Computing the Convex Frontier for Large-Scale Data Envelopment Analysis

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2025

Qianwei ZHUANG

This page intentionally left blank.

Abstract

Data Envelopment Analysis (DEA) is a widely used analytical tool that leverages linear programming (LP) to classify or benchmark decision-making units (DMUs) by establishing a convex frontier. Each DMU is represented as a point in $\mathbb{R}^d$, where d denotes the dimensionality. In DEA, lower values are preferred for input dimensions, while higher values are favored for output dimensions. The DEA frontier is defined as the oriented boundary of a polytope formed by extreme points.

Effective identification of this frontier is critical, particularly in large-scale contexts where the number of DMUs (n) is substantial. However, this process is computationally expensive, requiring the solution of numerous LP problems, each involving a large number of variables, which leads to significant time and memory challenges.

This study proposes methods to improve the computational performance of DEA frontier identification. First, we introduce a sequential categorization scheme that organizes extreme DEA points into initial blocks, enabling the early establishment of the frontier without processing the entire dataset. Next, we propose a novel reference set selection technique that minimizes the number of decision variables in the LP problems to the size of the dimension, thereby substantially reducing computational costs. Finally, we present an adaptive search and accumulation strategy that minimizes dependency on LP solutions, avoiding redundant computations. Regarding the number of dimension-sized LPs to be solved, these approaches collectively achieve a sublinear time complexity, ensuring computational efficiency superior to $O(n)$.

Rigorous proofs are provided for each proposed method. Experimental results demonstrate the scalability and robustness of the techniques, yielding substantial reductions in computational time and memory requirements while maintaining accurate frontier identification.

Keywords: Convex analysis; Algorithmic geometry; High-performance computing; Large-scale linear programming

This page intentionally left blank.

List of Publications

Journal Papers

1. Zhuang Q, Morita H. Accelerating the Identification of Efficient Frontier for Large-Scale Data Envelopment Analysis. JORSJ Vol. 68, No. 2.
2. Zhuang Q, Khezrimotlagh D, Morita H. Accelerating Large-Scale DEA Computation using Sequential Categorization and Dynamic Reference Set Selection. INFOR: Information Systems and Operational Research. 2024 Sep 19.

Conferences

1. Presentation: ”Reducing LPs and Variables in Large-Scale DEA Computation Through an Adaptive Shortest-Distance Search Scheme” at the INFORMS Annual Meeting, October 2024, US.
2. Presentation: ”Accelerating Large-Scale Data Envelopment Analysis with a Partition and Sequence Computation Scheme” at the International Conference of DEA45, September 2023, UK.
3. Presentation: “Computing the convex hull in high dimensions by solving small-size quadratic optimization problems”, Kansai Branch Young Researchers’ Workshop, Operations Research Society of Japan.
4. Presentation: “Identifying Inefficient DMUs Without Solving Linear Programming for Large-Scale DEA Computation” at the 2024 Autumn Research Presentation Meeting of the Operations Research Society of Japan.
5. Presentation: ”Streamlined DEA Computation Methods for Large-Scale DEA Problems” at the 98th Research Meeting of the Japan OR Society Evaluation OR Research Group.
6. Presentation: ”An Approximate Method for Large-Scale DEA Problems” at the 2022 Autumn Research Presentation Meeting of the Operations Research Society of Japan.
7. Presentation: ”Streamlined DEA Computation in the Big Data Context” at the 2022 Spring Research Presentation Meeting of the Operations Research Society of Japan.

This page intentionally left blank.

Contents

List of Figures

List of Tables

List of Algorithms

This page intentionally left blank.

Chapter 1

Introduction

Data Envelopment Analysis (DEA), introduced by Charnes et al. [6], is a widely applied data-driven analytical tool for the evaluation of decision-making units (DMUs), with multiple inputs (m) and outputs (s) [14]. Charnes et al. [6]'s foundational study established linear programming (LP) formulations for DEA and proposed a computational method for processing DEA data. In the standard procedure, one LP is formulated and solved for each DMU. For a dataset consisting of n DMUs, the LPs must be solved n times, with each LP containing decision variables necessary to classify all DMUs.

Each DMU is represented as a point in $\mathbb{R}^d$, where the dimension d is defined as $m + s$. The classification and evaluation of a DMU are based on its geometric position within $\mathbb{R}^d$ relative to the oriented boundary of a d-dimensional polyhedral region, commonly referred to as the DEA frontier. The DEA frontier is constructed through constrained linear operations on the data and tightly envelops the region. It is characterized by points that exhibit extreme properties of weighted dimensions, adhering to the principle of minimizing inputs while maximizing outputs. In essence, DEA integrates the requirements of dimensions.

This study focuses on identifying the set of extreme points in arbitrary dimensions that define the complete DEA frontier in large-scale contexts, where n is considered large. These extreme points form a subset of the convex hull—the smallest convex set that encloses all points. The convex hull is a foundational concept in mathematics and computational geometry [4, 28, 43, 39, 23], which is extensively utilized across various fields, including machine learning [34, 46], quantum computing [37], computer graphics [40], image processing [25], and computational biology [45, 41]. Notably, the convex hull problem imposes no specific requirements on dimensions, and advanced computational frameworks have been well studied [2].

The rapid advancement of information technology has facilitated access to large-scale datasets across various fields, where dimensional requirements are often specified. However, this convenience introduces new challenges, particularly when applying convex analysis through DEA, which increasingly demands high-speed computation to meet the requirements of certain scenarios.

One such example is the emerging *streaming data* environment [16]. Consider a practical real-time classification model designed to detect credit card fraud. A fraudster in possession of an invalid card aims to maximize their benefit while minimizing the risk of detection. They are aware that exceeding certain thresholds in specific transaction dimensions could reveal fraudulent activities. For instance, an unusually large transaction might trigger a red flag, prompting the credit card company to investigate potential fraud. Other indicators include transaction frequency, types of items purchased, and the relationship between transaction times and physical distances. A sophisticated fraudster, understanding these risks, would likely stay within safe boundaries for each dimension while still attempting to maximize their gain. Consequently, suspicious behavior may arise from extreme combinations of these dimensions.

As charges are processed in real-time, large-scale data streams into the credit card company at high speeds. Each new transaction corresponds to a data point within a polyhedral hull defined by all previous transactions. In Dulá [18], it was shown that the boundary of these hulls is equivalent to a variable-returns-to-scale (VRS) DEA frontier. Transactions deep within the hull are less likely to be flagged, while those near or on the frontier, or those that redefine it, are more likely to raise suspicion. To detect potential fraud, the company must rapidly verify transactions by identifying and updating the frontier in real-time as new data arrives. The key challenge is processing these transactions quickly enough to ensure timely detection of suspicious activities, making the expedited identification of the VRS boundary essential.

In addition to credit card fraud detection, DEA analysis is widely used across various fields that generate large-scale datasets, including transportation and logistics, energy planning, healthcare and medicine, as well as retail and e-commerce. For example, the evaluation of high-frequency data from market activities—such as stock markets [30, 31], e-commerce transactions [7, 15], and livestreaming commerce [8]—remains an active research area. These evaluations often require analyzing large numbers of DMUs. Furthermore, organizations have accumulated vast amounts of time series data over the years, generating significant scholarly interest in analyzing these datasets [44, 47]. Moreover, the challenges posed by big data have led to the adoption and further development of advanced DEA models, such as network DEA [24, 42], stochastic DEA [36], and slacks-based DEA [32], among others.

Nevertheless, as the number of DMUs n increases, applying DEA analysis becomes computationally intensive and time-consuming. As a result, addressing the computational challenges of DEA in large-scale contexts has received significant interest from scholars over the years.

1.1 Literature Review

As noted, DEA analysis relies on its established frontier, which is determined by extreme points. The process of identifying these extreme points is commonly referred to as the *frame* problem [20, 19], where the frame is the minimal subset of DEA points that collectively define the DEA frontier.

Ali [1] pioneered a method for detecting the DEA frontier, laying a foundational framework for subsequent advancements. Building on this, Barr et al. [5] introduced a parallel processing approach that significantly improved the computational performance of Ali's method, enabling faster processing of large datasets. Dulá et al. [21] further contributed by developing algorithms that utilize LP solutions to identify the DEA frame, which defines the DEA frontier. While these methods rely on solving LPs, the process can still be computationally intensive for large-scale contexts. To address this limitation, Dulá et al. [22] proposed preprocessing techniques based solely on arithmetic operations such as sorting, inner products, and translation. These preprocessors streamline computations without the need to solve LPs.

Building on earlier investigations, Dulá [19] introduced the Build Hull (BH) model, a pivotal advancement in large-scale DEA methodology. The BH model constructs the DEA frontier iteratively using only DMUs contributing to the construction of the DEA frame, making it highly practical and resource-conserving. Extending this foundation, Khezrimotlagh et al. [27] enhanced the Barr et al. [5] model by incorporating concepts from the BH model, further improving its performance and scalability. Similarly, Jie [26] refined the BH model by integrating a parallel computation scheme. This approach leverages parallel processing, achieving faster computation times as the number of CPU workers (computation cores) increases.

While models like BH focus on constructing the entire DEA frame, alternative methods have emerged that apply DEA without the need to construct the complete DEA frame. Chen et al. [9] proposed a small-size LP model based on the Karush-Kuhn-Tucker (KKT) conditions [29]. This method is particularly useful in scenarios with a high density of frontier units, as it significantly reduces computational demands by focusing only on decision variables associated with relevant DMUs. Subsequent advancements in this direction were made by Chen et al. [10] and Chu et al. [12]. These innovations have become essential in DEA analysis, offering streamlined solutions for analyzing massive datasets.

In addition to these purely algorithmic approaches, recent advancements have integrated machine learning techniques. For instance, Nemirko et al. [35] employed machine learning algorithms to classify points relative to the convex hull of a polyhedron, addressing the classification problem of identifying interior, boundary, or exterior points. Their method, implemented as a parallel algorithm, also calculates the distance of target points from the convex hull, effectively solving LP problems in parallel. Similarly, Muren et al. [33] proposed the angle-index synthesis method, which combines angular and indexed measures for prescoring DMUs. They used Monte Carlo simulations to assess their algorithm's performance on an exceptionally large dataset, consisting of one billion DMUs. These approaches underscore the increasing convergence of DEA methodologies with machine learning and simulation techniques, opening the door to innovative solutions for large-scale challenges in DEA analysis.

1.2 Structure of This Study

Dulá [19]'s method is widely recognized as the preferred approach in situations with limited computational resources. Building on this foundation, the objective of this study is to accelerate the identification of the DEA frame in such resource-constrained environments. Drawing inspiration from the techniques of Chen et al. [10] and Chu et al. [12], we introduce a novel shortest-distance-based reference set selection method to reduce the number of variables in the LPs of the BH procedure. This enables the identification of the DEA frame by solving LPs of minimal size (approximately equal to the dimension size). Building on this technique, we propose an enhanced version of the BH procedure, termed the small-size BH. To further improve computational performance, we introduce novel DEA preprocessing strategies that use arithmetic-based data preprocessing to eliminate the need for solving LPs during frame construction. Notably, our approach is purely algorithmic and does not rely on machine learning or statistical learning methods. We also provide theoretical proofs to support the proposed techniques.

The remainder of this study is organized as follows:

- **Chapter 2** introduces the notations, terminology, and key assumptions in DEA, with a specific emphasis on the VRS assumption. It also discusses the application of DEA analysis and the computational challenges encountered in large-scale contexts.

- **Chapter 3** emphasizes the importance of pre-identifying DEA frame points and reviews existing approaches for this task. The BH procedure is presented in detail.

- **Chapter 4** introduces a sequential categorization method to establish a systematic evaluation order for the BH procedure, potentially eliminating the need for LP solutions.

- **Chapter 5** presents a novel reference set selection method that enables a small-size BH procedure, further enhancing the VRS frame construction.
- **Chapter 6** extends the work from the previous chapter by further reducing both the size and number of LPs required for constructing the VRS frame, employing an adaptive method to generate virtual points for identifying non-dominated, non-frame points.
- **Chapter 7** concludes the study by summarizing the key findings and contributions. It also discusses potential directions for future research, offering insights into possible improvements, extensions, and adaptations of the methods introduced to advance DEA in large-scale contexts.

The techniques developed in this study are also applicable to the convex hull problem, although some adaptations may be required.

Chapter 2

Foundation Knowledge

This chapter begins by introducing the notation and terminology used throughout the study. It then provides an introduction of the convex hull problem, followed by a discussion of the convex frontier in DEA and several commonly used definitions in the field. The focus then shifts to the critical DEA assumption of variable returns-to-scale (VRS), with an illustration of the computational challenges involved in applying it to large-scale contexts.

2.1 Notation and Assumptions

The set $\mathbb{R}$ represents all real numbers, and $\mathbb{R}^d$ denotes the space of d-dimensional real-valued vectors, where $d \in \mathbb{N}_0$, the set of non-negative integers.

A DEA dataset comprises n decision-making units (DMUs), indexed by $j = 1, \ldots, n$. Each DMU j is described by d dimensions, $d \in \mathbb{N}_0 \setminus \{0\}$, where $d = m + s$, including:

- m-inputs, represented as $\boldsymbol{x}_j = (x_{1j}, \ldots, x_{mj})^T \in \mathbb{R}^m$,
- s-outputs, represented as $\boldsymbol{y}_j = (y_{1j}, \ldots, y_{sj})^T \in \mathbb{R}^s$,

where both inputs and outputs satisfy $\boldsymbol{x}_j \neq \boldsymbol{0}$, $\boldsymbol{x}_j \geq \boldsymbol{0}$, $\boldsymbol{y}_j \neq \boldsymbol{0}$, and $\boldsymbol{y}_j \geq \boldsymbol{0}$. In the DEA context, smaller values are more desirable for input dimensions, while larger values are more desirable for output dimensions, in accordance with practical requirements.

To align the requirements across dimensions, we define the DEA point for DMU j as:

$$\boldsymbol{a}_j = \begin{pmatrix} -\boldsymbol{x}_j \\ \boldsymbol{y}_j \end{pmatrix} \in \mathbb{R}^d, \quad j = 1, \ldots, n,$$

where the i-th coordinate of $\boldsymbol{a}_j$ is denoted as a_{ij}, $i = 1, \ldots, d$. This formulation ensures that all dimensions adhere to a *smaller is better* principle. The magnitude of $\boldsymbol{a}_j$ is quantified using the Euclidean norm:

$$\|\boldsymbol{a}_j\| = \sqrt{\sum_{i=1}^{d} a_{ij}^2}.$$

Let matrices

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{pmatrix} = \begin{pmatrix} X_1, \ldots, X_m \end{pmatrix} \text{ and } \boldsymbol{Y} = \begin{pmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_n^T \end{pmatrix} = \begin{pmatrix} Y_1, \ldots, Y_s \end{pmatrix}.$$

denote the input data and output data of all DMUs, respectively.

Let $\mathcal{A}$ denote the entire set of DEA points, $\mathcal{A} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$, where $|\mathcal{A}|$ is the number of elements in $\mathcal{A}$. Let $\mathcal{I}_{\mathcal{A}} = \{1, \ldots, n\}$ be the corresponding index set to the corresponding elements in $\mathcal{A}$. Let $\mathcal{S}$ be a subset of $\mathcal{A}$, such that $\mathcal{S} \subseteq \mathcal{A}$. $\mathcal{I}_{\mathcal{S}} \cup \{j\}$ is equivalent to $\mathcal{S} \cup \{\boldsymbol{a}_j\}$.

The DEA data are assumed to be *reduced*, which means that for any two distinct points $j_1 \neq j_2$, the relationship $\boldsymbol{a}_{j_1} \neq k \cdot \boldsymbol{a}_{j_2}$ holds for all $k \in \mathbb{R}$. In other words, there are no duplicate points in $\mathcal{A}$, and no two points are scalar multiples of each other.

The inner product of two vectors $\boldsymbol{\eta}$ and $\boldsymbol{z}$ of the same dimension is denoted by $\langle \boldsymbol{\eta}, \boldsymbol{z} \rangle$. The set $\mathcal{H}(\boldsymbol{\eta}, \xi) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathbb{R}^d, \langle \boldsymbol{\eta}, \boldsymbol{z} \rangle + \xi = 0\}$ corresponds to a hyperplane in $\mathbb{R}^d$ with an orthogonal vector $\boldsymbol{\eta}$ and a level value of $-\xi$. This hyperplane separates two subsets (corresponding to two half-spaces) of interest:

- $\mathcal{H}^{++}(\boldsymbol{\eta}, \xi) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathbb{R}^d, \langle \boldsymbol{\eta}, \boldsymbol{z} \rangle + \xi > 0\}$,
- $\mathcal{H}^{-}(\boldsymbol{\eta}, \xi) = \{\boldsymbol{z} \mid \boldsymbol{z} \in \mathbb{R}^d, \langle \boldsymbol{\eta}, \boldsymbol{z} \rangle + \xi \leq 0\}$.

We define $\mathcal{A}^{++}_{(\boldsymbol{\eta},\xi)} = \{\boldsymbol{a}_j \in \mathcal{A} \mid \boldsymbol{a}_j \in \mathcal{H}^{++}(\boldsymbol{\eta}, \xi)\}$ and $\mathcal{A}^{-}_{(\boldsymbol{\eta},\xi)} = \{\boldsymbol{a}_j \in \mathcal{A} \mid \boldsymbol{a}_j \in \mathcal{H}^{-}(\boldsymbol{\eta}, \xi)\}$.

This study is closely related to the convex hull problem in computational geometry. To establish a foundational understanding for this research, we start by introducing the convex hull problem.

2.2 The convex hull problem

Identifying the vertices of the polytope in $\mathbb{R}^d$ that define the convex hull is a fundamental and intriguing computational problem. These vertices correspond to the extreme points of the set.

Definition 2.1 (Extreme Points)**.** Given a finite set $\mathcal{A} = \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ of n unique points in $\mathbb{R}^d$, the point $\boldsymbol{a}_j$ is an extreme point of $\mathcal{A}$ if it cannot be represented as a convex combination of the points from the set $\mathcal{A} \setminus \boldsymbol{a}_j$. [38]

The extreme points represent the convex hull in the sense that if $\mathcal{E}$ is the set of extreme points of $\mathcal{A}$, then Conv$\mathcal{E}$ = Conv$\mathcal{A}$.

A key basic operation that is used to define and compute the convex hull is to determine the Euclidean distance of a point $\boldsymbol{z} \in \mathbb{R}^d$ to Conv$\mathcal{A}$. This function is represented as $d(\boldsymbol{z}, \mathcal{A})$, and its square can be computed via the quadratic program (QP) represented by Model 2.1:

$$\begin{aligned} d(\boldsymbol{z}, \mathcal{A})^2 = \min_{\lambda_j \geq 0} & \left\| \boldsymbol{z} - \sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j \boldsymbol{a}_j \right\|^2 \\ \text{s.t.} \quad & \sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j = 1 \end{aligned} \tag{2.1}$$

Note that $\boldsymbol{z} \in$ Conv$\mathcal{A}$ if and only if $d(\boldsymbol{z}, \mathcal{A}) = 0$. It is clear that if we have available the set $\mathcal{E}$, then $d(\boldsymbol{z}, \mathcal{A}) = d(\boldsymbol{z}, \mathcal{E})$, and one can use $\mathcal{E}$ for the calculation instead. We note that the extreme points are the smallest subset $\mathcal{E} \subseteq \mathcal{A}$ such that $d(\boldsymbol{z}, \mathcal{A}) = d(\boldsymbol{z}, \mathcal{E})$ for all $\boldsymbol{z} \in \mathbb{R}^d$. An intuitive understanding of the convex hull formed by the extreme points can be gained from Figure 2.1a, where the extreme points define the piecewise boundary enclosing all the points in $\mathcal{A}$.

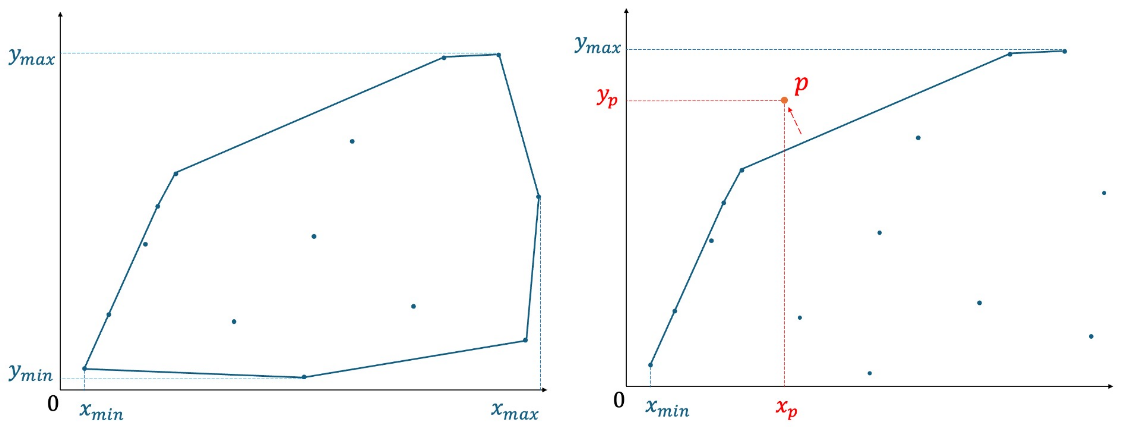
The fact that the extreme points are the smallest subset has a very practical importance in applications that use the convex hull as input, and require computations to be performed on each element of $\mathcal{E}$.

In the convex hull problem, there is no specific requirement for each dimension, and the entire polyhedral boundary is constructed. However, in certain tasks where prior knowledge about the dimensions exists, it becomes necessary to identify an oriented polyhedral boundary rather than the complete boundary. In this study, we focus on identifying this oriented boundary, which corresponds to a DEA frontier, as illustrated in Figure 2.1b.

2.3 The DEA frontier

In DEA, a convex frontier of $\mathcal{A}$ is formulated by the set of extreme points in $\mathbb{R}^d$ such that the relationship between inputs and outputs is theoretically admissible in the context of the economic assumptions [6]. A DEA dataset defines four standard convex frontiers: constant (CRS), variable (VRS), increasing (IRS), and decreasing returns-to-scale (DRS).

For each returns-to-scale assumption, any subset $\mathcal{S}$ generates the following four



(a) (b)

Figure 2.1: Convex Boundary

envelopment hulls:

$$\gamma_{\mathcal{S}}^{CRS} = \left\{ \boldsymbol{z} \;\middle|\; \boldsymbol{z} \in \mathbb{R}^d,\ \boldsymbol{z} \leq \sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j \boldsymbol{a}_j,\ \text{s.t. } \lambda_j \geq 0,\ \forall j \in \mathcal{I}_{\mathcal{S}} \right\},$$

$$\gamma_{\mathcal{S}}^{\{VRS,IRS,DRS\}} = \left\{ \boldsymbol{z} \;\middle|\; \boldsymbol{z} \in \mathbb{R}^d,\ \boldsymbol{z} \leq \sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j \boldsymbol{a}_j,\ \text{s.t.} \quad \begin{aligned} \lambda_j \geq 0,\ \forall j \in \mathcal{I}_{\mathcal{S}},& \\ \sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j = 1 \quad &(\text{for VRS}), \\ \sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j \leq 1 \quad &(\text{for IRS}), \\ \sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j \geq 1 \quad &(\text{for DRS}) \end{aligned} \right\}.$$

Following Dulá [19], we present the following definitions:

Definition 2.2 (DEA Hulls)**.** We define the four polyhedral sets associated with $\mathcal{S}$ as the *DEA hulls* of the points in $\mathcal{S}$. The points in $\mathcal{S}$ are referred to as the *generators* of the hulls.

The polyhedral set $\gamma_{\mathcal{S}}^{CRS}$ is called the CRS hull of $\mathcal{S}$, and similarly for the other types. The set $\gamma_{\mathcal{S}}^{CRS}$ is a cone, while the other three sets are more intricate. They usually consist of multiple extreme points, and their recession cone includes all d negative unit directions (i.e., their "slacks"). This indicates that these sets have full dimension. Finally, these sets are convex.

When $\mathcal{S} = \mathcal{A}$, these DEA hulls are referred to as *full* DEA hulls, corresponding to the familiar CRS, VRS, IRS, and DRS frontier in DEA. If $\mathcal{S} \subset \mathcal{A}$ (a strict subset) and the DEA hull is not a full DEA hull, it is called a *partial* DEA hull. It is important to note that if $\mathcal{S} \subset \mathcal{A}$, then for any of the four return-to-scale types, we have:

$$\gamma_{\mathcal{S}} \subseteq \gamma_{\mathcal{A}}.$$

In all four cases, the shape of the DEA hulls is entirely determined by the data points. Since these are finitely generated polyhedral sets, their minimal description requires identifying their extreme elements.

Definition 2.3 (DEA Frames)**.** We define a *DEA frame* $\mathcal{F}$ as the smallest subset of $\mathcal{A}$ necessary to describe a full DEA hull (or frontier). $\gamma_{\mathcal{F}} = \gamma_{\mathcal{A}}$.

It is evident that $\mathcal{F} \subseteq \mathcal{E}$. The extreme points defining the DEA frontier are a subset of the extreme points that define the convex hull.

Similarly, if $\mathcal{S} \subset \mathcal{F}$ and the frame is not a full DEA frame, it is called a *partial* DEA frame. For any of the four return-to-scale types, we also have:

$$\gamma_{\mathcal{S}} \subseteq \gamma_{\mathcal{F}}, \text{ where } \mathcal{S} \subset \mathcal{A}.$$

A DEA dataset can have up to four distinct frames, one for each returns-to-scale type. These correspond to extreme rays in the CRS case and extreme points in the VRS, IRS, and DRS cases. Further details of these results can be found in Dulá et al. [17].

The DEA frames used to construct the frontier are crucial for tasks such as classification, benchmarking, and related analyses. For example, the VRS frame represents the partial piece-wise convex boundary, as presented in Figure 2.1b. It enables the assessment of whether a streamed-in DEA point lies outside the VRS hull. Point p, where $p \notin \mathcal{I}_{\mathcal{A}}$, is located in the upper-left region of this oriented boundary, and $\boldsymbol{a}_p \notin \gamma_{\mathcal{F}}^{VRS}$.

2.3.1 The Role of VRS Frame

Among the four DEA assumptions, identifying a VRS frame provides concrete advantages:

1. The CRS frame is a subset of the VRS frame, and the VRS frame is the union of the IRS and DRS frames. This relationship allows the CRS, IRS, and DRS frames to be extracted directly from the VRS frame without the need to consider the entire dataset.

2. The VRS frame elements correspond to the extreme points. Knowing the VRS

frame ensures a complete classification of DMUs, as non-extreme boundary points are rare.

3. A VRS frame is independent of the DEA orientation or slack-based measures, providing greater flexibility in exploring different DEA LP formulations. The same frame can be used across multiple formulations. Computational results have demonstrated the practical benefits of this frame-based approach.

In the later sections, any reference to the DEA frame specifically refers to the VRS frame, which is the primary focus of this study.

2.3.2 Classification using the DEA Frontier

Let $\boldsymbol{u}$ be the vector of all ones, with its dimension determined by the context. Dulá [19] developed Model 2.2 and its dual Model 2.3 which can be used for multiple purposes such as classification. Indeed, the optimal result of Model 2.3 can be directly obtained when Model 2.2 is solved.

Assuming a reference set $\mathcal{R}$, where $\mathcal{R} \subseteq \mathcal{A}$, for a point $\boldsymbol{a}_l \notin \mathcal{R}$, the following definition applies:

Definition 2.4 (Exterior and Interior)**.** A point $\boldsymbol{a}_l$ is classified as an *exterior* with respect to the VRS hull of $\mathcal{R}$ if $\alpha_l^* > 0$. Otherwise, it is referred to as an *interior* when $\alpha_l^* = 0$.

Points on DEA frame are exclusively found among exteriors. This is because, when $\alpha_l^* = 0$, the inequality constraint of Model 2.2 implies $\sum_{j \in \mathcal{I}_\mathcal{R}} \lambda_{lj}^* \boldsymbol{a}_j \geq \boldsymbol{a}_l$. Thus, $\boldsymbol{a}_l \in \gamma_\mathcal{F}^{VRS}$, meaning that $\boldsymbol{a}_l$ does not contribute to the construction of the VRS frame of $\mathcal{R}$, and hence not to $\mathcal{F}$. Conversely, if $\alpha_l^* > 0$, we have $\sum_{j \in \mathcal{I}_\mathcal{R}} \lambda_{lj}^* \boldsymbol{a}_j < \boldsymbol{a}_l$.

$$\begin{aligned} \alpha_l = &\min_{\alpha, \lambda_{lj} \geq 0} \alpha \\ \text{s.t.} & \\ & \sum_{j \in \mathcal{I}_\mathcal{R}} \lambda_{lj} \boldsymbol{a}_j + \boldsymbol{u}\alpha \geq \boldsymbol{a}_l, \\ & \sum_{j \in \mathcal{I}_\mathcal{R}} \lambda_{lj} = 1. \end{aligned} \tag{2.2}$$

Let $\boldsymbol{\pi}_l^*, \xi_l^*$ represent the optimal solution to Model 2.3. The corresponding non-trivial hyperplane $\mathcal{H}(\boldsymbol{\pi}_l^*, \xi_l^*)$ separates the VRS hull from an exterior $\boldsymbol{a}_l$.

- If $\alpha_l^* > 0$, $\boldsymbol{a}_l$ resides in the half-space defined by $\mathcal{H}^{++}(\boldsymbol{\pi}_l^*, \xi_l^*)$.

- If $\alpha_l^* = 0$, $\boldsymbol{a}_l$ lies within the half-space defined by $\mathcal{H}^{-}(\boldsymbol{\pi}_l^*, \xi_l^*)$.

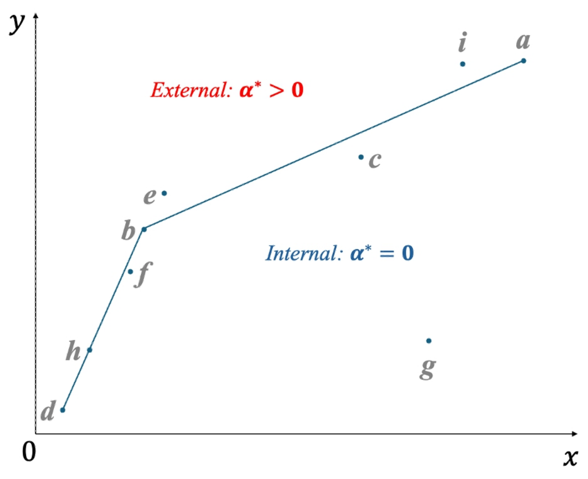


Figure 2.2: Exterior and Interior

We present a 1+1 dimensional DEA dataset with 9 DMUs, as shown in Table 2.1, for illustration purpose in the next sections.

Table 2.1: A 1+1 Dimension Instance Sample

DMU	a	b	c	d	e	f	g	h	i
x	72	16	48	4	19	14	58	8	63
y	124	68	92	8	80	54	31	28	123

An intuitive illustration of interiors and exteriors is shown in Figure 2.2. Assuming $\mathcal{R} = \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_c, \boldsymbol{a}_d, \boldsymbol{a}_f, \boldsymbol{a}_h\}$. When e, i, and g streams come in, we conclude that e and i are exterior points, located in the upper-left region relative to the frame of $\mathcal{R}$, while g is an interior point. The classification of a point as exterior or interior is actually determined by the VRS frontier of $\mathcal{R}$, which is defined by its frame a, b, and d.

$$\begin{aligned} \alpha_l = & \max_{\boldsymbol{\pi}_l \geq 0, \xi_l} \langle \boldsymbol{\pi}_l, \boldsymbol{a}_l \rangle + \xi_l \\ \text{s.t.} & \\ & \langle \boldsymbol{\pi}_l, \boldsymbol{a}_j \rangle + \xi_l \leq 0, \quad \forall j \in \mathcal{I}_{\mathcal{R}}, \\ & \langle \boldsymbol{\pi}_l, \boldsymbol{u} \rangle \leq 1. \end{aligned} \tag{2.3}$$

2.3.3 Benchmarking using the DEA Frontier

Frontier-based benchmarking is a critical topic in DEA. Consider the example in Table 2.1, it is clear the full frame is composed by the points a, e, b, i, and d. The input-oriented radial efficiency θ_l for DMU l is determined by solving the LP represented by Model (2.4), with the reference set $\mathcal{R}$ conventionally defined as the entire set of DMUs $\mathcal{A}$ (i.e., $\mathcal{R} \leftarrow \mathcal{A}$):

$$\begin{aligned} \theta_l = & \min_{\lambda \geq 0} \theta \\ \text{s.t.} & \\ & \sum_{j \in \mathcal{I}_\mathcal{R}} \lambda_{lj} \boldsymbol{x}_j \leq \theta \boldsymbol{x}_l, \\ & \sum_{j \in \mathcal{I}_\mathcal{R}} \lambda_{lj} \boldsymbol{y}_j \geq \boldsymbol{y}_l, \\ & \sum_{j \in \mathcal{I}_\mathcal{R}} \lambda_{lj} = 1. \end{aligned} \tag{2.4}$$

The optimal result indicates that DMU l is either *efficient* ($\theta_l^* = 1$) or *inefficient* ($\theta_l^* < 1$). The set of efficient DMUs, that is, the best-practices from an input-output perspective, denoted as $\mathcal{B}$, is a subset of $\mathcal{A}$, $\mathcal{B} \subseteq \mathcal{A}$. It is clear that $\gamma_\mathcal{F} = \gamma_\mathcal{B} = \gamma_\mathcal{A}$. Notably, Model (2.4) yields the same result when $\mathcal{F}$ or $\mathcal{B}$ is used as the reference set instead of $\mathcal{A}$ [1], i.e., $\mathcal{R} \leftarrow \mathcal{F}$ or $\mathcal{R} \leftarrow \mathcal{B}$. This means that we can use only the DEA frame as the benchmark for evaluating each DMU l in Model (2.4). Below, we provide an illustration demonstrating how the VRS frame serves as the benchmark for the evaluation of DMUs in DEA.

In the provided two-dimensional example, the piece-wise efficient frontier is formed by the segments connecting $d \rightarrow b$, $b \rightarrow e$, $e \rightarrow i$, and $i \rightarrow a$, as shown in Figure 2.3. DMUs c, f, and g are inefficient. Using Model 2.4, each DMU is matched to the facet that is closest to itself, and their efficiency scores are calculated based on this relevant facet. For example, DMU c is matched to the facet $e \rightarrow i$ (its benchmark facet), which has the shortest distance to it, resulting in an efficiency score of $\theta_c^* = 0.65$ using Model 2.4. According to the input-oriented projection assumption, this indicates that DMU c could become efficient by reducing its input usage to 65% of its current level while maintaining the same output.

From Figure 2.3, we see intuitively that efficient DMUs are those located on the upper-left boundary of the distribution of the DEA points. They exhibit extreme features in either input, output, or weighted combinations of input and output. This observation holds in dimensions higher than this two-dimensional example.

2.4 Computation Challenges

In this study, all results are derived under the VRS assumption [3]. The VRS assumption generally leads to a higher number of extreme points compared to other

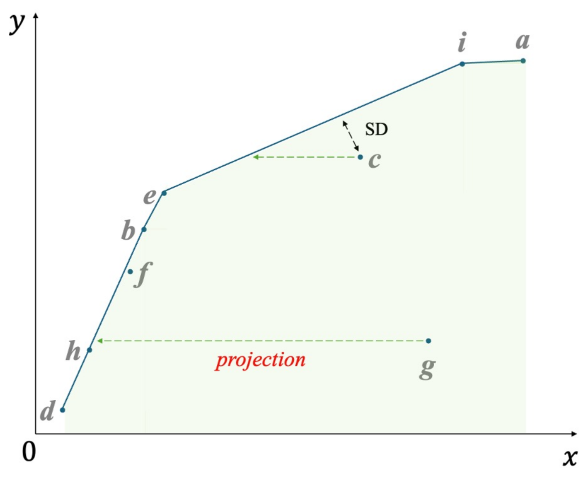


Figure 2.3: Shortest-Distance (SD) Matching

scale assumptions. The proposed approaches can be extended to the other three assumptions, although specific adaptations would be required.

As the size of the reference set $|\mathcal{R}|$ increases, the number of associated variables in the models grows correspondingly, resulting in a sharp rise in computational costs. This escalation is reflected in both increased time consumption and higher memory requirements for running DEA models with a larger number of variables.

However, since only the extreme points of the frame are relevant for classification or benchmarking purpose in DEA, as previously mentioned, we can potentially reduce the computational burden by eliminating variables associated with non-frame points. Therefore, it is advantageous to identify the DEA frame before solving any of the DEA models, i.e., Model 2.4, particularly for large-scale samples where n is substantial.

Summary

This chapter lays the foundation for understanding the key concepts and assumptions underpinning DEA, specifically within the context of datasets comprising n DMUs, each described by d dimensions, which is composed of m inputs and s outputs. A thorough introduction is provided to DEA assumptions: CRS, VRS, IRS and DRS. Additionally, the discussion introduces key components such as DEA hulls and frames, emphasizing their crucial role in simplifying computational processes and making large-scale DEA analysis more feasible and efficient.

The VRS assumption is specifically adopted in this study for its flexibility, as it accommodates a broader spectrum of DEA formulations and applications. Furthermore, the study utilizes Model 2.2 and Model 2.3 in varying capacities across the proposed computational frameworks. These models are strategically applied in different analytical contexts, showcasing their adaptability and robustness in solving diverse problems within DEA. This chapter sets the stage for a deeper exploration of the computational techniques developed throughout this study.

Chapter 3

Advances in Large-Scale DEA

As discussed, since only the DEA frame is essential for benchmarking in DEA, identifying the boundary points that define the DEA frame prior to applying Model 2.4 is highly beneficial. This is particularly critical in large-scale scenarios where the number of DMUs, n, is substantial, as pre-identification helps reducing computational burden and streamlines the analysis process.

In this chapter, we provide an detailed overview of the major existing methods developed for this purpose, highlighting their theoretical foundations, practical implementation, and relative strengths in addressing the challenges posed by large-scale datasets.

3.1 Existing Results

3.1.1 Discarding Non-frame Points

Before DEA computation, eliminating non-frame points from $\mathcal{A}$ helps minimize the set of candidate points that establishes DEA frame, $\mathcal{F}$. DEA Dominator [22] is one such method that directly uses dimension comparison to discard non-frame points, without the need of LP solutions.

For illustration purposes, we refer to the following definitions and conclusions [13, 22, 27]:

Definition 3.1 (Domination)**.** Point j dominates point l if and only if: $\boldsymbol{a}_j \geq \boldsymbol{a}_l$.

Definition 3.2 (Frame Points)**.** Point $l \in \mathcal{I}_\mathcal{F}$ if and only if it is dominated neither by any other existing points nor by any virtual points j', where $\boldsymbol{a}_{j'} = \sum_{j \neq l} \lambda_j \boldsymbol{a}_j$, $\sum_{j \neq l} \lambda_j = 1$, and $\lambda_j \geq 0, \forall j$.

From Definition 3.2, we conclude that point $l \notin \mathcal{I}_\mathcal{F}$ if it is dominated. Furthermore, we can conclude that every point $l \notin \mathcal{I}_\mathcal{F}$ is dominated. This can be easily proved by referring to Model 2.2.

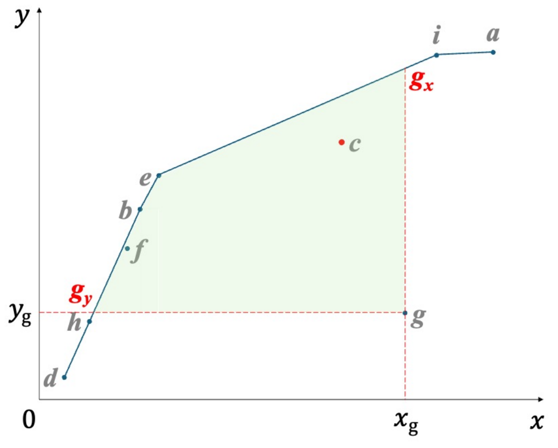
Let the set $\mathcal{D}$ denote the set of points that are dominated. The DEA Dominator can be described by Algorithm 3.1.

Algorithm 3.1 identifies non-frame points using DEA domination. It has the potential to identify a large subset of dominated points. However, it relies solely on existing points to establish DEA domination. For those non-frame points where their relevant dominating point does not exist among the given set of points, it fails to identify them.

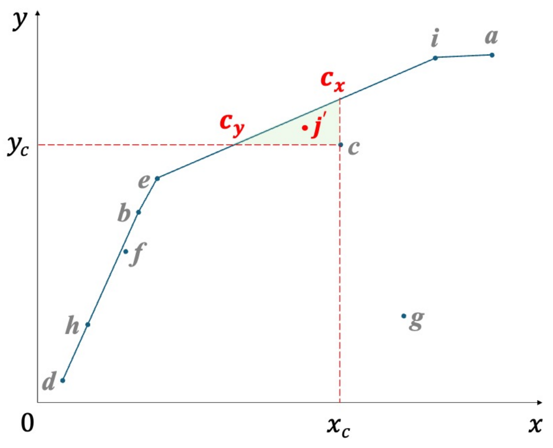
Algorithm 3.1 DEA Dominator

```
1: Initialize $\mathcal{D} \leftarrow \emptyset$
2: for each $l \in \mathcal{I}_\mathcal{A}$ do
3:     for each $j \in \mathcal{I}_\mathcal{A} \setminus \{l\}$ do
4:         if $\boldsymbol{a}_j \geq \boldsymbol{a}_l$ then          ▷ the domination criterion
5:             $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{a}_l\}$
6:             break
7:         end if
8:     end for
9: end for
10: return $\mathcal{D}$
```

For instance, in Figure 3.1a, point g is dominated by existing points f, b, e and any virtual points within the polygonal area of $g_y g g_x e b$. And g can be identified as non-frame point through Dominator. Conversely, in Figure 3.1b, point c is dominated by virtual points j' within the triangular area of $c_y c c_x$. However, there is no existing point that dominates c. As a result, the Dominator method fails to discriminate point c.



(a)



(b)

Figure 3.1: Domination

3.1.2 Identifying Frame Points

Several lemmas are useful for arithmetically detecting frame points by leveraging the extreme features of DMUs' dimensions, without the need to solve an LP. These lemmas are as follows:

Lemma 3.1. If point l is the unique point that satisfies

$$l = \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} a_{ij},$$

then $l \in \mathcal{I}_{\mathcal{F}}$, for $i = 1, \ldots, d$.

Lemma 3.1 [1] can identify at most d frame points by leveraging the extreme features from each dimension. In addition, frame points also exhibit extreme features of the weighted sum of dimensions. Assume an arbitrary weight vector $\boldsymbol{\pi} \in \mathbb{R}^d_{\geq 0}$ and $\boldsymbol{\pi} \neq \mathbf{0}$. We have the following result:

Lemma 3.2. If point l is the unique point that satisfies

$$l = \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}, \boldsymbol{a}_j \rangle,$$

then $l \in \mathcal{I}_{\mathcal{F}}$.

Relevant results using LP optimal solutions (i.e., Model 2.3) are discussed in existing studies [19, 12]. We provide a formal proof involving of arbitrary weights in Appendix A.1 under the DEA model assumptions.

Apart from leveraging the extreme features of dimensions to identify frame points, there are also lemmas based on the *output/input* ratio, which is expected to be higher under a production context.

Lemma 3.3. If point l is the unique point that satisfies

$$l = \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \frac{\sum_{r=1}^{s} y_{rj}}{\sum_{i=1}^{m} x_{ij}},$$

then $l \in \mathcal{I}_{\mathcal{F}}$.

Lemma 3.3 [1] can potentially identify one additional frame point. Assume weight vectors $\boldsymbol{w} = (w_1, \ldots, w_m)^T \in \mathbb{R}^m_{>0}$ and $\boldsymbol{v} = (v_1, \ldots, v_s)^T \in \mathbb{R}^s_{>0}$. Similarly, we can leverage a weighted *output/input* ratio [discussed in 11], as described in Lemma 3.4:

Lemma 3.4. If point l is the unique point that satisfies

$$l = \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \frac{\langle \boldsymbol{v}, \boldsymbol{y}_j \rangle}{\langle \boldsymbol{w}, \boldsymbol{x}_j \rangle},$$

then $l \in \mathcal{I}_{\mathcal{F}}$.

Lemmas 3.2 and 3.4 incorporate arbitrary weights, making it possible to identify as many frame points as needed when appropriate weights are selected. This is particularly helpful for establishing the VRS frame.

3.2 Build Hull

Dulá [19] developed the Build Hull (BH) framework, which constructs the DEA frame $\mathcal{F}$ step by step, retaining only frame points as references in Model 2.2. The BH procedure is described in Algorithm 3.2, where $\mathcal{T}$ represents a temporary workspace array.

Algorithm 3.2 Build Hull

```
1: Initialize $\mathcal{S} \subset \mathcal{F}$                    ▷ using the mentioned lemmas
2: $\mathcal{T} \leftarrow \mathcal{A}$
3: $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{S}$
4: while $\mathcal{T} \neq \emptyset$ do
5:     select $\boldsymbol{a}_l \in \mathcal{T}$
6:     $\mathcal{R} \leftarrow \mathcal{S}$
7:     $\alpha_l^*, \boldsymbol{\pi}_l^* \leftarrow$ solve Model 2.2 and Model 2.3 for $l$
8:     if $\alpha_l^* > 0$ then
9:         $l^* = \arg\max_{j \in \mathcal{I}_{\mathcal{T}}} \langle \boldsymbol{\pi}_l^*, \boldsymbol{a}_j \rangle$    ▷ in case of a tie, set $l^*$ to one extreme point
10:        $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{l^*}\}$
11:        $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\boldsymbol{a}_l, \boldsymbol{a}_{l^*}\}$
12:    else
13:        $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\boldsymbol{a}_l\}$
14:    end if
15: end while
16: $\mathcal{F} \leftarrow \mathcal{S}$                              ▷ the frame is established
17: return $\mathcal{F}$
```

To explain the BH procedure, let us consider the instance dataset presented in Table 2.1, and the possible implementation procedure depicted in Figure 3.2.

Firstly, to apply BH to this example, without loss of generality, assume a subsample $\mathcal{S}$ is initialized by Lemma 3.1 and Lemma 3.3, including frame points a, b, and d. In other words, $a = \arg\max x$, $e = \arg\max y$, and $b = \arg\max \frac{x}{y}$. The partial frame is formed with the subsample $\mathcal{S} = \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d\}$, as shown in Figure 3.2a. Then, the remaining points g, i, e, h, c, and f are evaluated one after another using Model 2.2 and Model 2.3. Note that the selection order of points for evaluation is random.

For example, without loss of generality, assume that g is selected in the first iteration and we obtain $\alpha_g^* = 0$, which indicates point g is an interior. Point i is

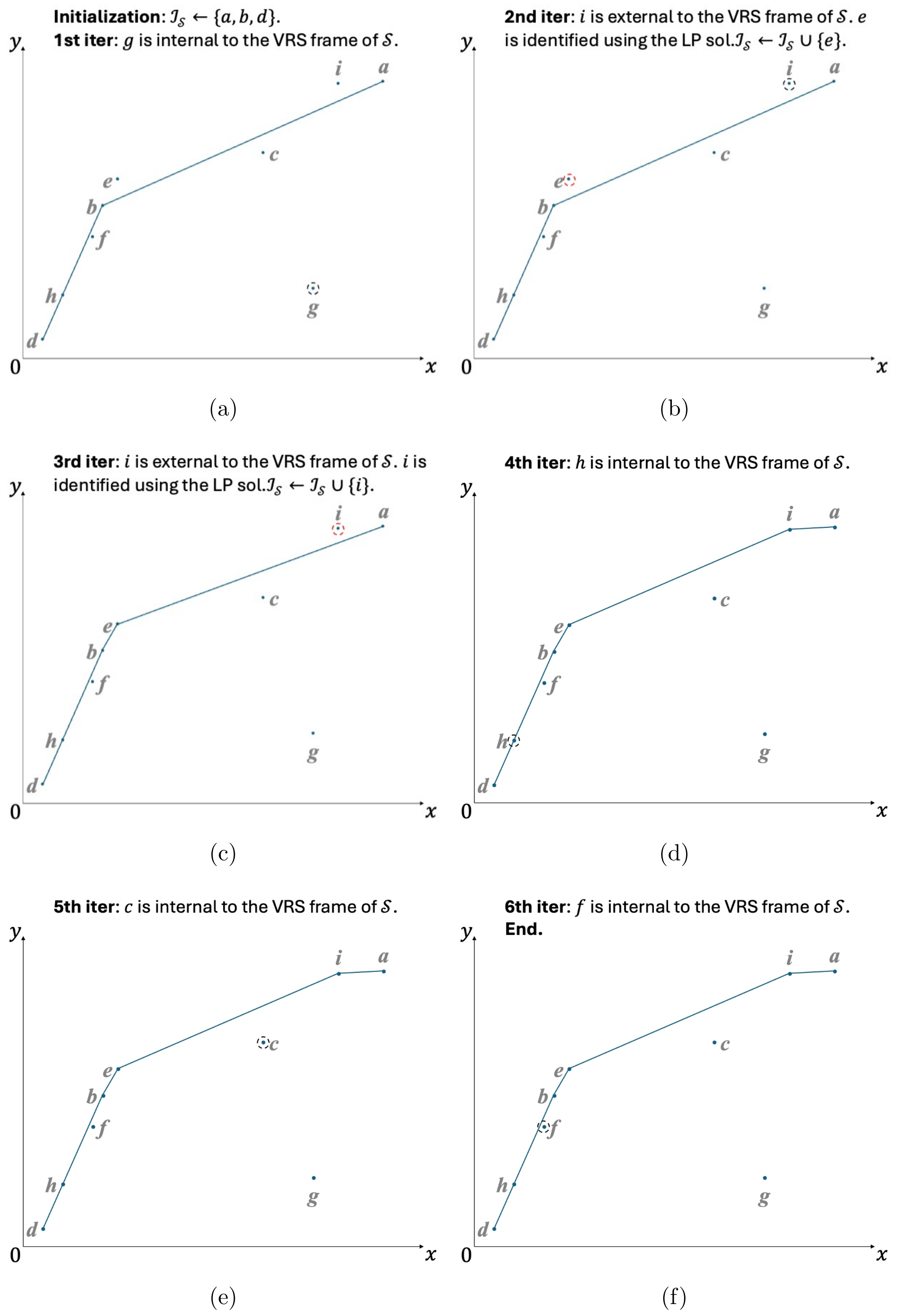

Initialization: $\mathcal{I}_S \leftarrow \{a, b, d\}$.
1st iter: g is internal to the VRS frame of $\mathcal{S}$.
(a)
2nd iter: i is external to the VRS frame of $\mathcal{S}$. e is identified using the LP sol.$\mathcal{I}_S \leftarrow \mathcal{I}_S \cup \{e\}$.
(b)
3rd iter: i is external to the VRS frame of $\mathcal{S}$. i is identified using the LP sol.$\mathcal{I}_S \leftarrow \mathcal{I}_S \cup \{i\}$.
(c)
4th iter: h is internal to the VRS frame of $\mathcal{S}$.
(d)
5th iter: c is internal to the VRS frame of $\mathcal{S}$.
(e)
6th iter: f is internal to the VRS frame of $\mathcal{S}$.
End.
(f)
y
x
0
a
b
c
d
e
f
g
h
i

Figure 3.2: BH Implementation

randomly selected in the next iteration. Then, it is evaluated and identified as an exterior in relation to the partial frame formed by $\mathcal{S}$, that is, $\alpha_i^* > 0$, as shown in Figure 3.2b. We obtain $\boldsymbol{\pi}_i^*$ from Model 2.3. Since e satisfies $e = \arg\max_j \langle \boldsymbol{\pi}_i^*, \boldsymbol{a}_j \rangle$, it is identified as a frame point, and $\mathcal{S}$ is updated, i.e., $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_e\}$. Next, in Figure 3.2c, point i is still exterior in relation to the VRS frame of the current $\mathcal{S}$, and since $i = \arg\max_j \langle \boldsymbol{\pi}_i^*, \boldsymbol{a}_j \rangle$, it is identified as a frame point, and $\mathcal{S}$ is updated again, i.e., $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_i\}$. In the next iterations, points h, c, and f are concluded as interior points, so no action is taken. Finally, the VRS frame is established as $\mathcal{F} \leftarrow \mathcal{S}$.

Summary

This chapter firstly covers techniques for eliminating non-frame points, such as the DEA Dominator method, which identifies dominated points. However, this method has limitations, particularly when dealing with non-frame points not dominated by any existing points.

Additionally, this chapter introduces several lemmas for identifying frame points, such as those based on extreme attributes, weighted sums, and output/input ratios. These methods provide streamlined, arithmetic-based tools for selecting boundary points without needing to solve LPs, crucial for large-scale DEA computation.

The famous BH framework is also reviewed in detail, which offers a sequential approach to constructing the DEA frame. This method combines identified frame points with iterative evaluations to classify other points. The integration of the discussed lemmas makes the BH algorithm effective in minimizing computational complexity. Practical examples and illustrations showcase the method's implementation.

Overall, this chapter provides an in-depth look at techniques for constructing DEA frame, establishing the foundation for the methodologies developed in the remainder of this study.

Chapter 4

Sequential Categorization

BH procedure demonstrates that frame points are identified only when an exterior to the partial frame is detected, as illustrated in Figures 3.2b and 3.2c. When an exterior is identified, the current partial frame is updated by incorporating a new extreme point, while the interior does not contribute to the update. This observation reveals an important insight: only exterior point contributes to the construction of the DEA frame $\mathcal{F}$. Therefore, instead of selecting points stochastically during the BH procedure, it is crucial to establish a systematic order for evaluating exterior points to swiftly build the VRS frame at an early stage.

Consider the example in Table 2.1, where the initial partial frame is $\mathcal{S} = \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d\}$. Let us assume there is a method that determines the evaluation order of the remaining points, as shown in Figure 3.2: i, e, h, c, f, g. After the first three iterations, the updated set becomes $\mathcal{S} = \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d, \boldsymbol{a}_e, \boldsymbol{a}_i\}$, at which point the update of $\mathcal{S}$ has converged. Consequently, $\mathcal{F}$ is constructed as $\mathcal{F} \leftarrow \mathcal{S}$, making it unnecessary to evaluate the remaining points.

This chapter introduces the method that establishes a systematic order for the BH procedure.

4.1 Mannual Distinct Weights

Zhuang et al. [48] proposed a method for manually formulating the positive weights. For a set of points with m inputs and s outputs, one can construct at most $2^m - 1$ proper subsets of inputs and $2^s - 1$ proper subsets of outputs. Let $\Upsilon(I)$ and $\Upsilon(O)$ denote the set of proper input subsets and proper output subsets, respectively.

For example, consider the case where $m = 3$. The set of inputs is $\{x_1, x_2, x_3\}$. Note here x_i denote the i-th input. In this scenario, the complete set of combinations yields:

$$\Upsilon(I) = \{\{x_1\}, \{x_2\}, \{x_3\}, \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\}, \{x_1, x_2, x_3\}\},$$

resulting in a total of $|\Upsilon(I)| = 7$.

Assume that I_p and O_q are one of the proper input subsets and proper output subsets, where $p = 1, \ldots, |\Upsilon(I)|$ and $q = 1, \ldots, |\Upsilon(O)|$. A bijection $f : \Upsilon(I) \to \boldsymbol{W}$ is defined such that each subset $I_p \in \Upsilon(I)$ maps to a unique weight vector in $\boldsymbol{W}$. For each subset $I_p \in \Upsilon(I)$, the corresponding weight vector $\boldsymbol{w}(I_p)$ is defined as:

$$\boldsymbol{w}(I_p) = \left(w_1 \cdot 1_{(x_1 \in I_p)}, \ldots, w_i \cdot 1_{(x_i \in I_p)}, \ldots, w_m \cdot 1_{(x_m \in I_p)}\right)^T,$$

where $w_i > 0$ for $i = 1, \ldots, m$, and $1_{(x_i \in I_p)}$ is an indicator function that is 1 if $x_i \in I_p$ and 0 otherwise.

Taking $m = 3$ as an example, the following explicit mapping is obtained:

$$\begin{aligned}
f(\{x_1\}) &= (w_1, 0, 0) \\
f(\{x_2\}) &= (0, w_2, 0) \\
f(\{x_3\}) &= (0, 0, w_3) \\
f(\{x_1, x_2\}) &= (w_1, w_2, 0) \\
f(\{x_1, x_3\}) &= (w_1, 0, w_3) \\
f(\{x_2, x_3\}) &= (0, w_2, w_3) \\
f(\{x_1, x_2, x_3\}) &= (w_1, w_2, w_3)
\end{aligned}$$

The bijection f maps each subset of inputs to a unique weight vector in $\boldsymbol{W}$. Thus, for $m = 3$, the following is obtained:

$$\boldsymbol{W} = \left(\ldots, \boldsymbol{w}(I_p), \ldots\right) = \begin{pmatrix} w_1 & 0 & 0 & w_1 & w_1 & 0 & w_1 \\ 0 & w_2 & 0 & w_2 & 0 & w_2 & w_2 \\ 0 & 0 & w_3 & 0 & w_3 & w_3 & w_3 \end{pmatrix}$$

Similarly, a bijection for outputs, $g : \Upsilon(O) \to \boldsymbol{V}$, can be established.

Furthermore, the values of w_i and v_r can be manually assigned any positive values, which implies that an infinite number of distinct weight vectors can be formulated and utilized. This flexibility allows us to apply Lemma 3.4 and Lemma 3.2 to identify a large portion of frame points, denoted as $\mathcal{S}$. However, achieving $\mathcal{S} = \mathcal{F}$ solely with manually selected weights is challenging. To identify the remaining points in $\mathcal{F} \setminus \mathcal{S}$, Model 2.2 and Model 2.3 are still needed. This established $\mathcal{S}$ can be used as a initial reference set when implementing BH.

4.1.1 Selection of Dimension Combinations

When the values of m and s are large, the number of dimension combinations can become substantial. For instance, if $m = 10$ and $s = 10$, the total number of combinations is

$$(2^{10} - 1) \times (2^{10} - 1) = 1,046,529,$$

assuming all possible combinations are considered.

To address this issue, instead of generating every possible combination, a selective approach is suggested that constructs $\Upsilon(I)$ using specific combinations $\binom{m}{i}$, where $i = 1, \ldots, m$. This method ensures that each input has an equal chance of being included in the subsets. For example, $\Upsilon(I)$ can be formed using $\binom{m}{m-1}$, which includes combinations of $m - 1$ inputs. When $m = 4$, the following holds:

$$\Upsilon(I) = \{\{x_1, x_2, x_3\}, \{x_1, x_2, x_4\}, \{x_1, x_3, x_4\}, \{x_2, x_3, x_4\}\},$$

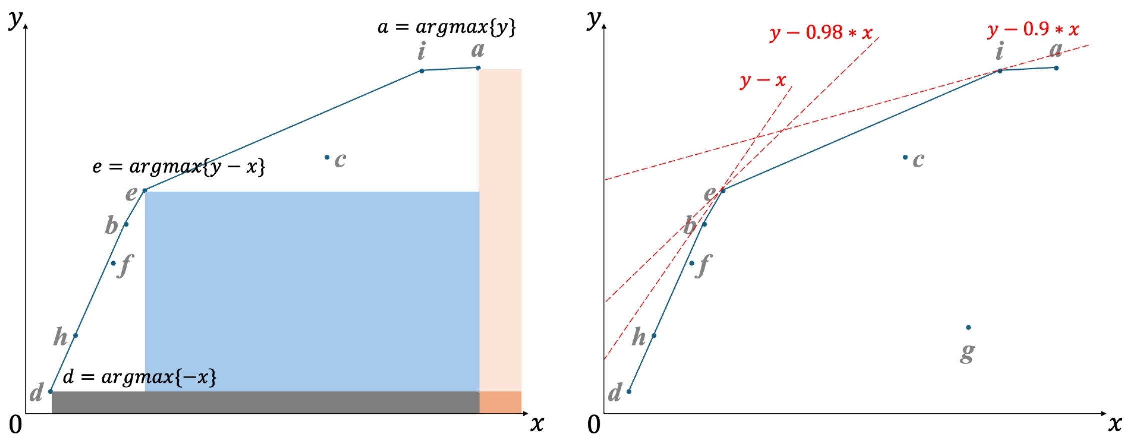
where each input is selected three $(m - 1)$ times. Alternatively, using $\binom{m}{\lceil m/2 \rceil}$ generates the maximum number of combinations, as $\binom{m}{\lceil m/2 \rceil} \geq \binom{m}{i}$ for $i = 1, \ldots, m$. For $m = 4$, the following holds:

$$\Upsilon(I) = \{\{x_1, x_2\}, \{x_1, x_3\}, \{x_1, x_4\}, \{x_2, x_3\}, \{x_2, x_4\}, \{x_3, x_4\}\}.$$

Similarly, $\Upsilon(O)$ can be constructed based on any of the combinations $\binom{s}{r}$, where $r = 1, \ldots, s$. This selective approach ensures that each input and output is selected equally.

Points that exhibit extreme features involving more dimensions are considered to be more geometrically *centralized* on the VRS frame than those involving fewer dimensions. For instance, in Figure 4.1a, point $e = \arg\max\{y - x\}$ is more centrally located than $a = \arg\max\{y\}$ and $d = \arg\max\{x\}$. A more centralized frame point spans a larger VRS hull (e.g., the blue rectangular area in Figure 4.1a), potentially enveloping more points as interiors. This is helpful for classification when using Models 2.2 and 2.3. In contrast, a frame point involving fewer dimensions is more marginalized, and the VRS hull it generates wraps fewer points. For instance, in Figure 4.1a, no points lie within the orange and black rectangular areas, as points a and d are located at the extremities of the frame.

Therefore, among all the input combinations corresponding to $\binom{m}{i}$, $i = 1, \ldots, m$, and output combinations corresponding to $\binom{s}{r}$, $r = 1, \ldots, s$, it is advisable to choose those that incorporate more dimensions in each column of the combination matrices. For instance, using input combinations of $\binom{m}{m-1}$ and output combinations of $\binom{s}{s-1}$ results in $m \times s$ columns, each including $m + s - 2$ dimensions. In contrast, using input combinations of $\binom{m}{1}$ and output combinations of $\binom{s}{1}$ results in $m \times s$ columns, each containing only two dimensions.



(a) Centralization (b) Relative Magnitude

Figure 4.1: Determining Manual Weights

4.1.2 Selection of Weight Values

A selective approach using $\binom{m}{i}$ and $\binom{s}{r}$ combinations helps prevent the combination explosion problem. However, reducing the number of input-output combinations may lead to fewer frame points being identified. To address this, the weights can be manually adjusted to increase the likelihood of identifying more frame points by applying Lemma 3.2 and Lemma 3.4 (as shown in Figure 4.1).

There are, in fact, infinite weight options available, but different weight choices can yield the same frame points. For instance, in Figure 4.1b, the weight combinations $(v, w) = (1.00, 1.00)$ and $(1.00, 0.98)$ both identify point e, while $(1.00, 0.90)$ identifies point i. Therefore, it is not necessary to test all weight combinations. What matters is the relative magnitude of the weights. In other words, the more distinct the weights are from one another, the higher the likelihood of identifying different frame points.

Considering both the relative size of weights and convenience of implementation, the weights can be set as $w_i = w, \forall i$, $v_r = v, \forall r$, where w and v are selected from a set of weights $\left\{\frac{k}{K} \mid k = 1, \ldots, K\right\}, K = \frac{\xi}{d}$. Here, ξ is a hyperparameter and is supposed to be a multiple of the value of d. For instance, when $m = 5$ and $s = 5$, setting $\xi = 10 \cdot (m + s) = 100$ results in $K = 10$ and weights $= \{0.1, 0.2, \ldots, 1.0\}$, leading to a total of 91 (v, w) combinations (excluding cases where $v = w$, except for $(1.0, 1.0)$). Notably, K decreases as d increases, indicating that fewer manual weights are needed for higher-dimensional cases. This determination of manual weights is based on the rationale that higher dimensions correspond to more potential combinations and thus more frame points. Consequently, the need for manually formulating weight values diminishes with higher dimensions.

4.2 Sequential Categorization

Distinct manual weight vectors are useful for identifying a subset of frame points, $\mathcal{S}$. However, it is challenging to achieve $\mathcal{S} = \mathcal{F}$ in a single attempt. To address this, a procedure is proposed that iteratively leverages the generated weight vectors, resulting in sequentially categorized blocks of points. Regarding the obtained blocks, points with smaller block indices have a higher probability of being frame points.

To facilitate the repeated use of the generated weights, the combination matrices are calculated before the categorization process. Let $\boldsymbol{X}'_{n\times|\Upsilon(I)|}$ denote the matrix of weighted inputs, where $\boldsymbol{X}' = \boldsymbol{XW}$, and X'_i represents the i-th column of $\boldsymbol{X}'$. Similarly, let $\boldsymbol{Y}'_{n\times|\Upsilon(O)|}$ denote the matrix of weighted outputs, where $\boldsymbol{Y}' = \boldsymbol{YV}$, and Y'_j is the j-th column of $\boldsymbol{Y}'$.

The combination matrices $\boldsymbol{C}^1_{n\times(|\Upsilon(I)|\times|\Upsilon(O)|)}$ and $\boldsymbol{C}^2_{n\times(|\Upsilon(I)|\times|\Upsilon(O)|)}$ are computed, where the number of columns is $|\Upsilon(I)| \times |\Upsilon(O)|$. These matrices are calculated as:

$$C^1_{i,j} = X'_i + Y'_j, \quad C^2_{i,j} = X'_i \oslash Y'_j.$$

The categorization process is described in Algorithm 4.1, where matrix $\boldsymbol{C}_{\mathcal{T}}$ contains the rows corresponding to the temporal work-array set $\mathcal{T}$. The operation $\arg\max(\boldsymbol{C}_{\mathcal{T}})$ identifies the set of point indices that yield the maximum value for each column.

Algorithm 4.1 Sequential Categorization

```
1: $\mathcal{T} \leftarrow \mathcal{A}, k \leftarrow 0$                              ▷ $k$ records the block index
2: while $\mathcal{T} \neq \emptyset$ do
3:     $\mathcal{B}_k \leftarrow \arg\max(\boldsymbol{C}^1_{\mathcal{T}}) \cup \arg\max(\boldsymbol{C}^2_{\mathcal{T}})$
4:     $k \leftarrow k + 1$
5:     $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{B}_k$
6: end while
7: return $\mathcal{B}_0, \ldots, \mathcal{B}_K$
```

Using Algorithm 4.1, a finite number (K) of sequential blocks $\mathcal{B}_0, \ldots, \mathcal{B}_K$ are obtained. These blocks share the characteristic that each point lies on the VRS frame of the points in the current and subsequent blocks, i.e., $\mathcal{B}_0 \subseteq \mathcal{F}$. This property can be easily deduced by referring to Lemmas 3.2 and 3.4.

4.2.1 How Algorithm 4.1 Works

Table 4.1 presents the combination matrix using the data from Table 2.1. To establish $\boldsymbol{C}^1_{\mathcal{T}}$, the values $(v, w) = (1.00, 1.00), (1.00, 0.10), (0.10, 1.00)$ are set for illustration convenience. Since there is only 1 input and 1 output in this case, $\boldsymbol{C}^2_{\mathcal{T}}$ leverages the ratio form and thus requires only 1 column. Both $\boldsymbol{C}^1_{\mathcal{T}}$ and $\boldsymbol{C}^2_{\mathcal{T}}$ are presented together in Table 4.1.

Table 4.1: Initial Combination Matrix

points	y/x	$y-x$	$y-0.1x$	$0.1y-x$
a	1.72	52	**116.8**	-59.6
b	**4.25**	52	66.4	-9.2
c	1.92	44	87.2	-38.8
d	2.00	4	7.6	**-3.2**
e	4.21	**61**	78.1	-11.0
f	3.86	40	52.6	-8.6
g	0.53	-27	25.2	-54.9
h	3.50	20	27.2	-5.2
i	1.95	60	116.7	-50.7

Let us assume the data in Table 4.1. We have $\mathcal{B}_0 = \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d, \boldsymbol{a}_e\}$ where

$$\begin{aligned} a &= \arg\max\{y - 0.1x\}, \\ b &= \arg\max\{y/x\}, \\ d &= \arg\max\{0.1y - x\}, \\ e &= \arg\max\{y - x\}. \end{aligned}$$

After obtaining $\mathcal{B}_0$ and excluding it from $\mathcal{T}$, we get $\mathcal{B}_1 = \{\boldsymbol{a}_f, \boldsymbol{a}_h, \boldsymbol{a}_i\}$ from Table 4.2 where

$$\begin{aligned} f &= \arg\max\{y/x\}, \\ h &= \arg\max\{0.1y - x\}, \\ i &= \arg\max\{y - x\} \text{ and } \arg\max\{y - 0.1x\}. \end{aligned}$$

Next, we have $\mathcal{B}_2 = \{\boldsymbol{a}_c\}$ from Table 4.3 since c holds the maximum values of all columns. Finally, $\mathcal{B}_3 = \{\boldsymbol{a}_g\}$.

Table 4.2: Combination Matrix after Excluding $\mathcal{B}_0$

points	y/x	$y-x$	$y-0.1x$	$0.1y-x$
c	1.92	44	87.2	-38.8
f	**3.86**	40	52.6	-8.6
g	0.53	-27	25.2	-54.9
h	3.50	20	27.2	**-5.2**
i	1.95	**60**	**116.7**	-50.7

From the above example, four sequential blocks $\mathcal{B}_0, \mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ are obtained by recursively leveraging Lemmas 3.2 and 3.4, with all frame points gathered in $\mathcal{B}_0$ and $\mathcal{B}_1$. As expected, frame points are likely to be concentrated in the initial blocks, as

Table 4.3: Combination Matrix after Excluding $\mathcal{B}_0$ and $\mathcal{B}_1$

points	y/x	$y-x$	$y-0.1x$	$0.1y-x$
c	**1.92**	**44**	**87.2**	**-38.8**
g	0.53	-27	25.2	-54.9

each block of points lies on the VRS frame of those in the current and subsequent blocks.

4.2.2 Data Transformation

In real-world applications, the scale of inputs and outputs can vary significantly. For example, some columns in $\boldsymbol{C}^1$ and $\boldsymbol{C}^2$ may be primarily affected by inputs and outputs with values ranging from $[1000, 10000]$, while others may range from $[1, 10]$, resulting in the same frame point. In the sample provided in Table 4.4, the maximum values (bolded) for the columns $y_1 - x_1$, $y_1 - (x_1 + x_2)$, and $y_1 - (x_1 + x_2 + x_3)$ all correspond to the frame point 1. These columns are primarily influenced by x_1, whose scale is much larger than x_2 and x_3.

To better utilize the combination matrices and achieve a greater number of distinct frame points, the unit-invariant property of DEA can be applied by normalizing the data. This process involves transforming the input and output values as follows:

$$x'_{il} = \frac{x_{il}}{\max\{x_{ij} \mid j = 1, \dots, n\}}, \quad i = 1, \dots, m,\ l = 1, \dots, n, \tag{4}$$

$$y'_{rl} = \frac{y_{rl}}{\max\{y_{rj} \mid j = 1, \dots, n\}}, \quad r = 1, \dots, s,\ l = 1, \dots, n. \tag{5}$$

This transformation does not affect the evaluation results of the DEA model. A formal proof of this invariance is provided in Appendix A.2.

Table 4.4 illustrates the impact of scale in the original data. After applying the normalization process, Table 4.5 demonstrates that columns $y_1^{'} - x_1^{'}, y_1^{'} - (x_1^{'} + x_2^{'})$, and $y_1^{'} - (x_1^{'} + x_2^{'} + x_3^{'})$ yield three distinct frame points: 1, 2, and 6.

Table 4.4: Example of Scale Impact (original data)

points	x_1	x_2	x_3	y_1	θ^*	$y_1 - x_1$	$y_1 - (x_1 + x_2)$	$y_1 - (x_1 + x_2 + x_3)$
1	40.93	0.62	0.71	1.00	1.00	**-39.93**	**-40.54**	**-41.25**
2	42.07	0.51	0.57	1.00	1.00	-41.07	-41.58	-42.15
3	89.50	0.80	0.90	1.00	0.57	-88.50	-89.29	-90.19
4	68.81	0.81	0.46	1.00	0.74	-67.81	-68.63	-69.08
5	42.42	0.74	0.29	1.00	1.00	-41.42	-42.16	-42.45
6	97.92	0.07	0.37	1.00	1.00	-96.92	-96.99	-97.36

Table 4.5: Example of Scale Impact (normalized data)

points	x'_1	x'_2	x'_3	y'_1	θ^*	$y'_1 - x'_1$	$y'_1 - (x'_1 + x'_2)$	$y'_1 - (x'_1 + x'_2 + x'_3)$
1	0.42	0.76	0.79	1.00	1.00	**0.58**	-0.17	-0.96
2	0.43	0.63	0.63	1.00	1.00	0.57	**-0.06**	-0.69
3	0.91	0.98	1.00	1.00	0.57	0.09	-0.89	-1.89
4	0.70	1.00	0.51	1.00	0.74	0.30	-0.70	-1.21
5	0.43	0.91	0.33	1.00	1.00	0.57	-0.34	-0.67
6	1.00	0.08	0.41	1.00	1.00	0.00	-0.08	**-0.49**

4.3 Categorization Validation

A total of 80 simulated samples are employed for empirical validation of the categorization effect. These samples are generated from a uniform distribution over the interval $[10, 100]$, rounded to 10 decimal places. Input-output ($m\&s$) pairs are selected from the set $\{2\&2, 3\&3, 4\&4, 5\&5\}$, and cardinality from $\{10000, 25000, 50000, 100000\}$, with densities varying between 10% and 25%. For each combination of dimension and cardinality, 5 samples are generated.

Table 4.6: Blocks Gathered Frame Points

$m\&s$	n	Sample index				
		1	2	3	4	5
2 & 2	10000	21/175	25/177	22/150	27/172	18/182
	25000	34/250	56/271	72/259	109/267	37/225
	50000	47/414	72/367	85/388	113/377	90/356
	100000	118/599	129/603	103/777	159/627	192/500
3 & 3	10000	24/127	21/116	24/100	23/104	21/109
	25000	24/168	33/136	32/229	39/218	31/195
	50000	48/339	72/263	64/208	46/284	68/193
	100000	68/301	73/310	88/481	80/437	73/433
4 & 4	10000	27/90	23/89	17/85	24/98	22/58
	25000	44/168	46/127	38/135	25/120	40/141
	50000	55/208	58/204	62/176	50/142	58/218
	100000	91/357	96/347	68/250	92/313	101/273
5 & 5	10000	24/65	26/65	29/56	35/77	22/50
	25000	44/99	40/101	40/109	49/128	43/129
	50000	67/141	79/189	72/163	55/141	62/126
	100000	108/311	125/291	113/312	104/254	87/186

As previously analyzed, the frame points are likely to be concentrated in the initial blocks. According to the lemmas and the categorization scheme, each block of points lies on the VRS frame of points in both the current and subsequent blocks, with B_0 being a subset of F. Table 4.7 provides a summary of the number of frame points contained within B_0. In this study, $\Upsilon(I)$ and $\Upsilon(O)$ are constructed based on the combinations of $\binom{m}{m-1}$ and $\binom{s}{s-1}$, respectively. The values in the column

"No. combinations" are calculated as $|\Upsilon(I)| \times |\Upsilon(O)| \times 2 = m \times s \times 2$, where two combination matrices, $\boldsymbol{C}^1$ and $\boldsymbol{C}^2$, are created.

As shown in Table 4.7, all values exceed the $d + 1$ (maximum) number of points suggested by Ali [1]. For instance, in the sample labeled 10000_2&2_1, 147 frame points are identified, which is significantly larger than the expected value of $d+1 = 5$. These frame points are determined by using 8 created combinations and varying the 91 pairs of w and v. Moreover, the trend in Table 4.7 indicates that $|B_0|$ increases as the dimension grows, owing to the greater number of possible dimension combinations and the corresponding increase in the number of extreme points that can be identified.

Table 4.7: Number of Points in B_0

m&s	n	No. Combinations	Sample Index 1	2	3	4	5
2 & 2	10000	8	147	111	232	153	216
	25000		272	159	152	68	229
	50000		280	283	227	322	302
	100000		244	305	341	179	277
3 & 3	10000	18	215	205	247	250	324
	25000		300	278	295	306	344
	50000		393	302	310	356	415
	100000		529	474	340	583	411
4 & 4	10000	32	257	191	300	287	275
	25000		384	445	235	542	437
	50000		490	556	596	387	566
	100000		609	715	548	604	493
5 & 5	10000	50	409	359	355	396	327
	25000		440	356	375	467	528
	50000		626	452	713	620	628
	100000		581	549	582	727	779

Table 4.6 shows the blocks in which the frame points are concentrated. For the sample 10000_2&2_1, the notation '21/175' means that the frame points are concentrated in the first 21 blocks out of a total of 175 blocks, as determined by the proposed categorization scheme. All other samples exhibit similar results, confirming the effectiveness of the categorization method. This suggests that it is not necessary to process all blocks to construct the VRS frame, which enables further reduction in the time required to build $\mathcal{F}$. Table 4.8 shows the number of points included in $\mathcal{F}$ for each sample.

When executing the BH procedure in the sequential order of blocks, the initial blocks are expected to contribute more extreme points due to the concentration effect. Figure 4.2 shows samples with 100,000 points. The horizontal axis of each subplot represents the block index, while the vertical axis indicates the number of

Table 4.8: Number of Points in $\mathcal{F}$

m&s	n	Sample Index				
		1	2	3	4	5
2 & 2	10000	1017	1021	1233	1713	1122
	25000	2521	2520	2544	2728	2802
	50000	5015	5010	5063	9342	7477
	100000	9952	9958	14087	10531	23437
3 & 3	10000	1101	1113	1186	1532	2315
	25000	2619	2635	2826	3035	4174
	50000	5146	5180	7000	5223	11382
	100000	10181	10230	12695	14559	13471
4 & 4	10000	1272	1252	1494	1378	2101
	25000	2959	2920	3473	4288	3549
	50000	5467	5639	7236	9234	9031
	100000	10583	10642	12204	12692	12029
5 & 5	10000	1560	1629	1991	2160	2417
	25000	3322	3253	4060	4030	4221
	50000	6235	6025	7552	10832	11379
	100000	11525	11329	12565	20787	16533

frame points identified from the relevant blocks. For clarity, only the first 16 blocks of samples with indices 1, 3, and 5 are shown.

As observed in Figure 4.2, block $\mathcal{B}_0$ contributes the most frame points, and this number decreases as the block index increases, eventually reaching zero after processing the first few blocks (e.g., the first 21 blocks for the sample 10000_2&2_1). This trend is consistent across all 80 samples, validating the effectiveness of systematically evaluating points to enable the early construction of the VRS frame.

Summary

In this chapter, we introduce the method that establishes a systematic order of selecting DEA points for the construction of DEA frontier using the BH procedure.

The first section describes a method for manually constructing distinct weights for a set of points with m inputs and s outputs. For each set of inputs and outputs, when all possible proper subsets are identified, with the number of subsets being $2^m - 1$ for inputs and $2^s - 1$ for outputs. A bijection is then defined between the proper input subsets and a set of weight vectors, where each subset maps to a unique weight vector. The selection of dimension combinations and values of the manual weights are then discussed. The weights can be manually assigned any positive values, allowing for an infinite number of distinct weight vectors. This flexibility facilitates the identification of a large portion of frame points, though achieving a complete set requires further methods.

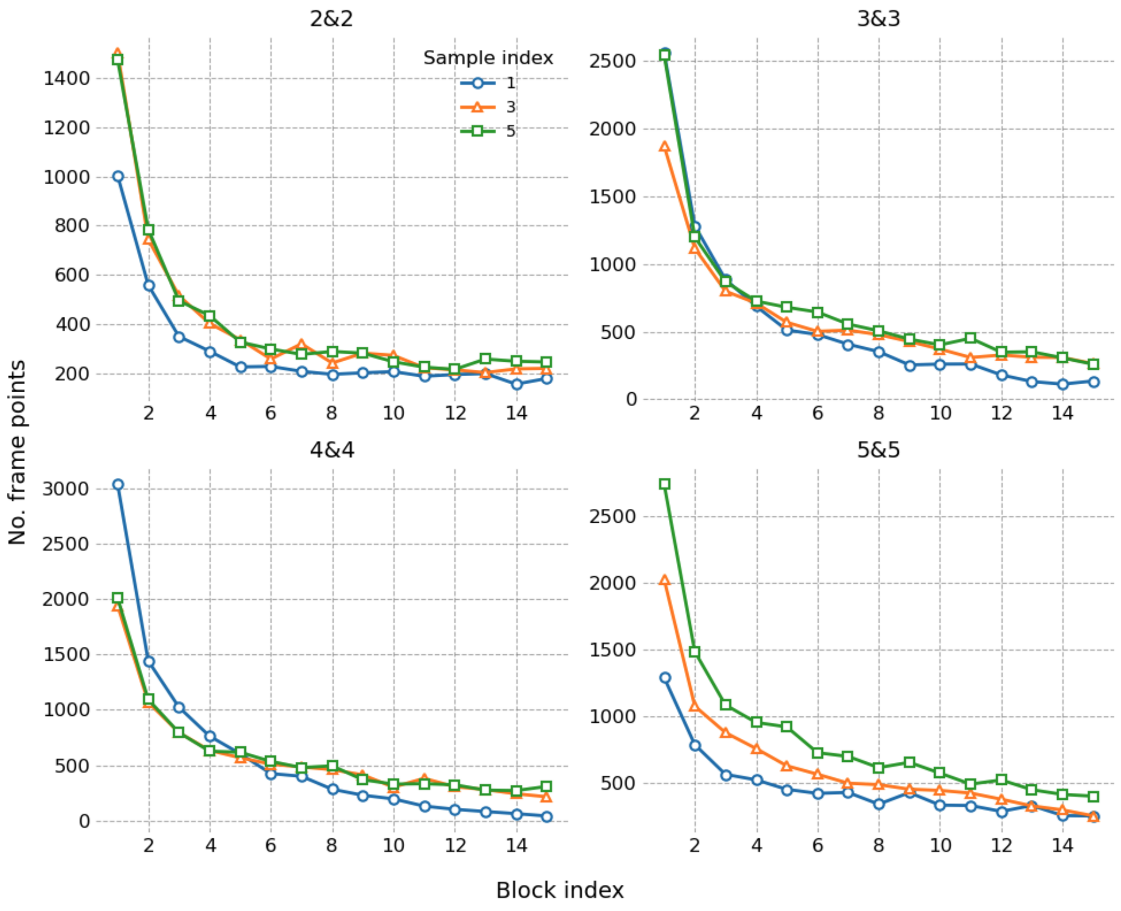


Figure 4.2: Frame Points Distribution $n = 100,000$

The second section outlines a sequential categorization process for gathering frame points, through iteratively using manual weight vectors. Combination matrices, $\boldsymbol{C}^1$ and $\boldsymbol{C}^2$, are generated from weighted inputs and outputs, and the categorization algorithm sequentially yields point blocks, with frame points concentrated in the initial blocks. A normalization procedure is proposed to eliminate the scale effect of inputs and outputs. Theoretical guarantees for the method are provided, and its effectiveness is demonstrated through an example using both the original and normalized data.

The third section presents the empirical validation. It shows the distribution of frame points across different blocks, implying that the initial blocks effectively concentrate all frame points, thus enabling a reduction in the number of blocks required to construct the VRS frame.

Chapter 5

Dimension-Wise Reference Selection

The systematic computation order of BH accelerates the construction of the DEA frame by sequentially gathering frame points. However, solving LPs with a large number of variables continues to be time consuming. Specifically, in LPs represented by Model 2.2 and Model 2.3, each point is matched only to the facets defined by the reference frame points closest to it, and only the variables associated with these frame points are necessary for the corresponding LPs.

In this chapter, a novel reference set selection method is introduced, applied prior to solving any of the models discussed earlier. Unlike existing selection techniques, this method enhances the likelihood of identifying the most relevant facet for each point, facilitating the small-size BH approach presented here. This approach significantly improves the performance of the BH computation, reducing both time consumption and memory requirements.

5.1 The Selection Technique

5.1.1 Geometric Insights

When point l is under evaluation in Model 2.4, at most d relevant benchmarks (frame points) are necessary to determine its efficiency result [9, 10, 12]. This can be intuitively observed from Figure 2.3. The model matched c to its closest facet ei. Only e and i provided the benchmarking role for c, using their convex combination, while other points on the frame did not contribute to the evaluation of c.

In Figure 3.2b, using Model 2.2 and Model 2.3, similarly, each point is assessed only in relation to its closest facet. For instance, when i is under evaluation, the optimal results $\boldsymbol{\pi}_i^*$ and ξ_i^* correspond to the parameters of the line where segment ab lies. The operation $\arg\max_{j \in \mathcal{I}_\mathcal{A}} \langle \boldsymbol{\pi}_i^*, \boldsymbol{a}_j \rangle$ translates the line towards the upper-left direction, identifying the extreme point e. Similarly, the interior point f in Figure 3.2c is evaluated in relation to the facet formed by a and b.

During the BH process, each point is matched to its closest facet, which is formed by at most d frame points. For each point, solving Model 2.2 and Model 2.3 using the subset of $\mathcal{S}$ that forms the relevant facet as the reference set, rather than all points in $\mathcal{S}$, reduces the number of variables involved in these LPs. This, in turn, decreases both memory requirements and computation time, thereby accelerating frame construction. Hence, it is helpful to develop a method for selecting the appropriate subset of reference points specifically for each point to be evaluated.

5.1.2 The Selection Method

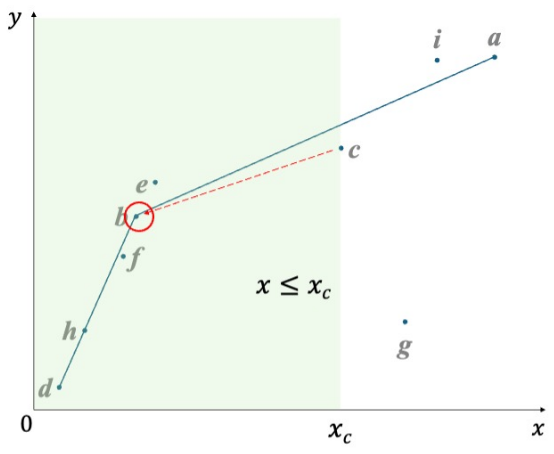
Assume $\mathcal{S} \subset \mathcal{F}$. Let $d_{l,j} = \|\boldsymbol{a}_l - \boldsymbol{a}_j\|_2 = \sqrt{\sum_i (a_{il} - a_{ij})^2}$ denote the Euclidean distance between points l and j, where $j \in \mathcal{I}_\mathcal{S}$ and $l \notin \mathcal{I}_\mathcal{S}$. For each $l \neq j$, define $D_l^i = \{d_{l,j} \mid a_{ij} \geq a_{il}, j \in \mathcal{I}_\mathcal{S}\}$, $i = 1, \ldots, d$. Then, define $\mathcal{R}_l^k = \bigcup_{i=1}^d \min^k(D_l^i)$, where the operation $\min^k(\cdot)$ obtains the points corresponding to the first k minimum values in D_l^i.

The subset $\mathcal{R}_l^k$ is selected as the reference set for point l in Model 2.2 and Model 2.3, with $\mathcal{R}_l^k \subseteq \mathcal{S}$. The size of $\mathcal{R}_l^k$ is at most $k \cdot d$. Generally, for point l, there is at least one frame point in its reference set $\mathcal{R}_l^k$ that is better than l in each dimension. $\mathcal{R}_l^k$ is expected to include points that form the relevant facet of the current partial frame to determine whether point l is an exterior or interior point. Figure 5.1 provides an illustrative example of this selection technique.

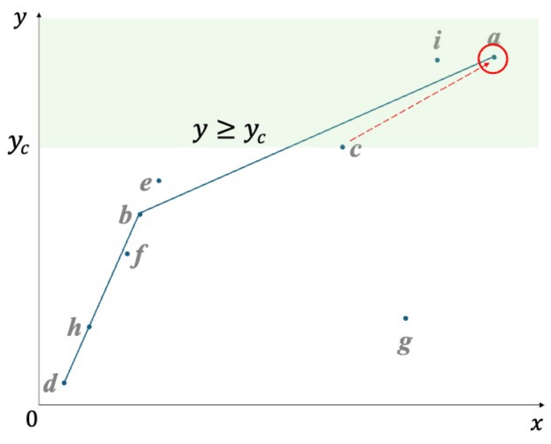
Without loss of generality, let $\mathcal{S} = \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d\}$ in Figure 5.1. Set $k = 1$, and let us first consider point c as an example. In Figure 5.1a, concerning the input, point b has the smallest distance to c among $\{a, b\}$. Then point b is selected into $\mathcal{R}_c^1$. For the output, only point a in $\mathcal{S}$ has a larger output value than c in Figure 5.1b, and it is selected into $\mathcal{R}_c^1$. Therefore, $\mathcal{R}_c^1 = \{\boldsymbol{a}_a, \boldsymbol{a}_b\}$. In Figure 5.1, the selected facet for point c is represented by the segment ab, and it is the exact facet matched to point c. Solving Model 2.2 and Model 2.3 using $\mathcal{R}_c^1$ as the reference set, we obtain $\alpha_c^* = 0$, which correctly concludes that point c is an interior. Similarly, we obtain $\mathcal{R}_i^1 = \{\boldsymbol{a}_a, \boldsymbol{a}_b\}$ for point i, and solving Model 2.2 and Model 2.3 using $\mathcal{R}_i^1$, we obtain $\alpha_i^* > 0$. Accordingly, a frame point $e = \arg\max_{j \in \mathcal{I}_\mathcal{A}} \langle \boldsymbol{\pi}_i^*, \boldsymbol{a}_j \rangle$ is obtained.

Next, let us consider the interior point g as an example in Figure 5.1c. In the output dimension, frame points a and b in $\mathcal{S}$ have larger output values than g, and b is selected into $\mathcal{R}_g^1$ as it is closer to g. Regarding the input, point b again has the smallest distance to g among the other candidate a. Thus, $\mathcal{R}_g^1 = \{\boldsymbol{a}_b\}$, and we have $\alpha_g^* = 0$ from Model 2.2 and Model 2.3 using $\mathcal{R}_g^1$, successfully identifying point g as an interior.

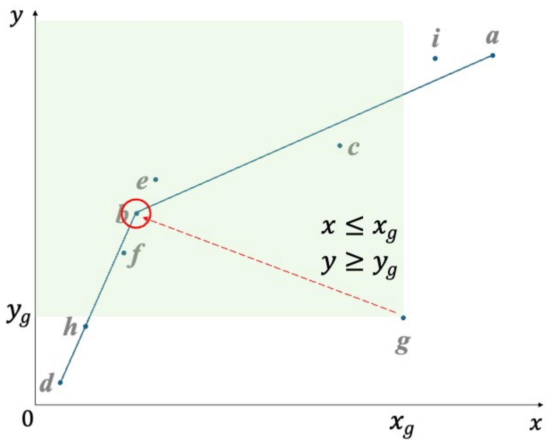
Consider special cases involving frame points in each $D_l^i, i = 1, \ldots, d$ that share the same distance to point l. In this scenario, we suggest selecting the frame point with the larger value of the input-output ratio $\sum_{r=1}^{s} y'_{r\cdot} / \sum_{i=1}^{m} x'_{i\cdot}$, this is in line with the philosophy of Lemma 3.3.



(a)



(b)



(c)

Figure 5.1: Dimension-Wise Selection

5.1.3 Improvements over Existing Techniques

To increase the likelihood of selecting the correct points as references, Chen et al. [10] propose the use of Equation 5.1 to construct a reference set of size d. Specifically, they calculate the angle between the projection direction of a point l and the direction to all other points on the current frame. Then, point l, where $l \notin \mathcal{I}_\mathcal{S}$, selects the top points j with the smallest angle $\phi_{l,j}$ to it as its initial reference set:

$$\phi_{l,j} = \arccos\left(\frac{(-\boldsymbol{x}_l, \boldsymbol{0}) \cdot (\boldsymbol{x}_j - \boldsymbol{x}_l, \boldsymbol{y}_j - \boldsymbol{y}_l)^T}{|(-\boldsymbol{x}_l, \boldsymbol{0})| \cdot |(\boldsymbol{x}_j - \boldsymbol{x}_l, \boldsymbol{y}_j - \boldsymbol{y}_l)|}\right), \quad \forall j \in \mathcal{I}_\mathcal{S}. \tag{5.1}$$

Instead of using Equation 5.1, Chu et al. [12] propose an alternative method for selecting the initial reference set, as represented by Equation 5.2:

$$\delta_{l,j} = \sqrt{\left|(\boldsymbol{x}_j - \boldsymbol{x}_l, \boldsymbol{y}_j - \boldsymbol{y}_l)\right|^2 - \frac{\left((-\boldsymbol{x}_l, \boldsymbol{0}) \cdot (\boldsymbol{x}_j - \boldsymbol{x}_l, \boldsymbol{y}_j - \boldsymbol{y}_l)^T\right)^2}{\left|(-\boldsymbol{x}_l, \boldsymbol{0})\right|^2}}, \quad \forall j \in \mathcal{I}_{\mathcal{S}}. \tag{5.2}$$

Here, $\delta_{l,j}$ denotes the vertical Euclidean distance from point j to the projection direction of point l. In this approach, the top $k \cdot d$ points that are closest to the projection line of point l are included in the initial reference set for l.

To implement both Equation 5.1 and Equation 5.2, the data needs to be standardized using the data transformation techniques represented by Equation 4 and Equation 5.

Table 5.1: Calculation Results of the Illustrative Example

point	x	y	$\phi_{l,j}$	$\delta_{l,j}$
a	72	124	2.21	32
b	16	68	0.64	24
d	4	8	1.09	84
e	19	80	0.39	12

To demonstrate the difference between our new technique and those proposed by Chen et al. [10] and Chu et al. [12], we use the example shown in Table 2.1. Consider point i as the target point for reference selection, using Equation 5.1 and Equation 5.2. The calculated values of $\phi_{i,j}$ $(j = a, b, d, e)$ and $\delta_{i,j}$ $(j = a, b, d, e)$ are presented in the fourth and fifth columns of Table 5.1, respectively. Assuming that two points need to be included in the initial reference set, both the methods by Chen et al. [10] and Chu et al. [12] select points b and e, while our technique selects points a and e.

Figure 5.2 visually demonstrates the differences between our proposed technique and the existing techniques. According to Chen et al. [10]'s technique, the directions ce and cb have the smallest angles with respect to point c's projection direction, leading to the selection of $\{\boldsymbol{a}_b, \boldsymbol{a}_e\}$ as the reference set. Similarly, Chu et al. [12]'s technique identifies points b and e as having smaller vertical distances to point c's projection direction. Consequently, point a, which is crucial for forming the closest facet to point c, is excluded from the initial reference set. Using points b and e as references in Model 2.2 and Model 2.3 incorrectly classifies the interior point c as an exterior point, potentially leading to the solution of more LPs in the BH process. In contrast, our new technique selects points closest to the current point in each dimension, successfully identifying $\{\boldsymbol{a}_a, \boldsymbol{a}_e\}$ as the reference set.

In an d-dimensional space, the closest VRS facet for point l is formed by at most d points. Our technique separately considers the relative magnitude of each dimen-

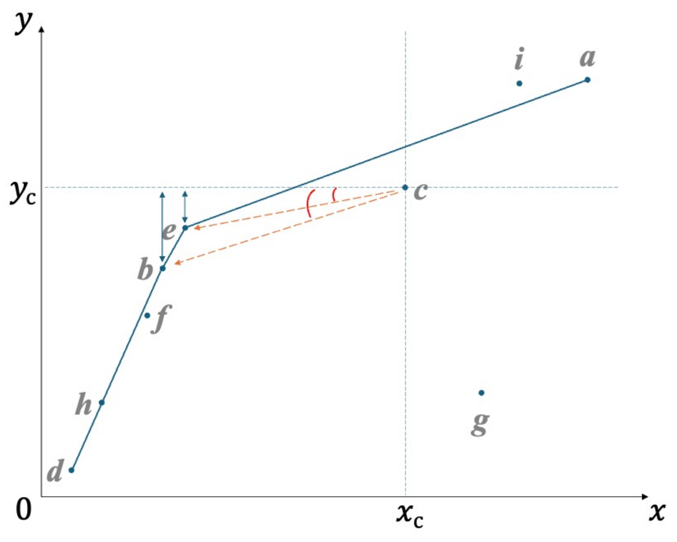


Figure 5.2: Comparison with Existing Techniques

sion between the current point and the points in the current subsample, selecting the closest point for each dimension. This approach is more likely to identify the correct reference points in arbitrary dimension. It is worth noting, however, that our technique does not provide a theoretical guarantee of selecting the correct points as the dimensionality increases and the structure of the facet becomes more complex.

5.2 Enhanced Build Hull

Based on the new selection technique, we propose a small-size BH procedure as outlined in Algorithm 5.1.

In line 2, $\mathcal{S}$ is initialized using the lemmas in Section 3.1.2. The algorithm then selects $l \notin \mathcal{I}_{\mathcal{S}}$. For each point l, the evaluation is first performed using Model 2.2 and Model 2.3 with the smallest ($\leq d$) reference set $\mathcal{R}_l^1$. If the exterior condition $\alpha_l^* > 0$ holds, a corresponding frame point j^* is identified, and $\mathcal{S}$ is updated by adding j^*, i.e., $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{j^*}\}$. The point l is then reevaluated with a larger reference set $\mathcal{R}_l^2$ selected from the updated $\mathcal{S}$. The rationale for increasing the size of $\mathcal{R}_l^k$ is twofold: first, the newly discovered extreme point j^* should be incorporated into $\mathcal{R}_l^k$, updating the reference facet for point l as shown in Figure 3.2b to Figure 3.2c; second, since $\alpha_l^* > 0$ may be obtained even if point l is interior, but $\mathcal{R}_l^k$ does not include the effective reference points, a larger $\mathcal{R}_l^k$ is necessary to confirm the status of point l. By starting with the smallest $\mathcal{R}_l^k$ and dynamically increasing its size only when a point is identified as exterior, the model is solved with the fewest variables, thus minimizing computational effort. The procedure outlined in lines 9 to 16 converges as established in Theorem 5.1, with the proof provided.

By removing irrelevant variables in Model 2.2 and Model 2.3 through a dynamic

Algorithm 5.1 Small-Size BH

```
1: Initialize $\mathcal{S} \subset \mathcal{F}$
2: $\mathcal{T} \leftarrow \mathcal{A}$
3: $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{S}$
4: while $\mathcal{T} \neq \emptyset$ do
5:     select $l \in \mathcal{A}$
6:     $k \leftarrow 1$                                   ▷ $k$ corresponds to size of selected reference set
7:     $\mathcal{R} \leftarrow \mathcal{R}_l^k = \bigcup_i^d min^k(D_l^i)$
8:     $\alpha_l^*, \boldsymbol{\pi}_l^* \leftarrow$ solve Model 2.2 and Model 2.3 for $l$
9:     while $\alpha_l^* > 0$ do
10:        $l^* = \arg\max_{j \in \mathcal{A}} \langle \boldsymbol{\pi}_l^*, \boldsymbol{a}_j \rangle$      ▷ in case of a tie, set $l^*$ to one extreme point
11:        $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{l^*}\}$
12:        $\mathcal{T} \setminus \{\boldsymbol{a}_l^*\}$
13:        $k \leftarrow k + 1$
14:        $\mathcal{R} \leftarrow \mathcal{R}_l^k$
15:        $\alpha_l^*, \boldsymbol{\pi}_l^* \leftarrow$ solve Model 2.2 and Model 2.3 for $l$
16:    end while
17:    $\mathcal{T} \setminus \{\boldsymbol{a}_l\}$
18: end while
19: $\mathcal{F} \leftarrow \mathcal{S}$                                     ▷ the frame is established
20: return $\mathcal{F}$
```

dimension-wise reference set selection procedure, this small-size BH procedure substantially decreases memory usage, without requiring a high memory as in the original BH. Hence, it becomes possible to enhance BH further by transitioning from a purely sequential computation to a parallel computation scheme to fully leverage the computation cores of a CPU.

Theorem 5.1. Given any random reference sample $\mathcal{R}_l \subset \mathcal{A}$ for a point l, where $l \notin \mathcal{I}_{\mathcal{R}_l}$, the value of α_l^* converges to 0 in the procedure outlined in lines 9 to 16.

Proof of Theorem 5.1. For each $l \in \mathcal{I}_{\mathcal{A}}$, we initialize a reference set $\mathcal{R}_l$ such that $l \notin \mathcal{I}_{\mathcal{R}_l}$ and $\mathcal{R}_l \subset \mathcal{A}$. The procedure's convergence requires the condition $\alpha_l^* = 0$ be satisfied. Denoting α_l^* at iteration k as $\alpha_l^{*(k)}$ $(k = 1, 2, \ldots)$, we aim to prove that α_l^* decreases with each iteration and converges to 0. At iteration k, the reference set $\mathcal{R}_l$ is updated as:

$$\mathcal{R}_l^{(k+1)} = \mathcal{R}_l^{(k)} \cup \{\boldsymbol{a}_{j^*}\},$$

where j^* is uniquely selected as:

$$j^* = \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}_l^{*(k)}, \boldsymbol{a}_j \rangle.$$

Since $\alpha_l^{*(k)} > 0$, the hyperplane defined by $\mathcal{H}(\boldsymbol{\pi}_l^{*(k)}, \xi_l^{*(k)})$ separates $\mathcal{A}$ into two subsets:

$$\mathcal{A}^{++}_{\boldsymbol{\pi}_l^{*(k)},\xi_l^{*(k)}} \quad \text{and} \quad \mathcal{A}^{-}_{\boldsymbol{\pi}_l^{*(k)},\xi_l^{*(k)}},$$

where $\boldsymbol{a}_l \in \mathcal{A}^{++}_{\boldsymbol{\pi}_l^{*(k)},\xi_l^{*(k)}}$. The choice of j^* ensures:

$$\boldsymbol{a}_{j^*} \in \mathcal{A}^{++}_{\boldsymbol{\pi}_l^{*(k)},\xi_l^{*(k)}}, \quad \text{and} \quad \langle \boldsymbol{\pi}_l^{*(k)}, \boldsymbol{a}_{j^*} \rangle > \langle \boldsymbol{\pi}_l^{*(k)}, \boldsymbol{a}_l \rangle.$$

From Model 2.3, we have:

$$\alpha_{j^*}^{*(k)} \geq \langle \boldsymbol{\pi}_l^{*(k)}, \boldsymbol{a}_{j^*} \rangle + \xi_l^{*(k)} > \alpha_l^{*(k)} > 0.$$

Similarly, referring to Model 2.2, it follows:

$$\boldsymbol{a}_{j^*} > \sum_{j \in \mathcal{I}_{\mathcal{R}_l^{(k)}}} \lambda_{lj}^{*(k)} \boldsymbol{a}_j.$$

The updated value of $\alpha_l^{*(k+1)}$ is given by:

$$\alpha_l^{*(k+1)} = \min_{\lambda \geq 0} \left\{ \boldsymbol{u}^T \left(\boldsymbol{a}_l - \left(\sum_{j \in \mathcal{I}_{\mathcal{R}_l^{(k)}}} \lambda_{lj} \boldsymbol{a}_j + \lambda_{lj^*} \boldsymbol{a}_{j^*} \right) \right) \right\},$$

where $\sum_{j \in \mathcal{I}_{\mathcal{R}_l^{(k+1)}}} \lambda_{lj} = 1$. Hence, we have:

$$\alpha_l^{*(k+1)} < \boldsymbol{u}^T \left(\boldsymbol{a}_l - \sum_{j \in \mathcal{I}_{\mathcal{R}_l^{(k)}}} \lambda_{lj}^{*(k)} \boldsymbol{a}_j \right) = \alpha_l^{*(k)}.$$

implying:

$$\alpha_l^{*(k+1)} < \alpha_l^{*(k)}.$$

By induction, $\alpha_l^{*(k)}$ decreases with each iteration and converges to 0 as $k \to \infty$:

$$\lim_{k \to \infty} \alpha_l^{*(k)} = 0.$$

This confirms that the procedure described in lines 9 to 16 is convergent. □

The proof of Theorem 5.1 can also be obtained through other ways. For instance, since $\mathcal{A}^{++}_{\boldsymbol{\pi}_l^{*(k)},\xi_l^{*(k)}}$ is a finite set and $|\mathcal{R}_l^{(k+1)}| = |\mathcal{R}_l^{(k)}| + 1$, the loop can iterate at most $|\mathcal{A}^{++}_{\boldsymbol{\pi}_l^{*(k)},\xi_l^{*(k)}}|$ times before $\boldsymbol{a}_l \in \gamma^{VRS}_{\mathcal{R}_l}$. At this stage:

$$\alpha_l^{*(k)} = 0, \quad \text{when} \quad \boldsymbol{a}_l \in \gamma^{VRS}_{\mathcal{R}_l}.$$

Parallel Small-Size BH

We can further enhance the small-size BH using parallel computation with the categorized blocks $\mathcal{B}_0, \dots, \mathcal{B}_K$, where frame points are gathered in initial blocks. This systematic computation increases the likelihood of detecting exteriors early in the BH process, resulting in a faster construction of the VRS frame.

Algorithm 5.2 Parallel BH with Sequential Blocks

```
1: Initialize $\mathcal{S} \subset \mathcal{F}$
2: $\mathcal{T} \leftarrow \mathcal{A}$
3: $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{S}$
4: while $\mathcal{T} \neq \emptyset$ do
5:     for $\mathcal{B}$ in $\mathcal{B}_1, \dots, \mathcal{B}_K$ do
6:         parallel for $l$ in $\mathcal{B}$
7:         if $l \notin \mathcal{I}_{\mathcal{S}}$ then
8:             $k \leftarrow 1$
9:             $\mathcal{R} \leftarrow \mathcal{R}_l^k = \bigcup_i^d \min^k(D_l^i)$
10:            $\alpha_l^*, \boldsymbol{\pi}_l^* \leftarrow$ solve Model 2.2 and Model 2.3 for $l$
11:            while $\alpha_l^* > 0$ do
12:                $l^* = \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}_l^*, \boldsymbol{a}_j \rangle$
13:                $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{l^*}\}$
14:                $k \leftarrow k + 1$
15:                $\mathcal{R} \leftarrow \mathcal{R}_l^k$
16:                $\alpha_l^*, \boldsymbol{\pi}_l^* \leftarrow$ solve Model 2.2 and Model 2.3 for $l$
17:            end while
18:        end if
19:        $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\boldsymbol{a}_l, \boldsymbol{a}_{l^*}\}$
20:    end for
21: end while
22: $\mathcal{F} \leftarrow \mathcal{S}$
23: return $\mathcal{F}$
```

Since points within the same block lie on the VRS frame of points in both the current and subsequent blocks, and are obtained using distinct positive weights (indicating they lie in different locations of the current partial VRS frame), calculations for points within the same block can be performed in parallel. This allows for the ongoing construction of the VRS frame in multiple locations. Algorithm 5.2 outlines the procedure to achieve these objectives.

Identified extreme points can originate from the current block or subsequent blocks. Therefore, line 7 checks whether the current point l is already included in $\mathcal{S}$. If it is, no action is taken in the current iteration, meaning that lines 9 to 17 are skipped. The updating of $\mathcal{S}$ may converge after processing the initial few blocks, at which point all frame points have been gathered. This allows for skipping lines 11 to 16 for all points in subsequent blocks, so they can be evaluated using Model 2.2 and Model 2.3 with the smallest $\mathcal{R}_l^{(1)}$ in line 10. As a result, the computation cost becomes negligible.

5.3 Empirical Experiments

We use a computer with an Intel Core i7-12700K CPU @3.60 GHz, 32.0 GB of memory, and Linux 20.04. The algorithm is implemented in Python 3. Note that the following results are obtained without parallel computation, to make a fair comparison with BH.

5.3.1 Time Comparison

Table 5.2 presents the CPU time for both BH and small-size BH, using a single CPU worker. For example, in the first sample of 10,000 DMUs with 2 inputs and 2 outputs, referred to as sample 10000_2&2_1, small-size BH takes only 65 seconds, while BH takes 268 seconds, as shown in Table 5.2 in the format '65 (268)'. Note here the same initial subsample $\mathcal{S} = \mathcal{B}_0$ is used for both BH and small-size BH.

Table 5.2: Time (seconds) Comparison

m&s	n	Sample Index				
		1	2	3	4	5
2 & 2	10000	65 (268)	66 (272)	70 (305)	78 (333)	67 (289)
	25000	193 (2144)	189 (1953)	190 (2069)	193 (2062)	197 (2099)
	50000	433 (8950)	427 (9477)	428 (9379)	553 (11941)	495 (10978)
	100000	1164 (48152)	1201 (51122)	1420 (59117)	1231 (51977)	2099 (88590)
3 & 3	10000	92 (409)	93 (390)	91 (396)	99 (428)	115 (494)
	25000	284 (3140)	272 (3017)	277 (3106)	285 (3124)	340 (3666)
	50000	682 (15302)	645 (13705)	740 (16533)	665 (13661)	941 (20025)
	100000	1731 (77519)	1806 (75569)	2047 (91561)	2138 (90284)	1997 (84662)
4 & 4	10000	127 (546)	129 (571)	143 (588)	136 (599)	162 (726)
	25000	405 (4413)	385 (4183)	408 (4299)	462 (4854)	411 (4292)
	50000	911 (18778)	966 (20837)	1086 (22847)	1236 (25567)	1243 (27273)
	100000	2564 (106209)	2559 (109143)	2769 (117406)	2684 (120188)	2833 (120982)
5 & 5	10000	193 (861)	207 (863)	212 (944)	218 (979)	244 (1085)
	25000	557 (5815)	528 (5478)	587 (6042)	580 (6366)	671 (7503)
	50000	1417 (29602)	1395 (28795)	1474 (32769)	1942 (42379)	1911 (40582)
	100000	3812 (156750)	3639 (150466)	4001 (173128)	5737 (240040)	4577 (195303)

Furthermore, Table 5.2 shows that as the samples' cardinality (n) and dimension increase, there is a noticeable reduction in computation time. Figure 5.3 illustrates this trend of decreasing time. It highlights that the time performance of small-size BH is considerably less sensitive than that of BH with respect to the portion of $\mathcal{F}$, cardinality, and dimension. As the portion of $\mathcal{F}$ (calculated by $\mathcal{F}/\mathcal{A}$) increases, the computation time for both small-size BH and BH also rises. However, the increase in computation time for BH is much greater than that for small-size BH. This trend is similarly observed with respect to cardinality and dimension, as shown in Figure 5.3.

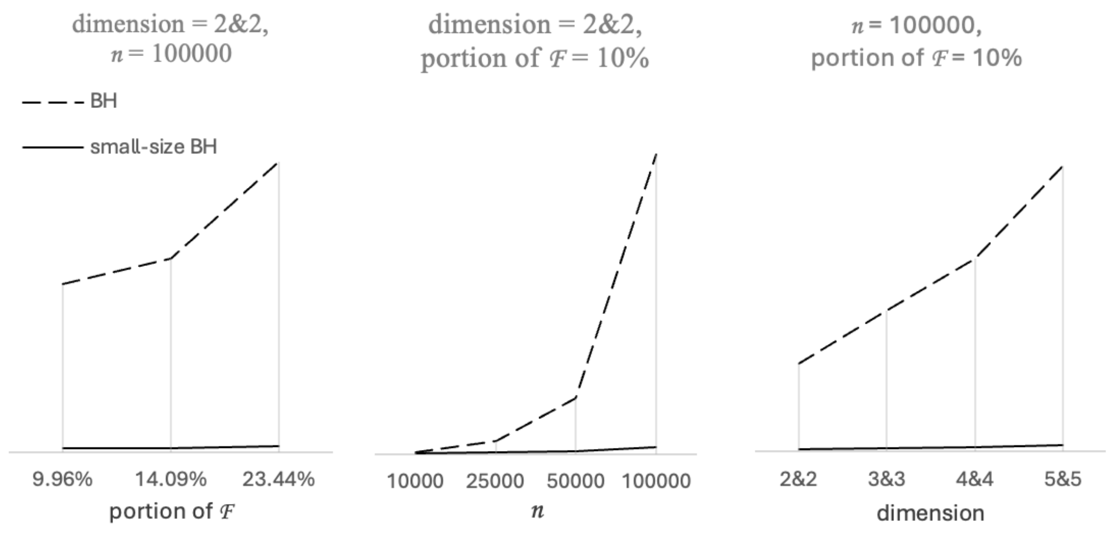


Figure 5.3: Time Comparison between BH and Small-Size BH

5.3.2 LP Statistics

The time reduction arises from solving small-size LPs using the subset $\mathcal{R}_l^k$ selected from $\mathcal{S}$. Table 5.3 summarizes the number of smallest-size LPs ($k = 1$) and the total number of LPs ($k = 1, \dots$) involved across all samples using our new technique.

In the sample 10000_2&2_1, a total of 10,775 LPs ($k = 1, \dots$) are computed, of which 8,872 LPs are solved using the smallest size ($\leq d$). This indicates that 88.72% of non-frame DMUs obtained their status (interior) with minimal computational cost. Similar results are observed across other samples, confirming the effectiveness of our proposed reference set selection technique.

Table 5.3 also suggests a positive correlation between the total number of LPs ($k = 1, \dots$) required during small-size BH and the dimensions $m\&s$. Consequently, computation time increases, as depicted in Figure 5.2 with respect to dimensions. However, the increase in time is negligible compared to the full BH.

Summary

In this chapter, we propose a novel reference set selection method and validate its effectiveness through empirical experiments.

The first section presents the geometric insights behind the development of our selection method, followed by a detailed description of the technique. To demonstrate its effectiveness, we compare our approach with existing methods for small-size LP problems. This selection procedure is intended to be applied before solving any of the LP models discussed in this study.

The second section introduces the enhanced BH procedure, incorporating our

Table 5.3: Number of LPs: $(k = 1)/(k = 1, \dots)$

$m\&s$	n	Sample Index				
		1	2	3	4	5
2 & 2	10000	8872/10775	8870/10850	8695/10772	8199/11495	8818/10675
	25000	22329/26882	22338/27026	22312/27014	22056/27356	22044/27246
	50000	44774/54091	44773/54029	44668/54071	40513/57253	42335/55872
	100000	89628/108340	89733/107932	85489/111021	89016/109143	75784/118983
3 & 3	10000	8636/11092	8620/11196	8627/11036	8251/11377	7522/12203
	25000	21841/28310	21822/27906	21826/27855	21525/28070	20362/30174
	50000	44161/55824	44147/55674	42058/59292	43886/56804	38012/62409
	100000	88681/110890	88425/112085	85968/116912	84514/116991	85637/114156
4 & 4	10000	8330/11762	8395/11756	8070/12132	8156/12150	7466/13372
	25000	21180/29897	21286/29400	20805/30109	19893/31753	20606/30196
	50000	43177/58051	42893/59501	41326/61526	39160/65513	39524/66102
	100000	87405/115781	87287/116334	85042/121457	85399/117590	85305/122673
5 & 5	10000	7855/12783	7736/13477	7457/13566	7347/13676	7034/14563
	25000	20521/31514	20647/30977	19702/32988	19938/31884	19482/34967
	50000	41659/64040	42029/62636	40422/65023	37206/72700	36937/68600
	100000	84854/124330	85235/123014	84033/125576	76023/138939	79984/132648

dimension-wise reference selection technique. We also provide a theoretical proof of the convergence of the proposed algorithm. With the small-size selection procedure in place, the BH method can be more easily extended to a parallel computation scheme, and a computational framework is designed to support this.

The third section empirically validates the effectiveness of the proposed methods. The majority of points are classified correctly by solving the minimal-size LPs, resulting in a significant reduction in computation time compared to the conventional BH method.

Chapter 6

Adaptive Shortest-Distance Search

In the previous chapter, we introduced a straightforward method to effectively reduce the size of LPs that need to be solved.

Beyond minimizing the size of the LPs required for computing the VRS frame, we can also reduce the number of LPs that need to be solved. The DEA Dominator, described in Section 3.1.1, provides an arithmetic approach to identify non-frame points through dimension comparison, thereby eliminating the need to solve LPs. However, certain non-frame points may not be dominated by any existing points (i.e., point c in Figure 3.1b), which limits the applicability of the direct dimension comparison. To resolve this issue, building upon the shortest distance approach introduced in Chapter 5, we propose a method for adaptively constructing virtual points for non-frame points. The frame points identified during this process can also function as the reference set for Models 2.2 and 2.3, thereby accelerating the construction of the VRS frame.

6.1 Search Strategy

For each $l \notin \mathcal{I}_{\mathcal{S}}$ and $\boldsymbol{a}_l \in \Upsilon_{\mathcal{S}}^{\text{VRS}}$, there exists a virtual point j' such that $\boldsymbol{a}_{j'} = \sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j \boldsymbol{a}_j$, where $\sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j = 1$ and $\boldsymbol{a}_{j'} \geq \boldsymbol{a}_l$. According to Definition 3.2, point l is dominated by the virtual point j' and does not belong to the set $\mathcal{F}$. Since virtual dominating points can be convex combinations of frame points closest to the current point l, we propose **Strategy 1** to identify frame points as candidates for constructing virtual dominating points for each non-frame point l:

Strategy 1: Consider a subset of frame points $\mathcal{S}$. For each point l (where $l \notin \mathcal{I}_{\mathcal{S}}$), select the points from $\mathcal{S}$ that are closest to point l, and construct the dominating point j' such that $\boldsymbol{a}_{j'} \geq \boldsymbol{a}_l$.

After formulating point j' using the selected candidate points, we propose **Strategy 2** to assign positive weights and identify frame points:

Strategy 2: Given the formulation of point j', we derive $\boldsymbol{\pi}_l = \boldsymbol{a}_{j'} - \boldsymbol{a}_l \geq \mathbf{0}$. Additional frame points can then possibly be identified by selecting $j^* \leftarrow \arg\max_{j \in \mathcal{I}_\mathcal{A}} \langle \boldsymbol{\pi}_l, \boldsymbol{a}_j \rangle$.

Based on these proposed strategies, we next introduce the corresponding algorithm to discriminate non-frame points using virtual domination and detect frame points through the derived positive weights.

6.2 Proposed Approach

First, assume that we have obtained a subset of frame points $\mathcal{S}$ using the lemmas from Section 3.1.2. For each point l, where $l \in \mathcal{I}_\mathcal{A} \setminus \mathcal{I}_\mathcal{S}$, we calculate $\boldsymbol{a}_j^l = \boldsymbol{a}_j - \boldsymbol{a}_l$ for $j \in \mathcal{I}_\mathcal{S}$, representing the domination scores of point j over point l. It is evident that if there exists $\boldsymbol{a}_j^l \geq \mathbf{0}$ for some $j \in \mathcal{I}_\mathcal{S}$, then point l is dominated by point j, indicating that point l is non-frame. This scenario corresponds to real domination.

Geometrically, the transformation $\boldsymbol{a}_j^l = \boldsymbol{a}_j - \boldsymbol{a}_l$ establishes a coordinate system letting $\boldsymbol{a}_l^l = \mathbf{0}$ be the origion, keeping the relative position of all other points unchanged, i.e., c is at the origion in Figure 6.1. Define point $-j$ as the virtual point, whose transformed dimensions are opposite of the transformed dimensions of j, that is, $\boldsymbol{a}_{-j}^l = -\boldsymbol{a}_j^l$. For instance, point $-b$ can be obtained as the reflection of point b with respect to the origin c. The purpose of this reflection is to identify the frame point closest to the reflected point, such as point a which is nearest to $-b$. This identified frame points are then used to construct j' that dominates l.

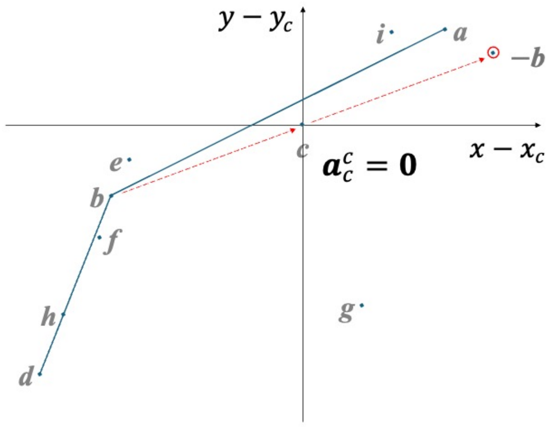


Figure 6.1: Establishing coordinate system for point c

Note that the proposed transformation keeps the relatie position of points unchanged. Following the previous chapter, let $d_{l,j}$ represent the Euclidean distance between point l and point j, where $j \in \mathcal{I}_\mathcal{S}$ and define $D_l = \{d_{l,j} \mid j \in \mathcal{I}_\mathcal{S}\}$. The operation $\min(D_l)$ selects the point corresponding to the minimum value in D_l. Similarly, let $\mathcal{D}$ be the set of points that are dominated under either the real or virtual scenario.

Based on these preparations, we now propose the following procedure described by Algorithm 6.1.

In line 7, k records the current iteration step and is initialized to 1 for each point l. In line 8, we first select from $\mathcal{S}$ the point j_1 that is closest to point l, following **Strategy 1**. Point $j^{\#}$ is a virtual point initialized as point j_1. The set $\mathcal{R}_l$ is the set of reference points selected from $\mathcal{S}$ for point l, which is initialized by $\mathcal{R}_l \leftarrow \{\boldsymbol{a}_{j_{k=1}}\}$.

It is worth noting that line 11 can be considered a refined version of the Dominator approach. The key difference is that we only need to compare point l with the current point $j^{\#}$, whereas Dominator requires comparing point l with all points in $\mathcal{A}$.

Next, let's see how the Algorithm 6.1 works using the instance dataset presented in Table 2.1.

6.2.1 The Dominated Scenario

In line 11, if $\boldsymbol{a}^l_{j^{\#}} \geq \boldsymbol{0}$, it implies that $\boldsymbol{a}_{j^{\#}} \geq \boldsymbol{a}_l$, indicating that point l is dominated by the current point $j^{\#}$. Point l is determined to be non-frame (according to Definition 3.2 and its corresponding corollaries) and is assigned to the set $\mathcal{D}$. At the same time, a positive weight vector $\boldsymbol{\pi}_l$ is formulated, and a frame point not yet included in $\mathcal{S}$ might be detected using $j^* \leftarrow \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}_l, \boldsymbol{a}_j \rangle$, after which the subset of frame points is updated by $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{j^*}\}$. This is the scenario of real domination.

For example, assume point g is being processed and the initial partial frontier is $\mathcal{S} \leftarrow \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d\}$. Point $j^{\#}$ is initialized as point b, as it is the closest to point g among a, b, d. We conclude that point g is non-frame because it is dominated by point b. Using $\boldsymbol{\pi}_g \leftarrow \boldsymbol{a}^g_b$, we have $e \leftarrow \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}_g, \boldsymbol{a}_j \rangle$, meaning that point e is found on the frame, and $\mathcal{S}$ is updated by $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_e\}$, which becomes $\{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d, \boldsymbol{a}_e\}$.

When the selected point j does not dominate l, we implement lines 17 to 27, where K is the hyperparameter that limits the maximum number of iterations, and can be set to any desired value. In this study, we recommend setting K as a multiple of d. In line 19, we first find the point j_k by $j_k \leftarrow \min(D_{-j^{\#}})$, which is closest to the virtual point $-j^{\#}$, then update the current $j^{\#}$ by accumulation: $\boldsymbol{a}^l_{j^{\#}} \leftarrow \boldsymbol{a}^l_{j^{\#}} + \boldsymbol{a}^l_{j_k}$. $\mathcal{R}_l$ is updated using $\mathcal{R}_l \leftarrow \mathcal{R}_l \cup \{\boldsymbol{a}_{j_k}\}$. Next, if $\boldsymbol{a}^l_{j^{\#}} \geq \boldsymbol{0}$ ($\boldsymbol{a}_{j^{\#}} \geq \boldsymbol{a}_l$), point l is assigned to $\mathcal{D}$. At the same time, a frame point j^* might be identified, and $\mathcal{S}$ is updated accordingly. The while loop is then terminated at this point. If $\boldsymbol{a}^l_{j^{\#}} \geq \boldsymbol{0}$ is not satisfied and the number of iterations k is less than K, the same procedure of updating $j^{\#}$ by finding the point closest to the current $-j^{\#}$ is repeated until $\boldsymbol{a}^l_{j^{\#}} \geq \boldsymbol{0}$ is satisfied or $k = K$.

The procedure outlined in lines 17 to 27 iteratively constructs a virtual dominat-

Algorithm 6.1 Shortest-Distance Adaptive Formulation

```
 1: Initialize $\mathcal{S} \subset \mathcal{F}$
 2: $\mathcal{T} \leftarrow \mathcal{A}$
 3: $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{S}$
 4: $\mathcal{D} \leftarrow \emptyset$
 5: while $\mathcal{T} \neq \emptyset$ do
 6:     select $l \in \mathcal{I}_{\mathcal{T}}$
 7:     $k \leftarrow 1$                                    $\triangleright$ $k$ records iteration steps
 8:     $j_k \leftarrow \min(D_l)$
 9:     $j^{\#} \leftarrow j_k$
10:     $\mathcal{R}_l \leftarrow \{\boldsymbol{a}_{j_k}\}$
11:     if $\boldsymbol{a}^l_{j^{\#}} \geq \boldsymbol{0}$ then
12:         $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{a}_l\}$
13:         $\boldsymbol{\pi}_l \leftarrow \boldsymbol{a}^l_{j^{\#}}$
14:         $j^* \leftarrow \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}_l, \boldsymbol{a}_j \rangle$
15:         $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{j^*}\}$
16:     else if $\boldsymbol{a}^l_{j^{\#}} \ngeq \boldsymbol{0}$ then
17:         while $k \leq K$ do
18:             $k \leftarrow k + 1$
19:             $j_k \leftarrow \min(D_{-j^{\#}})$
20:             $\boldsymbol{a}^l_{j^{\#}} \leftarrow \boldsymbol{a}^l_{j^{\#}} + \boldsymbol{a}^l_{j_k}$
21:             $\mathcal{R}_l \leftarrow \mathcal{R}_l \cup \{\boldsymbol{a}_{j_k}\}$
22:             if $\boldsymbol{a}^l_{j^{\#}} \geq \boldsymbol{0}$ then
23:                 $\mathcal{D} \leftarrow \mathcal{D} \cup \{\boldsymbol{a}_l\}$
24:                 $\boldsymbol{\pi}_l \leftarrow \boldsymbol{a}^l_{j^{\#}}$
25:                 $j^* \leftarrow \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}_l, \boldsymbol{a}_j \rangle$
26:                 $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{j^*}\}$
27:                 break
28:             end if
29:         end while
30:     else if $k = K$ and $\boldsymbol{a}^l_{j^{\#}} \ngeq \boldsymbol{0}$ then
31:         $\alpha^*_l, \boldsymbol{\pi}^*_l \leftarrow$ compute Model 2.2 and Model 2.3 with $\mathcal{R} \leftarrow \mathcal{R}_l$
32:         while $\alpha^*_l > 0$ do
33:             $j^* \leftarrow \arg\max_{j \in \mathcal{I}_{\mathcal{A}}} \langle \boldsymbol{\pi}^*_l, \boldsymbol{a}_j \rangle$
34:             $\mathcal{S} \leftarrow \mathcal{S} \cup \{\boldsymbol{a}_{j^*}\}$
35:             $\mathcal{R}_l \leftarrow \mathcal{R}_l \cup \{\boldsymbol{a}_{j^*}\}$
36:             $\alpha^*_l, \boldsymbol{\pi}^*_l \leftarrow$ compute Model 2.2 and Model 2.3 with $\mathcal{R} \leftarrow \mathcal{R}_l$
37:         end while
38:     end if
39:     $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\boldsymbol{a}_l, \boldsymbol{a}_{l^*}\}$
40: end while
41: $\mathcal{F} \leftarrow \mathcal{S}$
```

ing point using an adaptive search scheme guided by the shortest distance principle outlined in **Strategy 1**. For any point satisfying the condition in line 22 and being assigned to the set $\mathcal{D}$, it is dominated and explicitly non-frame. This result is formalized in Theorem 6.1, with the proof provided.

Theorem 6.1. Any point l assigned to the set $\mathcal{D}$ is clearly non-frame.

Proof of Theorem 6.1. Let l be a given point in $\mathcal{D}$. The combined representation of points in the reference set $\mathcal{R}_l$ is expressed as:

$$\boldsymbol{a}^l_{j\#} = \sum_k \boldsymbol{a}_{j_k} = \sum_k \boldsymbol{a}_{j_k} - \boldsymbol{a}_l \cdot k.$$

When $\boldsymbol{a}^l_{j\#} \geq \mathbf{0}$, it follows that:

$$\sum_k \boldsymbol{a}_{j_k} \geq \boldsymbol{a}_l \cdot k,$$

which implies:

$$\frac{\sum_k \boldsymbol{a}_{j_k}}{k} \geq \boldsymbol{a}_l.$$

Define a virtual point j' with:

$$\boldsymbol{a}_{j'} = \frac{\sum_k \boldsymbol{a}_{j_k}}{k}.$$

This representation of $\boldsymbol{a}_{j'}$ is a convex combination of points in the reference set $\mathcal{R}_l$. Since $\mathcal{R}_l \subseteq \mathcal{S}$, we deduce:

$$\boldsymbol{a}_{j'} \in \Upsilon^{VRS}_{\mathcal{S}},$$

where $\Upsilon^{VRS}_{\mathcal{S}}$ represents the convex hull generated by the points in $\mathcal{S}$. More generally, $\boldsymbol{a}_{j'}$ can be expressed as:

$$\boldsymbol{a}_{j'} = \sum_{j \in \mathcal{I}_{\mathcal{S}}} \lambda_j \boldsymbol{a}_j,$$

where:

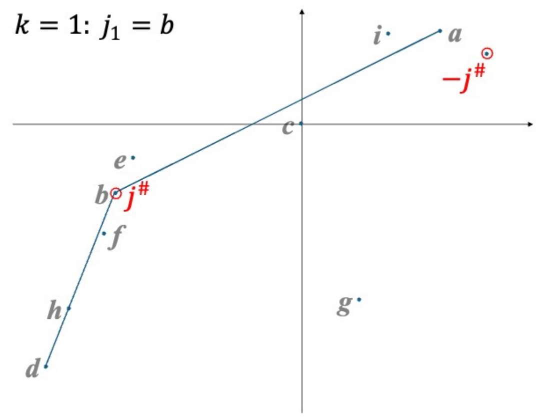
$$\lambda_j = \frac{|\{j_k \mid j_k = j\}|}{k}.$$

Here, λ_j represents the times of point j being selected into $\mathcal{R}_l$. From the inequality $\boldsymbol{a}_{j'} \geq \boldsymbol{a}_l$, we conclude that point l is dominated by the virtual point j'. Consequently, point l is non-frame. $\square$

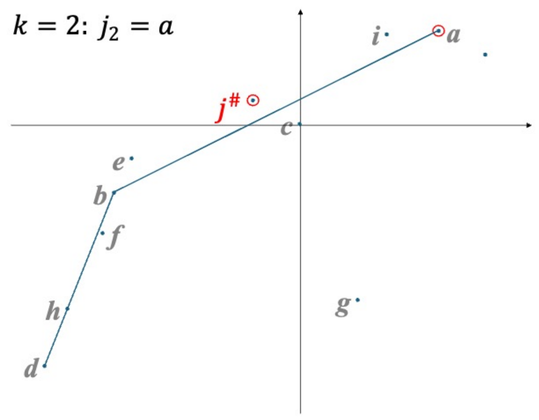
The convergence of the procedure outlined in lines 32 to 37 can be established by referencing Theorem 5.1.

Illustration Implementations

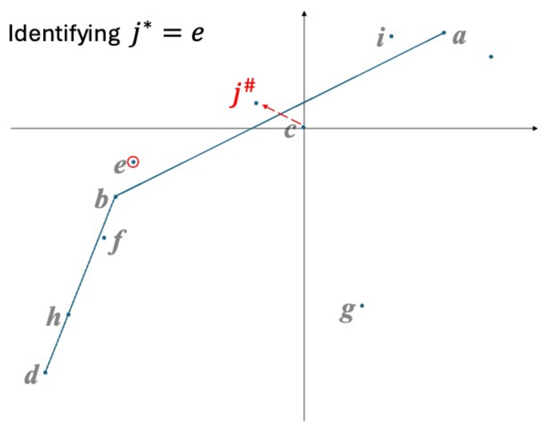
To intuitively understand how lines 17 to 27 work, let us assume that point c is being processed, and the current set $\mathcal{S} \leftarrow \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d\}$. Without loss of generality, this setup allows us to observe the mechanics of the algorithm. Figure 6.2 visually presents the implementation details, showing how a virtual dominating point j' is formulated step by step during the adaptive searching process.



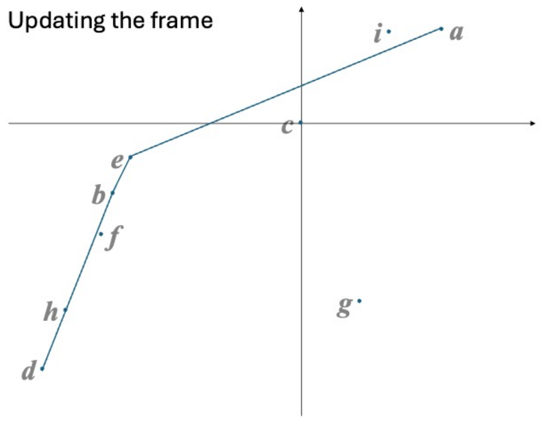
(a)



(b)



(c)



(d)

Figure 6.2: Implementation for point c

In the first iteration ($k = 1$) shown in Figure 6.2a, both a and b share the same minimal distance to c. In this case, we select the unit with the larger input-output ratio $\sum_r y_{rj} / \sum_i x_{ij}$. So, we select $j_1 \leftarrow b$, meaning b is chosen in the first iteration. We set $j^\# \leftarrow j_1$. Since $\boldsymbol{a}^c_{j^\#} \geq \mathbf{0}$ is not satisfied at this point, in Figure 6.2b, we first reflect $j^\#$ through the origin (point c's location) and find $-j^\#$. Next, a is the closest to $-j^\#$, so we set $j_2 \leftarrow a$. Then, $j^\#$ is updated by $\boldsymbol{a}^c_{j^\#} \leftarrow \boldsymbol{a}^c_{j^\#} + \boldsymbol{a}^c_{j_2}$, which is shown in Figure 6.2b.

After two iterations, $\boldsymbol{a}^c_{j^\#} \leftarrow \boldsymbol{a}^c_{j^\#} + \boldsymbol{a}^c_{j_k} \geq \mathbf{0}$ is established for c, where $j_1 \leftarrow b$

and $j_2 \leftarrow a$. The while loop is terminated by the *break* command as described in line 27. This indicates that point c is dominated by the virtual point j', with $\boldsymbol{a}_{j'} \in \Upsilon_{\mathcal{S}}^{VRS}$, where $\boldsymbol{a}_{j'} = \sum_{j=b,a} \lambda_j \boldsymbol{a}_j$ with $\lambda_a = |\{j_k \mid j_k = a\}|/k = 1/2$, and $\lambda_b = |\{j_k \mid j_k = b\}|/k = 1/2$. Therefore, point c is non-frame. Furthermore, we have $\boldsymbol{\pi}_c \leftarrow \boldsymbol{a}_{j'}$, and a frame point $e \leftarrow \arg\max_{j \in \mathcal{I}_\mathcal{A}} \langle \boldsymbol{\pi}_c, \boldsymbol{a}_j \rangle$ is identified, as shown in Figure 6.2c. Finally, the partial frontier is updated as shown in Figure 6.2d.

The implementation of point c requires only two iterations to establish the virtual dominating point, and the set $\mathcal{S}$ is updated to include $\{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d, \boldsymbol{a}_e\}$. Next, let us examine the implementation for point f, which requires more iterations to establish its corresponding dominating point. Figure 6.3 illustrates this process step by step.

In the first iteration shown in Figure 6.3a, we have $j_1 \leftarrow b$, which is the closest to f, and initialize $j^\#$ as b. Next, we find $-j^\#$. In Figure 6.3b, b is again selected, since it is the closest to $-j^\#$, so we set $j_2 \leftarrow b$, and $j^\#$ is updated by $\boldsymbol{a}^f_{j^\#} \leftarrow \boldsymbol{a}^f_{j^\#} + \boldsymbol{a}^f_{j_2}$. We then find $-j^\#$ in Figure 6.3b. Since $\boldsymbol{a}^f_{j^\#} \geq \boldsymbol{0}$ is still not satisfied, in Figure 6.3c, we find $j_3 \leftarrow d$, which is the closest to the current $-j^\#$. We repeat the same updating process for $j^\#$, and in Figure 6.3d, we find $j_4 \leftarrow b$. At this point, $\boldsymbol{a}^f_{j^\#} \leftarrow \boldsymbol{a}^f_{j^\#} + \boldsymbol{a}^f_{j_4} \geq \boldsymbol{0}$ is satisfied, meaning that the virtual dominating point j' for point f is established after four iterations. Then the while loop is terminated. The points involved in this process are: b, b, d, and b. We then have $\boldsymbol{a}_{j'} = \sum_{j=b,d} \lambda_j \boldsymbol{a}_j$, where $\lambda_b = |\{j_k \mid j_k = b\}|/k = 3/4$ and $\lambda_d = |\{j_k \mid j_k = d\}|/k = 1/4$. Furthermore, $\boldsymbol{\pi}_f \leftarrow \boldsymbol{a}_{j'}$, and a frame point $i \leftarrow \arg\max_{j \in \mathcal{I}_\mathcal{A}} \langle \boldsymbol{\pi}_f, \boldsymbol{a}_j \rangle$ is identified, as shown in Figure 6.3e. Finally, the partial frontier is updated, as illustrated in Figure 6.3f.

Based on the implementation for points c and f, the effectiveness of lines 17 to 27 can be visually observed: By iteratively reflecting the current $j^\#$ through the origin to $-j^\#$, we identify the point closest to $-j^\#$, and then update $j^\#$ using the accumulation operation $\boldsymbol{a}^l_{j^\#} \leftarrow \boldsymbol{a}^l_{j^\#} + \boldsymbol{a}^l_{j_k}$. As a result, $j^\#$ remains confined to the region near the origin and tends to move progressively closer to it.

6.2.2 The Undominated Scenario

As detailed, for any non-frame point l being processed, if the corresponding real dominating point does not exist, a virtual dominating point can potentially be constructed using lines 17 to 27, provided K is set sufficiently large. However, when point l is exterior to the convex hull of $\mathcal{S}$, the condition $\boldsymbol{a}^l_{j^\#} \geq \boldsymbol{0}$ in line 11 will never be satisfied, regardless of how large K is.

Consider the example provided in Table 2.1. Since the order in which points enter the procedure outlined in the pseudocode is not predetermined, we can assume, without loss of generality, that the partial frontier $\mathcal{S}$ is formed by points a, b, and d. For instance, in Figure 6.4a, points i and e, as well as other points such as o, which are positioned above the partial frontier, cannot have a virtual dominating point

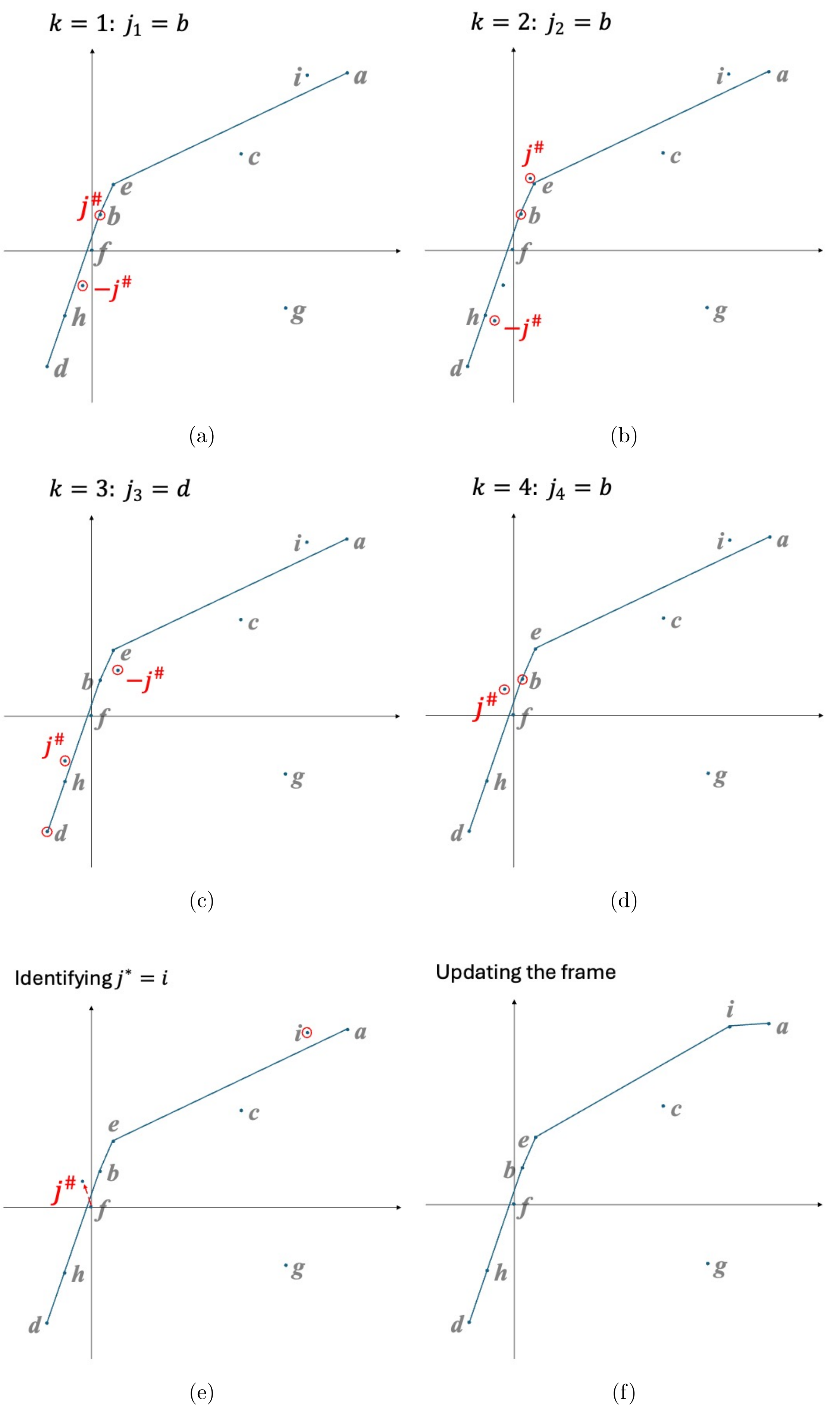
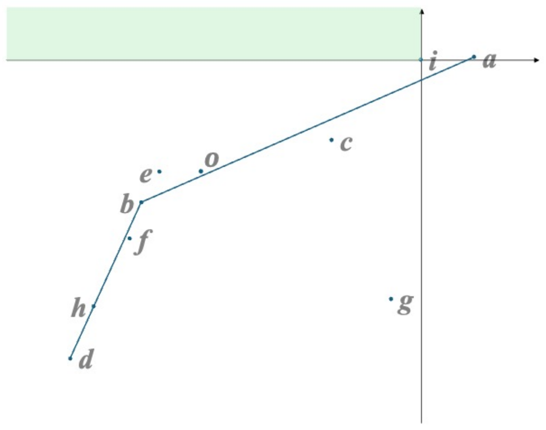
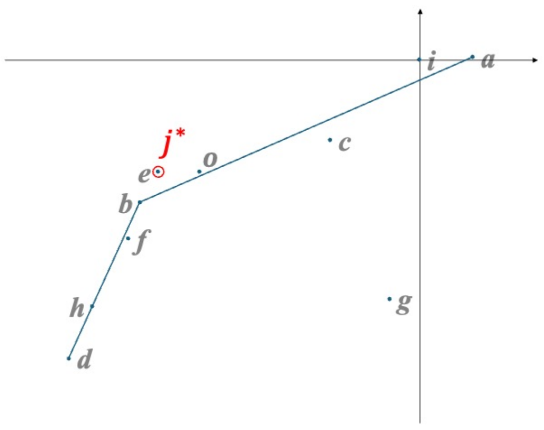


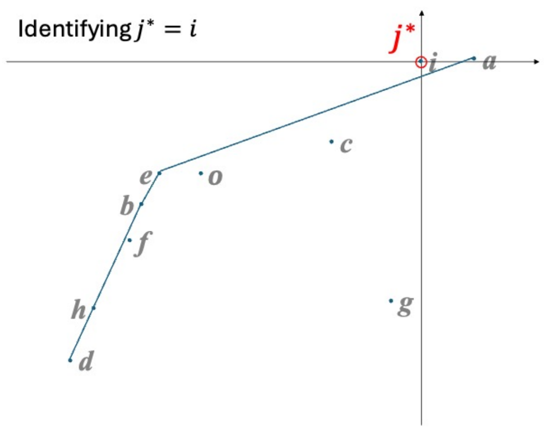
Figure 6.3: Implementation for point f



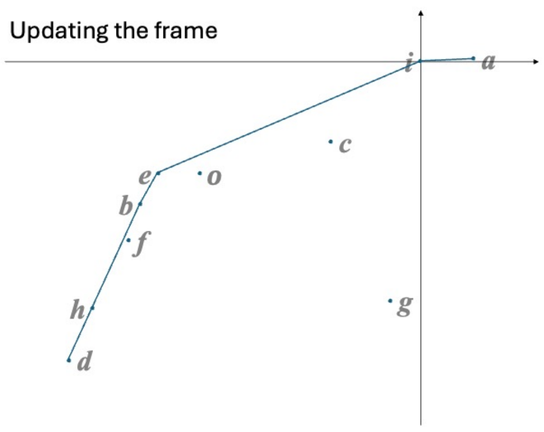
(a)



(b)



(c)



(d)

Figure 6.4: Case of Exteriors i.e., point i

constructed, regardless of the number of iterations performed. In other words, the status of these points cannot be determined even when $k = K$. In this situation, we implement lines 31 to 36 to differentiate the current point and construct the frame hull.

To understand how lines 31 to 36 work, let's take point i as an example. After executing lines 17 to 27, we obtain $\mathcal{R}_i \leftarrow \{\boldsymbol{a}_a, \boldsymbol{a}_b\}$ based on the shortest distance search. Solving Model 2.2 and Model 2.3 with the reference set $\mathcal{R}_i$, we find $\alpha_i^* > 0$, indicating that point i is exterior to $\mathcal{S}$. We then identify $e \leftarrow \arg\max_{j \in \mathcal{I}_\mathcal{A}} \langle \boldsymbol{\pi}_i^*, \boldsymbol{a}_j \rangle$. Thus, $\mathcal{R}_i$ is updated to $\{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_e\}$, and $\mathcal{S}$ is updated to $\{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d, \boldsymbol{a}_e\}$, as depicted in figure 6.4b.

Next, we solve Model 2.2 and Model 2.3 again, which results in $\alpha_i^* > 0$ and we identify $i \leftarrow \arg\max_{j \in \mathcal{I}_\mathcal{A}} \langle \boldsymbol{\pi}_i^*, \boldsymbol{a}_j \rangle$. Consequently, $\mathcal{S}$ is updated to $\{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d, \boldsymbol{a}_e, \boldsymbol{a}_i\}$, as shown in figure 6.4c. Finally, solving Model 2.2 and Model 2.3 once more with $\mathcal{R}_i \leftarrow \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_e, \boldsymbol{a}_i\}$ results in $\alpha_i^* = 0$, terminating the while loop for point i.

Note that, in this process, unlike the BH approach, Model 2.2 and Model 2.3 incorporates variables corresponding solely to points in $\mathcal{R}_i$, which is a subset of $\mathcal{S}$. This effectively excludes unnecessary variables. In large-scale DEA computations, especially when the dimensionality is significantly higher than the current two-dimensional example, the composition of the convex hull becomes increasingly complex, and $\mathcal{S}$ can become quite large. By establishing $\mathcal{R}_l$ for each exterior l such that $|\mathcal{R}_l| \ll |S|$, we reduce the number of variables in Model 2.2 and Model 2.3, thereby significantly decreasing both the memory requirements and computation time needed to obtain results.

Moreover, since the partial frontier is dynamically updated—for example, after processing point i, other exteriors like o become interiors of $\mathcal{S}$. When these points are processed later, the procedure outlined in lines 17 to 27 becomes valid for them, allowing their corresponding dominating point to be formulated without the need to solve any LPs.

For cases like point h, if $\mathcal{S} \leftarrow \{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d\}$, no virtual dominating point can be formulated when applying lines 17 to 27. When lines 31 to 37 are then implemented with $\mathcal{R}_h \leftarrow \{\boldsymbol{a}_b, \boldsymbol{a}_d\}$, the result is $\alpha_i^* = 0$, which terminates the while loop, leaving $\mathcal{S}$ unchanged at $\{\boldsymbol{a}_a, \boldsymbol{a}_b, \boldsymbol{a}_d\}$.

It is evident that after processing all points in $\mathcal{A} \setminus \mathcal{S}$, the set $\mathcal{F} \leftarrow \mathcal{S}$ is finalized, as described in line 41.

6.3 Empirical Experiments

To validate the effectiveness of Algorithm 6.1, K is set to be d, and $\mathcal{S}$ is initialized using Lemma 3.1 and Lemma 3.3. The computation environment is the same as in

Chapter 5, and the result is obtained using solely one worker of the CPU.

6.3.1 Running Time Analysis

Table 6.1 shows the computation time of this approach compared to the small-size BH method from Zhuang et al. (2024), with both methods run using a single CPU worker. For example, in the first sample of 10,000 points with 2 inputs and 2 outputs (sample 2&2_10000_1), this approach finishes in 24 seconds, while small-size BH requires 65 seconds, which is represented in Table 6.1 as '24 (65)'. Overall, the time required by our method is about one-third that of small-size BH across all simulated samples.

Figure 6.5 illustrates the trend in computation time of this approach, namely adaptive formulation as density, cardinality, and dimensionality increase. As shown, this method is less sensitive to these factors, with relatively stable time consumption despite the growth in density, cardinality, and dimensionality.

Table 6.1: Running Time (seconds)

m&s	n	Sample Index				
		1	2	3	4	5
2 & 2	10000	24 (65)	26 (66)	24 (70)	31 (78)	25 (67)
	25000	70 (193)	67 (189)	65 (190)	74 (193)	68 (197)
	50000	149 (433)	145 (427)	150 (428)	183 (553)	165 (495)
	100000	339 (1164)	356 (1201)	414 (1420)	349 (1231)	563 (2099)
3 & 3	10000	46 (92)	36 (93)	37 (91)	38 (99)	41 (115)
	25000	89 (284)	92 (272)	84 (277)	99 (285)	108 (340)
	50000	185 (682)	191 (645)	225 (740)	212 (665)	257 (941)
	100000	436 (1731)	445 (1806)	544 (2047)	575 (2138)	491 (1997)
4 & 4	10000	51 (127)	47 (129)	48 (143)	55 (136)	56 (162)
	25000	130 (405)	124 (385)	145 (408)	141 (462)	137 (411)
	50000	296 (911)	267 (966)	296 (1086)	315 (1236)	358 (1243)
	100000	653 (2564)	679 (2559)	711 (2769)	657 (2684)	695 (2833)
5 & 5	10000	74 (193)	74 (207)	70 (212)	73 (218)	77 (244)
	25000	179 (557)	160 (528)	183 (587)	188 (580)	195 (671)
	50000	395 (1417)	378 (1395)	414 (1474)	465 (1942)	450 (1911)
	100000	962 (3812)	923 (3639)	966 (4001)	1251 (5737)	1035 (4577)

The results highlight the efficiency of this approach in reducing computation time by minimizing the necessity of solving LPs and decreasing the number of variables when solving LPs is required. Table 6.2 presents the statistics on the number of LPs solved and the corresponding variables across all simulated samples. For example,

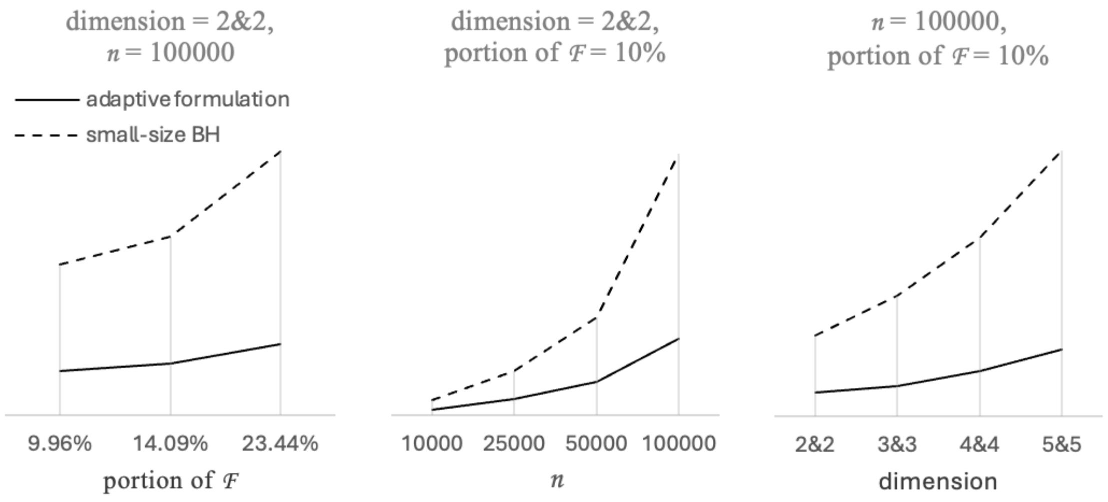


Figure 6.5: Time Comparison (seconds)

in sample 2&2_10000_1, only 1499 LPs are solved, with variables associated with an average of 5.05 relevant frame points, represented as '1499 @5.05' in Table 6.2.

Table 6.2: Number of LPs and Corresponding Variable Counts

m&*s*	*n*	Sample Index 1	2	3	4	5
2 & 2	10000	1499 @5.05	1923 @4.85	1829 @4.98	3532 @5.69	1505 @5.21
	25000	3998 @5.16	3949 @5.34	3829 @5.51	5327 @5.22	3972 @5.44
	50000	7831 @5.25	6786 @5.51	8081 @5.47	16042 @5.91	11437 @6.23
	100000	12762 @5.76	14348 @5.97	22655 @6.31	15502 @6.19	42462 @6.55
3 & 3	10000	5284 @4.81	3212 @5.35	3537 @5.39	3869 @5.54	5129 @6.71
	25000	7215 @6.28	7024 @5.79	6075 @5.72	8714 @5.77	11737 @6.96
	50000	11063 @5.9	11388 @5.99	19671 @6.93	14746 @6.37	26534 @7.32
	100000	19723 @6.52	22459 @6.65	38356 @6.82	39871 @7.58	26067 @6.65
4 & 4	10000	4753 @6.59	4265 @6.75	4717 @7.07	5756 @6.88	6685 @7.58
	25000	11401 @7	10272 @7	13451 @7.14	14240 @7.97	12236 @7.14
	50000	21880 @6.85	18869 @7.42	23328 @7.62	28277 @8.18	32688 @8.28
	100000	36624 @7.61	37746 @7.49	44427 @8.09	36431 @7.61	42995 @7.89
5 & 5	10000	7144 @7.73	7815 @8.22	7341 @8.38	7888 @8.28	8623 @8.56
	25000	14976 @7.92	12605 @8.07	16423 @8.35	16458 @7.97	18160 @8.87
	50000	27383 @8.37	25904 @8.11	29961 @8.78	38396 @9.58	34480 @8.99
	100000	52374 @8.28	47904 @8.11	51873 @8.53	72778 @9.42	56659 @9.26

Using standard computation [6], for sample 2&2_10000_1, 10,000 LPs with over 10,000 variables would need to be solved. This improvement has significantly reduced the LP solutions necessity. Similar results are observed across all samples, further confirming the effectiveness of our approach.

6.3.2 Classification Results

As demonstrated, the proposed approach exhibits significant performance improvements in terms of time performance when compared to the latest small-size BH approach. Table 6.3 summarizes the proportion of non-frame points, calculated as $|\mathcal{D}|/|\mathcal{A}|$, identified using our designated arithmetic dominating point formulation technique. Remarkably, 87.79% of non-frame points were successfully identified, regarding sample 2&2_10000_1, without the need to solve Model 2.2 and Model 2.3. It is evident that the majority of non-frame points are identified through arithmetic operations across all samples. The identified proportion tends to decline as the dimensionality increases, while it rises with increased cardinality. This trend suggests that our proposed approach can be particularly competitive in large-scale computational contexts (e.g., with 1,000,000 points).

Table 6.3: Identified Non-frame Points (%)

$m\&s$	n	Sample Index				
		1	2	3	4	5
2&2	10000	87.79	86.91	85.69	81.17	87.39
	25000	87.14	88.21	88.34	86.20	87.01
	50000	87.45	88.57	87.44	79.58	84.12
	100000	88.91	88.58	84.29	88.43	74.30
3&3	10000	68.12	79.09	75.78	74.66	73.30
	25000	83.64	80.99	83.56	78.78	77.42
	50000	84.88	83.72	80.78	83.22	74.22
	100000	86.12	86.07	80.86	78.56	83.33
4&4	10000	70.36	74.67	73.75	68.41	67.26
	25000	73.67	75.53	69.96	71.90	71.34
	50000	72.71	78.20	73.86	73.36	70.52
	100000	79.17	77.76	77.18	78.93	78.38
5&5	10000	59.46	61.42	62.77	59.77	58.03
	25000	64.40	70.02	65.30	62.63	65.16
	50000	68.37	68.79	68.06	65.16	63.91
	100000	70.14	70.46	70.69	64.48	70.41

Table 6.4 shows that the proposed method effectively identifies frame points by using positive weights when a non-frame point is identified arithmetically, with detection rates ranging from 47.84% to 93.34% across various configurations. Higher rates are observed in the cases of lower dimensions. Also, these detection rates tend to improve with larger cardinality, particularly for configurations with 100,000 points. Variations exist among different samples, indicating that sample characteristics can impact detection performance. Overall, the findings highlight the method's effectiveness while also pointing to challenges posed by increasing dimensionality.

Table 6.4: Identified Frame Points (%)

$m\&s$	n	Sample Index				
		1	2	3	4	5
2&2	10000	86.23	81.59	89.46	66.20	88.06
	25000	93.34	89.76	88.44	78.46	91.57
	50000	92.74	93.06	91.48	84.04	87.20
	100000	93.29	93.70	89.65	91.02	81.60
3&3	10000	68.67	74.84	79.16	72.37	60.59
	25000	72.31	81.06	79.75	69.39	56.00
	50000	82.78	85.32	55.72	71.24	60.16
	100000	88.27	83.78	56.38	75.85	80.66
4&4	10000	66.48	60.82	65.10	56.11	47.66
	25000	67.81	75.36	48.60	60.13	65.28
	50000	72.73	73.64	67.35	48.60	42.56
	100000	72.62	76.47	55.31	71.56	55.19
5&5	10000	63.67	55.62	59.75	56.67	49.56
	25000	71.06	72.06	57.29	61.18	51.43
	50000	70.48	70.87	61.92	48.80	59.02
	100000	63.45	70.07	60.78	47.84	63.51

The effect of K

As analyzed, allowing more iteration steps enables the establishment of additional virtual dominations, thereby reducing the number of LPs that need to be solved. This trend can be observed across all the samples used, and some results are provided to visually illustrate this in Figure 6.6. For example, in Figure 6.6, the x-axis represents the value of K. An increasing trend in the proportion of virtual dominations is evident as K increases, although this growth becomes less pronounced as K continues to rise.

Since the operations outlined in lines 17 to 27 involve only arithmetic computations rather than solving LPs, higher values of K are expected to improve computational performance. However, because the distribution of points around the frame is unknown prior to implementing the proposed procedure, the performance of lines 17 to 27 is output-sensitive. Despite this uncertainty, it is generally reasonable to set a large K in practice to minimize the need for LP solutions.

Summary

In this chapter, we present an adaptive search and formulation strategy aimed at arithmetically constructing virtual points to identify non-frame points that are not dominated by any existing points. The key idea is to formulate a virtual dominating point, which allows the identification of a non-frame point through dimension com-

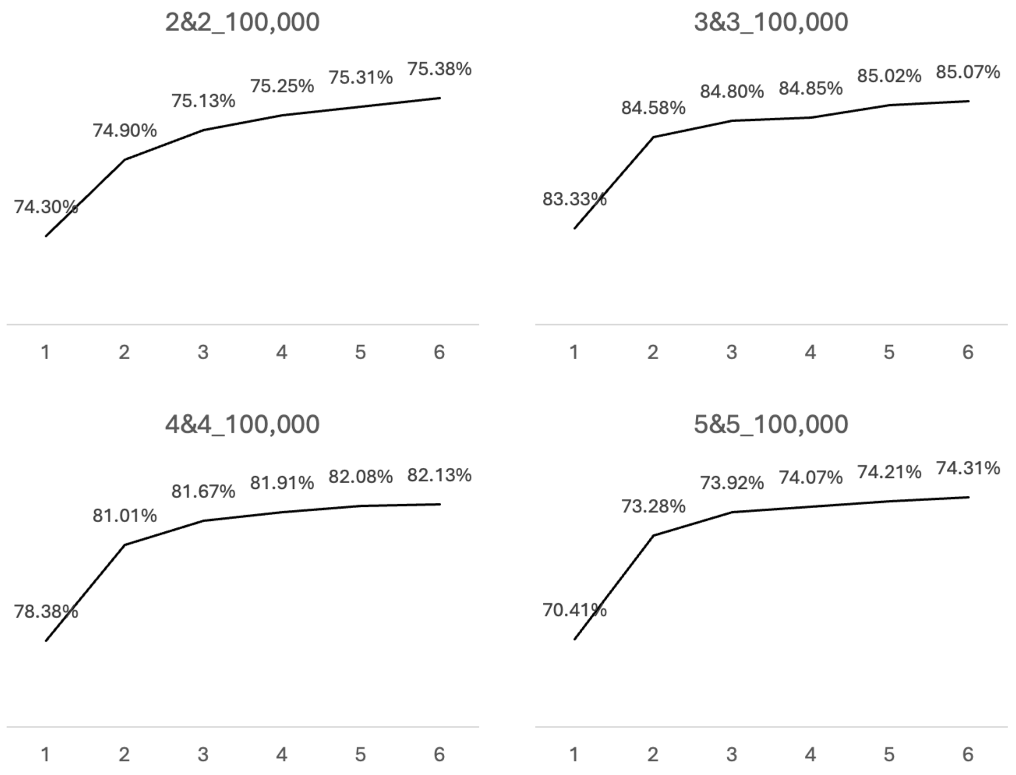


Figure 6.6: Domination Respect to K

parison, as described in the DEA Dominator method. This approach not only helps identify non-frame points but also enables the creation of a small-size reference set , as described in Chapter 5, which can be utilized for each point when solving Models 2.2 and 2.3.

To validate the effectiveness of our approach, we conduct a series of empirical experiments. The results demonstrate that the majority of non-frame points can be identified without the need to solve LP problems. For those points that remain unclassified and require LP solutions, the average number of variables involved is close to the dimension of the DEA dataset, indicating a significant reduction in computational complexity. Furthermore, the time required for solving these LPs is substantially lower compared to the small-size LP approach introduced in the previous chapter, highlighting the performance gains achieved by our proposed method.

Chapter 7

Conclusions and Discussions

This study presents a series of existing and proposed methodologies designed to enhance the computational efficiency of DEA frontier identification in large-scale scenarios.

Chapter 2 establishes the theoretical foundation with a discussion of DEA, highlighting the importance of the DEA frontier. These concepts provide the basis for the developments explored in subsequent chapters. Chapter 3 reviews existing methods and results, offering a detailed analysis that serves as a baseline for the approaches proposed in this study.

Chapter 4 introduces a weight-based categorization scheme to systematically order the computation of DMUs, facilitating the early construction of DEA frames. This approach iteratively gathers frame points by applying distinct weight vectors. Empirical validation demonstrates the method's effectiveness in reducing computational demands.

Building on research into small-scale linear programming (LP), Chapter 5 proposes a novel dimension-wise reference set selection procedure. This geometric method identifies reference DMUs from the nearest relevant facets for the DMU under evaluation, reducing the number of LP formulations and supporting parallel computation frameworks. Empirical results confirm that this approach significantly lowers computation time while preserving accuracy, by solving LPs with the least number of decision variables.

Inspired by the straightforward shortest-distance selection process in Chapter 5, Chapter 6 introduces an adaptive formulation scheme for constructing virtual domination. The benefit of this adaptive formulation is that it identifies non-frame points through direct dimensional comparisons, rather than LP solutions. When LPs are necessary, the average number of variables corresponds to the dataset's dimensionality.

Each of the proposed methodologies offers distinct advantages for large-scale

DEA applications. Their integration with other methods holds potential for further advancements, paving the way for even more robust and efficient DEA computational frameworks.

Key Contributions

The key contributions of this study can be summarized as follows:

1. The study introduces a set of innovative techniques—including weight-based categorization, shortest-distance-based selection, and adaptive formulation—that significantly reduce computational complexity.
2. Each proposed procedure is supported by rigorous theoretical proofs, establishing the validity of the methodologies and providing a solid foundation for further development and adaptation.
3. Beyond enhancing DEA frontier identification, the proposed techniques demonstrate potential applicability to other computational challenges, such as solving the convex hull problem.

Future Directions

There remain opportunities for further exploration:

1. Scalability in ultra-large datasets: Demonstrating the performance of these methods in ultra-large datasets with millions of DEA points would be valuable for real-world applications.
2. Investigation of DEA point distribution: The distribution of DEA points may influence the behavior of the proposed procedures. Further exploration in this area could lead to refinement and improved performance of the methods.
3. Integration with advanced models: Extending the proposed methods to other DEA models, such as dynamic or network DEA, could broaden their applicability.
4. Hybrid approaches: Combining the proposed methods with statistical learning techniques could open up new possibilities for handling highly complex problems, especially in cases involving stochastic or uncertain data.

In conclusion, this study contributes to the advancement of large-scale computation by proposing scalable methods aimed at improving efficiency and reducing resource consumption. While these methods effectively address computational challenges in DEA, they also offer potential for further exploration and refinement in the field of high-performance computing. The approaches presented here may serve as a starting point for developing more general techniques, with possible applications extending to areas such as machine learning, computational geometry, optimization, and other related fields.

Appendix A

Appendix

A.1 Obtaining Frame Points using Arbitary Weights

From Model 2.4, the constraints are:

$$\sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j \boldsymbol{x}_j \leq \theta_l \boldsymbol{x}_l,$$
$$\sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j \boldsymbol{y}_j \geq \boldsymbol{y}_l.$$

For arbitrary weights $v_r \geq 0$ and $w_i \geq 0$, the following holds:

$$w_i \sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j x_{ij} \leq w_i \theta_l x_{il}, \quad \forall i = 1, \dots, m,$$
$$v_r \sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j y_{rj} \geq v_r y_{rl}, \quad \forall r = 1, \dots, s.$$

By summing over all input and output dimensions, we derive:

$$\sum_{r=1}^{s} v_r y_{rl} - \theta_l \sum_{i=1}^{m} w_i x_{il} \leq \sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j \left(\sum_{r=1}^{s} v_r y_{rj} - \sum_{i=1}^{m} w_i x_{ij} \right). \tag{A.1}$$

If point l satisfies:

$$\Delta_l = \sum_{r=1}^{s} v_r y_{rl} - \sum_{i=1}^{m} w_i x_{il} = \max_{j \in \mathcal{I}_\mathcal{A}} \left\{ \sum_{r=1}^{s} v_r y_{rj} - \sum_{i=1}^{m} w_i x_{ij} \right\},$$

then:

$$\Delta_l \geq \sum_{j \in \mathcal{I}_\mathcal{A} \setminus \{l\}} \lambda_j \left(\sum_{r=1}^{s} v_r y_{rj} - \sum_{i=1}^{m} w_i x_{ij} \right) + \lambda_l \Delta_l.$$

Let $\boldsymbol{\pi}^T = \{\pi_1, \ldots, \pi_{m+s}\} = \{w_i, \ldots, w_m, v_r, \ldots, v_s\}$, subject to:

$$\sum_{r=1}^{s} v_r + \sum_{i=1}^{m} w_i > 0. \tag{A.2}$$

To satisfy inequality (A.1) under condition (A.2), the only feasible solution is:

$$\theta_l^* = 1, \quad \lambda_l^* = 1, \quad \lambda_j^* = 0 \quad \forall j \neq l.$$

This indicates that point l is neither dominated by any existing point nor by any virtual point. Thus, point l must lie on the VRS frame.

A.2 Invariance of Constraints Under Data Transformation

The initial DEA constraints are:

$$\sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j x_{ij} \leq \theta_l x_{il}, \quad i = 1, 2, \ldots, m, \tag{A.3}$$

$$\sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j y_{rj} \geq y_{rl}, \quad r = 1, 2, \ldots, s. \tag{A.4}$$

Applying the transformations:

$$x'_{il} = \frac{x_{il}}{\max_{h \in \mathcal{I}_\mathcal{A}} x_{ih}}, \quad y'_{rl} = \frac{y_{rl}}{\max_{h \in \mathcal{I}_\mathcal{A}} y_{rh}},$$

the constraints become:

$$\sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j x'_{ij} \leq \theta_l x'_{il}, \quad i = 1, 2, \ldots, m, \tag{A.5}$$

$$\sum_{j \in \mathcal{I}_\mathcal{A}} \lambda_j y'_{rj} \geq y'_{rl}, \quad r = 1, 2, \ldots, s. \tag{A.6}$$

Substituting the transformations:

$$\sum_{j \in \mathcal{I}_\mathcal{A}} \frac{\lambda_j x_{ij}}{\max_{h \in \mathcal{I}_\mathcal{A}} x_{ih}} \leq \frac{\theta_l x_{il}}{\max_{h \in \mathcal{I}_\mathcal{A}} x_{ih}}, \quad \sum_{j \in \mathcal{I}_\mathcal{A}} \frac{\lambda_j y_{rj}}{\max_{h \in \mathcal{I}_\mathcal{A}} y_{rh}} \geq \frac{y_{rl}}{\max_{h \in \mathcal{I}_\mathcal{A}} y_{rh}}.$$

By dividing both sides by the corresponding maximum values, the transformed constraints (A.5) and (A.6) are shown to be equivalent to the original constraints (A.3) and (A.4). Thus, the transformations preserve the DEA constraints, ensuring consistency between the original and transformed data evaluations.

Bibliography

[1] Agha Iqbal Ali. “Streamlined computation for data envelopment analysis”. In: *European journal of operational research* 64.1 (1993), pp. 61–67.

[2] Randall Balestriero, Zichao Wang, and Richard G Baraniuk. “Deephull: Fast convex hull approximation in high dimensions”. In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2022, pp. 3888–3892.

[3] Rajiv D Banker, Abraham Charnes, and William Wager Cooper. “Some models for estimating technical and scale inefficiencies in data envelopment analysis”. In: *Management science* 30.9 (1984), pp. 1078–1092.

[4] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. “The quickhull algorithm for convex hulls”. In: *ACM Transactions on Mathematical Software (TOMS)* 22.4 (1996), pp. 469–483.

[5] Richard S Barr and Matthew L Durchholz. “Parallel and hierarchical decomposition approaches for solving large-scale data envelopment analysis models”. In: *Annals of Operations Research* 73.0 (1997), pp. 339–372.

[6] Abraham Charnes, William W Cooper, and Edwardo Rhodes. “Measuring the efficiency of decision making units”. In: *European journal of operational research* 2.6 (1978), pp. 429–444.

[7] Chih Cheng Chen. “Measuring internet users’ online activity: An application of the superefficiency data envelopment analysis model”. In: *The Information Society* 31.4 (2015), pp. 315–345.

[8] Chun-Der Chen, Qun Zhao, and Jin-Long Wang. “How livestreaming increases product sales: role of trust transfer and elaboration likelihood model”. In: *Behaviour & Information Technology* 41.3 (2022), pp. 558–573.

[9] Wen-Chih Chen and Wei-Jen Cho. “A procedure for large-scale DEA computations”. In: *Computers & Operations Research* 36.6 (2009), pp. 1813–1824.

[10] Wen-Chih Chen and Sheng-Yung Lai. “Determining radial efficiency with a large data set by solving small-size linear programs”. In: *Annals of Operations Research* 250 (2017), pp. 147–166.

[11] Yao Chen and Agha Iqbal Ali. "Output–input ratio analysis and DEA frontier". In: *European Journal of Operational Research* 142.3 (2002), pp. 476–479.

[12] Junfei Chu et al. "A general computational framework and a hybrid algorithm for large-scale data envelopment analysis". In: *European Journal of Operational Research* 316.2 (2024), pp. 639–650.

[13] William W Cooper, Lawrence M Seiford, Kaoru Tone, et al. *Data envelopment analysis: a comprehensive text with models, applications, references and DEA-solver software*. Vol. 2. Springer, 2007.

[14] William W Cooper, Lawrence M Seiford, and Joe Zhu. "Handbook on data envelopment analysis". In: (2011).

[15] Debora Di Caprio and Francisco J Santos-Arteaga. "A novel perception-based DEA method to evaluate alternatives in uncertain online environments". In: *Computers & Industrial Engineering* 131 (2019), pp. 327–343.

[16] JH Dulá and FJ López. "DEA with streaming data". In: *Omega* 41.1 (2013), pp. 41–47.

[17] JH Dulá and RM Thrall. "A computational framework for accelerating DEA". In: *Journal of Productivity Analysis* 16 (2001), pp. 63–78.

[18] José H Dulá. "A geometrical approach for generalizing the production possibility set in DEA". In: *Journal of the Operational Research Society* 60.11 (2009), pp. 1546–1555.

[19] José H Dulá. "An algorithm for data envelopment analysis". In: *INFORMS Journal on Computing* 23.2 (2011), pp. 284–296.

[20] José H Dulá, Richard V Helgason, and Betty L Hickman. "Preprocessing schemes and a solution method for the convex hull problem in multidimensional space". In: *Computer Science and Operations Research*. Elsevier, 1992, pp. 59–70.

[21] José H Dulá and Francisco J López. "Algorithms for the frame of a finitely generated unbounded polyhedron". In: *INFORMS Journal on Computing* 18.1 (2006), pp. 97–110.

[22] José H Dulá and Francisco J López. "Preprocessing dea". In: *Computers & Operations Research* 36.4 (2009), pp. 1204–1220.

[23] Laura Escobar and Kiumars Kaveh. "Convex polytopes, algebraic geometry, and combinatorics". In: *Notices of the American Mathematical Society* 67.8 (2020).

[24] Rolf Färe, Shawna Grosskopf, and Gerald Whittaker. “Network dea”. In: *Modeling data irregularities and structural complexities in data envelopment analysis* (2007), pp. 209–240.

[25] MA Jayaram and Hasan Fleyeh. “Convex hulls in image processing: a scoping review”. In: *American Journal of Intelligent Systems* 6.2 (2016), pp. 48–58.

[26] Tao Jie. “Parallel processing of the Build Hull algorithm to address the large-scale DEA problem”. In: *Annals of Operations Research* 295.1 (2020), pp. 453–481.

[27] Dariush Khezrimotlagh et al. “Data envelopment analysis and big data”. In: *European Journal of Operational Research* 274.3 (2019), pp. 1047–1054.

[28] Christer Kiselman. “A semigroup of operators in convexity theory”. In: *Transactions of the American Mathematical Society* 354.5 (2002), pp. 2035–2053.

[29] H. W. Kuhn and A. W. Tucker. “Nonlinear programming”. In: *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability.* University of California Press, 1951, pp. 481–492.

[30] Sungmook Lim, Kwang Wuk Oh, and Joe Zhu. “Use of DEA cross-efficiency evaluation in portfolio selection: An application to Korean stock market”. In: *European journal of operational research* 236.1 (2014), pp. 361–368.

[31] Ali Mohtashami and Bahram Mohammadkhani Ghiasvand. “Z-ERM DEA integrated approach for evaluation of banks & financial institutes in stock exchange”. In: *Expert Systems with Applications* 147 (2020), p. 113218.

[32] Hiroshi Morita, Koichiro Hirokawa, and Joe Zhu. “A slack-based measure of efficiency in context-dependent data envelopment analysis”. In: *Omega* 33.4 (2005), pp. 357–362.

[33] Muren, Lining Hao, and Qingxian An. “Efficiency evaluation of very large-scale samples:: Data envelopment analysis with angle-index synthesis”. In: *Computers and Operations Research* (2024).

[34] AP Nemirko and JH Dulá. “Machine learning algorithm based on convex hull analysis”. In: *Procedia Computer Science* 186 (2021), pp. 381–386.

[35] AP Nemirko and JH Dulá. “Machine learning algorithm based on convex hull analysis”. In: *Procedia Computer Science* 186 (2021), pp. 381–386.

[36] Ole B Olesen and Niels Christian Petersen. “Stochastic data envelopment analysis—A review”. In: *European journal of operational research* 251.1 (2016), pp. 2–21.

[37] Eleanor G Rieffel and Wolfgang H Polak. *Quantum computing: A gentle introduction.* MIT press, 2011.

[38] Hossein Sartipizadeh and Tyrone L Vincent. "Computing the approximate convex hull in high dimensions". In: *arXiv preprint arXiv:1603.04422* (2016).

[39] Raimund Seidel. "Convex hull computations". In: *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, 2017, pp. 687–703.

[40] Min Tang et al. "GPU accelerated convex hull computation". In: *Computers & Graphics* 36.5 (2012), pp. 498–506.

[41] Kun Tian, Xin Zhao, and Stephen S-T Yau. "Convex hull analysis of evolutionary and phylogenetic relationships between biological groups". In: *Journal of theoretical biology* 456 (2018), pp. 34–40.

[42] Kaoru Tone and Miki Tsutsui. "Network DEA: A slacks-based measure approach". In: *European journal of operational research* 197.1 (2009), pp. 243–252.

[43] Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry*. CRC press, 2017.

[44] Yi Wu et al. "Efficiency estimation of urban metabolism via Emergy, DEA of time-series". In: *Ecological Indicators* 85 (2018), pp. 276–284.

[45] Zhengwei Yang and Fernand S Cohen. "Image registration and object recognition using affine invariants and convex hulls". In: *IEEE Transactions on Image Processing* 8.7 (1999), pp. 934–946.

[46] Roozbeh Yousefzadeh. "Deep learning generalization and the convex hull of training sets". In: *arXiv preprint arXiv:2101.09849* (2021).

[47] Xuyao Zhang and Dayu Xu. "Assessing the eco-efficiency of complex forestry enterprises using LCA/time-series DEA methodology". In: *Ecological Indicators* 142 (2022), p. 109166.

[48] Qianwei Zhuang, Dariush Khezrimotlagh, and Hiroshi Morita. "Accelerating large-scale DEA computation using sequential categorization and dynamic reference set selection". In: *INFOR: Information Systems and Operational Research* (2024).