_i = 0$. By letting $\lambda_i$ enter the basis, $z^*$ does not change in value. By Theorem 1, DMU$_i$ must be an efficient DMU. ∎

This corollary supports an extended search for members of E associated with the nonbasic variables. Our experience is that there are often a substantial number of alternate optimal solutions and nonbasic variables with zero reduced costs.

### 2.1.2.3. Restricted Basis Entry

Another consequence of Theorem 1 is that, since any $\lambda_j$ associated with an optimal basis corresponds to an efficient $\text{DMU}_j$, we need only consider those variables associated with $D-I$ when pricing. This was publically reported by the authors in [18,19] but discovered independently by Ali and published in [6]. This computationally powerful finding permits continual restriction of the pricing operation; as inefficient DMUs are identified, their corresponding decision variables are dropped from subsequent problems.

This narrowing of focus to the $D - I$ set has a ratchet-like effect on the size of the linear programs being solved. As the membership of I increases, the effective sizes of unsolved subproblems (in terms of number of variables) become progressively smaller as well. Hence the "later" subproblems in a series tend to solve more quickly than the "earlier" ones. As will be shown, this has a dramatic impact on the solution effort required, especially on large-scale problems.

### 2.1.2.4. Candidate List

Candidate lists, and other multi-pricing schemes, are standard computational procedures for expediting the solution of linear programs [64,67,71,72,74]. These basically involve pricing a set of nonbasic variables to form a short list of promising candidates for entry into the solution, and repeatedly selecting incoming variables from the list (re-pricing members between pivots) before replenishing with fresh candidates.

This mechanism may be specialized to the structure of data envelopment analyses as well. First, the restricted basis entry procedure eliminates from consideration variables associated with I. Further, priority for list inclusion can be given to members of E over U, since any basis will contain only slacks and members of $E^*$ [6].

### 2.1.2.5. Degeneracy and Anti-Cycling Logic

Note that if $DMU_j$ is efficient, the optimal basic solution for the envelopment model can be comprised of $\theta = \lambda_j = 1$, with all other variables equal to zero. Accompanying such highly degenerate solutions is a greater likelihood of degenerate pivots, as when the minimum ratio is zero. When such degeneracy exists, so does the possibility of cycling, or a repetition of bases for the same extreme point solution leading to non-convergence. Although practitioners often dismiss this possibility as remote, our experience indicates that cycling in DEA codes is not only prevalent but likely, in the absence of anti-cycling procedures.

Various methods have been proposed for the avoidance of cycling within the simplex method for linear programming. The leading approaches—lexicographic ordering [30], perturbation [30], and Bland's rule [26]—can be computationally demanding and hence are usually invoked only under certain conditions. Some researchers support the use of lexicographic ordering in order to avoid a temporary stalling of the solution process, noting a substantial decrease in the number of pivots when employed [46]. Ali [6] proposes a specialized ratio test that gives preference to $\boldsymbol{\lambda}$ variables over slacks when breaking ties for zero minimum ratios, and yields decreases in the number of pivots and solution times on reported test problems.

Our experience indicates that, for many problems, simple scaling of the problem data—by variables and by constraints, normalizing by averages [25]—is most effective in lowering the incidence of stalling and cycling [50,64,83]. In those cases where a lack of progress is detected, the invocation of the lexicographic ordering procedure quickly remedies the situation.

### 2.1.2.6. Non-Archimedean Infinitesimal, $\varepsilon$

The non-Archimedean infinitesimal, $\varepsilon$, in the objective functions of envelopment-form models was represented in early DEA codes using relatively small positive values (e.g., $10^{-6}$). Unfortunately, the effect for CCR and BCC models was to make the results of any analysis dependent on the value chosen, leading to incorrect results in many cases [9].

More recently, algorithmic changes have been proposed to avoid such data dependencies and ensure correctness of envelopment analyses. Specifically, the CCR and BCC objective functions are recast in a preemptive priority form: minimize $P_1\theta + P_2(-\mathbf{1}\mathbf{s}^i - \mathbf{1}\mathbf{s}^o)$. Proposed implementation schemes include two-stage optimization [3], and a multi-objective pricing form of the simplex method [46]. This approach appears to have resolved the infinitesimal issue in the DEA community.

### 2.1.2.7. Preprocessing of Observations

An analysis of the DMUs' observations can lead to identification of members of I and E prior to the application of optimization. Also, in some instances, the likelihood of encountering efficient or inefficient units early in the solution process can be increased when DMUs are processed in a particular order.

As shown in [6], DMU$_i$ is inefficient if it is dominated by any DMU$_j$, that is if $X_j \leq X_i$ and $Y_j \geq Y_i$. In the BCC and additive models, DMU$_j$ is efficient if one of its outputs (inputs) has a unique maximum (minimum) value, or it has the largest ratio $O_j = \mathbf{1}Y_j/\mathbf{1}X_j$.

Processing DMUs in $O_j$ order is reported to identify members of E* earlier than random order. This may be helpful, especially when the number of efficient DMUs is small, but it is unclear whether it is generally preferable to solve subproblems associated with E* or I* in the early part of the process, since solutions to inefficient DMU subproblems tend to reveal more members of E* than the solutions to efficient DMUs, and subproblems for members of E* tend to be more degenerate and are more prone to cycling.

#### 2.1.2.8. Advanced Starting Bases and Reoptimization

In many algorithms that involve the solution of a series of related linear programs, or subproblems, the use of *reoptimization* can lead to substantial time savings. This is accomplished by using $B_i^*$ as an initial basic feasible solution for the LP associated with $DMU_j$, as is possible when subproblems differ by one or more parameters (objective function or right-hand-side values) but otherwise have the same mathematical structure.

DEA subproblems can employ reoptimization, but its effectiveness is dependent on the mathematical "closeness" of the two solution points, $B_i^*$ and $B_j^*$. Ali [7] suggests the use of this technique only for $j \in I^*$.

## 2.2. PIONEER Code

We have developed and tested PIONEER, a new DEA code which embodies most of the above efficiency techniques. The objective was to build a flexible research code for testing and evaluating alternative solution approaches to large-scale problems.

### 2.2.1. Implementation Overview

PIONEER's optimization kernel is based on the XMP library, a collection of portable, reliable, and widely used Fortran subroutines for the solution of linear programs using the primal and dual simplex methods [70]. The LP basis is maintained in LU-factorized form with pivoting routines from the Harwell library. The availability of source code permitted customization of algorithmic steps for DEA problems, and the clean, documented, modularized design of the routines provided a flexibility that supported and simplified experimentation.

Anti-cycling logic was added to the XMP ratio test. While the aforementioned data scaling procedure substantially reduced the incidence of stalling, lexicographic ordering is invoked when lack of progress is detected.

The PIONEER code implements all varieties of DEA models described in [81]: CCR, BCC, and additive with both input and output orientations and two variable-returns-to-scale models allowing for either increasing or decreasing returns to scale. All of the efficiency techniques in Section 1.2 are employed, except for preprocessing and reoptimization. (Our testing showed that reoptimization was of uneven value—sometimes reducing, sometimes increasing, run times—hence was not used in the final testing.) The candidate list is not prioritized, and the two-stage optimization approach to the non-Archimedean infinitesimal issue is employed.

Auxiliary data structures maintain the E, I, or U status of each DMU, as indicated by the following outline of the code's logic:

**Procedure PIONEER**

```
1. Initialize $U = D, E = \emptyset, I = \emptyset$.
2. While $U \neq \emptyset$ :
      a. Select $i \in U$.
      b. Solve the subproblem for $DMU_i$.
      c. If $DMU_i$ is efficient, $E = E \cup \{i\}$, else $I = I \cup \{i\}$.
      d. Update: $E = E \cup \Lambda_i \cup \{j | \overline{\lambda}_j = 0\}$, and $U = D - E - I$.
```

The strength of step 2d depends on Theorem 1, Corollary 1, and the distribution of observations in $\Re^{(s+m)}$, relative to the efficient frontier.

It should be noted that while the XMP data structures are designed for sparse problems, DEA subproblems are almost totally dense. This results in an unnecessary processing overhead in the current implementation that can be eliminated with recoding. Hence the execution times reported in this paper should be considered worst case, or at least conservative, measures of performance, relative to a fully optimized and tuned implementation.

### 2.2.2. Baseline Computational Testing

To evaluate the efficiency of the PIONEER research code, and to provide a set of baseline measurements from which to compare algorithmic and implementational enhancements, a series of test runs were made on medium- and large-scale problems. Testing with and without individual efficiency techniques lent insight into their impact on solution speed.

#### 2.2.2.1. Test Data

Although the code was validated for solution accuracy on small problems from the open literature, of primary interest was its effectiveness on large-scale problem sets. For testing purposes, real life data, randomly generated data, and data based on the economic Cobb-Douglas functional form were employed.

First, the Federal Reserve Bank of Dallas provided a data set of 8,742 banks from its Southwest district, with variables consisting of the six inputs and three outputs described in [20,21]. This challenging problem from industry was clearly beyond the state of the art of DEA codes, and was a prime motivator for this research. Testing was performed on smaller subsets by selecting the first $n$ observations from the unordered file.

Randomly generated observations from a multinormal population provided a second source of test data. Using the DRNMVN routine from the IMSL library and the procedures given in Appendix A, large data sets could be easily created. The variables were partitioned into 3 inputs and 4 outputs so as to observe positive correlations between the "input" and "output" sets.

Finally, a random problem generator, DEA-GEN, was written to create observations based on the classic Cobb-Douglas form of production surfaces. As detailed in Appendix B, the program gives the user a measure of control over the proportion of efficient points, and creates data sets that more closely approximate realistic economic processes than the multinormal generator.

### 2.2.2.2. Test Environment

The PIONEER code was tested on Southern Methodist University's Sequent Symmetry S81B with 32MB of internal storage and processing units consisting of 16-MHz Intel 80386s with Weitek coprocessors. The software is written entirely in Fortran and executed under Dynix 3.0.12, a BSD-Unix-based operating system. While the processors are rated at 4 million operations per second, in terms of current technology they are equivalent to relatively slow personal computers.

### 2.2.2.3. Experimental Results

The PIONEER code was applied to problems from each of the three sources. The solution times reported below are "wall-clock" times, or elapsed real execution times, exclusive of problem input and output on the Sequent Symmetry. Unsolved subproblems associated with members of E (as determined by Step 2d) were bypassed.

Table 2.1 describes test set A, which was made up of banking, multinormal, and DEA-GEN problems. Three problems from Federal Reserve banking data were examined: the first 1,000, the first 2,000, and the first 8,000 DMUs of the 8,742-bank data set. Two multinormal data sets, for $n = 1{,}000$ and $n = 2{,}000$ were generated using the variance-covariance matrix and procedures given in Appendix B. Six problems, with $n = 1{,}000$ and $n = 2{,}000$, were created using DEA-GEN and the parameters given in Appendix C.

Table 2.1.-- PIONEER solution times on test set A

| Source | $s$ | $m$ | DMUs | No RBE Time | RBE Time | Ratio |
|---|---|---|---|---|---|---|
| FR Bank | 3 | 6 | 1000 | 33.29 min | 17.04 min | 0.526 |
| FR Bank | 3 | 6 | 2000 | 179.91 min | 95.59 min | 0.532 |
| FR Bank | 3 | 6 | 8000 | 44.21 hour | 19.80 hour | 0.448 |
| Multinormal | 4 | 6 | 1000 | 31.96 min | 18.72 min | 0.586 |
| Multinormal | 4 | 6 | 2000 | 106.90 min | 61.65 min | 0.577 |
| DEA-GENa | 4 | 6 | 1000 | 57.51 min | 33.13 min | 0.576 |
| DEA-GENa | 5 | 3 | 2000 | 130.02 min | 75.94 min | 0.584 |
| DEA-GENb | 7 | 4 | 1000 | 61.49 min | 37.96 min | 0.617 |
| DEA-GENb | 7 | 2 | 2000 | 151.27 min | 92.42 min | 0.611 |
| DEA-GENc | 6 | 6 | 1000 | 72.83 min | 43.29 min | 0.594 |
| DEA-GENc | 6 | 5 | 2000 | 213.61 min | 126.50 min | 0.592 |
| Average | | | | | | 0.567 |

Solution times for these problems with the PIONEER code and the CCR model are given in Table 2.1.

The code was run using the $CCR^o$ model with and without the use of restricted basis entry (RBE) and early identification of efficient units (EIE) to examine its impact on solution time, as shown in Table 2.1. In all cases, the RBE procedure had a strong impact, cutting solution times roughly in half. The 17.04-minute time for the 1,000-DMU problem indicated that the PIONEER code is reasonably efficient for medium-sized problems. But, even with the help of RBE, the 19.8-hour solution time for the 8,000-DMU problem is excessive for practical usage.

A closer examination of the 8000-DMU bank problem (BANK-8) solution process gives insight into the sources of the speed improvements. Figure 2.1 gives the time to solve each set of 1,000 subproblems in the 8,000 total, both with and without RBE logic. Note that when RBE is not used, the time to solve each set of 1,000 subproblems is roughly the same (around 4.1 hours). But when RBE is employed, the last group of 1,000 subproblems is solved almost four times faster than the first group. By restricting the known inefficient DMUs from entering the basis, the later subproblems are smaller, and easier to solve. In addition, the early identification of efficient DMUs results in fewer subproblems to be solved. Typically, fewer than 20% of all DMUs are efficient, so the faster solution time can be attributed mainly to restricted basis entry.

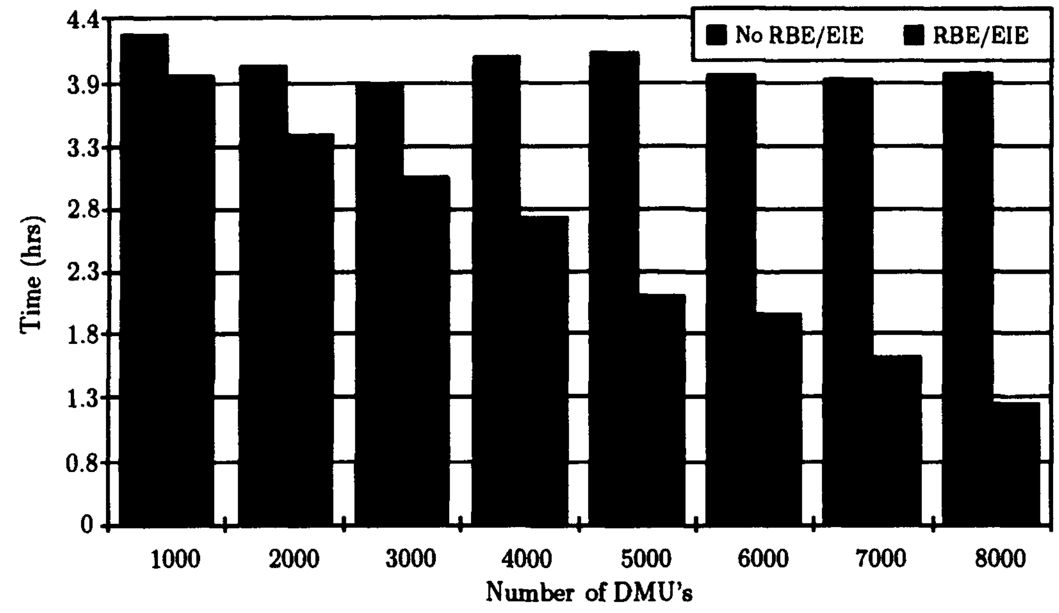

Figure 2.1. Solution time for subsets of BANK-8

Figure 2.2 shows the cumulative effect of RBE/EIE on solution time for the 8,000-DMU bank problem. While the performance improvement is less pronounced in the earlier subproblem groups, the 2:1 ratio becomes evident in the last few groups, and trends indicate an even greater disparity might result for larger problem sets.

Although PIONEER's solution times are encouraging and indicate that the code may be comparable to others described in the literature, other performance improvements are possible. The next section describes the use of parallel processing to further decrease solution times for these and other problems.

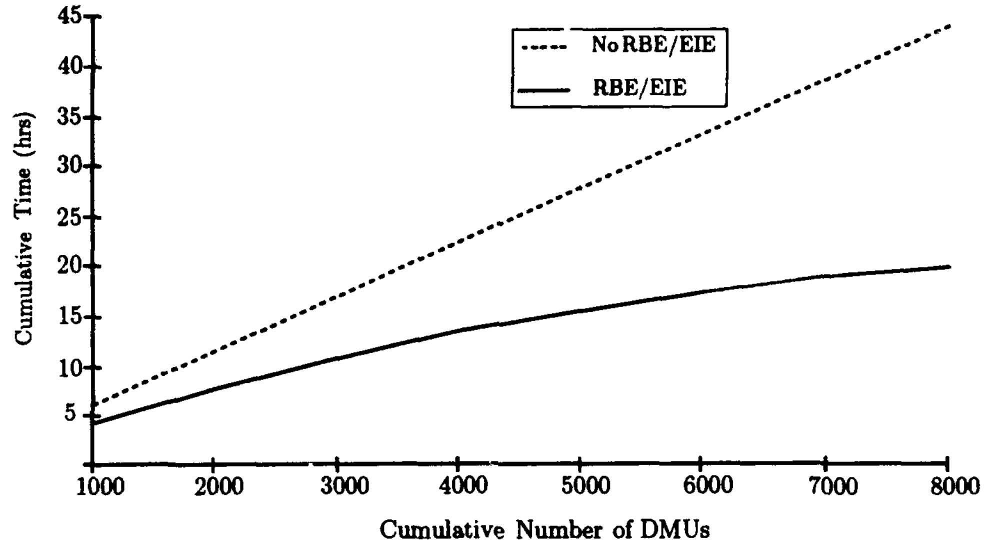

Figure 2.2. Cumulative solution time for subsets of BANK-8

## 2.3. Parallel PIONEER Code

Because of the nature of the DEA formulation, an advance in computer technology, parallel processing, can be utilized to achieve dramatic reductions in solution times. With *parallel processing*, the solution for a given problem is found by simultaneously coordinating the efforts of multiple processors to accomplish a given task on a common body of data. If the algorithm associated with the problem solution process consists of multiple independent tasks that can be properly assigned to separate processors, dramatic reduction in solution times may be possible.

Of the numerous varieties of parallel machine architectures, the most prevalent commercial design is *multiple-instruction, multiple-data* (MIMD) [19,56]. Parallel computer systems are composed of multiple processors which have the ability to act independently on problem data sets. To arrive at a correct solution, coordination of all processors and their tasks is necessary. Two basic techniques are used to accomplish this coordination activity in the parallel environment. The processors can communicate through shared memory. If the time to access a particular memory location is identical for all of the processors, the system is called *tightly coupled*, or else, the system is termed *loosely coupled*. If the parallel machine contains no shared memory capabilities, coordination of processor activities must be accomplished through message passing between all of the processors.

As with traditional single-processor (serial) machines, solution efficiencies are directly tied to how well the algorithmic steps match the architecture of the underlying machine. Because a DEA problem involves the optimization of many separate linear-programming subproblems, the use of MIMD-style parallelism to speed solution appears, on the surface, to be a "natural" one. In fact, the mapping of the DEA solution process to a tightly coupled MIMD architecture turns out to be an ideal example of the use of parallel processing.

### 2.3.1. Parallel Code Design

The application of parallel processing to DEA problems was first reported in Phillips, Parsons, Donoho [76], where four transputers were run from a Macintosh IIcx on a 54-DMU problem. Times were reduced by a factor of three in this loosely coupled MIMD implementation. The next section describes a very different computing environment and how the PIONEER code was modified to use this form of parallelism.

#### 2.3.1.1. Target Machine Environment

As with software designed for vector processors, parallel codes must be structured to match the architecture of the target machine. Our test machine was the same Sequent Symmetry S81B that was employed for serial testing, but which can be programmed for parallel processing. The system has a tightly coupled MIMD design, with 20 16-MHz 80386 processors, Weitek coprocessors, and 32MB of sharable memory.

The Sequent's operating system permits the programmer to create multiple, independent processes which it schedules on the available processors. The processes can have both private and shared data-spaces. The shared spaces can be read and written by all designated processes, and can be used for interprocess communication and to avoid duplication of common data.

#### 2.3.1.2. Parallelization of the DEA Algorithm

This type of parallel machine is designed to be most effective with work that can be decomposed into large, independent tasks. The DEA solution process can be organized in this manner through the use of data partitioning, where multiple identical processes apply the same program steps to different data. By considering each DMU's LP subproblem to be a separate task, such large-granularity work decomposition is possible.

In the parallel PIONEER code, a *self-scheduling* approach is used, where processes select and execute tasks from a shared work list, on a first-come-first-served basis. Although incurring a minor amount of coordination overhead, such self-scheduling permits a balanced distribution of the workload across processes, an important characteristic when individual task times vary.

Each process solves its LP subproblems in its private memory; shared memory stores the original problem data and the status of each DMU. Since a DMU's status—in terms of membership in E, I, or U—may change, restricted basis entry becomes dynamic and time-based. At one moment, a given $\lambda_j$ variable may be part of the $E \cup U$ pricing set and, an instant later, be found to be inefficient and ineligible for basis entry. The shared status array automatically communicates this information to all processes. This is an instance of a *race condition*, or timing-dependent code, which can result in stochastic solution statistics when the order of events differ from run to run due to minute timing differences.

### 2.3.2. Testing Parallel PIONEER

To test the parallel implementation of the PIONEER code, a variety of problems were chosen for analysis. Table 2.5 describes the characteristics of the problems chosen. The original bank data consisted of 8,742 DMUs. For each of the bank problems, the DMUs composing the data set were randomly chosen from the original data set. The multi-normal and DEA-GEN data sets are described in Appendices A and B. These problems were generated to simulate real life data of large-scale DEA problems. The $CCR^i_j$ model was used for all runs.

The "wall clock" time, measured in minutes, to solve each DEA problem, exclusive of input output times, are given in Table 2.2. For the problems consisting of 8,000 and 4,000 DMUs, the number of processors used was limited by the available memory. Tables 2.3 and 2.4 reveal the speedup and efficiency of the parallel approach. In all cases, relative speedup was nearly linear. Isolated examples indicate that superlinear speedup may be possible because of the use of RBE/EIE in the PIONEER code. Since the PIONEER code is asynchronous, i.e., each LP can be solved independently of all other LPs, the parallelization is highly effective at improving solution times.

Table 2.2. – Test problems parallel run times (min.)

| # Procs | Bank | | | Multi-Normal | DEA-GENa | DEA-GENb | DEA-GENc |
|---|---|---|---|---|---|---|---|
| | 1000 | 4000 | 8000 | 4000 | 2000 | 2000 | 2000 |
| 1 | 17.04 | 283.22 | 1189.81 | 242.96 | 75.94 | 92.42 | 126.50 |
| 2 | 8.70 | 151.23 | 589.22 | 122.35 | 38.20 | 46.52 | 63.63 |
| 3 | 5.68 | 101.42 | 389.46 | 81.81 | 25.58 | 31.06 | 42.48 |
| 4 | 4.27 | 73.26 | | 61.39 | 19.19 | 23.39 | 31.91 |
| 5 | 3.40 | 61.27 | | 49.14 | 15.35 | 18.71 | 25.57 |
| 6 | 2.86 | | | | 12.82 | 15.60 | 21.32 |
| 7 | 2.45 | | | | 11.02 | 13.38 | 18.28 |
| 8 | 2.15 | | | | 9.65 | 11.74 | 16.02 |
| 9 | 1.96 | | | | 8.57 | 10.45 | 14.26 |
| 10 | 1.72 | | | | 7.72 | 9.40 | 12.80 |
| 15 | 1.17 | | | | 5.19 | 6.33 | 8.61 |

Table 2.3. – Test problems parallel speedups

| # Procs | Bank | | | Multi-Normal | DEA-GENa | DEA-GENb | DEA-GENc |
|---|---|---|---|---|---|---|---|
| | 1000 | 4000 | 8000 | 4000 | 2000 | 2000 | 2000 |
| 1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| 2 | 1.96 | 1.87 | 2.02 | 1.99 | 1.99 | 1.99 | 1.99 |
| 3 | 3.00 | 2.79 | 3.06 | 2.97 | 2.97 | 2.98 | 2.98 |
| 4 | 3.99 | 3.87 | | 3.96 | 3.96 | 3.95 | 3.96 |
| 5 | 5.01 | 4.62 | | 4.94 | 4.95 | 4.94 | 4.95 |
| 6 | 5.96 | | | | 5.92 | 5.92 | 5.93 |
| 7 | 6.96 | | | | 6.89 | 6.91 | 6.92 |
| 8 | 7.93 | | | | 7.87 | 7.87 | 7.90 |
| 9 | 8.69 | | | | 8.86 | 8.84 | 8.87 |
| 10 | 9.91 | | | | 9.84 | 9.83 | 9.88 |
| 15 | 14.56 | | | | 14.63 | 14.60 | 14.69 |

Table 2.4. – Test problems parallel efficiencies

| # Procs | Bank | | | Multi-Normal | DEA-GENa | DEA-GENb | DEA-GENc |
|---|---|---|---|---|---|---|---|
| | 1000 | 4000 | 8000 | 4000 | 2000 | 2000 | 2000 |
| 1 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| 2 | 0.979 | 0.936 | 0.979 | 0.993 | 0.994 | 0.993 | 0.994 |
| 3 | 1.000 | 0.931 | 1.000 | 0.990 | 0.990 | 0.992 | 0.993 |
| 4 | 0.998 | 0.966 | | 0.989 | 0.989 | 0.988 | 0.991 |
| 5 | 1.002 | 0.924 | | 0.989 | 0.989 | 0.988 | 0.989 |
| 6 | 0.993 | | | | 0.987 | 0.987 | 0.989 |
| 7 | 0.994 | | | | 0.984 | 0.987 | 0.989 |
| 8 | 0.991 | | | | 0.984 | 0.984 | 0.987 |
| 9 | 0.966 | | | | 0.985 | 0.983 | 0.986 |
| 10 | 0.991 | | | | 0.984 | 0.983 | 0.988 |
| 15 | 0.971 | | | | 0.975 | 0.973 | 0.979 |

### 2.3.3. Limits of Parallelism

The computational testing indicated that the parallel PIONEER code was a highly scalable MIMD application that fully utilized the multiprocessing capability of the given computer system. We feel that the software would also exhibit much of the same efficiency if implemented in a loosely coupled MIMD environment. In the latter setting, changes in a DMU's status would have to be broadcast to all processors, thus incurring additional overhead and a latency in communicating the information. While unnecessary work (i.e., avoidable in a shared-memory environment) might result, the use of a larger number of processors could more than offset this disadvantage.

In our testing, memory size limited the dimensions of problems that could use the full parallelism of this system. We believe that additional internal storage would permit the same excellent speedups on the larger problems as was observed on the smaller ones. (This notion was verified by preliminary testing on a larger system.)

Even with these encouraging results, we felt that further improvements were needed and possible. In fact, a close examination of the parallel solution statistics led to a new procedure which further reduced all times—both serial and parallel—by up to an order of magnitude.

## 2.4. Hierarchical Decomposition

Experimentation with sets of problems that were identical except for the number of DMUs yielded solution times such as those given in Figure 2.3. Not only did the memory requirements of larger problems limit the amount of usable parallelism, but run times grew exponentially in $n$. Hence if a larger problem could be decomposed into a series of smaller ones, lower memory requirements, greater parallelism, and faster individual solution times might offset any additional work that might be required.

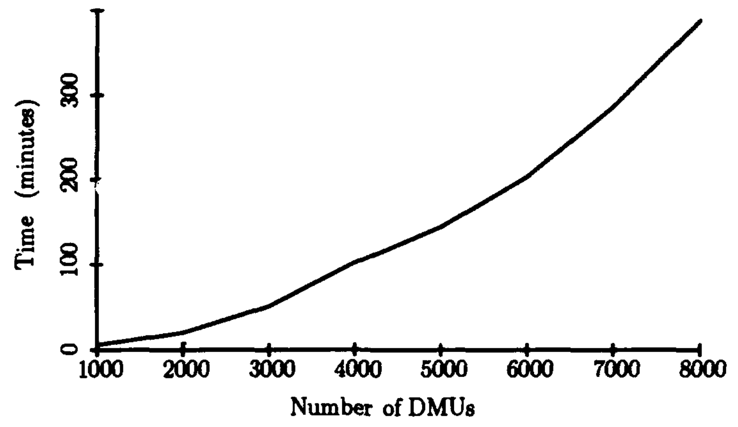

Figure 2.3. Solution time versus problem size

### 2.4.1. DEA Decomposition Background

Consider a partitioning of the set of DMUs into a series of $k$ mutually exclusive and collectively exhaustive subsets $D_i \subset D$, where $D = \bigcup_{i \in K} D_i, \bigcap_{i \in K} D_i = \emptyset$, and $K = \{1, \ldots, k\}$. Define $E(D_i)$ and $I(D_i)$ to be the index sets of DMUs in $D_i$ that are efficient and inefficient, respectively, relative to $D_i$ (i.e., $D_i = E(D_i) \cup I(D_i)$), based on the meaning of efficiency in the DEA model of interest.

**Theorem 2.** *If* $D_i \subseteq D$, *then* $I(D_i) \subseteq I^*$ *and* $E^* \subseteq \bigcup_{i \in K} E(D_i)$.

**Proof** ($\overline{CCR}^i$ version). Define $z_j^*(D_i)$ to be the optimal value of $z$ for DMU $j$ relative to the set $D_i$. We know that $z_j^*(D) \leq z_j^*(D_i), \forall j \in D_i$ in this maximization problem, since there are more model constraints associated with D than $D_i$.

*Assertion* $I(D_i) \subseteq I^*$:

DMU $j$ is inefficient (efficient) relative to set S if $z_j^*(S) < 1$ $(= 1)$. Assume $j \in D_i$ has $z_j^*(D_i) < 1$, and hence is inefficient. Since $z_j^*(D) \leq z_j^*(D_i)$, $j$ will be inefficient relative to D.

*Assertion* $E^* \subseteq \bigcup_{i \in K} E(D_i)$:

Since $E^* \subseteq D = \cup_{i \in K} D_i$, then $E^* \subseteq \cup_i (E(D_i) \cup I(D_i))$. From above, $E^* \cap \cup_i I(D_i) = \emptyset$, therefore the assertion must be true. ∎

Hence if a DMU is inefficient in a subproblem, it is inefficient for the full problem; if it is efficient for a subproblem, it may or may not be efficient overall. These relationships provide the foundation for the following decomposition procedure for DEA models.

### 2.4.2. The Hierarchical Decomposition Procedure

The following approach to DEA problems solves a series of subproblems which are arranged in a hierarchy. The subproblems are themselves DEA problems created from subsets of the DMUs in D. They can be solved independently, but information about set I membership generated from the subproblems can accelerate the solution of others.

First we define a procedure for creating and solving the subproblems. It partitions the set of DMUs whose efficiency is unknown (U) into a series of subsets, or *blocks*, of size $b$. Recall that $U = D - E - I$ and $|U|$ is the cardinality of U.

**Procedure SolveBlocks**($b, \ell$, I):

1. Partition U into $k = \lceil |U|/b \rceil$ mutually exclusive, approximately equal-sized blocks of DMUs.
2. For each block $B_i, i \in K, K = \{1, \ldots, k\}$:
    (a) Apply a DEA envelopment model to compute $E_i^\ell = E(B_i)$, using early identification of efficiency and restricted basis entry.
    (b) Set $I = I \cup I(B_i)$.

This is used in the following hierarchical decomposition procedure (HDEA). A graphical representation of the process is given in Figure 2.4.

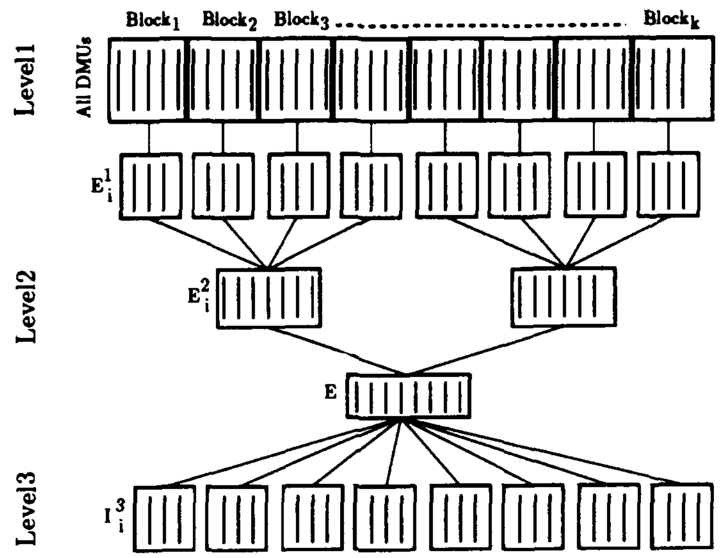

Figure 2.4. Hierarchical decomposition schematic

```
Procedure HDEA($b, \alpha, \beta$):
Level 1: (Initial pass)
    1. $I = \emptyset, U = D, \ell \leftarrow 1$.
    2. SolveBlocks($b, \ell$,I).
Level 2: (Identify $E^*$ and $I^*$)
    while ( $U \neq \emptyset$ ) do:
        1. $\ell \leftarrow \ell + 1$.
        2. $u \leftarrow |U|$.
        3. SolveBlocks($b, \ell$,I).
        4. if $|U|/u > \alpha, b \leftarrow |U|$, else $b \leftarrow \beta b$.
Level 3: (Resolution of $I^*$)
```

Re-solve the DEA model (with basis entry restricted to $E^*$) for members of $I^*$ to compute correct solution values.

### 2.4.3. Implementation Considerations

HDEA is a divide-and-conquer algorithm similar in nature to merge sorting and sorting networks [49,66]: the original problem is broken into several smaller subproblems, which are solved recursively and the results combined to solve the original problem. The blocksize parameter, $b$, defines sets of linear programs with the same number of constraints as the original problem, but many fewer variables. This results in lower memory requirements and faster solutions for the subproblems, although more linear programs may be involved. Because of this increased speed and Theorem 2, these easier problems can eliminate inefficient-DMU variables earlier than with non-hierarchical approaches.

The HDEA procedure focuses initially on isolation of $E^*$ (in levels 1 and 2), so as to expedite solution of the subproblems associated with $I^*$ (in level 3). The method should be particularly effective when $|E^*| \ll |I^*|$, as is typically the case. The decomposition into subproblems with minimal data communication requirements is highly attractive from a parallel processing standpoint.

Memory requirements are a function of $\max\{b, |E^*|\}$. If primary storage is at a premium, its use can be minimized by paging in DMU data from external storage for each subproblem separately. In parallel implementations, participating processes might place in shared memory a copy of the **X** and **Y** data, plus a DMU-status array; each process would also need its own private storage for solver-specific information such as a simplex basis inverse and candidate list.

The choice of blocksize also affects overall solution time. Since $b$ influences the tradeoff between the size and the number of subproblems solved, computational testing for an appropriate setting is required. Figure 2.5 shows the total solution time in minutes for various values of $b$ on an example problem.

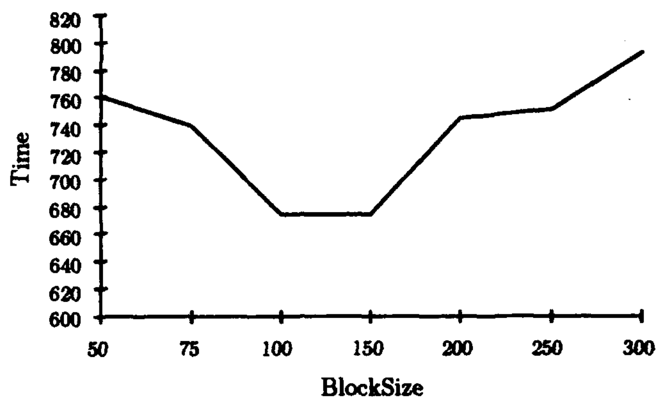

Figure 2.5. Total solution time vs. blocksize for BANK-8

### 2.4.4. Computational Testing

For testing the HDEA procedure, real life data, randomly generated data, and data based on the economic Cobb-Douglas functional form were employed. For each test problem, an array containing the status of each DMU was maintained in shared memory. A private copy of the appropriate data was given to each processor in parallel so that each DMU could be solved independently. Since the status of each DMU could be updated without conflict from other processors, no locks were needed during the status update and as a result, the parallel implementation is purely asynchronous.

All times reported are wall clock times, in minutes, to solve the problems exclusive of any input/output. The single processor times represent the best achieved serial times across the differing-sized DEA problems. The parallel case is a direct implementation of the serial formulation, requiring only a few modifications for the parallel environment to accomplish the self-scheduling parallel implementation. It is important to note, none of the runs were made in a dedicated environment, and hence, times are subject to system load. However, precautions were taken to conduct the runs during non-peak hours so as to minimize the confounding of outside factors on the solution times.

Table 2.5. -- Hierarchical test problem results

| | DMUs | Variables | | Eff. | Number of Pivots | | | Number Priced | | | Time to Solve(min.) | | | Speedup | | Efficiency | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Out | In | | 1 | 5 | 10 | 1 | 5 | 10 | 1 | 5 | 10 | 5 | 10 | 5 | 10 |
| Bank | 1000 | 3 | 6 | 67 | 43992 | 44532 | 45207 | 1590581 | 1632351 | 1680000 | 12.41 | 2.78 | 1.70 | 4.46 | 7.30 | 0.89 | 0.73 |
| | 2000 | 3 | 6 | 70 | 92374 | 93154 | 94219 | 4572332 | 4959926 | 4757618 | 32.81 | 7.17 | 4.19 | 4.58 | 7.83 | 0.92 | 0.78 |
| | 4000 | 3 | 6 | 87 | 189784 | 190340 | 190987 | 12093187 | 12232628 | 12242949 | 85.31 | 17.64 | 9.40 | 4.84 | 9.08 | 0.97 | 0.91 |
| | 8000 | 3 | 6 | 77 | 382755 | 374088 | 383872 | 22890453 | 26946497 | 23429491 | 166.00 | 37.68 | 18.18 | 4.41 | 9.13 | 0.88 | 0.91 |
| Multi-Normal | 1000 | 3 | 4 | 79 | 43992 | 44532 | 45207 | 1590581 | 1632351 | 1680000 | 12.41 | 2.78 | 1.70 | 4.46 | 7.30 | 0.89 | 0.73 |
| | 2000 | 3 | 4 | 89 | 62418 | 62502 | 62674 | 4984732 | 5029902 | 5089368 | 28.03 | 5.85 | 3.14 | 4.81 | 8.92 | 0.96 | 0.89 |
| | 4000 | 3 | 4 | 85 | 132884 | 133149 | 133566 | 10490470 | 10595619 | 10723294 | 59.41 | 12.36 | 6.66 | 4.79 | 8.93 | 0.96 | 0.89 |
| | 8000 | 3 | 4 | 104 | 282679 | 283219 | 284001 | 22740407 | 22934235 | 23190802 | 128.10 | 26.64 | 14.30 | 4.81 | 8.96 | 0.96 | 0.90 |
| DEA-GENa | 1000 | 6 | 4 | 96 | 51102 | 51645 | 52392 | 1922161 | 1964649 | 2019596 | 15.81 | 3.50 | 2.04 | 4.52 | 7.75 | 0.90 | 0.78 |
| | 2000 | 3 | 5 | 147 | 87709 | 88084 | 88746 | 5211167 | 5275631 | 5364496 | 33.64 | 7.12 | 4.00 | 4.72 | 8.41 | 0.94 | 0.84 |
| | 4000 | 5 | 4 | 82 | 168813 | 169281 | 169856 | 12682231 | 12816378 | 12986010 | 83.38 | 17.18 | 9.11 | 4.85 | 9.15 | 0.97 | 0.92 |
| | 8000 | 2 | 5 | 70 | 233108 | 233585 | 234293 | 18772046 | 18962220 | 19215932 | 107.60 | 22.40 | 12.08 | 4.80 | 8.91 | 0.96 | 0.89 |
| DEA-GENb | 1000 | 4 | 7 | 214 | 72961 | 73706 | 74457 | 3530627 | 3583495 | 3638872 | 30.60 | 6.43 | 3.51 | 4.76 | 8.72 | 0.95 | 0.87 |
| | 2000 | 2 | 7 | 275 | 119211 | 113755 | 114479 | 8040444 | 8143142 | 8239181 | 55.37 | 11.54 | 6.21 | 4.80 | 8.92 | 0.96 | 0.89 |
| | 4000 | 3 | 3 | 64 | 98087 | 99237 | 99378 | 7397992 | 7474611 | 7560169 | 39.77 | 8.29 | 4.49 | 4.80 | 8.86 | 0.96 | 0.89 |
| | 8000 | 4 | 6 | 153 | 16418 | 16443 | 16493 | 36671079 | 36943789 | 37360152 | 248.90 | 51.63 | 27.53 | 4.82 | 9.04 | 0.96 | 0.90 |
| DEA-GENc | 1000 | 6 | 6 | 201 | 79180 | 79976 | 81034 | 3693590 | 3749286 | 3824666 | 33.97 | 7.19 | 3.91 | 4.72 | 8.69 | 0.94 | 0.87 |
| | 2000 | 5 | 6 | 265 | 151704 | 152727 | 154118 | 10210540 | 10327905 | 10479963 | 79.73 | 16.58 | 8.82 | 4.81 | 9.04 | 0.96 | 0.90 |
| | 4000 | 4 | 4 | 66 | 127473 | 127684 | 127827 | 9348034 | 9440561 | 9547832 | 57.72 | 11.97 | 6.38 | 4.82 | 9.05 | 0.96 | 0.90 |
| | 8000 | 4 | 5 | 216 | 400772 | 401413 | 402133 | 37030392 | 37277393 | 37572989 | 231.20 | 48.24 | 25.57 | 4.79 | 9.04 | 0.96 | 0.90 |

For the test problems shown, a blocksize of 250 was used for the 8,000 and 4,000 DMU cases, 125 for the 2,000 DMU cases, and 100 for the 1,000 DMU cases. The effects of different blocksizes must be investigated, hence, these results should not be viewed as the best possible. During level 2 of the HDEA procedure, $\beta = 1.5$. As a result, each subsequent block at level 2 grows by 50% until all DMUs have been classified as either efficient or inefficient.

Table 2.5 shows that for all test runs the speedup was nearly linear. Larger more difficult problems resulted in better speedup than smaller problems. With the smaller problems, the overall solution time in parallel is so small that the overhead of maintaining the self-scheduling tasks, as well as possible workload imbalance between processors, becomes apparent. Consequently, although the parallel results for small problems are quite good, the parallel implementation will be more efficient for large, difficult problems.

Table 2.5 also gives the total number of pivots and pricing operations executed to find the DEA scores for each data set. These numbers vary with the number of processors because of the method used to update the status of each DMU. As expected, problems with a greater number of DMUs, and those with a larger density of efficient DMUs, require more pivots and pricing operations and, consequently, take longer to solve.

Tables 2.6 through 2.10 contain the results of each test problem for each level of the HDEA procedure. The speedup remains relatively consistent across all problems across all levels and is nearly linear.

The effects of using early identification of efficient DMUs to reduce the number of linear programming problems that must be solved are shown in Tables 2.11 through 2.15. With the HDEA procedure, the total number of linear programs that must be solved is slightly more than twice the original number of DMUs contained in each data set. However, since the HDEA linear programs are smaller than other DEA methods, overall performance is improved. EIE improves performance further by reducing the number of

Table 2.6. -- Bank hierarchical level test problem results

| DMUs | | Number of Pivots | | | Number Priced | | | Time to Solve(min.) | | | Speedup | | Efficiency | | Fraction of Time | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 | 1 | 5 | 10 | 5 | 10 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level 1 | 14739 | 14968 | 15458 | 504349 | 528431 | 562621 | 3.92 | 0.87 | 0.53 | 4.51 | 7.40 | 0.90 | 0.74 | .32 | .31 | .31 |
| | Level 2 | 7865 | 8176 | 8361 | 322718 | 340406 | 353865 | 2.41 | 0.53 | 0.30 | 4.55 | 8.03 | 0.91 | 0.80 | .19 | .19 | .18 |
| | Level 3 | 21388 | 21388 | 21388 | 763514 | 763514 | 763514 | 6.08 | 1.38 | 0.87 | 4.41 | 6.99 | 0.88 | 0.70 | .49 | .50 | .51 |
| 2000 | Level 1 | 30770 | 31234 | 31679 | 1493465 | 1558862 | 1628509 | 10.53 | 2.30 | 1.31 | 4.58 | 8.04 | 0.92 | 0.80 | .32 | .32 | .27 |
| | Level 2 | 14486 | 14702 | 14922 | 820049 | 842246 | 870291 | 5.82 | 1.22 | 0.67 | 4.77 | 8.69 | 0.95 | 0.87 | .18 | .18 | .16 |
| | Level 3 | 47218 | 47218 | 47218 | 2258818 | 2558818 | 2258818 | 16.46 | 3.65 | 2.21 | 4.51 | 7.45 | 0.90 | 0.74 | .50 | .50 | .53 |
| 4000 | Level 1 | 80874 | 81060 | 81454 | 5347913 | 5438858 | 5441015 | 36.49 | 7.52 | 3.97 | 4.85 | 9.19 | 0.97 | 0.92 | .43 | .43 | .42 |
| | Level 2 | 20475 | 20845 | 21098 | 1614961 | 1663457 | 1671621 | 10.95 | 2.28 | 1.20 | 4.80 | 9.13 | 0.96 | 0.91 | .13 | .13 | .13 |
| | Level 3 | 88435 | 88435 | 88435 | 5130313 | 5130313 | 5130313 | 37.87 | 7.84 | 4.23 | 4.83 | 8.95 | 0.97 | 0.90 | .44 | .44 | .45 |
| 8000 | Level 1 | 165189 | 165432 | 165996 | 10774504 | 11051147 | 11232790 | 73.65 | 16.53 | 8.05 | 4.46 | 9.15 | 0.89 | 0.91 | .44 | .44 | .44 |
| | Level 2 | 41814 | 41985 | 42124 | 3252093 | 3295040 | 3332845 | 22.26 | 4.81 | 2.34 | 4.63 | 9.51 | 0.93 | 0.95 | .13 | .13 | .13 |
| | Level 3 | 175752 | 175752 | 175752 | 8863856 | 8863856 | 8863856 | 70.10 | 16.34 | 7.79 | 4.29 | 9.00 | 0.86 | 0.90 | .43 | .43 | .43 |

Table 2.7. -- Muli-normal hierarchical level test problem results

| DMUs | | Number of Pivots | | | Number Priced | | | Time to Solve(min.) | | | Speedup | | Efficiency | | Fraction of Time | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 | 1 | 5 | 10 | 5 | 10 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 14739 | 14968 | 15458 | 504349 | 528431 | 562621 | 3.92 | 0.87 | 0.53 | 4.51 | 7.40 | 0.90 | 0.74 | .32 | .31 | .31 |
| | Level2 | 7865 | 8176 | 8361 | 322718 | 340406 | 353865 | 2.41 | 0.53 | 0.30 | 4.55 | 8.03 | 0.91 | 0.80 | .19 | .19 | .18 |
| | Level3 | 21388 | 21388 | 21388 | 763514 | 763514 | 763514 | 6.08 | 1.38 | 0.87 | 4.41 | 6.99 | 0.88 | 0.70 | .49 | .50 | .51 |
| 2000 | Level1 | 27897 | 27969 | 28102 | 2239921 | 2279134 | 2330107 | 12.36 | 2.57 | 1.37 | 4.81 | 9.02 | 0.96 | 0.90 | .44 | .44 | .44 |
| | Level2 | 3928 | 3940 | 3979 | 366038 | 371995 | 380488 | 1.98 | 0.41 | 0.22 | 4.83 | 9.00 | 0.97 | 0.90 | .07 | .07 | .07 |
| | Level3 | 30593 | 30593 | 30593 | 2378773 | 2378773 | 2378773 | 13.69 | 2.87 | 1.55 | 4.77 | 8.83 | 0.95 | 0.88 | .49 | .49 | .49 |
| 4000 | Level1 | 55307 | 55535 | 55790 | 4420400 | 4511812 | 4610712 | 24.29 | 5.09 | 2.71 | 4.77 | 8.96 | 0.95 | 0.90 | .41 | .41 | .41 |
| | Level2 | 12400 | 12437 | 12599 | 1131057 | 1144794 | 1173569 | 6.31 | 1.30 | 0.70 | 4.85 | 9.01 | 0.97 | 0.90 | .11 | .11 | .11 |
| | Level3 | 65177 | 65177 | 65177 | 4939013 | 4939013 | 4939013 | 28.81 | 5.97 | 3.25 | 4.83 | 8.86 | 0.97 | 0.89 | .48 | .48 | .48 |
| 8000 | Level1 | 112154 | 112564 | 113192 | 8900632 | 9068521 | 9286181 | 49.50 | 10.28 | 5.48 | 4.82 | 9.03 | 0.96 | 0.90 | .39 | .39 | .38 |
| | Level2 | 23633 | 23763 | 23917 | 2149424 | 2175363 | 2214270 | 12.17 | 2.49 | 1.30 | 4.89 | 9.36 | 0.98 | 0.94 | .10 | .09 | .09 |
| | Level3 | 146892 | 146892 | 146892 | 11690351 | 11690351 | 11690351 | 66.40 | 13.87 | 7.52 | 4.79 | 8.83 | 0.96 | 0.88 | .51 | .52 | .53 |

Table 2.8. -- DEA-GENa hierarchical level test problem results

| DMUs | | Number of Pivots | | | Number Priced | | | Time to Solve(min.) | | | Speedup | | Efficiency | | Fraction of Time | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 | 1 | 5 | 10 | 5 | 10 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 16360 | 16749 | 17042 | 538284 | 569692 | 599536 | 4.54 | 1.02 | 0.60 | 4.45 | 7.57 | 0.89 | 0.76 | .29 | .29 | .29 |
| | Level2 | 7960 | 8114 | 8568 | 340961 | 352041 | 377144 | 2.66 | 0.58 | 0.34 | 4.59 | 7.82 | 0.92 | 0.78 | .17 | .17 | .17 |
| | Level3 | 26782 | 26782 | 26782 | 1042916 | 1042916 | 1042916 | 8.61 | 1.90 | 1.10 | 4.53 | 7.83 | 0.91 | 0.78 | .54 | .54 | .54 |
| 2000 | Level1 | 25295 | 25565 | 26010 | 1331525 | 1377928 | 1442319 | 8.58 | 1.86 | 1.08 | 4.61 | 7.94 | 0.92 | 0.79 | .26 | .26 | .27 |
| | Level2 | 16902 | 17007 | 17224 | 1072520 | 1090581 | 1115055 | 6.77 | 1.39 | 0.75 | 4.87 | 9.03 | 0.97 | 0.90 | .20 | .20 | .19 |
| | Level3 | 45512 | 45512 | 45512 | 2807122 | 2807122 | 2807122 | 18.29 | 3.87 | 2.17 | 4.73 | 8.43 | 0.95 | 0.84 | .54 | .54 | .54 |
| 4000 | Level1 | 69334 | 69591 | 69879 | 5397968 | 5501412 | 5623544 | 33.95 | 7.08 | 3.72 | 4.80 | 9.13 | 0.96 | 0.91 | .41 | .41 | .41 |
| | Level2 | 18130 | 18341 | 18628 | 1935924 | 1966627 | 2014127 | 12.07 | 2.47 | 1.29 | 4.89 | 9.36 | 0.98 | 0.94 | .14 | .14 | .14 |
| | Level3 | 81349 | 81349 | 81349 | 5348339 | 5348339 | 5348339 | 37.36 | 7.63 | 4.10 | 4.90 | 9.11 | 0.98 | 0.91 | .45 | .45 | .45 |
| 8000 | Level1 | 93421 | 93782 | 94278 | 7692497 | 7842514 | 8035761 | 42.46 | 8.90 | 4.79 | 4.77 | 8.86 | 0.95 | 0.89 | .39 | .40 | .40 |
| | Level2 | 26747 | 26863 | 27075 | 3274462 | 3314619 | 3375084 | 17.28 | 3.53 | 1.84 | 4.90 | 9.39 | 0.98 | 0.94 | .16 | .16 | .15 |
| | Level3 | 112940 | 112940 | 112940 | 7805087 | 7805087 | 7805087 | 47.87 | 9.97 | 5.45 | 4.80 | 8.78 | 0.96 | 0.88 | .45 | .44 | .45 |

Table 2.9. -- DEA-GENb hierarchical level test problem results

| DMUs | | Number of Pivots | | | Number Priced | | | Time to Solve (min.) | | | Speedup | | Efficiency | | Fraction of Time | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 | 1 | 5 | 10 | 5 | 10 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 14439 | 14815 | 15240 | 556774 | 584561 | 615245 | 4.92 | 1.09 | 0.65 | 4.51 | 7.57 | 0.90 | 0.76 | .16 | .17 | .19 |
| | Level2 | 24853 | 25222 | 25548 | 1302084 | 1327165 | 1351858 | 10.76 | 2.25 | 1.21 | 4.78 | 8.89 | 0.96 | 0.89 | .35 | .35 | .34 |
| | Level3 | 33669 | 33669 | 33669 | 1671769 | 1671769 | 1671769 | 14.92 | 3.09 | 1.65 | 4.83 | 9.04 | 0.97 | 0.90 | .49 | .48 | .47 |
| 2000 | Level1 | 24953 | 25337 | 25788 | 1404447 | 1456216 | 1514152 | 9.64 | 2.09 | 1.20 | 4.61 | 8.03 | 0.92 | 0.80 | .17 | .18 | .19 |
| | Level2 | 32561 | 33021 | 33294 | 2456025 | 2506954 | 2545057 | 16.21 | 3.36 | 1.79 | 4.82 | 9.06 | 0.96 | 0.91 | .29 | .29 | .29 |
| | Level3 | 55397 | 55397 | 55397 | 4179972 | 4179972 | 4179972 | 29.52 | 6.09 | 3.22 | 4.85 | 9.17 | 0.97 | 0.92 | .54 | .53 | .52 |
| 4000 | Level1 | 44515 | 45676 | 45831 | 3647181 | 3720602 | 3801151 | 18.56 | 3.86 | 2.08 | 4.81 | 8.92 | 0.96 | 0.89 | .47 | .47 | .46 |
| | Level2 | 7166 | 7155 | 7141 | 674808 | 678006 | 683015 | 3.47 | 0.70 | 0.36 | 4.96 | 9.64 | 0.99 | 0.96 | .09 | .08 | .08 |
| | Level3 | 46406 | 46406 | 46406 | 3076003 | 3076003 | 3076003 | 17.74 | 3.73 | 2.05 | 4.76 | 8.65 | 0.95 | 0.87 | .44 | .45 | .46 |
| 8000 | Level1 | 136962 | 137406 | 138254 | 11300447 | 11493854 | 11764096 | 75.94 | 15.67 | 8.28 | 4.85 | 9.17 | 0.97 | 0.92 | .31 | .30 | .30 |
| | Level2 | 66459 | 66735 | 67432 | 7522680 | 7601983 | 7748104 | 50.08 | 10.19 | 5.24 | 4.91 | 9.56 | 0.98 | 0.96 | .20 | .20 | .19 |
| | Level3 | 215839 | 215839 | 215839 | 17847952 | 17847952 | 17847952 | 122.90 | 25.77 | 14.02 | 4.77 | 8.76 | 0.95 | 0.88 | .49 | .50 | .51 |

Table 2.10. – DEA-GENc hierarchical level test problem results

| DMUs | | Number of Pivots | | | Number Priced | | | Time to Solve (min.) | | | Speedup | | Efficiency | | Fraction of Time | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 | 1 | 5 | 10 | 5 | 10 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 16890 | 17438 | 17904 | 632567 | 668930 | 705190 | 5.87 | 1.36 | 0.76 | 4.32 | 7.72 | 0.86 | 0.77 | .17 | .19 | .19 |
| | Level2 | 26070 | 26318 | 26910 | 1310594 | 1329927 | 1369047 | 11.53 | 2.40 | 1.31 | 4.80 | 8.80 | 0.96 | 0.88 | .34 | .33 | .34 |
| | Level3 | 36220 | 36220 | 36220 | 1750429 | 1750429 | 1750429 | 16.57 | 3.43 | 1.84 | 4.83 | 9.01 | 0.97 | 0.90 | .49 | .48 | .47 |
| 2000 | Level1 | 30545 | 31123 | 31848 | 1663487 | 1732443 | 1815061 | 13.02 | 2.81 | 1.59 | 4.63 | 8.19 | 0.93 | 0.82 | .16 | .17 | .18 |
| | Level2 | 41269 | 41714 | 42380 | 2975857 | 3024266 | 3093706 | 22.37 | 4.63 | 2.45 | 4.83 | 9.13 | 0.97 | 0.91 | .28 | .28 | .28 |
| | Level3 | 79890 | 79890 | 79890 | 5571196 | 5571196 | 5571196 | 44.34 | 9.14 | 4.78 | 4.85 | 9.28 | 0.97 | 0.93 | .56 | .55 | .54 |
| 4000 | Level1 | 59940 | 60099 | 60225 | 4627122 | 4708855 | 4805916 | 27.30 | 5.66 | 2.99 | 4.82 | 9.13 | 0.96 | 0.91 | .47 | .47 | .47 |
| | Level2 | 9627 | 9679 | 9696 | 979096 | 989890 | 1000100 | 5.82 | 1.19 | 0.61 | 4.89 | 9.54 | 0.98 | 0.95 | .10 | .10 | .10 |
| | Level3 | 57906 | 57906 | 57906 | 3741816 | 3741816 | 3741816 | 24.60 | 5.12 | 2.78 | 4.80 | 8.85 | 0.96 | 0.88 | .43 | .43 | .43 |
| 8000 | Level1 | 128400 | 128880 | 129358 | 10534301 | 10730605 | 10964250 | 65.30 | 13.66 | 7.24 | 4.78 | 9.02 | 0.96 | 0.90 | .28 | .28 | .28 |
| | Level2 | 54664 | 54825 | 55067 | 5780407 | 5831104 | 5893055 | 35.75 | 7.30 | 3.75 | 4.90 | 9.53 | 0.98 | 0.95 | .15 | .15 | .15 |
| | Level3 | 217708 | 217708 | 217708 | 20715684 | 20715684 | 20715684 | 130.10 | 27.28 | 14.58 | 4.77 | 8.92 | 0.95 | 0.89 | .57 | .57 | .57 |

linear programs at level 1 and level 2 by 10–25%. Early identification has no effect at level 3 since only DEA scores for inefficient DMUs are found at that level.

Table 2.16 shows the dramatic effect of the hierarchical procedure on improving solution time for the 8,000 DMU cases over the best DEA approach reported to date. Only three processors are used in the parallel implementation because of memory limitations of the non-hierarchical procedures. The HDEA procedure results in a 6- to 12-fold increase in speed over the non-hierarchical procedure when both cases are run with the same number of processors. Because the HDEA procedure requires less memory, permitting use of more processors, the cases with 15 processors are 75 to 125 times faster than the serial non-hierarchical procedure.

Although the test problems were limited to 8,000 DMUs because of the availability of real life data, the HDEA procedure can accommodate much larger problems. A Cobb-Douglas test case (Appendix B) consisting of 25,000 DMUs with 3 inputs and 3 outputs was solved using 15 processors in 19.26 minutes. Because of memory limitations, it was not possible to solve this problem with the non-hierarchical procedure.

As seen with the test problems, the HDEA procedure allows for the solution of large-scale DEA problems much faster than previously reported DEA methods. When coupled with a parallel environment, the HDEA procedure yields solutions to problems in a matter of minutes which previously, would have taken days. Additionally, the HDEA procedure allows for the solution of very large-scale DEA problems that remain unsolvable (in a practical sense) with any other reported approaches.

### 2.4.5. Further Advantages of the HDEA Approach

Accompanying the HDEA method's advances in sheer solution speed are a variety of additional computational advantages in other settings. The hierarchical structure can be exploited when solving multiple DEA models and further streamlining opportunities await exploration.

Table 2.11. -- Bank data results – linear programs required

| DMUs | | LPs No Early Identification | | | LPs With Early Identification | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 1000 | 1000 | 1000 | 795 | 807 | 827 |
| | Level2 | 408 | 408 | 408 | 294 | 304 | 312 |
| | Level3 | 922 | 922 | 922 | 922 | 922 | 922 |
| | Total | 2330 | 2330 | 2330 | 2011 | 2033 | 2061 |
| 2000 | Level1 | 2000 | 2000 | 2000 | 1649 | 1670 | 1694 |
| | Level2 | 653 | 653 | 653 | 509 | 519 | 530 |
| | Level3 | 1918 | 1918 | 1918 | 1918 | 1918 | 1918 |
| | Total | 4571 | 4571 | 4571 | 4076 | 4107 | 4142 |
| 4000 | Level1 | 4000 | 4000 | 4000 | 3552 | 3562 | 3577 |
| | Level2 | 779 | 779 | 779 | 638 | 648 | 653 |
| | Level3 | 3902 | 3902 | 3902 | 3902 | 3902 | 3902 |
| | Total | 8681 | 8681 | 8681 | 8092 | 8112 | 8132 |
| 8000 | Level1 | 8000 | 8000 | 8000 | 7092 | 7116 | 7149 |
| | Level2 | 1487 | 1487 | 1487 | 1278 | 1288 | 1289 |
| | Level3 | 7923 | 7923 | 7923 | 7923 | 7923 | 7923 |
| | Total | 17410 | 17410 | 17410 | 16293 | 16327 | 16361 |

Table 2.12. – Multi-normal data results – linear programs required

| DMUs | | LPs No Early Identification | | | LPs With Early Identification | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 1000 | 1000 | 100 | 795 | 807 | 827 |
| | Level2 | 408 | 408 | 408 | 294 | 304 | 312 |
| | Level3 | 922 | 922 | 922 | 922 | 922 | 922 |
| | Total | 2330 | 2330 | 2330 | 2011 | 2033 | 2061 |
| 2000 | Level1 | 2000 | 2000 | 2000 | 1781 | 1783 | 1788 |
| | Level2 | 282 | 282 | 282 | 211 | 212 | 213 |
| | Level3 | 1911 | 1911 | 1911 | 1911 | 1911 | 1911 |
| | Total | 4193 | 4193 | 4193 | 3903 | 3906 | 3912 |
| 4000 | Level1 | 4000 | 4000 | 4000 | 3530 | 3542 | 3549 |
| | Level2 | 709 | 709 | 709 | 569 | 570 | 579 |
| | Level3 | 3915 | 3915 | 3915 | 3915 | 3915 | 3915 |
| | Total | 8024 | 8624 | 8624 | 8014 | 8027 | 8043 |
| 8000 | Level1 | 8000 | 8000 | 8000 | 7106 | 7120 | 7145 |
| | Level2 | 1295 | 1295 | 1295 | 1074 | 1080 | 1091 |
| | Level3 | 7896 | 7896 | 7896 | 7896 | 7896 | 7896 |
| | Total | 17191 | 17191 | 17191 | 16076 | 16096 | 16132 |

Table 2.13. -- DEA-GENa results – linear programs required

| DMUs | | LPs No Early Identification | | | LPs With Early Identification | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 1000 | 1000 | 1000 | 774 | 791 | 797 |
| | Level2 | 435 | 435 | 435 | 288 | 291 | 300 |
| | Level3 | 904 | 904 | 904 | 904 | 904 | 904 |
| | Total | 2339 | 2339 | 2339 | 1966 | 1986 | 2001 |
| 2000 | Level1 | 2000 | 2000 | 2000 | 1520 | 1533 | 1566 |
| | Level2 | 972 | 972 | 972 | 682 | 687 | 692 |
| | Level3 | 1853 | 1853 | 1853 | 1853 | 1853 | 1853 |
| | Total | 4825 | 4825 | 4825 | 4055 | 4073 | 4111 |
| 4000 | Level1 | 4000 | 4000 | 4000 | 3522 | 3530 | 3540 |
| | Level2 | 729 | 729 | 729 | 612 | 616 | 623 |
| | Level3 | 3918 | 3918 | 3918 | 3918 | 3918 | 3918 |
| | Total | 8647 | 8647 | 8647 | 8052 | 8064 | 8081 |
| 8000 | Level1 | 8000 | 8000 | 8000 | 7029 | 7050 | 7079 |
| | Level2 | 1465 | 1465 | 1465 | 1271 | 1276 | 1285 |
| | Level3 | 7930 | 7930 | 7930 | 7930 | 7930 | 7930 |
| | Total | 17395 | 17395 | 17395 | 16230 | 16256 | 16294 |

Table 2.14. – DEA-GENb results – linear programs required

| DMUs | | LPs No Early Identification | | | LPs With Early Identification | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 1000 | 1000 | 1000 | 670 | 688 | 709 |
| | Level2 | 1105 | 1105 | 1105 | 716 | 729 | 740 |
| | Level3 | 786 | 786 | 786 | 786 | 786 | 786 |
| | Total | 2891 | 2891 | 2891 | 2172 | 2203 | 2235 |
| 2000 | Level1 | 2000 | 2000 | 2000 | 1436 | 1455 | 1475 |
| | Level2 | 1785 | 1785 | 1785 | 1173 | 1190 | 1201 |
| | Level3 | 1725 | 1725 | 1725 | 1725 | 1725 | 1725 |
| | Total | 5510 | 5510 | 5510 | 4334 | 4370 | 4401 |
| 4000 | Level1 | 4000 | 4000 | 4000 | 3636 | 3644 | 3650 |
| | Level2 | 438 | 438 | 438 | 384 | 384 | 383 |
| | Level3 | 3936 | 3936 | 3936 | 3936 | 3936 | 3936 |
| | Total | 8374 | 8374 | 8374 | 7956 | 7964 | 7969 |
| 8000 | Level1 | 8000 | 8000 | 8000 | 6678 | 6698 | 6507 |
| | Level2 | 2326 | 2326 | 2326 | 1893 | 1898 | 2506 |
| | Level3 | 7847 | 7847 | 7847 | 7847 | 7847 | 7886 |
| | Total | 18173 | 18173 | 18173 | 16418 | 16443 | 16899 |

Table 2.15. – DEA-GENc results – linear programs required

| DMUs | | LPs No Early Identification | | | LPs With Early Identification | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 10 | 1 | 5 | 10 |
| 1000 | Level1 | 1000 | 1000 | 1000 | 684 | 704 | 726 |
| | Level2 | 1052 | 1052 | 1052 | 670 | 684 | 700 |
| | Level3 | 799 | 799 | 799 | 799 | 799 | 799 |
| | Total | 2851 | 2851 | 2851 | 2153 | 2187 | 2225 |
| 2000 | Level1 | 2000 | 2000 | 2000 | 1415 | 1446 | 1480 |
| | Level2 | 1806 | 1806 | 1806 | 1181 | 1194 | 1213 |
| | Level3 | 1735 | 1735 | 1735 | 1735 | 1735 | 1735 |
| | Total | 5541 | 5541 | 5541 | 4331 | 4375 | 4428 |
| 4000 | Level1 | 4000 | 4000 | 4000 | 3611 | 3616 | 3624 |
| | Level2 | 462 | 462 | 462 | 413 | 413 | 413 |
| | Level3 | 3934 | 3934 | 3934 | 3934 | 3934 | 3934 |
| | Total | 8396 | 8396 | 8396 | 7958 | 7963 | 7971 |
| 8000 | Level1 | 8000 | 8000 | 8000 | 6730 | 6757 | 6782 |
| | Level2 | 2181 | 2181 | 2181 | 1712 | 1716 | 1724 |
| | Level3 | 7784 | 7784 | 7784 | 7784 | 7784 | 7784 |
| | Total | 17965 | 17965 | 17965 | 16226 | 16257 | 16290 |

Table 2.16. -- Non-hierarchical and hierarchical comparisons: 8,000 DMU problems

| | Solution Times (min.) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | No Hierarchical | | Hierarchical | | | | Speed Improvement | | | | |
| | 1 | 3 | 1 | 3 | 10 | 15 | 1:1 | 3:3 | 1:3 | 1:10 | 1:15 |
| Bank | 1294.07 | 434.58 | 166.00 | 61.73 | 18.18 | 14.27 | 7.80 | 7.04 | 20.96 | 71.18 | 90.68 |
| Multi-Normal | 931.98 | 313.15 | 128.10 | 43.72 | 14.30 | 10.54 | 7.28 | 7.16 | 21.32 | 65.17 | 88.42 |
| DEA-GENa | 1088.50 | 366.51 | 107.60 | 31.92 | 12.08 | 8.83 | 10.12 | 11.48 | 34.10 | 90.11 | 123.27 |
| DEA-GENb | 2091.20 | 704.13 | 248.90 | 84.47 | 27.53 | 19.89 | 8.40 | 8.34 | 24.76 | 75.96 | 105.14 |
| DEA-GENc | 1437.32 | 482.84 | 231.20 | 78.88 | 25.57 | 18.55 | 6.22 | 6.12 | 18.22 | 56.21 | 77.48 |

### 2.4.5.1. Solving Multiple Models

There are numerous instances in which multiple models must be solved for each DMU under consideration. The HDEA approach can be easily modified and extended to efficiently determine all of the required values.

- To solve for technical, scale, and overall efficiency (inefficiency), both the BCC and CCR models must be solved [85]. With HDEA, We can use the BCC or ADD model through level 2 to find $E^*_{bcc}$. On only the $E^*_{bcc}$ set we can apply the CCR model formulation to find the $E^*_{ccr}$ ($\subseteq E^*_{bcc}$) set of DMUs. During level 3, while determining the inefficient scores, we can solve the values for $I^*_{ccr}$. Then, as an advanced starting basis, we can allow all $E^*_{bcc}$ DMUs to enter the basis and solve for $I^*_{bcc}$. (Alternately, we could solve for the $I_{bcc}$ values first, then relax the BCC $\mathbf{1\lambda} = 1$ constraint, tighten the basis entry rules, and solve for the $I_{ccr}$ scores). Once the BCC and CCR values are found, we can rapidly solve for the three efficiency values desired. Note that this prevents a duplication of effort to find the inefficient values.

- Byrnes, Färe, and Grosskopf [28] argue that a "no increasing returns to scale" (NIRS) model must be solved to determine if the DMU (or the virtual DMU if inefficient) is in a region of increasing or decreasing returns to scale. Knowing this may help indicate the direction the DMU should go to become efficient, either expand operations through output augmentation, or downsize through input contraction. We know that $E^*_{ccr} \subseteq E^*_{nirs} \subseteq E^*_{bcc}$. Consequently, the NIRS strategy could be added to the strategy above to find these scores.

- Besides solving for BCC, CCR, NIRS models, we can also apply assurance region restrictions [84,85] once the $E^*$ for each model is found at level 2. The various assurance region approaches can be solved at level 2 then extended to level 3. In this way, a series of models can be solved rapidly without duplication of effort.

We know $E^*_{ar} \subseteq E^*$ for any of the above models. Note: this same approach can be used for the cone-ratio model of Charnes, et al [34,35].

- For the CCR and BCC models, there is an associated orientation, either input or output. The inefficiency scores will vary depending on the orientation, but the $E^*_{bcc}$ and $E^*_{ccr}$ sets will remain the same regardless of the orientation. Consequently, if both orientations are needed, then only the level 3 values must be resolved to obtain both.

- Some versions of the model may provide faster solutions than others depending on the number of output and input variables that are used. For example, if the number of outputs exceeds the number of inputs, the output-oriented model solves faster. Also, the $CCR^o$ model has an initial feasible solution, thus bypassing Phase I of the simplex method. Since the output model may solve faster, this can be used through level 2, then the input model applied in level 3 calculations if the input values are needed. HDEA provides the flexibility to incorporate the fastest model to identify $E^*$ through level 2, then at level 3 can use the model of choice to determine the inefficiency scores.

### 2.4.5.2. Implementation Issues

- As noted above, any method can be used to find the set of DMUs belonging to $E^*$ and $I^*$ such as sorting, preprocessing, domination theory, etc. These can possibly enhance the HDEA procedure to expedite the level 2 and level 3 efficiency score values.

- Computational concerns have arisen over the two-phase [4] or [46] approaches, which can be lessened via HDEA. We know that the efficiency scores do not vary with the sum-of-slack solutions. So in HDEA, there is no need to solve for the sum of slacks during level 1 or most of level 2. Once potential $E^*$ DMUs are identified, the sum of slacks must be solved only for these DMUs to determine if they are weakly efficient. If there are any weakly efficient DMUs, they are removed from E*. During level 3, there is no need to solve for the sum of slacks for the inefficient DMUs since a proper projection to the efficient surface is obtained since no weakly efficient DMUs can enter the basis. Solving for the sum of slacks for the inefficient DMUs may simply identify possible alternate optimal solutions that offer no new information when weakly efficient DMUs are not in the basis.

- As the number of input/output variables grows, the likelihood of cycling also increases because of severe degeneracy involved with the solution of some DMUs (especially efficient DMUs). Anti-cycling procedures invoke a high computational cost. Also, the possibility of cycling can increase with a larger number of DMUs (variables) in the model. HDEA can help reduce the cost of cycling by maintaining smaller problems. Additionally, at level 1 and most of level 2 there is no need to invoke the anti-cycling rules. Any DMUs that exhibit cycling can be deferred. Only when all potential members of E* are identified at level 2, must the anticycling rules be invoked. By passing over the DMUs that cycle, the basic variables of other DMUs' solutions may later reveal the problematic DMUs as efficient. In this manner, the DMUs will not need to be solved and the cost of applying anti-cycling rules is reduced.

#### 2.4.5.3. Exploiting the Levels

- As we progress through level 1 to level 2 to level 3, information at each level can be used to enhance the solutions at the next level. DMUs that frequently show up in efficient reference sets can be given priority during the pricing procedure to enter the basis. By choosing the most attractive variables to enter the basis, the number of pivots to solve the LPs may be reduced. This may also reduce the potential for cycling. This also indicates the possibility of using different heuristics for the entering and leaving variables for the different levels of the HDEA process.

- One advantage of the HDEA process over domination and other preprocessing techniques is that it can be used to find DEA scores for categorical variables and window analysis enroute to determining overall DEA scores. By blocking appropriately, the efficient DMUs for each category or window can be recorded along the way. In this way, solving overall scores as well as those within particular categories or across various categories can be accomplished without duplication of effort.

2.4.5.4. Parallel Implementation

- The HDEA procedure can expand the parallel approach across many platforms. For example, a SIMD distributed network can be used to solve each block of DMUs. Since the blocks are smaller, memory requirements are smaller. The advantage of updated restricted basis entry and early identification of DMU status within blocks will be lost, but the basic process will still lead to a solution. The basis entry and identification schemes will hold between levels.
- For the parallel case, Amdahl's law assumes that the same information available for the single processor as for multiple processors. But with RBE and early identification, the interaction between processors can enhance the solution process resulting in high levels of efficiency when multiple processors are used. This is not an HDEA-exclusive advantage, but it does hold for the HDEA process.

## 2.5. Conclusions

We have described a new hierarchical decomposition procedure for solving DEA problems that advances the state of the art for computational data envelopment analysis. As demonstrated with medium- and large-scale test sets, this approach can have dramatic benefits over traditional methods—in both single-processor and parallel settings—and permits enormous problems to be optimized in a modest amount of time. The ability to routinely solve problems with thousands of DMUs permits researchers and practitioners to be more ambitious in their application of this important class of models and, we hope, will encourage new and even more exciting applications of DEA.

# CHAPTER III

# RANK ORDERING DMUS USING TIERED DEA

Data envelopment analysis was originally developed for efficiency measurement. The primary focus to date has been on the development of a set of DEA models to identify the efficient and inefficient DMUs under different sets of assumptions designed to measure various types of inefficiencies. In spite of the tremendous growth and success of the DEA methodology, there has been little research into methodologies for ordering the DMUs according to the efficiency of their production and management practices. The inability to rank DMUs by the comparative degree of efficiency or inefficiency limits the potential for DEA to fully characterize successful and unsuccessful management practices.

Although inefficient DMUs receive a DEA score which reflects the degree of inefficiency, a direct comparison of DMUs is problematic unless both DMUs have the same *efficient reference set* (ERS). To illustrate, Figure 3.1 depicts a unit isoquant plot of a set of firms with one output and two inputs. Since each point represents a firm and all firms produce the same level of output, the more efficient units are closer to the origin. The efficient frontier is formed by DMUs A, B, C and D. DMUs E, F, and G are all inefficient. The efficiency of each point is determined by a ratio whose denominator is the length of the line segment from the point to the origin and the numerator is the length of the line segment from the origin to the efficient boundary. The line connecting an inefficient DMU's point and the origin will intersect one of the line segments forming the efficient frontier. The end points of this line segment, composed of efficient DMU points, form the efficient reference set for the inefficient DMU.

According to Charnes and Cooper [32], to compare DMUs with different efficient reference sets would require assumptions of the weighting (or pricing) scheme used by DEA. But it is precisely this lack of restrictions on the weighting scheme that makes the DEA methodology so attractive. Consequently, in general such assumptions are undesirable. In our example, DMU$_F$ can be compared to DMU$_G$ because they share the same ERS consisting of DMU$_C$ and DMU$_D$. In this case DMU$_G$ with an efficiency score of 0.800 is more efficient than DMU$_F$ with an efficiency score of 0.774. However, neither of these DMUs should be compared with DMU$_E$, with an efficiency score of 0.733, which has a different ERS composed of DMUs B and C. Hence, for inefficient DMUs, a new approach is necessary to further discriminate and allow comparisons across all inefficient DMUs.

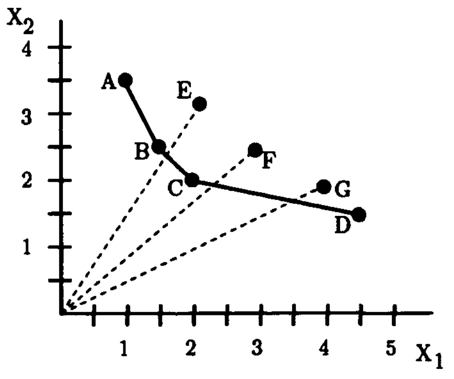

Figure 3.1. Normalized data isoquant plot

For efficient DMUs, the relative importance of each DMU is difficult to discern. Because each such unit has a DEA score of 1, there exists no variation in scores to determine a relative value. Charnes and Cooper [32] suggested a tool which they called the *envelopment map* to characterize the magnitude of the importance of each efficient DMU. This method consists of counting the number of times each efficient DMU appeared as a member of an ERS. Those DMUs occurring more often would be considered more "consistently efficient." However, there are at least two problems with this measure. First, to correctly count all occurrences of an efficient DMU in an ERS, all alternate optimal solutions of the DEA models would need to be identified. This can be computationally expensive and difficult to track. Secondly, this counting offers only a limited amount of useful information. An efficient DMU that occurs often in an ERS merely indicates that the DMU helps define part of the efficient surface which overshadows a high concentration of inefficient DMUs. Firms utilizing new production techniques may be extremely efficient, yet operate far from the "crowd" of other DMUs. As a result, these efficient firms do not occur often in the efficient reference sets of the inefficient units. Consequently, these *maverick* DMUs may not be deemed as important as they should be.

To discriminate between and identify the successful and unsuccessful production practices of the DMUs, a new procedure is necessary that provides a more detailed classification of DMUs than the ordinary DEA efficiency scores offer. This procedure should result in a rank ordering of DMUs which serves as a proxy measure for managerial efficiency. In this way, managers and analysts will have a useful tool to identify management practices that both accentuate and detract from productive efficiency by observing the practices of the higher and lower ranked DMUs. Additionally, experts may want to state, a priori to the DEA analysis, what they believe to be the most efficient firms in the industry. The rank ordering procedure can then be used to determine how the DEA results compare with the experts' opinions. As in standard DEA analysis, the ordering should allow either constant or variable returns-to-scale envelopments. The remainder of this chapter presents just such a rank ordering procedure which is easy to implement and meets all of the above criteria.

## 3.1. Need for a Ranking Procedure

Until recently, most modern writing on production theory assumed that all producers were efficient. The premise was that in a competitive market place the inefficient producers would realize the direct costs and indirect opportunity costs of continued production and would leave the market to pursue more profitable adventures. However, economic analysts have come to accept that inefficient production occurs in the market place and its causes vary. Inefficient production can occur because information on the most productive methods is neither perfect nor free. As a result, some firms may be slower to respond to changing market conditions than others. Along with imperfect information, market uncertainty influences the production process. The organization's (or the manager's) position towards risk will dictate the rapidity with which the firm will respond to change in the shadow of this uncertainty. Additionally, because perfect competition is rarely (if ever) seen, regulations, and other exogenous constraints may induce inefficiencies in the production process. Because of the social costs associated with inefficient conversion of input resources to output goods, there has been a growing interest in identifying and quantifying the inefficient processes. Data envelopment analysis has proven useful in measuring various types of the production inefficiencies which may be attributed to inefficient managerial practices.

The original DEA models focused on identifying technical inefficiency or scale inefficiency [11]. Färe, Grosskopf, and Lovell [53], relaxed the usual DEA assumption of strong disposability of inputs and outputs to further identify inefficiencies due to congestion of resources.[1] In addition, Färe, Grosskopf, and Lovell, following the lead of Farrell [51], introduced prices of inputs and outputs to identify allocative inefficiencies.

[1] Congestion of inputs implies that as at least one input increases, at least one output is reduced. That is, there is not a positive correlation between all outputs and inputs.

A firm demonstrates allocative inefficiency when it departs from its predefined goal such as maximizing profits or minimizing costs.

Even though technical, scale, and allocative inefficiencies can be measured, little has been written to formulate a means of ranking the DMUs based on the types of inefficiencies they demonstrate. As a result, even though the inefficiencies can be identified, they have not fully been related to various aspects of producer behavior. With a ranking system, the DMUs exhibiting the best production processes, in terms of efficiency, could be compared to those characterized by the worst production techniques. The management practices of the best producing DMUs could then be compared to the worst DMUs in order to identify the underlying managerial inefficiencies. Once identified, the less efficient firms could adopt the practices of the best firms to improve the productivity of their operations.

The purpose of this study is to present a new approach to rank order or stratify the DMUs to more clearly relate the efficiency (or inefficiency) of a given DMU to all others in the set. The intent is not to suggest this approach as the only valid rank ordering scheme. Indeed, any set of items can be ranked by any subjective means. However, the purpose is to present a methodology, with theoretical underpinnings, which can result in a meaningful ordering system. It is hoped that this methodology may stimulate research into other possible ranking procedures so that the richness of the DEA methodology can be more fully utilized.

### 3.1.1. Basis of the Ranking Procedure

Data envelopment analysis defines efficiency based on empirical observations of each evaluation unit's behavior. Each DMU consumes multiple inputs in order to produce one or more outputs. The implicit assumption in DEA is that the DMUs transform the inputs into outputs by means of a well-behaved production technology. A feasible input-output vector for a given DMU is technically efficient (Pareto-Koopmans efficient) if it is technologically impossible to increase any output and/or to reduce any input without simultaneously reducing at least one other output and/or increasing at least one other input. The empirical models of DEA use observed data to identify the DMUs which form the Pareto-Koopmans efficient surface, and hence, are best at using the current technology to convert the input resources to output goods. The level of technological achievement revealed by the efficient surface will be highly dependent upon the choice of DMUs used in the study since the methodology only measures relative (not absolute) efficiency.

The ranking method presented in this study separates DMUs into groups based on the level of technological achievement which they demonstrate. It will be shown that this aggregation reveals additional aspects of inefficiencies not available with traditional DEA measures. For example, a DMU that appears to be very inefficient by the standard DEA measures, may rank well, compared to other firms, when viewed in terms of the DMU's ability to employ the most recent technological advances. Once the DMUs are separated into these achievement levels, a procedure will be presented to rank the DMUs within each level. The ranking within a level will be determined by the contribution the DMU makes to defining the shape of the efficient surface of that level.

An attractive feature of the proposed ranking procedure is that it can be used across the many formulations of the DEA models. In this paper, the ranking procedure is introduced and applied to both the input- and output-oriented BCC (variable-returns-to-scale) models. The input- and output-oriented models achieve identical stratification of DMUs into tiers of common technological achievement levels. However, the ranking within each tier will differ according to the orientation used. Next, the ranking procedure is applied to the CCR (constant returns to scale) model. In this case, the input- and output-oriented schemes result in identical rankings. The CCR model adds interesting interpretations of the most productive scale size to achieve each technological level.

## 3.2. Tiered DEA

At the heart of data envelopment analysis is the separation of evaluation units into (relatively) efficient and inefficient sets. The models' objective function values, $\theta$ or $z$, have been used as metrics for the *degree* of inefficiency for comparative and predictive purposes [21,22]. Since these values may be incompatible, from an economic point-of-view, we present a different approach to comparing DMUs that has significant appeal, from both intuitive and economic-theoretic standpoints.

### 3.2.1. Tiering Algorithm

The following *tiered DEA* (TDEA) procedure stratifies decision units into tiers, or layers, of comparable productive efficiency, as measured by any standard DEA model.

**Procedure TDEA**

```
1. Initialize: $t \leftarrow 1, \mathrm{D}_{[1]} \leftarrow \mathrm{D}$.
2. While $\mathrm{D}_{[t]} \neq \emptyset$ do:
    a. Apply a DEA model to the DMUs in set $\mathrm{D}_{[t]}$ to identify $\mathrm{E}^*_{[t]}$.
    b. $\mathrm{I}^*_{[t]} = \mathrm{D}_{[t]} - \mathrm{E}^*_{[t]}$.
    c. $t \leftarrow t + 1$.
    d. $\mathrm{D}_{[t]} = \mathrm{I}^*_{[t]}$.
```

where $t$ is a tier index and $\mathrm{E}^*_{[t]}$ and $\mathrm{I}^*_{[t]}$ are the sets of efficient and inefficient DMUs on tier $t$, respectively, relative to set $\mathrm{D}_{[t]}$.

The TDEA procedure begins with a traditional data envelopment analysis, then progressively strips away production surfaces, revealing a series of frontiers of decreasing productivity. Specifically, at tier 1 all of the DMUs in the data set are analyzed using a standard DEA model, thus separating them into efficient and inefficient sets. The efficient units are then assigned to the current tier and the inefficient ones become the data set of interest for the next higher tier; this process is applied recursively until all

DMUs are assigned to a tier. In this way, the tier levels represent successive layers of relatively efficient production surfaces where the DMUs at any given tier are less productively efficient than those of "outer" (lower-numbered) tiers and more efficient than DMUs at "inner" (higher-numbered) tiers.

An example application of the TDEA procedure can be seen in Figure 3.2 where 10 DMUs, each with 1 output and 2 inputs, are plotted. All DMUs have the same output level so only the inputs are shown. DMUs A, B, and C are DEA efficient since the line segments joining these DMUs envelop the other DMUs from below. If firms A, B, and C were removed from the data set, a new efficient frontier would be formed by DMUs D, E, F, and G. The TDEA procedure reveals that the data contains three production surface layers.

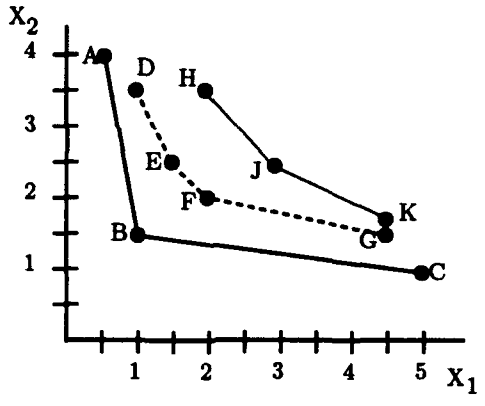

Figure 3.2. Example tiered results – two inputs, one output

Figure 3.3 shows 10 DMUs each with 1 input and 2 outputs. In this case, all DMUs use the same level of input so only the outputs are shown. Again, DMUs A, B, and C are efficient since they form a boundary that envelops the other DMUs from above. TDEA reveals three production surface layers.

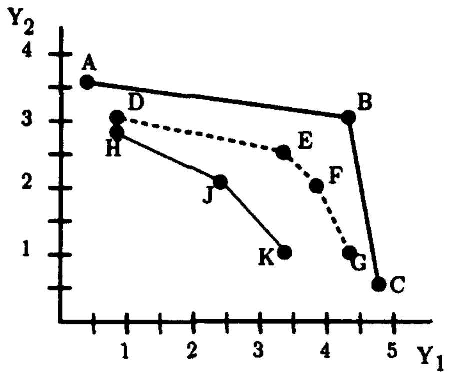

Figure 3.3. Example tiered results – one input, two outputs

### 3.2.2. Example TDEA Applications

The TDEA procedure was applied to three 8,000-DMU data sets, one from industry and the others randomly generated. The "Banking" data represents a selection of banks from the Federal Reserve Bank's Southwest district, with 6 input and 3 output values, as described in [21]. A set of "Cobb-Douglas" data with 5 input and 4 output values per observation was created with DEA-GEN using constant returns to scale and the parameter set $\alpha = (.20, .20, .20, .20, .20)$. (See Appendix B for the generation procedure.) Also, a "Multi-Normal" data set, with 4 inputs and 3 outputs and the variance-covariance matrix given in Appendix A, was generated using the DRNMVN routine from the IMSL library.

For this test bed, the TDEA procedure used the $CCR_j^i$ model to assign DMUs to layers. Table 3.1 reports, for the first 20 tiers, the number of DMUs assigned to each layer ("$|E^*_{[t]}|$") and their maximum, minimum, and mean *tier 1* efficiency scores ($\theta$). Note that, in each case, that the mean and maximum efficiencies drop with each successive interior layer, as might be expected. However, the TDEA provides insight that DEA cannot provide. As noted by the minimum values, some DMUs possess a low tier 1 efficiency score, yet fall on an outer tier, indicating an efficient use of current

Table 3.1. -- Large data set tiered DEA results

| Tier | Banking | | | | Cobb-Douglas | | | | Multi-Normal | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $t$ | Max | Min | Mean | $\lvert E_{[t]} \rvert$ | Max | Min | Mean | $\lvert E_{[t]} \rvert$ | Max | Min | Mean | $\lvert E_{[t]} \rvert$ |
| 1 | 1.00 | 1.00 | 1.00 | 77 | 1.00 | 1.00 | 1.00 | 70 | 1.00 | 1.00 | 1.00 | 104 |
| 2 | 1.00 | 0.60 | 0.87 | 242 | 0.98 | 0.62 | 0.81 | 91 | 1.00 | 0.58 | 0.93 | 215 |
| 3 | 0.92 | 0.39 | 0.78 | 373 | 0.80 | 0.34 | 0.60 | 109 | 0.96 | 0.64 | 0.86 | 312 |
| 4 | 0.88 | 0.11 | 0.73 | 552 | 0.66 | 0.25 | 0.47 | 114 | 0.95 | 0.51 | 0.82 | 410 |
| 5 | 0.84 | 0.42 | 0.70 | 694 | 0.52 | 0.20 | 0.40 | 106 | 0.92 | 0.61 | 0.80 | 471 |
| 6 | 0.80 | 0.41 | 0.68 | 826 | 0.43 | 0.16 | 0.30 | 95 | 0.88 | 0.52 | 0.77 | 526 |
| 7 | 0.78 | 0.39 | 0.65 | 855 | 0.34 | 0.10 | 0.24 | 118 | 0.86 | 0.53 | 0.74 | 556 |
| 8 | 0.76 | 0.25 | 0.64 | 872 | 0.31 | 0.09 | 0.19 | 133 | 0.84 | 0.52 | 0.73 | 579 |
| 9 | 0.73 | 0.40 | 0.62 | 812 | 0.25 | 0.08 | 0.15 | 150 | 0.82 | 0.49 | 0.71 | 612 |
| 10 | 0.72 | 0.35 | 0.60 | 739 | 0.20 | 0.06 | 0.12 | 198 | 0.80 | 0.43 | 0.69 | 543 |
| 11 | 0.69 | 0.40 | 0.59 | 619 | 0.17 | 0.05 | 0.10 | 233 | 0.78 | 0.48 | 0.68 | 555 |
| 12 | 0.67 | 0.41 | 0.57 | 477 | 0.15 | 0.04 | 0.09 | 285 | 0.76 | 0.45 | 0.66 | 560 |
| 13 | 0.66 | 0.42 | 0.57 | 372 | 0.12 | 0.04 | 0.08 | 323 | 0.75 | 0.45 | 0.64 | 463 |
| 14 | 0.64 | 0.40 | 0.55 | 242 | 0.10 | 0.04 | 0.07 | 363 | 0.73 | 0.40 | 0.63 | 410 |
| 15 | 0.61 | 0.36 | 0.53 | 151 | 0.09 | 0.03 | 0.06 | 388 | 0.72 | 0.37 | 0.61 | 384 |
| 16 | 0.59 | 0.40 | 0.51 | 60 | 0.08 | 0.03 | 0.06 | 395 | 0.69 | 0.40 | 0.60 | 283 |
| 17 | 0.56 | 0.38 | 0.47 | 31 | 0.07 | 0.03 | 0.05 | 424 | 0.67 | 0.44 | 0.58 | 270 |
| 18 | 0.49 | 0.28 | 0.42 | 6 | 0.06 | 0.02 | 0.05 | 391 | 0.64 | 0.42 | 0.56 | 240 |
| 19 | 0.00 | 0.00 | 0.00 | 0 | 0.06 | 0.02 | 0.04 | 361 | 0.63 | 0.39 | 0.55 | 159 |
| 20 | 0.00 | 0.00 | 0.00 | 0 | 0.05 | 0.02 | 0.04 | 367 | 0.61 | 0.19 | 0.52 | 126 |
| $> 20$ | 0.00 | 0.00 | 0.00 | 0 | 0.05 | 0.00 | 0.02 | 3286 | 0.58 | 0.34 | 0.49 | 222 |

technology. Conversely, the maximum values indicate that while other DMUs may seem fairly efficient, from a $\theta$-standpoint, they fall on an inner tier because of less productive uses of current technology. It is precisely this behavior that should prove useful in identifying a DMU's true level of efficiency.

Of interest also are the differences between these three problems. The frequency counts reveal that the banking data has only 18 tiers, while the Cobb-Douglas data still has 3,286 of its 8,000 points still un-stratified after 20 tiers. The multi-normal problem has more populous tiers than the Cobb-Douglas, but less than most of the bank's. The banking and multi-normal data sets have tier-1-efficient DMUs on interior layers, but Cobb-Douglas does not. The banking and Cobb-Douglas data have much larger mean-efficiency drops between tiers 1 and 2, relative to the multi-normal. Each problem seems to have a very different structure from the others, as uncovered by the TDEA process.

### 3.2.3. Tiered DEA and the BCC Model

The examples of the figures above illustrate the result of applying the tiering procedure to the $BCC_j^i$ and $BCC_j^o$ models. These models, used to identify the Pareto-Koopmans efficient production surface, with no restrictions on returns to scale, were first proposed by Banker, Charnes, and Cooper [11]. The $BCC_j^i$ model can be written as:

$$
(BCC_j^i) \qquad
\begin{aligned}
&\min\ \theta - \varepsilon(\mathbf{1}\boldsymbol{s}^i + \mathbf{1}\boldsymbol{s}^o) && (3.1)\\
&\text{s.t.}\quad \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = \mathbf{Y}_j && (3.2)\\
&\theta\mathbf{X}_j - \mathbf{X}\boldsymbol{\lambda} - \boldsymbol{s}^i = \mathbf{0} && (3.3)\\
&\mathbf{1}\boldsymbol{\lambda} = 1 && (3.4)\\
&\boldsymbol{\lambda}, \boldsymbol{s}^i, \boldsymbol{s}^o \geq \mathbf{0} && (3.5)\\
&\theta \text{ free} && (3.6)
\end{aligned}
$$

In Figure 3.2 and Figure 3.3, the outer layers designated by $\text{DMU}_A$, $\text{DMU}_B$, and $\text{DMU}_C$ represent the efficient set which demonstrate best practice in the production process. However, this empirical production surface is dependent on the data observed.

Suppose the original set had not included B. The models would have identified a different, yet legitimate, production surface revealing the current technology consisting of DMUs A, E, F, and C. If data on B later became available and was added to the set of observations, it would markedly alter the shape of the efficient production surface. B now reveals a new production process with a corresponding level of technological achievement that was previously unseen. It is precisely this realization that motivates the tiered ranking procedure.

In DEA, all of the efficient DMUs share a common characteristic: they all demonstrate an ability to make the best use of the current technology to conduct their production process (i.e., they demonstrate best practice behavior). Once these DMUs have been identified, they can be temporarily removed, and the remaining DMUs form a new, valid DEA data set. When the $BCC_j^i$ model is applied to this new data set a new efficient production surface is revealed. Had the data for DMUs of the outer tier not been available originally, this new production surface would legitimately characterize the best technological achievement level observed. Consequently, the DMUs comprising this new efficient surface share a common level of success of utilizing the currently revealed technology. Repeating this process groups the DMUs according to common achievement levels. DMUs on outer tiers reveal a technological advance not realized by DMUs on inner tiers.

Of primary significance is that the tiering procedure provides greater discriminatory power in determining managerial efficiency than previous DEA measures. To appreciate the importance of the new measure, an understanding of what causes a DMU to be inefficient is helpful. By stratifying across different tiers, the new measure provides a more complete description of the DMU's managerial efficiency relative to its contemporaries. One reason a DMU may be characterized as relatively inefficient in DEA is that it is dominated by a few highly efficient DMUs. A high concentration of other DMUs, with similar, but superior, management practices can cause the inferior DMUs to attain a low tier assignment. Figure 3.4 depicts a set of DMUs with one output and one input. The tiering procedure produces three layers. In this case, J is enveloped by a concentration of DMUs using nearly the same level of inputs to produce a similar level of outputs. However, J consistently under-achieves compared to the other DMUs, consequently, it would receive a relatively low ranking. Even though the original DEA scores for DMUs F, G, and J are similar, TDEA further discriminates J as less productive given the available technology than DMUs F and G.

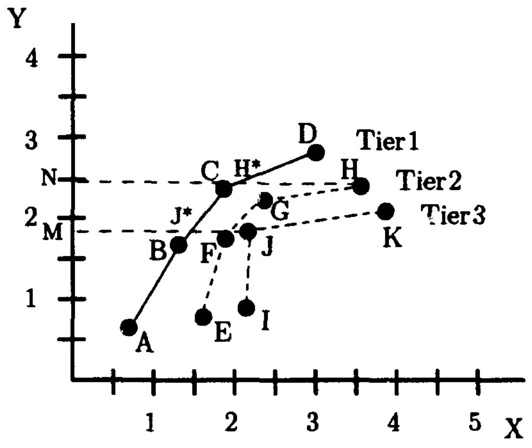

Figure 3.4. Example 1 tiered results – one input, one output

A second reason an inefficient DMU may result in a low efficiency score is that it operates in a production process "away from the crowd." These maverick DMUs may be leaders in the introduction of new production technologies or management methods into the market place. The transition to these new procedures is penalized under traditional DEA analysis because it appears that the DMUs introducing the new inputs consume more than the peers. The dominant DMU of this shift may be efficient, but the other DMUs can appear to be very inefficient. This situation can be seen with H. Notice, in this case H has a technical efficiency score seen as the ratio $\frac{NH^*}{NH}$. J has a higher technical efficiency score seen as the ratio $\frac{MJ^*}{MJ}$ but is ranked lower than H in terms of tier assignment. H may represent a risk-taker that shows a short-term reduction in its DEA efficiency score, to take advantage of new technology, by introducing new inputs. However, even with a short term loss of efficiency, the DMU could see a rise in its rank ordering by moving to a higher tier level. Once the new technology is fully integrated into the production process, the DMU may witness substantial increases in its DEA efficiency score. Traditional DEA analysis would penalize the management decision, in the short run, to introduce the new technology into the production process. Consequently, the decision to incur possible short term losses to achieve long term gains would appear unfavorable in a traditional DEA analysis. However, the stratified ranking of DMUs reveals the success of this management decision.

The above scenario can occur often in a free competitive market. Under dynamic conditions, the firms in the competitive market must adapt to maintain market share. A DMU that adopts a management style and production process similar to other DMUs, but consistently under-performs, may result in a seemingly high relative efficiency score, but with a low rank ordering. The inability of this competitive firm to find its niche and distinguish itself from the "competition" may result in its failure in the market place. Current DEA methodologies are inadequate to reveal such conditions; TDEA offers an attractive means to discriminate between these managerial behaviors.

DMUs with low efficiency scores but with relatively high rank order are not restricted to DMUs at "fringe" production levels. Data outliers can strongly affect the shape of the efficient surface. Figure 3.5 shows the effect that outlier B has on the production surface. The outlier significantly distorts the surface making F seem relatively inefficient. Had B not been in the data set, F would be efficient. In spite of F's low efficiency score, TDEA ranks F relatively high. Consequently, the performance of F may not be as poor as indicated by the DEA score.

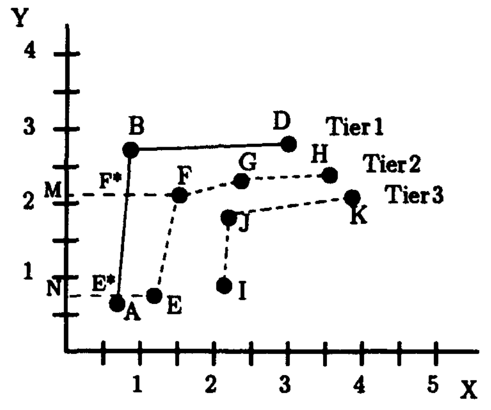

Figure 3.5. Example 2 tiered results – one input, one output

A key advantage of stratifying DMUs into tiers is that it allows the DEA methodology to more closely describe true managerial efficiency that may be masked by traditional DEA analysis. As a result, managers of inefficient DMUs have increased flexibility in improving production operations. The manager's long-term goal may be to achieve efficiency. The DMU can strive to accomplish this by improving the short run rank ordering without a myopic focus on its DEA efficiency score.

## 3.2.4. Tiered DEA and the CCR Model

The ranking procedure can also be applied to the CCR models. For these models, which assume constant returns to scale, the efficient DMUs not only are operating most efficiently (with the greatest level of technological achievement) but they are also operating at the most productive scale size (MPSS). For the single input and single output model, the MPSS is determined by the DMUs yielding the highest ratio of quantity output to quantity input. In economic terms, this equates to the DMUs yielding the highest average product. Banker [10] demonstrated that the CCR model revealed the efficient DMUs operating at the MPSS in multi-dimensional processes.

The MPSS units which form the frontier for the CCR models are dependent on the observed set of data. In a traditional DEA analysis, the CCR scores for the inefficient DMUs reflect both technical and scale inefficiencies. To separate the technical from the scale inefficiency, the BCC model must also be run. The tiered procedure, when applied to the CCR model, presents a different picture of scale inefficiency. As each outer tier is removed from the data set, the set of DMUs which composes the most productive scale size changes. As a result, the values of the scale inefficiencies for the other DMUs also change. Therefore traditional DEA measures may overstate the amount of scale inefficiency that a DMU demonstrates.

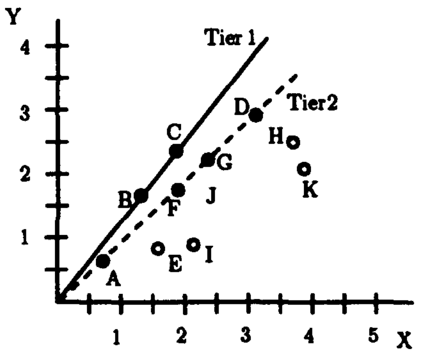

Figure 3.6. Example CCR tiered results – one input, one output

Figure 3.6 shows two tiers of scale-efficient DMUs for the same set of data used in Figure 3.4, where DMUs A, B, C, and D were shown to be technically efficient. However, Fig 3.6 indicates B and C are scale efficient while A and D are both scale inefficient. Because D falls far from the most productive scale boundary, the $CCR_j^i$ DEA measure would indicate it is more scale inefficient than A. Yet, with the tiering procedure, both become scale efficient at a common level. Consequently, the two DMUs may not be as different, in terms of scale, as first suggested by the DEA scores.

## 3.3. Ranking Within Tiers

The TDEA approach extends traditional DEA to allow for a stratification of DMUs reflecting different productive efficiency layers. However, a ranking system within each layer is still needed. This section describes a new procedure that can provide a useful ranking.

Each tier defines a set of DMUs that forms a production surface (layer). To determine the relative rank ordering among the DMUs at each tier, one could measure the contribution of the DMU to the shape of its layer. DMUs that significantly distort the production surface layer on which they are assigned play a more prominent role in determining the shape of the production surface. Consequently, if these distortions can be measured, the DMUs could be ranked by the degree of distortion they contribute to the production layer.

The following model describes an *extremal DEA* (EDEA) approach for the $CCR_j^i$ model. However, the formulation also applies to the output-oriented models and to either constant- or variable-returns-to-scale formulations. With this methodology, the DEA formulation is modified slightly to measure how far a DMU is from the resulting production surface when the DMU itself is removed from the data set of interest. The EDEA procedure can be described as follows. Let $\mathbf{X}$ be an $(m \times n)$ matrix and $\mathbf{Y}$ be an $(s \times n)$ matrix of the observed input and output values, respectively, of all the DMUs of interest. Select a DMU to be observed, in this case DMU$_o$. Let $\mathbf{X}^{[o]}$ and $\mathbf{Y}^{[o]}$ be $\mathbf{X}$ and $\mathbf{Y}$, respectively, with the observations of DMU$_o$ removed. Then the extremal DEA measure $\hat{\theta}$ is computed by the following model.

(EDEA Model)

$$\min\ \hat{\theta} \tag{3.7}$$

$$\text{s.t.}\quad \mathbf{Y}^{[o]}\boldsymbol{\lambda} - \boldsymbol{s}^{o} = Y_o \tag{3.8}$$

$$\hat{\theta}X_o - \mathbf{X}^{[o]}\boldsymbol{\lambda} - \boldsymbol{s}^{i} = \mathbf{0} \tag{3.9}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^{i}, \boldsymbol{s}^{o} \geq \mathbf{0} \tag{3.10}$$

$$\hat{\theta}\ \text{free} \tag{3.11}$$

In contrast to a traditional DEA analysis, $\hat{\theta}$ can now be greater than one. For $\hat{\theta} \geq 1.0$ the DMU will be efficient and $\hat{\theta}$ measures the allowable proportional increase in inputs for the DMU to remain efficient. If $\hat{\theta} = 1$, the corresponding DMU may be weakly efficient from a DEA standpoint. However, for the inefficient DMUs, the EDEA scores will be identical to traditional DEA scores, (e.g., $\hat{\theta} = \theta$), since the inefficient DMU itself will never be a member of an optimal basis (this can be shown in the same manner as Theorem 1 of Chapter II). Consequently, EDEA has an advantage over traditional DEA models in that it provides greater meaning to the scores for the efficient DMUs by allowing additional variability in the efficient score values.

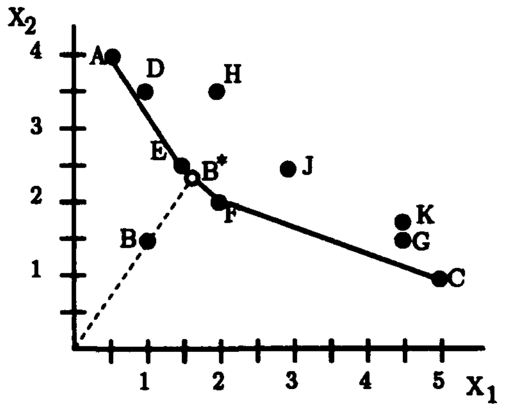

Figure 3.7. Extremal DEA example

Figure 3.7 illustrates the EDEA procedure for B when applied to the data in Table 3.2. Here, B is projected onto the "new" efficient surface using EDEA. The resultant objective value, $\hat{\theta}^* = \frac{0B^*}{0B}$ measures how "far" B is from the efficient surface. Viewing the problem from a different perspective, $\hat{\theta}^*$ reflects the degree to which B would contribute to the shape of the new efficiency surface if it was added to the data set.

DMUs that cause significant and important distortions of the efficiency surface will result in a high EDEA objective value. Those DMUs that have little influence on the shape of the production surface will have objective values close to 1. Consequently, the DMUs can be ranked by order of influence at each tier level based on the EDEA scores.

An overall ranking of all DMUs can be achieved by: (1) using TDEA to to stratify all DMUs into tiers, (2) applying EDEA to each tier level, (3) ranking each tier's DMUs by $\hat{\theta}$, (4) then finding each unit's overall rank by ranking those DMUs on outer tiers as more important than those on inner tiers. The following section presents an illustrative example of this procedure and provides some numerical interpretations.

## 3.4. Example of Tiered DEA

The illustration shown in Figure 3.2 is based on the data in Table 3.2. As illustrated in Figure 3.2, the TDEA approach yields three production surfaces. Those DMUs on tier 1 are the same efficient units found in a standard envelopment analysis. The DMUs of succeeding inner tiers compose less efficient production surface layers. Notice though, the inefficient DMUs can now be further distinguished by the production surface on which they fall. DMUs on outer tier levels represent more successful production processes. Consequently, the DMUs on outer tiers can be ranked more efficient than DMUs on inner tiers. As will be demonstrated, this does not necessarily coincide with

Table 3.2. – Tiered DEA example data

| | DMU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | H | J | K |
| $y$ | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| $x_1$ | 0.50 | 1.00 | 5.00 | 1.00 | 1.50 | 2.00 | 4.50 | 2.00 | 3.00 | 4.50 |
| $x_2$ | 4.00 | 1.50 | 1.00 | 3.50 | 2.50 | 2.00 | 1.50 | 3.50 | 2.50 | 1.80 |

Table 3.3. – Results of the tiered rank ordering procedure

| | DMU | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | H | J | K |
| DEA | 1.00 | 1.00 | 1.00 | .765 | .650 | .722 | .788 | .482 | .565 | .688 |
| TDEA | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 |
| TEDEA1 | 2.00 | 1.60 | 1.50 | .765 | .650 | .722 | .788 | .482 | .565 | .688 |
| TEDEA2 | — | — | — | 1.50 | 1.05 | 1.13 | 1.33 | .733 | .774 | .889 |
| TEDEA3 | — | — | — | — | — | — | — | 1.50 | 1.13 | 1.39 |
| RANK1 | 1 | 2 | 3 | — | — | — | — | — | — | — |
| RANK2 | — | — | — | 1 | 4 | 3 | 2 | — | — | — |
| RANK3 | — | — | — | — | — | — | — | 1 | 3 | 2 |
| RANK | 1 | 2 | 3 | 4 | 7 | 6 | 5 | 8 | 10 | 9 |
| DEARANK | 1 | 1 | 1 | 5 | 8 | 6 | 4 | 10 | 9 | 7 |

the traditional DEA score. The results of the TDEA coupled with the EDEA are shown in Table 3.3 and labeled as TEDEA1 through TEDEA3.

The DEA row lists the $CCR^i$ $\theta$ scores for each unit in the data set. The TDEA row indicates the tier level to which each DMU is assigned as a result of the TDEA procedure. TEDEA$t$ rows give the EDEA $\hat{\theta}$ values for all DMUs on tier $t$ or higher. DMUs in tier $t$ are ranked within tier in the RANK$t$ rows. The overall rank for the entire set of DMUs is given in the RANK row. This can be compared to the ranking the DMUs would have been given, listed in DEARANK, had they been ordered by the $CCR^i$ $\theta$ value.

Some important observations can be made concerning these results. Notice, the DEA and the TEDEA1 results are identical for the inefficient DMUs. Likewise, the efficient DMUs are appropriately identified with values greater or equal to 1. Notice that the higher inefficiency scores do not necessarily indicate on what tier level a DMU may fall. For example, K has a higher $\theta$ than E but falls on a lower tier level. By observing the results at TEDEA3 one notices that the ranking of H, J, and K does not correspond with the ranking the DMUs would receive if the DEA efficiency scores were used. In fact, H has the lowest DEA efficiency score, but the highest rank of tier 3. Figure 3.8 illustrates why this is so. DMU J does not significantly distort the efficiency surface that exists when it is not present. This is not true for H which significantly distorts the shape of the production surface; DMU H is more influential than DMU J and thus is ranked higher.

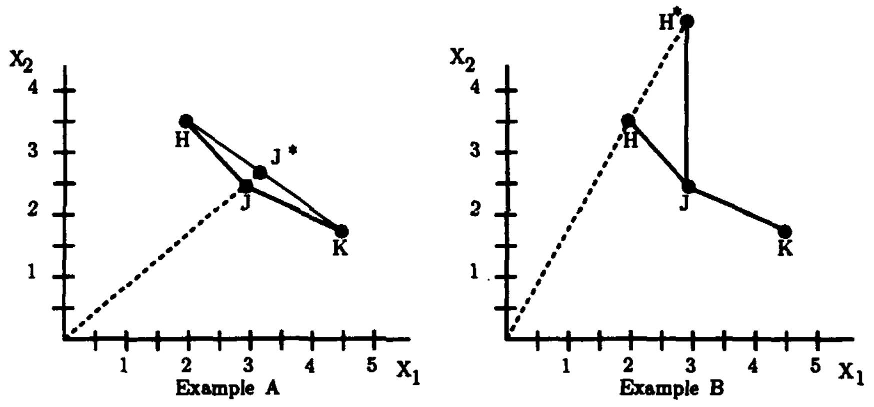

Figure 3.8. TEDEA examples

A major advantage of TEDEA, besides ranking each DMU by its influence, is that it can help paint a numerical picture of the environment in which a DMU operates. As each tier level and associated production surface is removed, the new DEA scores for the remaining DMUs can be calculated. In this way, the *migration* of the DEA scores for a particular DMU can be traced through a series of tiers. A rapid rise in the scores may indicate that the DMU is in a region with a relative low density of other DMUs but is dominated by a few highly efficient ones. A slow rise in the score may indicate that the DMU is surrounded by a larger density of other DMUs which may have similar management styles or environments but are operating more efficiently. This information can assist the analyst and managers in determining appropriate courses of actions to improve either the rank ordering or the efficiency score.

The rank ordering procedure may also prove useful in window analysis. If DMU behavior is tracked over time, the changes in rank ordering should reflect the relative effectiveness of on-going managerial decision-making. These managerial changes may remain hidden from traditional DEA analysis unless the changing practices result in a change in efficient or inefficient status of the DMU. Consequently, the rank ordering methodology may provide prompt managerial feedback as to how a DMU compares with the competition as a result of implemented changes.

## 3.5. Computational Considerations

It is important to note, unlike CCR or BCC models, with the EDEA method the basic elements in the optimal solution need not represent efficient DMUs. Figure 3.7 depicts such a situation. When B is removed from the data set, the resulting efficient surface includes E and F, both of which are inefficient according to a traditional DEA. Therefore, the advantage of computational efficient techniques for DEA, such as restricted basis entry and early identification of efficient DMUs, cannot be maintained for the EDEA models. However, other computationally efficient procedures are possible for the EDEA model. Computational efficiency can be maintained by combining TDEA and EDEA into a single formulation, TEDEA$^*$, based on the following observation.

*When a DMU is removed from tier $t$'s data set, the resulting production surface will consist of DMUs belonging to either tier $t$ or $t+1$.*

In this way, the stratification of DMUs proves to be a valuable computational tool for the EDEA method. Since the number of DMUs at *tier* $t$ and $t+1$ is typically small compared to the entire set, the linear programming problems remain small when determining the EDEA scores. As a result, TDEA and EDEA can be combined to form a computationally efficient rank ordering method. Let $\mathbf{X}_{t\bullet}$ and $\mathbf{Y}_{t\bullet}$ be the matrices of input and output vectors of DMUs belonging to *tier* $t$ or $t+1$. Choose a DMU from tier $t$ for analysis, let this be DMU$_o$. The TEDEA* model can be written as:

$$\text{(TEDEA*)} \qquad \min\ \hat{\theta} \tag{3.12}$$

$$\text{s.t.}\quad \mathbf{Y}_{t\bullet}^{[o]}\boldsymbol{\lambda} - \boldsymbol{s}^{o} = Y_o \tag{3.13}$$

$$\hat{\theta}X_o - \mathbf{X}_{t\bullet}^{[o]}\boldsymbol{\lambda} - \boldsymbol{s}^{i} = \mathbf{0} \tag{3.14}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^{i}, \boldsymbol{s}^{o} \geq \mathbf{0} \tag{3.15}$$

$$\hat{\theta}\ \text{free} \tag{3.16}$$

Because the TEDEA* problem consists of LP's much smaller than EDEA, computational efficiency is maintained.

## 3.6. Additional Benefits of Ranking

As mentioned previously, the empirical production surface of the DEA model is greatly influenced by possible outliers in the data. In 1971, Timmer [86] assumed a Cobb-Douglas form to develop a probabilistic frontier production function for observed data. Using the Cobb-Douglas form, Timmer translated the problem into a linear programming model with a striking similarity to the DEA model. Since he assumed a single output, multi-input case, he was able to compare the frontier analysis with the traditional econometric model. He showed that by eliminating the top 2% of observations which appeared to be outliers, the linear programming frontier model yielded results that could be supported by the econometric models.

In DEA, no functional form is assumed. Consequently, potential outliers have been difficult to identify. In this case, the efficient DMUs may not represent normal production activities and therefore may not serve as a legitimate basis of comparison.

Consequently, DEA efficiency scores of outliers could distort the true picture of managerial efficiency of the inefficient DMUs. As a result, identification of possible outlier or extremely dominant DMUs is essential. With TEDEA*, the outliers will be located on outer tiers. If possible outliers are revealed, the problematic DMUs can be removed from the analysis. By following this procedure, a better estimation of managerial efficiency can be achieved.

## 3.7. Conclusions

This chapter has outlined a rank ordering of DMUs based on the influence they serve in forming empirical production layers for the data set. The intent is not to present the only valid system or rank ordering, but to stimulate thought as to how DEA methods may be modified to make the results more meaningful to practitioners. One concern with the rank ordering approach is the large computational requirement necessary to achieve the results. On some layers the density of efficient DMUs will be high and on other layers the density may be low. The code must be computationally efficient across this wide variety of conditions. In addition, the code must be flexible to allow for constant or variable returns to scale in either the input or output models. Flexibility to switch between models at different layers may also be desirable. The TEDEA* approach meets the requirements to effectively and efficiently perform such tasks.

The rank order procedure presented in this discussion classifies DMUs by their ability to use current technology to efficiently conduct the production process. This classification can serve as a proxy measure of management efficiency that provides more detailed information than offered by standard DEA models. By identifying the best and worst performing DMUs, analysts can pinpoint which DMUs should serve as a basis of comparison from which to determine which management practices enhance and which deter from productive efficiency. In addition, the rank ordering methodology presented here reveals the need to further characterize the computational issues of DEA and provide efficient codes to accomplish these new tasks.

# CHAPTER IV

# DEA SENSITIVITY ANALYSIS

## 4.1. Motivation

In 1978, Charnes, Cooper, and Rhodes (CCR) [37] developed a technique, which they named data envelopment analysis (DEA), to determine the relative efficiency of a production unit (commonly known as a decision making unit, or DMU) compared to other units using the same kinds of inputs to produce the same kinds of outputs. There are three popular linear programming formulations which indicate whether a DMU is relatively efficient or inefficient. The CCR model [37] assesses whether the DMU is efficient at the most-productive scale-size (MPSS). This model is perhaps the most widely used in DEA applications. The Banker, Charnes, and Cooper (BCC) model [11] and the additive model [33] determine whether the DMU is Pareto-Koopmans efficient. In economic contexts, the BCC and additive models find the entire efficient empirical production surface while the CCR model finds a proper subset of that surface which is composed of efficient units at the MPSS. For a single-output, single-input DMU set, the MPSS is the size that yields the greatest average product (i.e, the greatest ratio of output to input).

Within the context of one of these models, a DMU's *status* is either efficient or not efficient. To determine this status, a separate linear program must be solved, either directly of implicitly. Since the DEA methodology is empirically based, values of the inputs and outputs must be measured or estimated at a given point in time for each DMU. A concern then is the accuracy of the measures or estimates, the comparability of the decision units (particularly on a temporal basis), and any uncertainty associated with the observed values. Any of these factors can affect the validity of an efficiency analysis and influence the status of all of the decision units under consideration.

Sensitivity analysis is an important element of any DEA study because it assesses the robustness of each DMU's status when variations in the inputs and outputs are possible. This is more than a simple extension of traditional LP sensitivity theory because of the nature of the DEA models.

Traditional sensitivity analysis in linear programming concentrates on finding a new optimal solution value to a problem, as updated estimates of some of the sample data become available, without the expense of resolving the problem from scratch. This may take the form of determining ranges within which data may be varied without requiring a change in the set of vectors composing the optimal basis. For the DEA model formulation, this traditional analysis is inappropriate because a change in the values of the inputs and outputs of a particular DMU simultaneously affects structural and right-hand-side coefficients for all LPs in the analysis. Additionally, the assumptions of non-degeneracy of the optimal basis and uniqueness of the optimal solutions are violated for the DEA problems. Also, in DEA, the primary concern is the effect that perturbations may have on the status of a DMU, not simply on the solution value of the underlying linear programming problem. Consequently, new approaches for sensitivity analysis must be developed when considering DEA models.

To date, a number of studies address sensitivity analysis in DEA [33,36,43,44], most with respect to the additive model. Charnes, Cooper, Lewin, Morey, and Rousseau [36] present a limited study of perturbations to a single output or single input using the CCR model. This study was extended by Charnes, Haag, Jaska, and Semple (CHJS) [43] to identify regions of stability using the additive model, in which allowable perturbations cause the DMU to retain its status. Because the additive model finds the Pareto-Koopmans efficient frontier without respect to an input or output orientation, the CHJS example is restricted to simultaneous perturbations of both the inputs and outputs of each DMU in the model.

This chapter presents a new methodology that allows for sensitivity analysis in the case of proportional changes to either the outputs alone or the inputs alone for the BCC and CCR models. The approach will identify a range of values such that all allowable perturbations for the DMU within the range, ceteris paribus, will preserve the DMU's current status. Since the analysis can be conducted in conjunction with determining the DEA efficiency score, minimal additional computational costs are required. Also presented is an update to the CHJS approach that overcomes inherent problems with their additive model approaches.

The remainder of this chapter is divided into four sections. Section 4.2 develops the framework for conducting sensitivity analysis on DEA problems. Section 4.3 introduces the methodology to conduct sensitivity analysis on each DMU for the BCC and CCR models. Although the sensitivity analysis is based on proportional changes in either outputs or inputs, nonproportional changes will also be investigated. Section 4.4 compares the methodology presented in Section 4.3 to the CHJS sensitivity analysis approach. A problematic concern of the CHJS method is presented and a demonstration of how the new methodology compliments the CHJS to meet that concern. Section 4.5 introduces an expanded sensitivity analysis approach based on the models introduced in Section 4.3. Although the expanded sensitivity analysis may be computationally intensive, it can reveal information to guide the management in determining what future actions should be taken to maintain or establish efficiency. Conclusions are given in Section 4.6.

## 4.2. BCC and CCR DEA Sensitivity Analysis

A new method for sensitivity analysis in DEA can be constructed from simple modifications to the CCR and BCC DEA formulations. In this approach, the sensitivity of a DMU's status is measured in terms of either proportional changes in outputs or inputs. The concerns in DEA sensitivity analysis differ dramatically between DMUs which are efficient and those that are inefficient. For efficient DMUs, an increase of any output cannot result in a lower efficiency rating; likewise, a decrease in any input cannot worsen the DEA efficiency score. Therefore, with efficient DMUs, primary attention is given to determining what reductions in outputs are allowable, or what increases in inputs are permissible in order to insure the DMU retains its efficient status. For efficient DMUs one is concerned with identifying how bad the production process can get before the DMU loses its efficient status.

This motivation fundamentally differs for the inefficient DMUs, for which the primary concern is the identification of changes that will improve the efficiency rating. Generally, one would like to identify those areas that require the smallest improvements which would cause a given DMU to become relatively efficient. For inefficient DMUs this would consist of either increasing outputs or decreasing inputs. We now present an elegant formulation for analyzing both efficient and inefficient DMUs in the CCR and BCC models.

### 4.2.1. CCR and BCC Sensitivity: A Background

For the sensitivity analysis, let DMU$_k$ be the DMU of interest. In this model, we define D $= \{1, \ldots, n\}$, the index set of DMUs, with $\mathbf{X} = [X_1, \ldots, X_n]$ and $\mathbf{Y} = [Y_1, \ldots, Y_n]$. Vectors $(X_j, Y_j)$ denote the observed data where $X_j = (x_{1j}, \ldots, x_{ij}, \ldots, x_{mj})$ is a column vector of observed input values with $x_{ij} \geq 0$ (at least one positive) and $Y_j = (y_{1j}, \ldots, y_{rj}, \ldots, y_{sj})$ is a column vector of observed outputs with $y_{rj} \geq 0$ (at least one positive). Let $\mathbf{X}^{[k]}$ be the $\mathbf{X}$ matrix with the $X_k$ column removed. Similarly define $\mathbf{Y}^{[k]}$ to be $\mathbf{Y}$ with the $Y_k$ column removed. A new approach called *extremal DEA* (EDEA) will form the basis for the DEA sensitivity analysis. The approach allows for both an input and output oriented analysis and can be used within the CCR and BCC modeling contexts. The CCR-based EDEA models can be formulated as follows, with $ECCR^i$ and $ECCR^o$ having input and output orientations, respectively.

$(ECCR_k^i)$

$$\min\ w = \hat{\theta} \tag{4.1}$$

$$\text{s.t.} \quad \mathbf{Y}^{[k]}\boldsymbol{\lambda} - \mathbf{s}^o = \mathbf{Y}_k \tag{4.2}$$

$$\hat{\theta}\mathbf{X}_k - \mathbf{X}^{[k]}\boldsymbol{\lambda} - \mathbf{s}^i = \mathbf{0} \tag{4.3}$$

$$\boldsymbol{\lambda}, \mathbf{s}^i, \mathbf{s}^o \geq \mathbf{0} \tag{4.4}$$

$$\hat{\theta} \text{ free} \tag{4.5}$$

$(ECCR_k^o)$

$$\min\ w = \hat{\phi} \tag{4.6}$$

$$\text{s.t.} \quad \mathbf{X}^{[k]}\boldsymbol{\lambda} + \mathbf{s}^i = \mathbf{X}_k \tag{4.7}$$

$$\hat{\phi}\mathbf{Y}_k - \mathbf{Y}^{[k]}\boldsymbol{\lambda} + \mathbf{s}^o = \mathbf{0} \tag{4.8}$$

$$\boldsymbol{\lambda}, \mathbf{s}^i, \mathbf{s}^o \geq \mathbf{0} \tag{4.9}$$

$$\hat{\phi} \text{ free} \tag{4.10}$$

The BCC versions of the above EDEA models — $EBCC^i$ and $EBCC^o$, respectively — can be obtained by appending the constraint: $\mathbf{1}\boldsymbol{\lambda} = 1$.

In the discussion that follows, the term *DEA models* refers to the original CCR, BCC, and additive models. In standard DEA form, the index set of frontier-efficient DMUs of the BCC and CCR models which are Pareto-Koopmans-efficient is designated as $\overline{\mathrm{E}}^* = \{j \in \mathrm{D} | \theta^* = 1 \text{ (or } \phi^* = 1), \mathbf{s}^{o*} = \mathbf{0}, \text{ and } \mathbf{s}^{i*} = \mathbf{0}\}$. The set F is the index set of frontier-efficient but not Pareto-Koopmans efficient DMUs, where $\mathrm{F} = \{j \in D | \theta^* = 1$ or $(\phi^* = 1)$ and $\mathbf{s}^{o*} \neq \mathbf{0}$ or $\mathbf{s}^{i*} \neq \mathbf{0}$. For a DMU to be Pareto-Koopmans efficient, it must not be possible to decrease any input value without also decreasing at least one output or increasing at least one other input value. In addition, to be Pareto-Koopmans-efficient, it must not be possible to increase any output value of $\mathrm{DMU}_k$ without also increasing at least one input value or decreasing at least one other output. All DMUs on the frontier, but which are not Pareto-Koopmans efficient, i.e., the members of F, are labeled *weakly efficient.* Inefficient DMUs have $\theta^* < 1$ for the input-oriented model and $\phi^* > 1$ for the output-oriented model. With EDEA, $\hat{\theta}^*$ and $\hat{\phi}^*$ values for inefficient DMUs are identical to their corresponding variables, $\theta^*$ and $\phi^*$, in the DEA models. However, frontier efficient DMUs in EDEA are characterized by $\hat{\theta}^* \geq 1$ or $\hat{\phi}^* \leq 1$. To understand this, a discussion of the DEA formulations is in order.

The description of the DEA formulations gives insight into the meaning of $\theta^*$ ($\phi^*$). The DEA models identify the set of efficient DMUs, $\overline{\mathrm{E}}^*$, as those forming an empirical production surface which envelops all DMUs in D, the set under investigation, [11,31,33,39,41]. For the CCR, BCC, and additive models, all DMUs in D are evaluated against the DMUs which are members of $\overline{\mathrm{E}}^*$. For the ECCR and EBCC models, the problem reference set is denoted as $\hat{\mathrm{D}} = \mathrm{D} - k$, where $\mathrm{DMU}_k$ is the unit under evaluation. If $k \notin \overline{\mathrm{E}}^* \cup \mathrm{F}$, then $\mathrm{DMU}_k$ is inefficient and $\hat{\theta}^* = \theta^*$ or $\hat{\phi}^* = \phi^*$. However, if $k \in \overline{\mathrm{E}}^* \cup \mathrm{F}$, and since $k \notin \hat{\mathrm{D}}$, then $\hat{\theta}^* \geq 1$ or $\hat{\phi}^* \leq 1$. This results because $\mathrm{DMU}_k$ will compare to the remaining frontier formed by the efficient DMUs of $\hat{\mathrm{D}}$. In this case $\hat{\theta}^*$ can be interpreted in terms of the Farrell measure [51]; $\mathrm{DMU}_k$ may increase its inputs by a factor of $\hat{\theta}^*$ and remain frontier efficient. Correspondingly, for the output-oriented model, $\mathrm{DMU}_k$ may decrease its outputs by a factor of $\hat{\phi}^*$ and remain frontier efficient.

Charnes, Cooper, and Thrall place the scale-efficient DMUs for the CCR model into three categories which they label E, E′, and F [41]. For the input-oriented DEA model, $\mathrm{DMU}_k \in \mathrm{E} \cup \mathrm{E}' \cup \mathrm{F}$ if and only if $\theta^* = 1$ (i.e., $\mathrm{DMU}_k$ is frontier efficient). Mathematically, the members of E are extreme rays (CCR model) or extreme points

(BCC model) which, when taken with their associated facets, form a piecewise-empirical production surface, or Pareto-Koopmans frontier [33]. For the *ECCR* and *EBCC* models, only members of E will have $\hat{\theta}^* > 1$. Members of E$'$ are non-extreme rays (or non-extreme points) but depict DMUs which fall on the frontier and are Pareto-Koopmans efficient. For the *ECCR* and *EBCC* models, only members of E$'$ will have $\hat{\theta}^* = 1$, $\boldsymbol{s}^i = \mathbf{0}$, and $\boldsymbol{s}^o = \mathbf{0}$ with an efficient reference set consisting of DMUs which are members of $\overline{\mathrm{E}}^*$. DMUs which are elements of F represent DMUs on the frontier but which are not Pareto-Koopmans efficient. For the *ECCR* and *EBCC* models only members of F will have $\hat{\theta}^* = 1$ and $\boldsymbol{s}^i \neq \mathbf{0}$ or $\boldsymbol{s}^o \neq \mathbf{0}$ with an efficient reference set consisting of DMUs which are members of $\overline{\mathrm{E}}^*$. It is important to note that if DMU$_k$ is a member of F with an efficient reference set containing other members of F then, for the *ECCR* and *EBCC* models, the optimal solution for DMU$_k$ may result in $\hat{\theta}^* = 1$, $\boldsymbol{s}^i = \mathbf{0}$, and $\boldsymbol{s}^o = \mathbf{0}$. In this case, members of E$'$ and F can be distinguished if the efficient reference set is retained in the solution of *ECCR* and *EBCC*, for once the solution for all DMUs in D are found, all members of F, E, and $\overline{\mathrm{E}}^*$ can be identified. Therefore, the two-phase approach suggested by Ali [3] to identify members of F is not necessary.

Some important observations can be made concerning the above classification scheme. With input-oriented DEA analysis, the transformation

$$X_k^* \leftarrow \theta^* X_k - \boldsymbol{s}^{i*}, \quad Y_k^* \leftarrow Y_k + \boldsymbol{s}^{o*} \tag{4.11}$$

projects DMU$_k$ onto an associated Pareto-Koopmans frontier-efficient facet. Correspondingly, the transformation

$$X_k^* \leftarrow X_k - \boldsymbol{s}^{i*}, \quad Y_k^* \leftarrow \phi^* Y_k + \boldsymbol{s}^{o*} \tag{4.12}$$

projects $\text{DMU}_k$ onto an associated Pareto-Koopmans frontier-efficient facet for the output-oriented models [33]. In either case, the projected $\text{DMU}_k^*$ consisting of $(Y_k^*, X_k^*)$ is called the *virtual representation* of $\text{DMU}_k$ [37].

The virtual representation $\text{DMU}_k^*$ at $(Y_k^*, X_k^*)$ determines an efficient projection that is a member of $\text{E} \cup \text{E}'$, i.e., Pareto-Koopmans efficient. However, other transformations are possible that provide important insights. The transformation

$$X_k^* \leftarrow \hat{\theta}^* X_k, \quad Y_k^* \leftarrow Y_k \tag{4.13}$$

will produce an efficient projection for the $ECCR^i$ and $EBCC^i$ input-oriented models, which will be a member of $\text{E} \cup \text{E}' \cup \text{F}$ (i.e., be frontier efficient). Likewise, for the output-oriented models $ECCR^o$ and $EBCC^o$, the transformation

$$X_k^* \leftarrow X_k, \quad Y_k^* \leftarrow \hat{\phi}^* Y_k \tag{4.14}$$

will also produce an efficient projection, $\text{DMU}_k^*$, which is a member of $\text{E} \cup \text{E}' \cup \text{F}$. Consequently, the primary purpose of the slacks in DEA models is to insure that the virtual transformation $\text{DMU}_k^*$ is Pareto-Koopmans efficient and not simply a DMU on the efficient frontier.

However, the restriction to limit projections to Pareto-Koopmans-efficient points is not necessary when determining the allowable perturbations that will cause a DMU to change status, for any weakly efficient DMU that has all inputs or all outputs perturbed by a small amount will become either Pareto-Koopmans efficient or completely inefficient. This has important implications for the EDEA models and corresponding sensitivity analysis. For the $ECCR^i$ and $EBCC^i$ input models, $\hat{\theta}^*$ determines the maximum proportional changes in all inputs of $\text{DMU}_k$ which allow it to maintain its current status.

For $ECCR^o$ and $EBCC^o$, the output oriented models, $\hat{\phi}^*$ determines the maximum proportional changes in all outputs of DMU$_k$ which allow it to maintain its current status. Consequently, these facts can be beneficial in determining the allowable variations in inputs or outputs which will not change the status of DMU$_k$.

## 4.3. CCR and BCC Sensitivity Procedures

The following procedures are used to determine the maximum allowable changes to inputs and outputs before a DMU changes its status. The procedures differ slightly for the CCR and BCC input- and output-oriented models. The procedure must be applied to each DMU in the set in which sensitivity analysis is desired.

Set $\zeta = X_j$ and $\eta = Y_j$ of the DMU $j$ under consideration.

```
Procedure SensCCR^i(j, ζ, η):
    1. Set $X_j \leftarrow \zeta$ and $Y_j \leftarrow \eta$.
    2. Solve $ECCR^i_j$.
    3. Let $X^*_j \leftarrow \hat{\theta}^* X_j$.
    4. $\Delta X_j = X^*_j - X_j$.
Procedure SensCCR^o(j, ζ, η):
    1. Set $X_j \leftarrow \zeta$ and $Y_j \leftarrow \eta$.
    2. Solve $ECCR^o_j$.
    3. Let $Y^*_j \leftarrow \hat{\phi}^* Y_j$.
    4. $\Delta Y_j = Y^*_j - Y_j$.
Procedure SensBCC^i(j, ζ, η):
    1. Set $X_j \leftarrow \zeta$ and $Y_j \leftarrow \eta$.
    2. Solve $EBCC^i_j$.
    3. If $EBCC^i_j$ is infeasible then:
        a. DMU$_j$ is efficient.
        b. $\Delta X_j = \infty$.
    Else:
        a. Let $X_j^\star \leftarrow \hat{\theta}^* X_j$.
        b. $\Delta X_j = X_j^\star - X_j$.
```

**Procedure SensBCC$^o(j, \zeta, \eta)$:**

```
    1. Set $X_j \leftarrow \zeta$ and $Y_j \leftarrow \eta$.
    2. Solve $EBCC_j^o$.
    3. If $EBCC_j^o$ is infeasible then:
        a. DMU$_j$ is efficient.
        b. $\Delta Y_j = Y_j$.
    Else:
        a. Let $Y_j^\star \leftarrow \hat{\phi}^* Y_j$.
        b. $\Delta Y_j = Y_j^\star - Y_j$.
```

In the EDEA input models, if $\hat{\theta}^* \geq 1$, then DMU$_k$ is efficient, and the maximum increase of inputs that allows the DMU to remain efficient is $\Delta X_k = \hat{\theta}^* X_k - X_k$. If $\hat{\theta}^* < 1$, then DMU$_k$ is inefficient and $\Delta X_k$ determines the necessary decrease of inputs to cause the DMU to become efficient. Note that $\Delta X_k$ identifies ranges such that if all perturbations remain less than the $\Delta X_k$ values, the DMU will retain its current status. Therefore, the effects of non-proportional variations of the inputs on the DMU's status can be evaluated. It is also important to note that only when all perturbations exceed the $\Delta X_k$ ranges will the DMU be guaranteed to change its status. If even one input perturbation is smaller than the allowable range, a change in status is not assured for the DMU. This will be discussed further in Section 4.4.

A similar interpretation can be made concerning the EDEA output-oriented models. In these models, allowable proportional changes in outputs are determined by

$\Delta Y_k = \hat{\phi}^* Y_k - Y_k$. If $\hat{\phi}^* \leq 1$, DMU$_k$ is efficient and the maximum allowable decrease of outputs is given by $\Delta Y_k$. For $\hat{\phi}^* > 1$, DMU$_k$ is inefficient and $\Delta Y_k$ identifies the required increases of the outputs of DMU$_k$ which will cause the DMU status to change.

In the BCC model, the EDEA linear programming formulations may be infeasible for the DMU of interest, DMU$_k$, in either the input or the output model. For the input-oriented model, this would mean that DMU$_k$ is producing one or more outputs in an amount greater than any other DMU. Consequently, regardless of how much the inputs of DMU$_k$ are increased, DMU$_k$ will remain Pareto-Koopmans efficient at its own scale level. For the output-oriented model, an infeasible solution indicates there is at least one input for which DMU$_k$ consumes less than any other DMU. In this case, outputs can be reduced to zero and the DMU will remain efficient at its own scale level. Because of their special characteristics, these DMUs will be termed *critically efficient* because eliminating them from the data set critically alters the feasible region. Pre-processing of the data can reveal these DMUs prior to performing any sensitivity analysis. It is important to note, that the problem of infeasibility is unique to the BCC model and not the CCR model because of the feasible regions: the feasible region of the BCC model is a closed convex set, whereas the feasible region of the CCR model is a convex cone.

The advantage of EDEA over previous models for conducting sensitivity analysis is that it separates the effects of perturbations of inputs from those of the perturbations of outputs. In general, the results are only meaningful for measuring effects of perturbations in inputs alone, or outputs alone, but not both simultaneously. Additionally, the method requires few computational resources and can be used simultaneously with determining the desired DEA efficiency scores.

## 4.4. Additive Model Sensitivity Analysis

Charnes, Haag, Jaska and Semple [43] (CHJS) presented the following procedure for performing sensitivity analysis in DEA based on the additive model. Given the empirical points $(X_j, Y_j)$, $j = 1, \ldots, n$, with input vector $X_j \geq \mathbf{0}$ with at least one $x_{ij}$ positive and output vector $Y_j \geq \mathbf{0}$ with at least one $y_{rj}$ positive, Charnes, et. al. [43], define the *empirical production set* as

$$\mathrm{PE} = \left\{ \boldsymbol{y} \in \Re^s, \boldsymbol{x} \in \Re^m | (\boldsymbol{y}, \boldsymbol{x}) = \sum_{i=1}^{n} \lambda_i (Y_i, X_i); \ \sum_{i=1}^{n} \lambda_i = 1, \ \lambda_i \geq 0 \right\}$$

where $\lambda_i$ are real scalar coefficients.

The efficiency of a DMU is determined by comparing its vector of outputs and inputs with the empirical production set. The DMU of interest is efficient if its vector is Pareto-Koopmans efficient with respect to PE.

A unique feature of the additive model is that it does not require an input or output orientation. The additive model is given as:

$$(ADD_k) \qquad \max\ z = \mathbf{1}\boldsymbol{s}^o + \mathbf{1}\boldsymbol{s}^i \tag{4.15}$$

$$\text{s.t.} \quad \mathbf{Y}\boldsymbol{\lambda} - \boldsymbol{s}^o = \mathbf{Y}_k \tag{4.16}$$

$$\mathbf{X}\boldsymbol{\lambda} + \boldsymbol{s}^i = \mathbf{X}_k \tag{4.17}$$

$$\mathbf{1}\boldsymbol{\lambda} = 1 \tag{4.18}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^o, \boldsymbol{s}^i \geq \mathbf{0}. \tag{4.19}$$

DMU$_k$ is efficient if and only if $z^* = 0$ at optimality. As in all DEA formulations, with this method, a linear program must be solved for each DMU tested.

The purpose of the CHJS paper is to present a new procedure to conduct sensitivity analysis for a specific class of DEA models which allows for a simultaneous proportional change in outputs and inputs. For each DMU, CHJS calculate a "region of stability," that is, a symmetric cell such that all perturbations within the cell maintain the DMU's status. The shape of the cell is affected by the norm used. The authors present an analysis based on the $L_\infty$ norm as well as the $L_1$ norm. Only the $L_\infty$ norm will be discussed since the same conclusions can be reached for the $L_1$ norm.

To understand the CHJS procedure, a few definitions are necessary. Given $\text{DMU}_k$, the radius of stability is the largest number, $r_p$, such that arbitrary perturbations to $\text{DMU}_k$ with $p$-norm strictly less than $r_p$ preserve the DMU's current state. The $L_\infty$ norm is defined as $\|z\|_\infty = \max_i |z_i|$.

A characteristic of the CHJS approach is that the analysis must be carried out differently for efficient DMUs and inefficient DMUs. In the CHJS approach, a procedure is needed to first determine the efficiency status of each DMU. Once this status is determined, sensitivity analysis can be conducted on each DMU with an appropriate procedure, depending if the DMU is efficient or inefficient. Note, with the EDEA formulation, computational efficiency is attained because one process determines the efficiency status and the necessary information to conduct sensitivity analysis for each DMU. The CHJS sensititivity analysis can be described as follows. For efficient $\text{DMU}_k$, the authors describe the $L_{\infty e}$ model as:

($L_{\infty e}$-norm Model)

$$\min\ \theta \tag{4.20}$$

$$\text{s.t.}\quad \mathbf{Y}^{[k]}\boldsymbol{\lambda} - \boldsymbol{s}^{o} + \mathbf{1}\theta = \mathbf{Y}_k \tag{4.21}$$

$$\mathbf{X}^{[k]}\boldsymbol{\lambda} + \boldsymbol{s}^{i} - \mathbf{1}\theta = \mathbf{X}_k \tag{4.22}$$

$$\mathbf{1}\boldsymbol{\lambda} = 1 \tag{4.23}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^{o}, \boldsymbol{s}^{i}, \theta \geq \mathbf{0}. \tag{4.24}$$

If $\text{DMU}_k$ is inefficient, the $L_{\infty i}$ model must be used. It is written as:

($L_{\infty i}$-norm Model)

$$\max\ \theta \tag{4.25}$$

$$\text{s.t.}\quad \mathbf{Y}^{[k]}\boldsymbol{\lambda} - \boldsymbol{s}^{o} - \mathbf{1}\theta = \mathbf{Y}_k \tag{4.26}$$

$$\mathbf{X}^{[k]}\boldsymbol{\lambda} + \boldsymbol{s}^{i} + \mathbf{1}\theta = \mathbf{X}_k \tag{4.27}$$

$$\mathbf{1}\boldsymbol{\lambda} = 1 \tag{4.28}$$

$$\boldsymbol{\lambda}, \boldsymbol{s}^{o}, \boldsymbol{s}^{i}, \theta \geq \mathbf{0}. \tag{4.29}$$

The $L_{\infty e}$ model determines the maximum permissible decrease in any output with a simultaneous increase in any input before the efficient DMU becomes inefficient. On the other hand, the $L_{\infty i}$ model finds the required decrease of inputs along with an increase in outputs that the inefficient DMU must witness before it changes status and becomes efficient.

The authors present proofs demonstrating that $\theta^*$ for both models is the radius of stability $r_\infty$. Having determined $\theta^*$, the authors claim that allowable perturbations to an efficient DMU of the form

$$\begin{bmatrix} Y_k^* \\ X_k^* \end{bmatrix} = \begin{bmatrix} Y_k - \mathbf{1}\tau \\ X_k + \mathbf{1}\tau \end{bmatrix}.$$

where $\tau \in [0, \theta^*)$, preserve efficiency for the DMU. No corresponding analysis is presented for the inefficient DMUs, but the allowable perturbations for the inefficient DMUs would take the form

$$\begin{bmatrix} Y_k^* \\ X_k^* \end{bmatrix} = \begin{bmatrix} Y_k + \mathbf{1}\tau \\ X_k - \mathbf{1}\tau \end{bmatrix}.$$

where $\tau \in [0, \vartheta^*)$. Only when $\tau > \theta^*$ will a change in status be assured for the inefficient DMU.

The authors claim that the projected points found when $\tau = \theta^*$ always form an *unstable point*, and therefore prove the region of stability is valid. A point $(Y_j, X_j)$ is unstable if and only if for all $\epsilon > 0$, the open ball centered at $(Y_j, X_j)$ with radius $\epsilon$ contains efficient and inefficient points with respect to the associated empirical production possibility set.

### 4.4.1. Sensitivity Analysis Example

The results of the CHJS approach will now be compared with the $ECCR^i$, $ECCR^o$, $EBCC^i$ and $EBCC^o$ models' results using the data presented in the CHJS paper. In this example, there are three DMUs, each with one output and two inputs, as shown in Table 4.1. The results of an analysis of efficient DMU$_A$ are given in Table 4.2.

Table 4.1. -- Sensitivity analysis example

| | DMU$_A$ | DMU$_B$ | DMU$_C$ |
|---|---|---|---|
| $y_{1j}$ | 1.00 | 0.25 | 0.25 |
| $x_{1j}$ | 1.00 | 0.25 | 1.00 |
| $x_{2j}$ | 1.00 | 1.00 | 0.25 |

Table 4.2. -- Comparative sensitivity results

| | EDEAi | | | | |
|---|---|---|---|---|---|
| | $ECCR^i$ | $EBCC^i$ | $ECCR^o$ | $EBCC^o$ | CHJS |
| $\Delta y_{1A}$ | — | — | - 0.643 | -0.750 | -0.750 |
| $\Delta x_{1A}$ | 1.800 | infeas. | — | — | 0.750 |
| $\Delta x_{2A}$ | 1.800 | infeas. | — | — | 0.750 |

To make proper comparisons between the EDEA models and the CHJS approach, the $EBCC$ models should be observed. The infeasibility of the $EBCC^i$ model indicates that DMU$_A$ is efficient at its own scale size, therefore, the inputs can be increased indefinitely and the DMU will remain efficient. The $EBCC^o$ model indicates how far the output can be reduced for DMU$_A$ to remain efficient. In this case, as long as $y_{1A} > 0.25$, DMU$_A$ will remain efficient. With the information from both the input and output models, one could conclude that simultaneous changes can occur in both the inputs and outputs and DMU$_A$ will remain efficient. As long as $y_{1A} > 0.25$, inputs can increase indefinitely and DMU$_A$'s status will remain unchanged. Notice, the CHJS approach yields similar, although more limited, results. The CHJS results indicate that the inputs and outputs can simultaneously be perturbed; as long as $y_{1A} > 0.25$ and $x_{1A} < 1.75$, and $x_{2A} < 1.75$, DMU$_A$ will be efficient. Compared to the EDEA models, it is evident that the CHJS results place an artificial limit on the allowable perturbations of the inputs.

An additional benefit of the EDEA sensitivity analysis approach is that the $ECCR^i$ and $ECCR^o$ models reveal information on the perturbations which allow DMU$_A$ to remain scale efficient. Information on any aspects of scale efficiency are unavailable with the CHJS approach.

To compare the CHJS approach to the $ECCR^i$ and $ECCR^o$ models, a simple scaling of the data, so that all DMUs are at the same scale size, is useful. Since there is only one output, this scaling can be accomplished by dividing all data for each DMU by its output value. The *normed* data are presented in Table 4.3.

Since all DMUs produce a single output at the same level, a unit isoquant plot of the data is possible as depicted in Figure 4.1. In this example, DMU A is efficient, B is weakly efficient, and C is inefficient.

Table 4.3. -- Normed data example

| | DMU$_A$ | DMU$_B$ | DMU$_C$ |
|---|---|---|---|
| $y_{1j}$ | 1.00 | 1.00 | 1.00 |
| $x_{1j}$ | 1.00 | 1.00 | 4.00 |
| $x_{2j}$ | 1.00 | 4.00 | 2.00 |

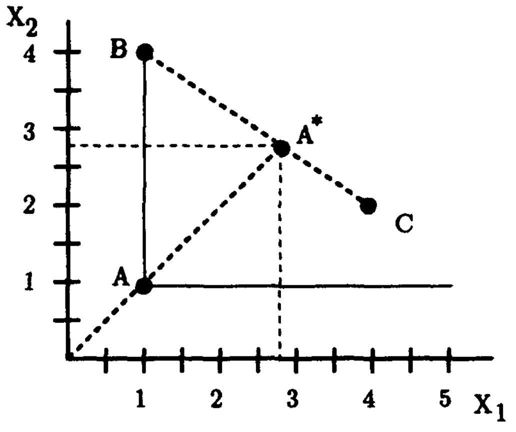

Figure 4.1. Normalized data isoquant plot

Table 4.4.-- Normed data sensitivity results

| | EDEA | | | | |
|---|---|---|---|---|---|
| | $ECCR^i$ | $EBCC^i$ | $ECCR^o$ | $EBCC^o$ | CHJS |
| $\Delta y_{1A}$ | — | — | -0.643 | infeas. | -1.800 |
| $\Delta x_{1A}$ | 1.800 | 1.800 | — | — | 1.800 |
| $\Delta x_{2A}$ | 1.800 | 1.800 | — | — | 1.800 |

Results for the EDEA and CHJS sensitivity models are given in Table 4.4. The infeasibility of the $EBCC^o$ model indicates that reductions in output are possible; as long as output is greater than zero, the DMU will remain Pareto-Koopmans efficient. Note however, if $y_{1A} < (1 - 0.643) = 0.357$, DMU$_A$ will no longer remain scale efficient. The $ECCR^i$ input-oriented models indicate that as long as $Y_A$ is held fixed, inputs can be increased to 2.8 and DMU A will remain both scale and Pareto-Koopmans efficient. In fact, when both the $ECCR^i$ and $ECCR^o$ models are paired, one would conclude that simultaneous perturbations of both outputs and inputs are possible. As long as inputs remain less than 2.8, output can be reduced to 0.357 and DMU$_A$ will remain scale efficient. DMU$_A$ will remain Pareto-Koopmans efficient even if output is reduced to 0, as long as inputs remains less than 2.8.

The results of CHJS models reveal a potential problem with their approach. The results indicate as long as the inputs remain less than 2.8, DMU A will remain Pareto-Koopmans efficient, even with simultaneous reductions in outputs. Further, A will remain efficient, according to CHJS, even if output is reduced to $y_{1A} = -0.800$. This clearly is not true. The causes of the problem can be identified. The authors claim the projection

$$\begin{bmatrix} y_{1o}^* \\ x_{1o}^* \\ x_{2o}^* \end{bmatrix} = \begin{bmatrix} y_{1o} - \theta^* \\ x_{1o} + \theta^* \\ x_{2o} + \theta^* \end{bmatrix} = \begin{bmatrix} -0.8 \\ 2.8 \\ 2.8 \end{bmatrix}$$

is an unstable point. The projection must be an unstable point for the region of stability to be legitimate. Generally, it is assumed that at least one output must be positive in efficient production analysis, such as DEA. Therefore, the projected point cannot be unstable. In fact, the projected point cannot be a valid member of the modified empirical production possibility set PE. Consequently, although the CHJS approach provides important insights into sensitivity analysis for DEA, there are special cases that are problematic with that approach.

## 4.5. Expanded Sensitivity Analysis

An attractive feature of the EDEA model is its simple implementation using existing DEA methodologies. However, a concern of this approach is that the resulting region of stability may be a small subset of the true allowable perturbations for the DMU to maintain its status. Figure 4.2 illustrates this. The shaded area above points A, E, E′ and below points A, B, C, D represents the allowable perturbations for which DMU

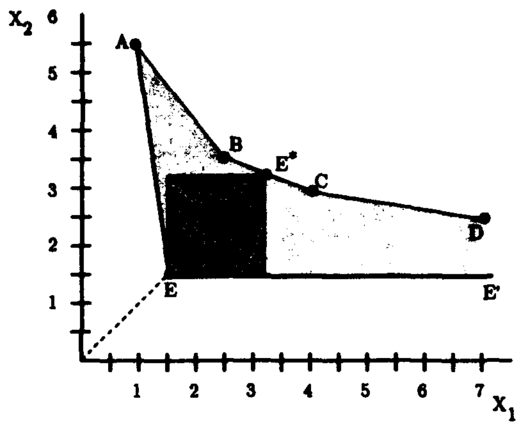

Figure 4.2. Total region of stability

E remains efficient. The EDEA method reveals a region of stability represented by the shaded box.

A larger region of stability is revealed by repeated application of the EDEA methodology, as described below. Procedures ESA-I and ESA-O characterize a series of stable regions for changes in the input and output vectors, respectively, for a given DMU $j$ to the degree of refinement desired. Recall that R $= \{1, \ldots, s\}$ and Q $= \{1, \ldots, m\}$ are the respective index sets for output and input measures. Initially, let $\boldsymbol{\zeta} = X_j$ and $\boldsymbol{\eta} = Y_j$.

**Procedure ESA-I($j, \boldsymbol{\zeta}, \boldsymbol{\eta}$):**

1. Apply procedure SensCCR$^i(j, \boldsymbol{\zeta}, \boldsymbol{\eta})$ or SensBCC$^i(j, \boldsymbol{\zeta}, \boldsymbol{\eta})$ to compute $\Delta X_j$.
2. If the region of interest is not fully characterized, then:

   For all $q \in$ Q: solve ESA-I($j, \boldsymbol{\zeta}, \boldsymbol{\eta} + \mathbf{e}_q \Delta x_{qj}$)

**Procedure ESA-O($j, \boldsymbol{\zeta}, \boldsymbol{\eta}$):**

1. Apply procedure SensCCR$^o(j, \boldsymbol{\zeta}, \boldsymbol{\eta})$ or SensBCC$^o(j, \boldsymbol{\zeta}, \boldsymbol{\eta})$ to compute $\Delta Y_j$.
2. If the region of interest is not fully characterized, then:

   For all $r \in$ R: solve ESA-O($j, \boldsymbol{\zeta}, \boldsymbol{\eta} + \mathbf{e}_r \Delta y_{rj}$)

where $\mathbf{e}_k$ is a column vector of zeros with a 1 in the $k^{th}$ row.

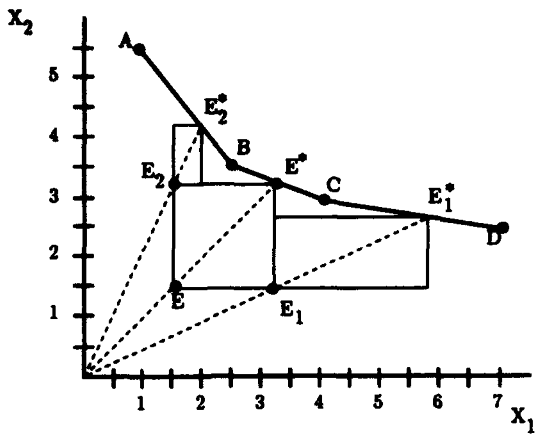

Figure 4.3. Expanded sensitivity analysis

Figure 4.3 reveals how one iteration of this procedure expands the region of stability for an efficient DMU, E. Because this DMU contains only one output, only the ESA-I portion of the algorithm is relevant. Notice, the ESA algorithm can be applied iteratively to yield larger regions of stability if desired. However, the number of linear programs required to trace larger regions grows rapidly. Table 4.5 shows the input and output values for the DMUs depicted in Figure 4.3.

Table 4.5. -- Expanded sensitivity example results

| | $\text{DMU}_A$ | $\text{DMU}_B$ | $\text{DMU}_C$ | $\text{DMU}_D$ | $\text{DMU}_E$ | $\text{DMU}_{E^*}$ | $\text{DMU}_{E_1^*}$ | $\text{DMU}_{E_2^*}$ |
|---|---|---|---|---|---|---|---|---|
| $y_{1j}$ | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| $x_{1j}$ | 1.00 | 2.50 | 4.00 | 7.00 | 1.50 | 3.25 | 5.84 | 2.00 |
| $x_{2j}$ | 6.00 | 3.50 | 3.00 | 2.50 | 1.50 | 3.25 | 2.69 | 4.33 |

The results of the expanded analysis now yield greater information than the EDEA sensitivity analysis. With straight EDEA sensitivity analysis, results indicate that for DMU E, both inputs can be increased from 1.5 to 3.25 before the DMU becomes inefficient. However, the expanded analysis yields more detailed information. DMU $E_1^*$ indicates that as long as $x_{2E} \leq 2.69$, $x_{1E}$ can be increased to 5.84 and E will remain efficient. Likewise, DMU $E_2^*$ indicates $x_{2E}$ can be increased to 4.33 as long as $x_{1E} \leq 2.00$. As can be seen in this example, the expanded sensitivity analysis reveals a greater region of stability in which non-proportional increases in inputs can be witnessed while the DMU maintains its status. This same analysis can apply to an inefficient DMUs as depicted by DMU I in Figure 4.4.

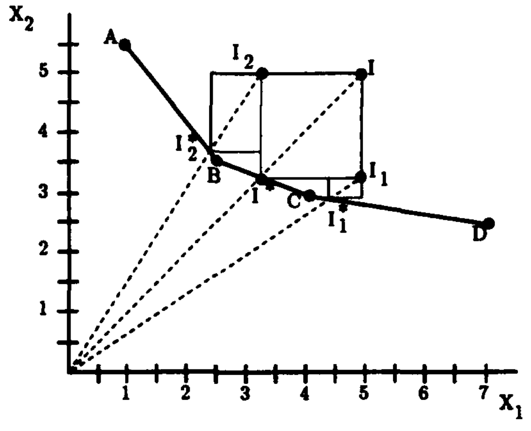

Figure 4.4. Inefficient DMU expanded sensitivity analysis

Because both input and output models must be solved for each DMU, the number of linear programs required to trace the expanded region grows rapidly. For each DMU, $(s + m)$ additional LPs must be solved for each additional increase in the region of stability that is measured. Note however, the CHJS $L_1$ norm technique for inefficient DMUs also requires $(s + m)$ LPs to be solved.

## 4.6. Summary

The motivation for the sensitivity analysis for the efficient DMUs differs substantially from that for the inefficient DMUs. The managers of the efficient DMUs desire to know how much loss in productive efficiency is allowable before the DMU becomes relatively inefficient compared to its peers. The inputs that permit only small increases before efficiency is lost play a different role in the production process than the inputs which can suffer large perturbations and not result in a change of state. This information can give the manager a sense of the relative worth, or price value, of each input. In this way, managers can plan for future markets by shifting to production technologies that provide greater regions of stability. From the output-oriented model, the manager is able to gain a more complete picture of the role of each output. By knowing which outputs are most critical to maintain relative efficiency, the manager is more able to evaluate the opportunity costs associated with producing each output.

For the inefficient DMUs, the manager wants to identify the smallest region of stability. This will reveal how best to allocate resources to move to efficiency. Instead of down-sizing across all inputs with proportional reductions, the manager may be able to focus on reducing those resources that will more rapidly bring the DMU to the efficient frontier. Likewise, by focusing on output expansion, the manager may be able to identify a market niche with a few of outputs in which the DMU can achieve relative efficiency.

The sensitivity analysis presented in this chapter presents the manager with useful information about the role each output and input variable plays in determining the efficiency of the production process. In this manner, the manager can concentrate attention on those factors which can more quickly result in increased productive efficiency when considering policy decisions.

# CHAPTER V

# SUMMARY AND CONCLUSIONS

This dissertation contributes to the development of solution techniques for solving large-scale data envelopment analysis problems which involve the solution of thousands of linear programs. Although the focus is on large-scale problems, the new algorithms presented can be used to effectively solve a wide range of DEA problems.

The analysis presented in Chapter II describes a new problem decomposition procedure which dramatically expedites the solution of the computationally intensive DEA problems and fully exploits parallel processing environments. Computational experiments indicate that by combining linear programming theory with known characteristics of the DEA formulations, solution times can be cut in half compared to general linear programming approaches. When utilizing a parallel processing environment, the efficiency remains above 0.97 indicating that the solution time will improve by the same magnitude as the number of processors used. A new decomposition algorithm is presented which causes a 6- to 10-fold improvement in the solution time for the serial case. When the decomposition procedure is coupled with the parallel processing environment, solution times can be increased by up to two orders of magnitude.

Besides effectively reducing the time to solve large-scale DEA problems, the decomposition approach offers many additional advantages over prior DEA solution methods. It allows for the solution of a large number of different DEA formulations in rapid succession. The decomposition approach opens a new means of conducting window analysis as well as studying categorical variables. Additionally, the decomposition approach helps avoid problems, such as cycling, which frequently occur in larger DEA problems.

Building on these improvements to the DEA solution process, Chapter III outlines a new rank ordering procedure to classify DMUs by their ability to apply current technology to the production process. This new ranking procedure acts as a proxy to measure managerial effectiveness in a way not possible by traditional DEA methods. Best and worst performing DMUs are easily classified with the new ordering procedure. In this way, the analyst can compare the management practices of the DMUs from both classifications to assist in determining which policies improve and detract from productive efficiency.

In Chapter IV, we presented a computationally intensive means of conducting sensitivity analysis on the DEA results. In the chapter, the new sensitivity analysis procedure is compared with previous approaches. The new procedure is shown to avoid problems that may arise with the previously reported methods. The key advantage of the sensitivity analysis is that it helps clarify which factors can cause significant improvements to the production process.

The study of large-scale DEA problems reveals many new issues that have gone unnoticed with the study of smaller problems. The insights gained by investigating these new problems can continue to result in new analytical tools to broaden the usefulness of DEA. The techniques presented in this dissertation open the door to new research which will further expand the range of applications for data envelopment analysis.

# APPENDIX A
# MULTINORMAL DATA GENERATOR

Since few large-scale DEA problems are currently available, randomly generated data sets were needed to simulate possible real life data. One procedure to generate such data is to draw the samples from a multinormal distribution. To this end, the DRNMVN routine from the IMSL library was used to generate data for 10,000 DMUs. The variance covariance matrix of Table A.1 was used to generate the data for the input and output variables for each DMU. The code to generate this data is described by Hickman[62].

Table A.1. – Variance-Covariance matrix for multinormal data

| | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ |
|---|---|---|---|---|---|---|---|---|
| $x_1$ | 303.4 | | | | | | | |
| $x_2$ | 2.5 | 2.5 | | | | | | |
| $x_3$ | 12493.9 | 6093.9 | 112750625.0 | | | | | |
| $x_4$ | 4062.6 | 484.2 | -2890405.7 | 31033992.2 | | | | |
| $x_5$ | 265.2 | 45.3 | -1346076.8 | -455158.7 | 7629765.5 | | | |
| $x_6$ | 19597.2 | -195.0 | 3436081.9 | -270405.4 | -674642.7 | 86492323.0 | | |
| $x_7$ | 36418.8 | 6428.4 | 111950224.4 | 27415022.3 | 5153887.3 | 88983356.8 | 233502490.7 | |
| $x_8$ | 3647.0 | 1001.0 | 10815783.1 | 112293.8 | -64824.7 | - 319883.0 | 10543369.3 | 7812141.0 |

From the randomly generated data, the variables representing inputs and outputs needed to be carefully chosen. To coincide with sound economic theory, all outputs should be positively correlated with all the inputs. In Fig. A.1, all negatively correlated variables are connected with a line.

By eliminating variable $x_4$, variables $x_1$, $x_3$, and $x_7$ could represent outputs since they would be positively correlated with all the other remaining variables that

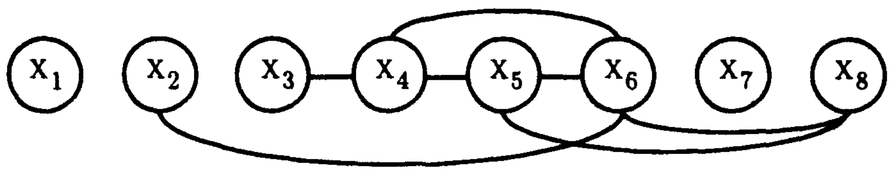

Figure A.1. Correlations of multinormal data variables

would represent inputs. In this way, each DMU would be comprised of 3 outputs and 4 inputs. Notice also that inputs positively correlated would be compliments in the production process and those negatively correlated would be substitutes. Since the input variables cover a wide range of compliment and substitute relationship cases, the randomly generated data further simulates possible real life data.

# APPENDIX B

# DEA-GEN

While random data generators can provide large problems, a more systematic approach is need to represent real life scenarios. To this end, economic theory was used to generate large-scale DEA problem sets. In production economics, the most widely used functional form is known as the Cobb-Douglas production function [24]. This function is written as:

$$y_j = a_o \prod_{i=1}^{m} x_{ij}^{\alpha_i}, \; x_{ij} > 0, \; j = 1, \ldots, n$$

Here, $y_j$ is the single aggregate output produced by DMU$_j$, $x_{ij}$ is the value of input $i$ used by DMU$_j$ in the production process, $\alpha_i$ is the elasticity factor for input $i$, and $a_o$ is a constant scale factor. If $\sum_{i=1}^{m} \alpha_i = 1$ then only constant returns to scale exist in the production process. For $\sum_{i=1}^{m} \alpha_i < 1$ decreasing returns to scale are present, while $\sum_{i=1}^{m} \alpha_i > 1$ indicates increasing returns to scale. In the DEA studies, increasing returns to scale are not used because the function results in only a few DEA efficient points. Although this does not pose a problem, it does not realistically represent true life data.

For the single output model, this production function has many desirable properties. If the inputs are randomly generated, the function generates output values that will always lie on the production possibility frontier, i.e., they will be DEA efficient for $\sum_{i=1}^{m} \alpha_i \leq 1$. This frontier is central to the theory of economic growth and measures the rate of technological progress. The Cobb-Douglas frontier represents the best use of technology as well as the best management practices to achieve efficient production. This coincides with the practical use of DEA. Unlike DEA, the quest in economic studies is to attempt to estimate the values of $\alpha_i$ by fitting the function to the observed data. By choosing the $\alpha_i$ values a priori and then randomly generating the input values, the output values can be determined so that they coincide with widely accepted economic theory. To insure that not all generated data sets fall on the efficient frontier surface, the $a_o$ scale factor can be randomly generated. The efficient DMUs will consist of those generated where $a_o$ takes on it maximum value. Control over the number of DMUs that are efficient in the data set can be maintained by limiting the number of DMUs generated with $\max a_o$.

In DEA analysis, the single-output, multiple-input scenario is not of primary interest. DEA was developed to analyze the case of multiple-output, multiple-input studies. Färe and Grosskopf [54] point out that the existence of a joint production function has not been established. That is, the multiple-output, multiple-input model does not produce values strictly on the efficient production frontier. To do so, would require strict assumptions that would not provide the desired realism for the DEA study. However, a joint model without the strict assumptions would simulate economically sound production processes. Consequently, a joint model was developed for the DEA-GEN problem generator.

For each of the problem sets, the input values were generated from a uniform distribution. The $\alpha_i$ values were chosen to simulate constant returns to scale processes as well as a variety of cases representing different levels of decreasing returns to scale. The $a_o$ value was randomly generated, but with a modest control of the number of efficient DMUs that could be present in the problem data.

Once the single aggregate output level was calculated, the individual output levels were determined by assigning each individual output as a percentage of the aggregate.

The percentages for each individual output were drawn from normal distributions with predetermined means and standard deviations. The means of the normal distributions were chosen so that the percentages sum to one. Table B.1 lists the $\alpha_i$ values as well as the means and standard deviations that were used to generate twelve different Cobb-Douglas data sets.

Table B.1. -- Parameter values for DEA-GEN code

| | #DMUs | $\alpha$ Values | Mean Values | Standard Dev. |
|---|---|---|---|---|
| DEA-GENa | 1,000 | .3,.2,.2,.3 | .12,.13,.13,.14,.11 | .03,.04,.02,.01,.01 |
| DEA-GENa | 2,000 | .3,.2,.2,.2,.1 | .20,.30,.15 | .03,.04,.02 |
| DEA-GENa | 4,000 | .3,.3,.2,.2 | .12,.12,.15,.14 | .03,.04,.02,.01 |
| DEA-GENa | 8,000 | .4,.2,.1,.1,.1 | .5 | .1 |
| DEA-GENb | 1,000 | .03,.07,.12,.03,.12,.1,.02 | .2,.3,.15 | .01,.05,.01 |
| DEA-GENb | 2,000 | .1,.1,.1,.1,.1,.1,.1 | .52,.13 | .03,.04 |
| DEA-GENb | 4,000 | .4,.3,.1 | .4,.3,.15 | .13,.08,.02 |
| DEA-GENb | 8,000 | .3,.17,.12,.03,.12,.2 | .2,.3,.15,.14 | .01,.05,.01,.01 |
| DEA-GENc | 1,000 | .18,.2,.14,.1,.1,.1 | .2,.12,.14,.11,.11 | .03,.04,.02,.01..01 |
| DEA-GENc | 2,000 | .13,.18,.2,.14,.1,.1 | .2,.18,.15,.19 | .03,.04,.02,.01 |
| DEA-GENc | 4,000 | .13,.2,.12,.13 | .2,.3,.15,.14,.11 | .03,.04,.02,.01,.01 |
| DEA-GENc | 8,000 | .08,.12,.21,.07,.11 | .3,.3,.3,.1 | .04,.04,.1,.03 |
| DEA-GENd | 25,000 | .3,.3,.3 | .3,.3 | .01,.01 |

Because of the economic foundations of the DEA-GEN code, the data generated resembles a class of problems that should more closely simulate realistic economic data than what would be possible from the data sampled from a multinormal distribution.

# BIBLIOGRAPHY

[1] D.L. Adolphson, G.C. Cornia and L.C. Walters, A unified framework for classifying DEA models, Operational Research '90 (1990)647–657.

[2] D.L. Adolphson, G.C. Cornia and L.C. Walters, Railroad property valuations using data envelopment analysis, Interfaces 19(1989)18–26.

[3] A.I. Ali, IDEAS: Integrated data envelopment analysis system, Technical Report (Department of General Business and Finance, University of Massachusetts Amherst, MA, 1989).

[4] A.I. Ali, Data envelopment analysis: computational issues, Comput. Environ. and Urban Systems 14(1990)157–165.

[5] A.I. Ali, Facilitating DEA methodology, ORSA/TIMS Joint National Meeting, San Francisco, CA, 1992.

[6] A.I. Ali, Streamlined computation for data envelopment analysis, European Journal of Operational Research 64(1993)61–67.

[7] A.I. Ali, Computational aspects of data envelopment analysis, to appear in A. Charnes, W.W. Cooper, A. Lewin and L.M. Seiford, *DEA: Theory, Methodology, and Applications* (Kluwer Academic Publishers, Boston, MA, 1994).

[8] A.I. Ali, W.D. Cook and L.M. Seiford, Strict vs. weak ordinal relations for multipliers in data envelopment analysis, Management Science 37(1991)733–738.

[9] A.I. Ali and L.M. Seiford, Computational accuracy and infinitesimals in data envelopment analysis, Technical Report (Department of General Business and Finance, University of Massachusetts, Amherst, MA, 1991).

[10] R.D. Banker, Estimating most productive scale size using data envelopment analysis, European Journal of Operations Research 17(1984)35–44.

[11] R.D. Banker, A. Charnes and W.W. Cooper, Some models for estimating technical and scale inefficiencies in data envelopment analysis, Management Science 30(1984)1078–1094.

[12] R.D. Banker, A. Charnes, W.W. Cooper and A.P. Schinner, A bi-extremal principle for frontier estimation and efficiency evaluations, Management Science 27(1981)1370–1389.

[13] R.D. Banker, R.F. Conrad and R.P. Strauss, A comparative application of data envelopment analysis and translog methods: an illustrative study of hospital production, Management Science 32(1986)30–44.

[14] R.D. Banker, S.M. Datar and C.F. Kemerer, A model to evaluate variables impacting the productivity of software maintenance projects, Management Science 37(1991)1–18.

[15] R.D. Banker and A. Maindiratta, Piecewise loglinear estimation of efficient production surfaces, Management Science 32(1986)126-135.

[16] R.D. Banker and R.C. Morey, Efficiency analysis for exogenously fixed inputs and outputs, Operations Research (1985)513–521.

[17] R.D. Banker and R.C. Morey, The use of categorical variables in data envelopment analysis, Management Science 32(1986)1613–1627.

[18] R.S. Barr and M.L. Durchholz, Parallel software for large-scale data envelopment analysis, ORSA/TIMS Joint National Meeting, San Francisco, CA, 1992.

[19] R.S. Barr and M.L. Durchholz, Parallel and hierarchical decomposition approaches for solving large-scale DEA models, ORSA/TIMS Joint National Meeting, Chicago, IL, 1993.

[20] R.S. Barr and B.L. Hickman, Reporting computational experiments with parallel algorithms: issues, measures, and experts' opinions, ORSA Journal on Computing 5(1993)2–18.

[21] R.S. Barr, L.M. Seiford and T. Siems, An envelopment-analysis approach to measuring the managerial quality of banks, Annals of Operations Research 42(1993)1-19.

[22] R.S. Barr and T. Siems, Predicting bank failure using DEA to quantify management quality, Technical Report (Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX, 1993).

[23] P.W. Bauer, A.N. Berger and D.B. Humphrey, Inefficiency and productivity growth in banking: a comparison of stochastic econometric and thick frontier methods, Financial Studies Section, Federal Reserve Board Washington DC (1991).

[24] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali, *Linear Programming and Network Flows* (John Wiley & Sons, NY, 1990).

[25] R.E. Bixby, Implementing the simplex method: the initial basis, ORSA Journal on Computing 4(1992)267-284.

[26] R.G. Bland, New finite pivoting rules for the simplex method, Mathematics of Operations Research 2(1977)103–107.

[27] W.F. Bowlin, A. Charnes, W.W. Cooper and H.D. Sherman, Data envelopment analysis and regression approaches to efficiency estimation and evaluation, Annals of Operations Research 2(1985)113–138.

[28] P. Byrnes, R. Färe and S. Grosskopf, Measuring production efficiency: an application to Illinois strip mines, Management Science 30(1984)671–680.

[29] P. Byrnes, S. Grosskopf and K. Hayes, Efficiency and ownership: further evidence, The Review of Economics and Statistics 68(1986)337–341.

[30] A. Charnes and W.W. Cooper, *Management models and industrial applications of linear programming* (John Wiley & Sons, NY, 1961).

[31] A. Charnes and W.W. Cooper, Preface to topics in data envelopment analysis, Annals of Operations Research 2(1985)59–94.

[32] A. Charnes and W.W. Cooper, Data envelopment analysis, Operational Research '90 (1990)641–646.

[33] A. Charnes, W.W. Cooper, B. Golany, L. Seiford and J. Stutz, Foundations of data envelopment analysis for Pareto-Koopmans efficient empirical production functions, Journal of Econometrics 30(1985)91–107.

[34] A. Charnes, W.W. Cooper and Z.M. Huang, Polyhedral cone-ratio DEA models with an illustrative application to large commercial banks, Journal of Econometrics 46(1990)73–91.

[35] A. Charnes, W.W. Cooper, Z.M. Huang and D.B. Sun, Relations between half-space and finitely generated cones in polyhedral cone-ratio DEA models, International Journal of Systems Science 22(1991)2057–2077.

[36] A. Charnes, W.W. Cooper, A.Y. Lewin, R.C. Morey and J. Rousseau, Sensitivity and stability analysis in DEA, Annals of Operations Research 2(1985)139–156.

[37] A. Charnes, W.W. Cooper and E. Rhodes, Measuring the efficiency of decision making units, European Journal of Operational Research 2(1978)429–444.

[38] A. Charnes, W.W. Cooper and E. Rhodes, Short communication: measuring the efficiency of decision making units, European Journal of Operational Research (1979)339.

[39] A. Charnes, W.W. Cooper and E. Rhodes, Evaluating program and managerial efficiency: an application of data envelopment analysis to program follow through, Management Science 27(1981)668–697.

[40] A. Charnes, W.W. Cooper, L. Seiford and J. Stutz, A multiplicative model for efficiency analysis, Journal of Socio-Economic Planning Science 16(1981)223–235.

[41] A. Charnes, W.W. Cooper and R.M. Thrall, Classifying and characterizing efficiencies and inefficiencies in data envelopment analysis, Operations Research Letters 5(1986)105–110.

[42] A. Charnes, W.W. Cooper and R.M. Thrall, A structure for classifying and characterizing efficiency and inefficiency in data envelopment analysis, The Journal of Productivity Analysis 2(1991)197–237.

[43] A. Charnes, S. Haag, P. Jaska and J. Semple, Sensitivity of efficiency classifications in the additive model of data envelopment analysis, International Journal of Systems Science 23(1992)789–798.

[44] A. Charnes and L. Neralić, Sensitivity analysis of the additive model in data envelopment analysis, European Journal of Operational Research 48(1990)332–341.

[45] A. Charnes and L. Neralić, Sensitivity analysis of the simultaneous proportionate change of inputs and outputs in data envelopment analysis, Research Report CCS 667 (Center for Cybernetic Studies, University of Texas at Austin, 1991).

[46] A. Charnes, J. Rousseau and J. Semple, An effective non-archimedean anti-degeneracy/cycling linear programming method especially for data envelopment analysis and like models, Annals of Operations Research 32(to appear 1994).

[47] C.W. Cobb and P.H. Douglas, A Theory of Production, American Economic Review (1928)139-165.

[48] W.D. Cook and M. Kress, A data envelopment model for aggregation preference rankings, Management Science 36(1990)1302–1310.

[49] T.H. Cormen, C.E. Leiverson and R.L. Rivest, *Introduction to Algorithms* (MIT Press, Cambridge, MA, 1990).

[50] CPLEX Optimization Inc., *Using the CPLEX Linear Optimizer* (CPLEX Optimization Inc., Incline Village, NV, 1990).

[51] M.J. Farrell, The measurement of productive efficiency, Journal of Royal Statistical Society Series A 120(1957)253–290.

[52] M.J. Farrell and M. Fieldhouse, Estimating efficient production functions under increasing returns to scale, Journal of Royal Statistical Society Series A (1962)252–267.

[53] R. Färe, S. Grosskopf, C.A.K. Lovell, The measurement of efficiency of production Kluwer-Nijhoff Publishing, MA, (1985)1–216.

[54] R. Färe and W. Hunsaker, Notions of efficiency and their reference sets, Management Science 32(1986)237–243.

[55] G.D. Ferrier and C.A.K. Lovell, Measuring cost efficiency in banking, econometric and linear programming evidence, Journal of Econometrics 46(1990)229–245.

[56] M.J. Flynn, Very high-speed computing systems, Proceedings of the IEEE 54(1966)1901–1909.

[57] F.R. Førsund, C.A.K. Lovell and P. Schmidt, A survey of frontier production functions and of their relationship to efficiency measurement, Journal of Econometrics 13(1980)5–25.

[58] T. Gal, H. Kruse and P. Zönig, Survey of solved and open problems in the degeneracy phenomenon, Mathematical Programming 42(1988)125–133.

[59] S.I. Gass, Comments on the possibility of cycling with the simplex method, Operations Research 27(1979)848–852.

[60] P. Hall, W. Härdle and L. Simar, Iterated bootstrap with applications to frontier models, CORE Discussion Paper No. 9121 (Center for Operations Research and Econometrics, Belgium, 1991).

[61] B. Hattersley and J. Wilson, A dual approach to primal degeneracy, Mathematical Programming 42(1988)135–145.

[62] B.L. Hickman, Parallel algorithms for pure network problems and related applications, Ph.D. thesis (Southern Methodist University, Dallas, TX, 1991).

[63] IBM Corporation, *Mathematical Programming System Extended/370 Program Reference Manual* (IBM Corporation, Kingston, NY, 1979).

[64] IBM Corporation, *Optimization Subroutine Library Reference* (IBM Corporation, Kingston, NY, 1979).

[65] W.A. Kamakura, A note on the use of categorical variables in data envelopment analysis, Management Science 34(1988)1273–1276.

[66] D.E. Knuth, *The Art of Computer Programming Volume 3 / Sorting and Searching* (Addison-Wesley, Reading, MA, 1973).

[67] L.S. Lasdon, *Optimization Theory for Large Systems* (Macmillan, NY, 1970).

[68] A.M. Law and W.D. Kelton, *Simulation Modeling and Analysis*, 2nd ed. (McGraw-Hill, NY, 1990).

[69] A. Maindiratta, Largest size-efficient scale and size efficiencies of decision-making units in data envelopment analysis, Journal of Econometrics 46(1990)57–72.

[70] R. Marsten, The design of the XMP linear programming library, ACM Transactions on Mathematical Software 7(1981)481–297.

[71] J.M. Mulvey, Pivot strategies for primal-simplex network codes, Journal of the ACM 25(1978)206-270.

[72] B.A. Murtagh, *Advanced Linear Programming* (McGraw-Hill, NY, 1981).

[73] E.D. Nering and A.W. Tucker, *Linear Programs and Related Problems* (Harcourt Brace Jovanovich, Boston, MA, 1993).

[74] Orchard-Hays, *Advanced Linear Programming Computing Techniques* (McGraw-Hill, NY, 1968).

[75] A. Osterhaug, *Guide to Parallel Programming On Sequent Computer Systems* (Sequent Computer Systems, Inc., OR, 1986).

[76] F. Phillips, R.G. Parsons and A. Donoho, Parallel microcomputing for data envelopment analysis, Computing, Environment, and Urban Systems 14(1990)167–170.

[77] D.L. Retzlaff-Roberts and R.C. Morey, A goal-programming method of stochastic allocative data envelopment analysis, Technical Report (Memphis State University, TN, 1992).

[78] J.J. Rousseau and J.H. Semple, Categorical outputs in data envelopment analysis, Management Science (forthcoming).

[79] J.J. Rousseau and J.H. Semple, Non-archimedean infinitesimals and categorical inputs in data envelopment analysis, International Journal of Systems Science (forthcoming).

[80] L.M. Seiford, A DEA bibliography (1978-1992), to appear in Charnes, A., W. W. Cooper, A. Y. Lewin, and L. M. Seiford, *DEA: Theory, Methodology, and Applications* (Kluwer Academic Publishers, Boston, MA, 1994).

[81] L.M. Seiford and R.M. Thrall, Recent developments in DEA: the mathematical programming approach to frontier analysis, Journal of Econometrics 46(1990)7–38.

[82] L. Simar, Estimating efficiencies from frontier models with panel data: a comparison of parametric, non-parametric and semi-parametric methods with bootstrapping, CORE Discussion Paper No. 9126 (Center for Operations Research and Econometrics, Belgium, 1991).

[83] Sperry Univac Corporation, *Functional Mathematical Programming System Programmer Reference* (Sperry Univac Corporation, St. Paul, MN, 1974).

[84] R.G. Thompson, L.N. Langemeier, C. Lee, E. Lee and R. M. Thrall, The role of multiplier bounds in efficiency analysis with application to Kansas farming, Journal of Econometrics 46(1990)93–108.

[85] R.G. Thompson, F.D. Singleton, Jr., R.M. Thrall and B.A. Smith, Comparative site evaluations for locating a high-energy physics lab in Texas, Interfaces 16(1986)35–49.

[86] C.P. Timmer, Using a probabilistic frontier production function to measure technical efficiency, Journal of Political Economy 79(1971)776–794.