

May 1, 1999.

Detecting the impact of including and omitting an attribute in DEA.

J.H. Dulá¹ and F.J. López²

The University of Mississippi

University, MS 38677

The University of Texas at El Paso

El Paso, TX 79968-0544

November 2001

ABSTRACT. We present procedures that detect the impact of including or omitting an input or output attribute in a DEA model. These procedures can serve several purposes including providing information about the robustness of a model; helping decide whether to include or omit specific attributes in subsequent models; and preprocess the data prior to a full analysis with a modified model. The procedure for the case of the inclusion of a new attribute provides information for the full set of standard DEA models; that is, DEA under the constant, variable, increasing, and decreasing returns assumptions. Computational testing reveals that, with a fraction of the work required for a full re-solving of LPs, we can obtain a good idea of the impact on a new model of the modification in the list of input and outputs.

Key Words: DEA, DEA computations, linear programming, and convex analysis.

1. Introduction. After the completion of a DEA study which uses a specific list of input and output attributes for the DMUs, the question may arise about the impact on the efficiency classifications of including a new attribute or omitting one already in the model. The question can be answered, of course, by re-solving the modified LPs. However, the analyst may want to obtain

¹ Corresponding Author: School of Business, The University of Mississippi, University, MS 38677; email: jdula@olemiss.edu. This author's contribution was funded by grant N00014-98-1-0766 from the Office of Naval Research.

²fjlopez@utep.edu. College of Business Administration, University of Texas at El Paso, El Paso, TX 79912.

an idea of the impact of including a new attribute from a long list of possible choices or about the omission of any of the current ones. Testing this impact requires considerable computational effort if there are many candidates for inclusion or omission. The same occurs if the application is so dynamic that it requires adding or omitting attributes frequently. This is why preprocessors can be useful. Efficient preprocessors are computationally cheap, yet they provide valuable information. Examples of preprocessors applicable to DEA include Dulá, Helgason, and Hickman [1992] and Rosen, Xue, and Phillips [1992]. Barr and Durchholz [1997] also preprocess the information to identify and remove inefficient DMUs from the LP systems before obtaining the list of efficient DMUs for a large scale DEA analysis.

This paper presents preprocessors for the cases of adding or omitting an attribute from a DEA study. We will proceed as follows. We will present theoretical foundations for the cases of inclusion and exclusion of an attribute and then we will offer a preprocessor for each case.

2. Theoretical underpinnings.

2.1. Case 1: Including a new attribute. We will derive the procedure for the case of the variable returns to scale model. Later we show how to extract information relative to the constant, increasing, and decreasing returns to scale DEA models.

The data consist of n points, a^j , $j = 1, \dots, n$; one for each DMU in the model. The data points are composed of two parts, as follows:

$$a^j = \begin{bmatrix} -X^j \\ Y^j \end{bmatrix} \in \mathfrak{R}^m; \quad j = 1, \dots, n;$$

where $0 \neq X^j \geq 0$ and $0 \neq Y^j \geq 0$ are the input and output data vectors, respectively, for Decision Making Unit (DMU) j . The i th coordinate of a^j is denoted by a_i^j . Let us collect the data points in the set \mathcal{A} ; i.e., $\mathcal{A} = \{a^1, \dots, a^n\}$. Recall that the same data set \mathcal{A} defines the four standard production possibility sets corresponding to the four economic assumptions governing the returns to scale of the transformation process. These four production possibility sets are given next:

Constant Returns:

$$\mathcal{P}^{\text{CR}} = \left\{ z \in \mathfrak{R}^m \mid z \leq \sum_j a^j \lambda_j; \quad \text{s.t. } \lambda_j \geq 0; \forall j \right\};$$

Variable Returns:

$$\mathcal{P}^{\text{VR}} = \left\{ z \in \mathbb{R}^m \mid z \leq \sum_j a^j \lambda_j; \text{ s.t. } \sum_j \lambda_j = 1, \lambda_j \geq 0; \forall j \right\};$$

Increasing Returns:

$$\mathcal{P}^{\text{IR}} = \left\{ z \in \mathbb{R}^m \mid z \leq \sum_j a^j \lambda_j; \text{ s.t. } \sum_j \lambda_j \leq 1, \lambda_j \geq 0; \forall j \right\};$$

Decreasing Returns:

$$\mathcal{P}^{\text{DR}} = \left\{ z \in \mathbb{R}^m \mid z \leq \sum_j a^j \lambda_j; \text{ s.t. } \sum_j \lambda_j \geq 1, \lambda_j \geq 0; \forall j \right\}.$$

These sets are finitely generated polyhedral sets different from each other.

Without loss of generality, suppose that the DEA data for a new input or output attribute appears as the $(m + 1)$ st coordinate of the data points; that is, the new data points become: $(a^j, a_{m+1}^j)^T \in \mathbb{R}^{m+1}; j = 1, \dots, n$.

In this work $\langle a, b \rangle$ stands for the inner product between vectors a and b and $\mathcal{H}(\pi, \beta)$ is the hyperplane defined by vector π with level value β . Now we proceed with a mathematical demonstration.

RESULT 1. *If a DMU has an efficiency score of one in dimension m , then it has an efficiency score of one in dimension $m + 1$.*

PROOF.

Let \mathcal{A} be a set of n DMUs in \mathcal{R}^m . Assume the efficiency score of $a^{\hat{j}} \in \mathcal{A}$ in dimension m is one. Then, there exist $\hat{\pi} \in \mathcal{R}^m$ and $\rho \in \mathcal{R}$ such that $\langle \hat{\pi}, a^{\hat{j}} \rangle + \rho = 1$ and $\langle \hat{\pi}, a^j \rangle + \rho \leq 1, j = 1, \dots, n$ (remember that ρ is just a translation parameter associated to the variable returns to scale model). Thus, $\langle (\hat{\pi}^T, 0)^T, (a^{\hat{j}}, a_{m+1}^{\hat{j}})^T \rangle + \rho = 1$, and $\langle (\hat{\pi}^T, 0)^T, (a^j, a_{m+1}^j)^T \rangle + \rho \leq 1, j = 1, \dots, n$. Then, the efficiency score of $a^{\hat{j}}$ in dimension $m + 1$ is one. □

Inefficient DMUs in dimension m may remain inefficient or may become efficient (see Figure 1 for an example). Therefore, the set of efficient DMUs in dimension $m + 1$ contains all efficient DMUs in \mathcal{R}^m and, possibly DMUs that are inefficient in \mathcal{R}^m .

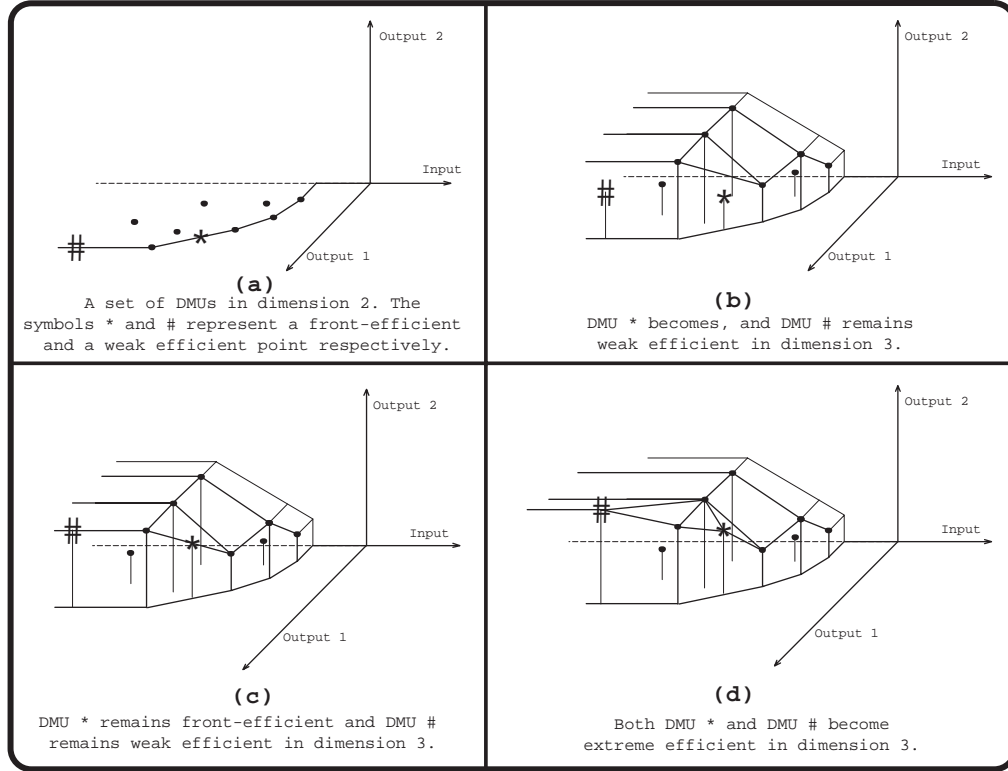


Figure 1. A set of DMUs and the corresponding efficient frontier.

At this point we need to differentiate among points with efficiency score of one in \mathcal{R}^m . These points belong to one of three mutually exclusive classes: *i*) points that correspond to extreme points of the production possibility set, *ii*) points that are technically efficient but are not extreme (points on the efficient frontier that can be expressed as a convex combination of extreme points) and, *iii*) weakly efficient points. We will call points in these categories *extreme-efficient points*, *front-efficient points*, and *weakly efficient points* respectively. In the remainder of this work, an efficient point will be a point in any of these categories, even though, when necessary, we will precisely indicate the corresponding class. Some characteristics of each of these categories follow.

RESULT 2. An extreme-efficient point in \mathcal{R}^m is extreme-efficient in \mathcal{R}^{m+1} .

PROOF.

Assume $a^{\hat{j}} \in \mathcal{A}$ is extreme-efficient in \mathcal{R}^m . Then the following system for any k points in \mathcal{A} ($2 \leq k \leq n - 1$) different from $a^{\hat{j}}$ and from each other is infeasible.

$$\begin{aligned} a^{\hat{j}} &\leq a^{j_1}x_1 + a^{j_2}x_2 + \dots + a^{j_k}x_k, \\ x_i &\geq 0, i = 1, \dots, k; \text{ and } \sum_{i=1}^k x_i = 1. \end{aligned}$$

Hence, the following system is also infeasible.

$$\begin{aligned} a^{\hat{j}} &\leq a^{j_1}x_1 + a^{j_2}x_2 + \dots + a^{j_k}x_k, \\ a_{m+1}^{\hat{j}} &\leq a_{m+1}^{j_1}x_1 + a_{m+1}^{j_2}x_2 + \dots + a_{m+1}^{j_k}x_k, \\ x_i &\geq 0, i = 1, \dots, k; \text{ and } \sum_{i=1}^k x_i = 1. \end{aligned}$$

With this, and RESULT 1 we can conclude that $a^{\hat{j}}$ is extreme efficient in \mathcal{R}^{m+1} . □

Front-efficient points in \mathcal{R}^m may become weakly efficient, remain front-efficient, or change into extreme-efficient points; that is, they remain efficient points. This is depicted in Figure 1 parts (b), (c), and (d) respectively.

Finally, weakly efficient points either remain weakly efficient or become extreme-efficient, which is illustrated in Figure 1 parts (b), (c), and (d). A condition for a point to be front-efficient is that it can be expressed as a convex combination of extreme-efficient points. An argument similar to the one employed for RESULT 2 can be used to prove that, since weakly efficient points in \mathcal{R}^m are not a convex combination of extreme-efficient points, they cannot become a convex combination of extreme-efficient points in \mathcal{R}^{m+1} and thus they cannot be front-efficient in \mathcal{R}^{m+1} .

We proceed with the case of omitting an attribute from a DEA model, that is, when passing from dimension m to dimension $m - 1$.

2.2. Case 2: Omitting an attribute. Analyzing the impact of omitting an attribute is easier than analyzing the case of adding an attribute. The reason is that only efficient points in dimension m need to be re-tested. As can be seen in Figure 1, efficient points in \mathcal{R}^m may remain efficient or become inefficient points in \mathcal{R}^{m-1} . On the other hand, the contrapositive of RESULT 1 can be used to show that inefficient DMUs in dimension m must be inefficient in dimension $m - 1$. This

fact may reduce the computational efforts dramatically, especially when the proportion of efficient DMUs (referred to as the “density” of the data set in other work) is low. Thus, not only the number of LPs to resolve is likely to decrease, but their size may also reduce since the LPs need only contain the data of efficient points in dimension m .

We continue now with the presentation of the preprocessors.

3. Preprocessors For Detecting the Impact of Adding or Omitting an Attribute.

As above, we will start with the case of adding a new attribute. We will present the corresponding procedure and then we will discuss implementation issues; computational results from our simulations; and how to adapt the preprocessor to the constant returns, increasing returns, and decreasing returns to scale models. After presenting all this, we will continue with the procedure for the case of omitting an attribute from a DEA model.

3.1. Case 1: Including a new attribute.

Consider an extreme-efficient DEA data point $a^{j^*} \in \mathcal{R}^m$. It follows from RESULT 2 that a^{j^*} is extreme efficient in \mathcal{R}^{m+1} . A supporting hyperplane in \mathfrak{R}^{m+1} , $\mathcal{H}((\tilde{\pi}^T, \tilde{\gamma})^T, \tilde{\beta})$ for the polyhedral set \mathcal{P}^{VR} at the new point $(a^{j^*T}, a_{m+1}^{j^*})^T$, is such that

$$\langle \tilde{\pi}, a^{j^*} \rangle + \tilde{\gamma} a_{m+1}^{j^*} = \tilde{\beta} \quad (1)$$

$$\langle \tilde{\pi}, a^j \rangle + \tilde{\gamma} a_{m+1}^j \leq \tilde{\beta}; \quad j = 1, \dots, n. \quad (2)$$

Any “rotation” of this hyperplane with respect to the $(m+1)$ st axis at the point $(a^{j^*T}, a_{m+1}^{j^*})^T$, such that the hyperplane remains a support, has the form

$$\langle \tilde{\pi}, a^{j^*} \rangle + \gamma^* a_{m+1}^{j^*} = \beta^*; \quad (3)$$

$$\langle \tilde{\pi}, a^j \rangle + \gamma^* a_{m+1}^j \leq \beta^*; \quad j = 1, \dots, n; \quad (4)$$

where β^* and γ^* are controllable parameters (although not necessarily completely free). From (3) and (4), it follows that

$$\langle \tilde{\pi}, a^j \rangle + \gamma^* a_{m+1}^j \leq \langle \tilde{\pi}, a^{j^*} \rangle + \gamma^* a_{m+1}^{j^*}; \quad j = 1, \dots, n.$$

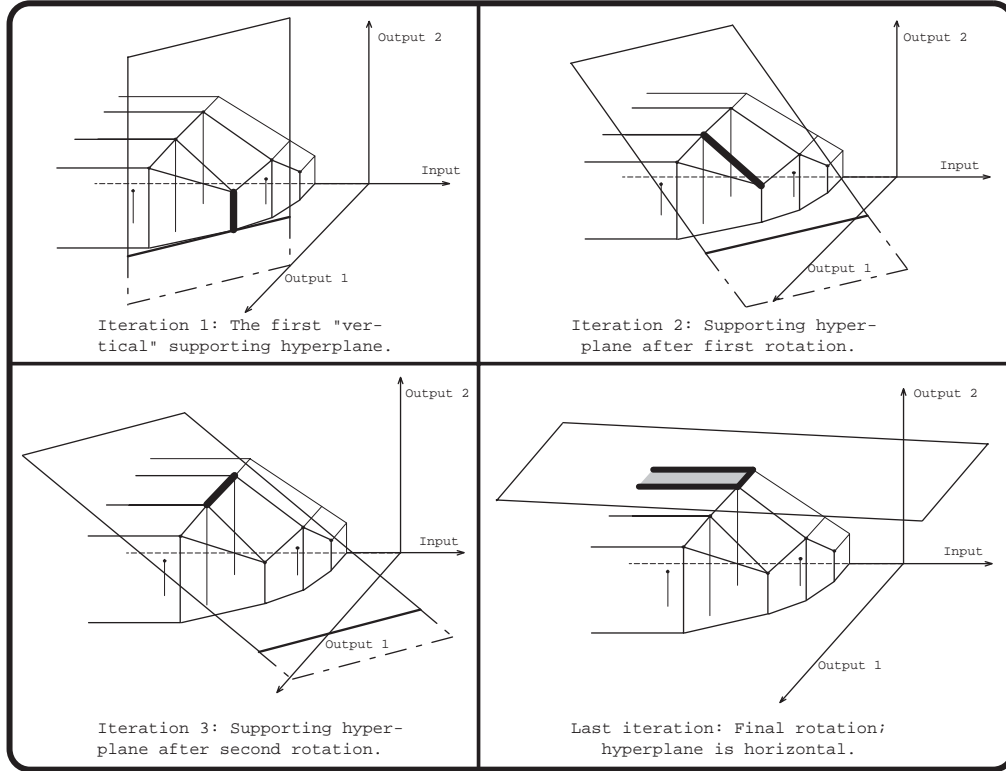


Figure 2. A sequence of supporting hyperplanes.

Solving for γ^* , when $(a_{m+1}^j - a_{m+1}^{j^*}) > 0$, we obtain:

$$\gamma^* \leq \frac{\langle \tilde{\pi}, a^{j^*} \rangle - \langle \tilde{\pi}, a^j \rangle}{(a_{m+1}^j - a_{m+1}^{j^*})}, \quad (5)$$

The maximum rotation occurs at a point $a^{\hat{j}} \neq a^{j^*}$ where γ^* equals the right-hand side in (5), so long as (5) holds for every point $a^j \in \mathcal{A}$ for which $(a_{m+1}^j - a_{m+1}^{j^*}) > 0$ (if there does not exist such a point, the maximum rotation occurs when the hyperplane supports the polyhedral set at a face orthogonal to the $(m + 1)$ st axis defined by one or more directions of recession). The second parameter, β^* , is now uniquely specified by (3). The new hyperplane, $\mathcal{H}((\tilde{\pi}^T, \gamma^*)^T, \beta^*)$ supports the production possibility set both at a^{j^*} and $a^{\hat{j}}$.

Our procedure is based on the idea of generating a sequence of supporting hyperplanes that “climb-up” the polyhedral set with respect to the new dimension. Figure 2 illustrates this, considering “output 2” as the $(m + 1)$ st dimension.

As can be seen in this figure, new extreme points are “visited” by the supporting hyperplane as it climbs. Also, each rotation takes place at a different extreme point.

Next we present a pseudo-code of this procedure, which we call **HyperClimb**.

3.1.1. Procedure HyperClimb.

PROCEDURE **HyperClimb**

[DATA:] The n DEA data points $a^j \in \mathfrak{R}^m$ and an n -dimensional vector $(a_{m+1}^1, \dots, a_{m+1}^n)^T$ (containing the $(m+1)$ st attribute).

Step 0. Initialization.

- i. $k \leftarrow 0$.
- ii. Select arbitrary non-zero finite multipliers $0 \leq \tilde{\pi} \in \mathfrak{R}^m$.
- iii. $j_k^* \leftarrow \underset{j=1, \dots, n}{\operatorname{argmax}}(\langle \tilde{\pi}, a^j \rangle)$.

In case of ties, select j_k^* such that $a_{m+1}^{j_k^*}$ is max.

Break further ties arbitrarily.

Note: $a^{j_k^*}$ is efficient in original model and hence also in dimension $(m+1)$.

Step 1. $k \leftarrow k + 1$.

$$\gamma_k^* = \begin{cases} \min_{j=1, \dots, n} \frac{\langle \tilde{\pi}, a^{j_{k-1}^*} \rangle - \langle \tilde{\pi}, a^j \rangle}{(a_{m+1}^j - a_{m+1}^{j_{k-1}^*})}, & \text{if } (a_{m+1}^j - a_{m+1}^{j_{k-1}^*}) > 0; \\ M \text{ (large number)}, & \text{otherwise.} \end{cases}$$

If $\gamma_k^* = M$, STOP. Otherwise go to Step 2.

Step 2. Find RHS: $\beta^k \leftarrow \langle \tilde{\pi}, a^{j_{k-1}^*} \rangle + \gamma_k^* a_{m+1}^{j_{k-1}^*}$.

Step 3. Define $J_k = \{j | \langle \tilde{\pi}, a^j \rangle + \gamma_k^* a_{m+1}^j = \beta^k\}$.

Note: a^j for $j \in J_k$ is on the boundary in dimension $(m+1)$.

Step 4. Select j_k^* such that $a_{m+1}^{j_k^*}$ is max for all $j \in J_k$. Break ties arbitrarily.

Step 5. Go to step 1.

END PROCEDURE HyperClimb.

3.1.2. Implementation. Since the selection of the multipliers π is arbitrary, it is possible to repeat the procedure over and over using different supporting hyperplanes, approaching the polyhedral set and climbing it from different directions.

To implement **HyperClimb**, we propose using the m unit vectors and the vector $(1, 1, \dots, 1)^T$ as multipliers π (each π defining a different hyperplane). Thus, we approach and climb the production possibility set from $m+1$ different directions. Our findings in this work, which we start discussing next, are based on this implementation approach.

3.1.3. Computational results. Given our implementation of the procedure **HyperClimb**, the next objective of this work was to test its performance. The way we measured this performance was by comparing the results of **HyperClimb** to those of re-solving the modified LPs considering two dimensions: time consumed, and percentage of new efficient DMUs found by each approach. In the remainder of this work, we will understand that **Naive** is an approach that consists in re-solving the LPs which are modified (expanded) by adding the data corresponding to the new attribute in the model.

The performance of **HyperClimb** and **Naive** were compared using randomly generated data as follows.

The sets of data we considered have cardinalities of 500, 1000, 2500, 5000, and 10000 DMUs, and increase their dimension from four ($m = 4$) to five ($m + 1 = 5$). For each of these cardinalities we had 15 different data sets; 75 sets in total.

We conducted our analysis on a SGI Origin 2000 computer at the University of Mississippi with 64 R12000 processing units, 32 gigabytes of memory, a CPU clock speed of 300 MHz, and a word size of 64 bits. To do this, we developed a Fortran code to measure the performance of both

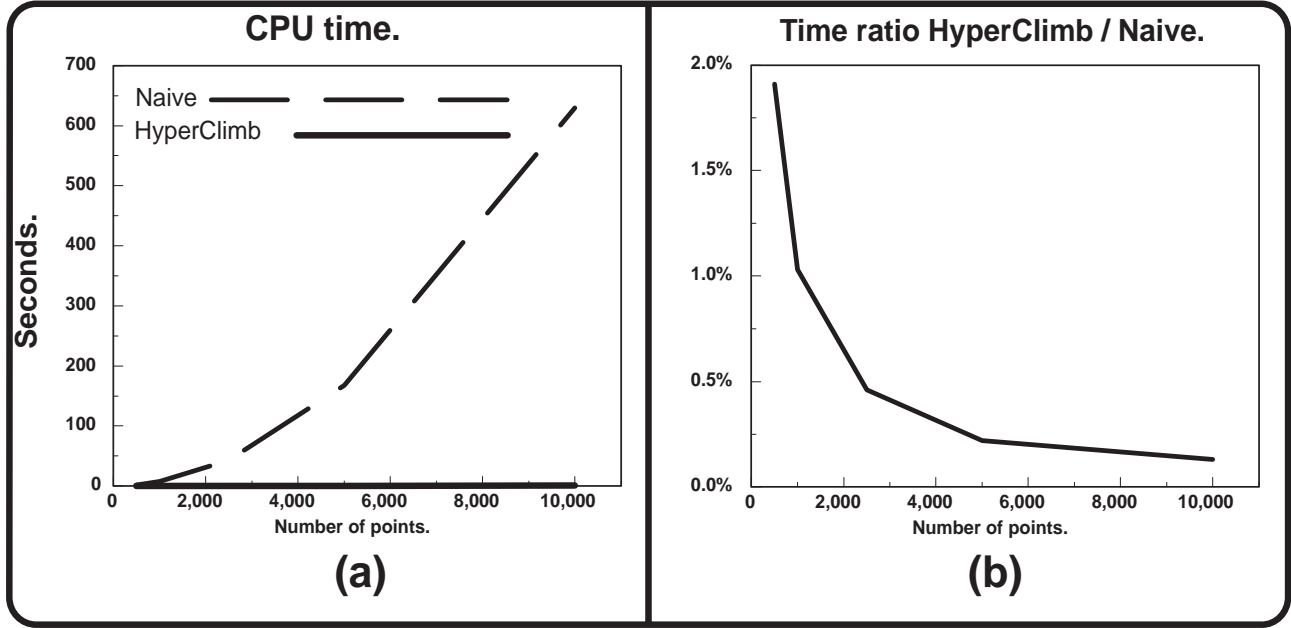


Figure 3. Times comparisons.

HyperClimb and **Naive**. To solve the LPs, we used the subroutine “ddlprs” from the IMSL library (1994). This Fortran code registers both the number of new efficient DMUs found by both procedures and the time required in each case. Recall that **Naive** always finds all new extreme points. The times we report here are the average of three different runs for each set. These runs (75 sets x 3 runs = 225) were executed in no particular order (randomly).

Table 1 shows the characteristics of our data files and the results of running the Fortran code. The cardinality of each set is indicated after the word “by” in the name of the files.

We generated Figures 3, 4, 5, and 6 with the average of the 15 data sets corresponding to each cardinality. Figure 3 refers to computer time. Part (a) presents the average CPU seconds consumed by both **Naive** and **HyperClimb** depending on the cardinality of the sets. **HyperClimb** clearly outperforms **Naive**, which is further shown in part (b) where one can see that for sets with 500 points, **HyperClimb** only needs about 2% of the time required by **Naive**. For cardinalities of 10,000 points, the difference is near three orders of magnitude (close to 0.1% of time).

A caveat for **HyperClimb** is that it does not identify all the extreme points that become efficient in \mathcal{R}^{m+1} . Figure 4 shows that, on average, **HyperClimb** identifies about

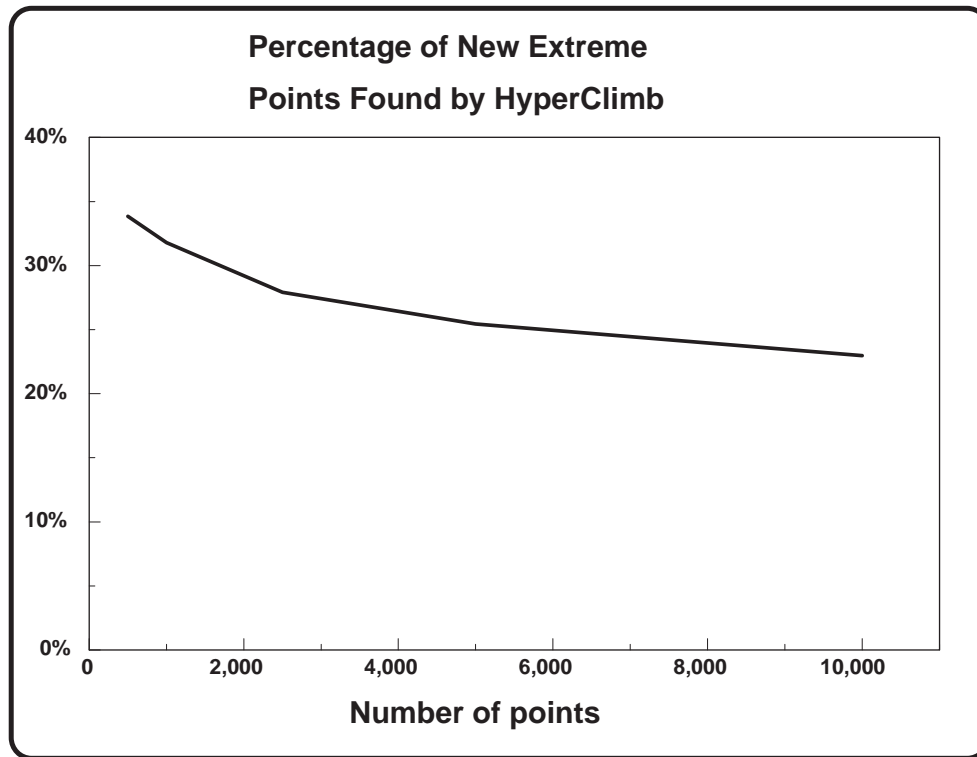


Figure 4. Percentage of new extreme points found by **HyperClimb**.

34% of the new extreme points on data sets with 500 points. This percentage decreases at a decreasing rate to become about 23% when the cardinality of the set is 10,000. Roughly, as the cardinality doubles, the percentage of new extreme points found drops to about 90% of the previous value; i.e., 25.43% for 5000 points and 22.97% for 10,000 points (This seems to resemble the behavior of learning curves). Despite this decay, given the relative little time that **HyperClimb** consumes, we consider the cost of finding extreme points using this procedure as “free.”

In order to further explore the ideas above, we calculated the ratio number-of-new-extreme-points-found to time-required-to-execute-the-procedure for both **HyperClimb** and **Naive**. We refer to this ratio as the *yield* of the corresponding procedure. Figure 5 part (a), shows that **HyperClimb** yields about 130 and 22 new extreme points per CPU second, on data sets with 500 and 10,000 points respectively. Likewise, **Naive** yields 8 and 0.12 new extreme points per CPU second, on data sets with 500 and 10,000 points respectively. Part (b) of the graph shows that

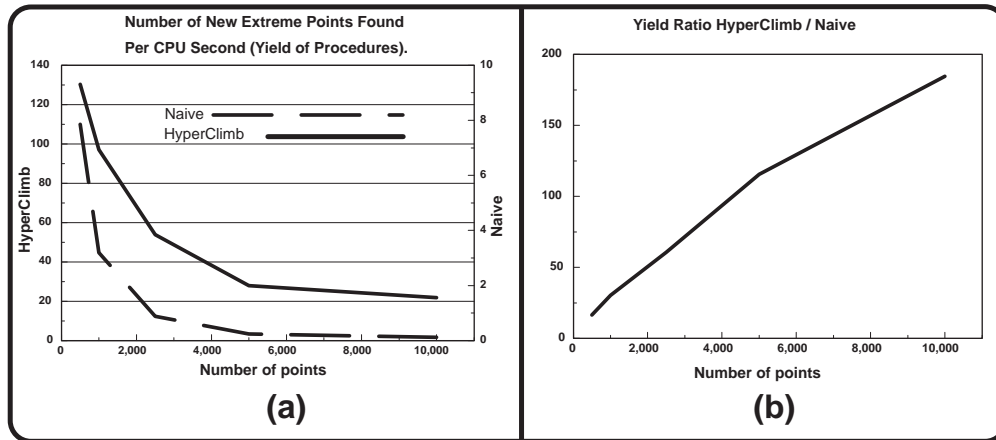


Figure 5. Yield of procedures.

the yield of **HyperClimb** compared to that of **Naive**, increases at an almost constant rate as the cardinality of the data sets grows.

Figure 6 shows the CPU time that is necessary, on average, to evaluate each DMU in the data set when applying either procedure. The graph suggests that the average CPU time per point in the case of **HyperClimb** is almost constant (the lowest is 0.0728 thousands of a second whereas the highest is 0.0803). As a result, we can expect the total time needed by **HyperClimb** to be an almost linear relation of the type $t = \hat{c}n$, where t , \hat{c} , and n are the total time, a constant, and the cardinality of the set. The case of **Naive** is different. The average CPU time per point seems to be a linear function of the form $\bar{t} = \tilde{c}n$, where \bar{t} , \tilde{c} , and n are the average time, a constant, and the cardinality of the data set respectively. This means that total CPU times to solve LPs are quadratic on n when both the size and the number of LPs to solve increase: $t = \tilde{c}n^2$. This observation agrees with what we reported in a previous paper (Dulá and López [2002?] ***MATH PROGRAMMING, HOPEFULLY***).

3.1.4. The case of Constant Returns to Scale (CRS), Increasing Returns to Scale (IRS), and Decreasing Returns to Scale (DRS). The parameter β^* from (3) is the key to identify new extreme points when applying **HyperClimb** to CRS, IRS, or DRS. This is because the value of β^* increases with each rotation of the climbing hyperplane and, at some point, either it changes from negative to positive, or it takes the value zero. In the first case, illustrated in Figure 7, extreme points under CRS are the points that belong to both the hyperplane corresponding to the last negative β^* , and the hyperplane corresponding to the first positive β^* . Extreme points under

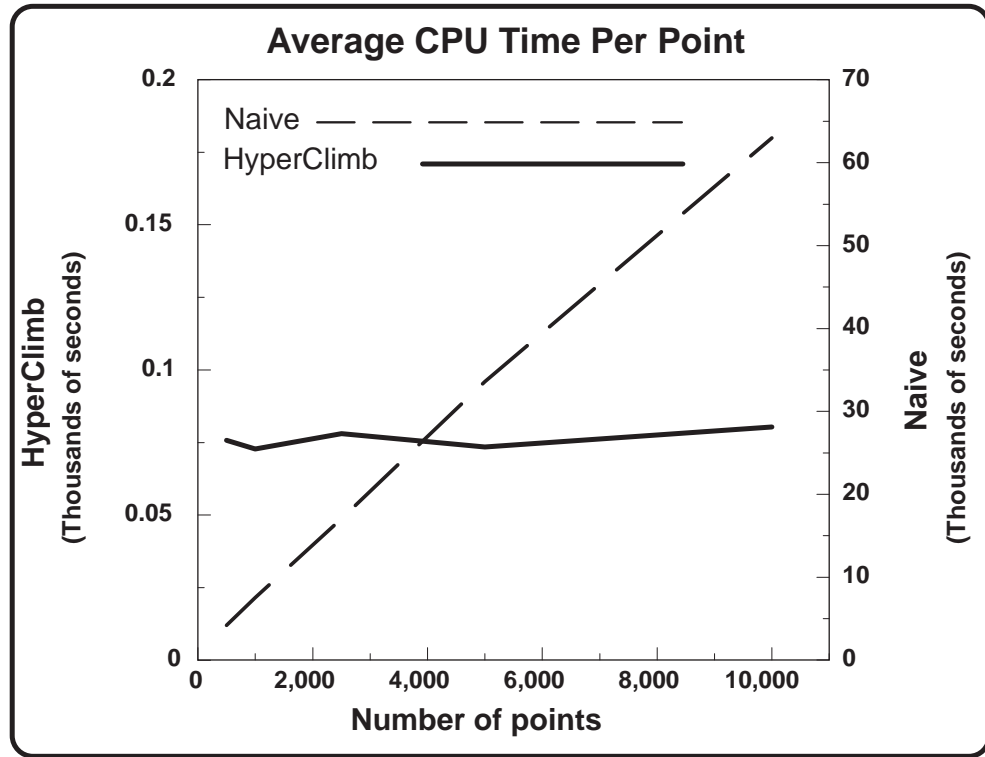


Figure 6. Average Time Per Point.

IRS are the extreme points under CRS plus all extreme points visited by **HyperClimb** while β^* is positive. Finally, extreme points under DRS are all extreme points under CRS plus extreme points touched by **HyperClimb** hyperplanes while β^* is negative.

For the latter case, when $\beta^* = 0$, the extreme points under CRS are the points that belong to the hyperplane $\mathcal{H}(\gamma^*, \beta^*)$, and the cases for IRS and DRS are the same as above.

With this we conclude our study of the case corresponding to adding a new attribute. Next we discuss the opposite situation: deleting a current attribute from a model.

3.2. Case 2: Omitting an attribute. Often, the optimal solution corresponding to an efficient DMU in $\mathcal{R}^m, a^{\hat{j}}$, readily supplies information about the efficiency status of $a^{\hat{j}}$ in \mathcal{R}^{m-1} as is shown next.

RESULT 3. Assume $a^{\hat{j}} \in \mathcal{A}$ is efficient and the corresponding optimal solution is such that the k th optimal multiplier is zero. Then $a^{\hat{j}}$ is efficient when removing the k th attribute from the DEA model.

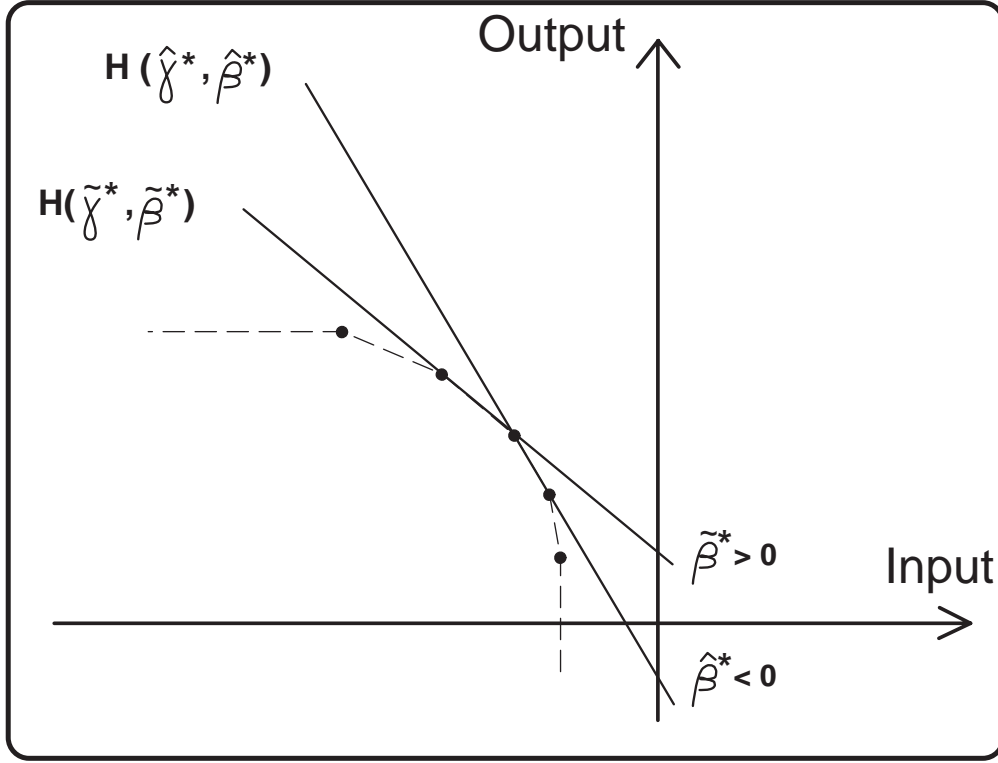


Figure 7. A rotation of a supporting hyperplane where the value of β^* changes from negative to positive.

PROOF.

Let $a^{\hat{j}} \in \mathcal{A}$ be an efficient DMU in \mathcal{R}^m with optimal multipliers vector π^* and suppose that $\pi_k^* = 0$. Then

$$\langle \pi^*, a^{\hat{j}} \rangle + \rho = 1; \langle \pi^*, a^j \rangle + \rho \leq 1, j = 1, \dots, n; \text{ and } \pi_i^* \geq 0, i = 1, \dots, m.$$

Thus

$$\begin{aligned} \pi_1^* a_1^{\hat{j}} + \dots + \pi_{k-1}^* a_{k-1}^{\hat{j}} + \pi_{k+1}^* a_{k+1}^{\hat{j}} + \dots + \pi_m^* a_m^{\hat{j}} + \rho &= 1; \\ \pi_1^* a_1^j + \dots + \pi_{k-1}^* a_{k-1}^j + \pi_{k+1}^* a_{k+1}^j + \dots + \pi_m^* a_m^j + \rho &\leq 1; j = 1, \dots, n; \quad \text{and} \\ \pi_1^* \geq 0, \quad \dots \quad \pi_{k-1}^* \geq 0, \quad \pi_{k+1}^* \geq 0, \quad \dots \quad \pi_m^* \geq 0. \end{aligned}$$

Hence $a^{\hat{j}}$ is efficient when removing the k th attribute from the model. \square

Based on this result, we propose the following simple and straightforward preprocessor for the case of removing the k th attribute from a DEA model. Identify all efficient DMUs in \mathcal{R}^m which optimal k th multiplier is zero. All of them are efficient in \mathcal{R}^{m-1} .

For example, suppose that Table 2 contains the list of efficient DMUs and their corresponding optimal multipliers from a DEA study with three inputs and two outputs ($\mathcal{R}^m = \mathcal{R}^5$).

Table 2. Efficient DMUs and their optimal multipliers.

DMU number	Input 1	Input 2	Input 3	Output 1	Output 2
⋮	⋮	⋮	⋮	⋮	⋮
17	0.0000	0.0182	0.4122	0.0379	0.0007
42	0.0661	0.0000	0.0000	0.0000	0.0258
93	0.0000	0.0539	0.0000	0.0417	0.0332
156	0.0248	0.0362	0.0000	0.0308	0.0000
⋮	⋮	⋮	⋮	⋮	⋮

Assume that we are interested in analyzing the impact of removing Input 1. RESULT 3 guarantees that DMUs 17 and 93 remain efficient.

To complete the list of efficient DMUs in \mathcal{R}^{m-1} , only yet unclassified DMUs (remember that inefficient DMUs in \mathcal{R}^m remain inefficient in \mathcal{R}^{m-1}) need to be tested. This should be done generating the LP systems with efficient DMUs in \mathcal{R}^m only (\mathcal{R}^5 in our example).

It is obvious that the same criterion can be applied to the cases corresponding to omitting another input or one of the outputs (i.e., when removing Output 1 from the example above, DMU 42 remains efficient).

3. Concluding remarks. The selection of attributes for a DEA analysis may not be an easy task. Sometimes it may be desirable either to add a new attribute from a long list of new variables or to eliminate one already in the model. The contributions of this paper include a preprocessing procedure called **HyperClimb** for the first case, and some useful observations that may save, potentially, a lot of computational time in the second case.

HyperClimb only requires a fraction of the time (around 2% and 0.1% for 500 and 10,000 DMUs respectively) needed to re-solve the DEA LPs. Because of this, it can be applied to many attributes, one at a time, that are candidates for inclusion without becoming a heavy burden.

The main reason why **HyperClimb** is faster than re-solving the LPs is because it is based on inner product and division calculations. In addition, or as a result, when the cardinality of the DEA data set (n) increases, the time that **HyperClimb** requires increases almost linearly whereas the time for re-solving the LPs is quadratic on n .

The outcome of **HyperClimb** is a list of efficient DMUs from which it is possible to identify DMUs that are not efficient before adding the new attribute but become efficient with the new inclusion. This provides an idea of the impact of each candidate on the list of efficient DMUs, and if some candidate is selected for definite inclusion, the new efficient DMUs found by **HyperClimb** do not need to be tested again. In our simulations this procedure finds around 34% of the new DMUs when applied to data sets with 500 DMUs and around 22% on data sets with 10,000 DMUs.

In this paper we also explain how to apply **HyperClimb** to any of the four returns to scale DEA models: variable, constant, increasing, or decreasing returns to scale.

References.

- Ali, I., [1993], "Streamlined computation for data envelopment analysis," *European Journal of Operational Research*, Vol. 64, pp. 61-67.
- Arnold, V., I. Bardham, W.W. Cooper, and A. Gallegos, [1998], "Primal and dual optimality in computer codes using two-stage solution procedures in DEA," in *Operations Research: Methods, Models, and Applications*, J.E. Aronson and S. Zionts, eds., Quorum Books, Westport, CT.
- Banker, R.D., A. Charnes, and W.W. Cooper, [1984], "Some models for estimating technological and scale inefficiencies in data envelopment analysis," *Management Science*, Vol. 30, No. 9, pp. 1078–1092.
- Banker, R.D., and R.M. Thrall, [1992], "Estimation of returns to scale using Data Envelopment Analysis," *European Journal of Operational Research*, Vol. 62, pp. 74–84.
- Barr, R.S. and M.L. Durchholz, [1997], "Parallel and hierarchical decomposition approaches for solving large-scale Data Envelopment Analysis models," *Annals of Operations Research*, 73, pp. 339–372.
- Charnes, A., W.W. Cooper, and E. Rhodes, [1978], "Measuring the efficiency of decision making units," *European Journal of Operational Research*, Vol. 2, No. 6, pp. 429-444.
- Charnes, A., W.W. Cooper, B. Golany, L. Seiford, and J. Stutz, [1985], "Foundations of data envelopment analysis for Pareto-Koopmans efficient empirical production functions," *Journal of Econometrics*, Vol. 30, pp. 91–107.
- Charnes, A., W.W. Cooper, and R.M. Thrall, [1986], "Identifying and classifying scale and technical efficiency in Data Envelopment Analysis," *Operations Research Letters*, Vol. 5, pp. 105–116.
- Charnes, A., W.W. Cooper, and R.M. Thrall, [1991], "A structure for classifying and characterizing efficiency and inefficiency in data envelopment analysis," *The Journal of Productivity Analysis*, Vol. 2, pp. 197–237.
- Cooper, W.W., K.S. Park, and J.T. Pastor, [1998], "RAM: A range adjusted measure of inefficiency for use with additive models and relations to other models and measures in DEA," Working Paper, IC² Institute, The University of Texas, Austin, TX 78712-1174. Accepted in *Journal of Productivity Analysis*.

Dulá, J.H., [1998], “An algorithm for Data Envelopment Analysis (DEA).” Submitted to *INFORMS Journal on Computing*.

Dulá, J.H. and N. Venugopal, [1995], “On characterizing the production possibility set for the CCR ratio model in DEA,” *International Journal of Systems Science* Vol. 26, No. 12, pp. 2319-2325.

Dulá, J.H. and R.V. Helgason, [1996], “A new procedure for identifying the frame of the convex hull of a finite collection of points in multidimensional space,” *European Journal of Operational Research*, Vol. 92, pp. 352–367.

Dulá, J.H., R.V. Helgason, and N. Venugopal [1998], “An algorithm for identifying the frame of a pointed finite conical hull,” *INFORMS Journal on Computing*, Vol. 10, No. 3.

Dulá, J.H. and B. L. Hickman, [1997], “Effects of excluding the column being scored from the DEA envelopment LP technology matrix,” *Journal of the Operational Research Society*, Vol. 48, pp. 1001–1012.

Färe, R., S. Grosskopf, C.A.K. Lovell, [1985], *The Measurement of Efficiency Production*. Kluwer Academic Publishers, Boston.

Oppa, G. and M. Yue [1997], “On measuring scale efficiency in DEA,” unpublished manuscript, London School of Economics, London, UK.

Pastor, J., [1996], “Translation invariance in Data Envelopment Analysis: A generalization,” *Annals of Operations Research*, Vol. 66, pp. 93–102.

Koopman, T., [1951], *Activity Analysis of Production and Allocation*, John Wiley & Sons, Inc., New York.

Rockafellar, R.T., [1970], *Convex Analysis*, Princeton University Press, Princeton, NJ.

Seiford, L.M., and R.M. Thrall, [1990], “Recent developments in DEA; the mathematical programming approach to frontier analysis,” *Journal of Econometrics*, Vol. 46, pp. 7–38.

Thrall, R.M., [1996a], “Duality, classification and slacks in DEA,” *Annals of Operations Research*, Vol. 66, pp. 109–138.

Thrall, R.M., [1996b], “Goal vectors for DEA efficiency and inefficiency,” Working Paper No. 128, J.H. Jones Graduate School of Administration, Rice University, Houston, TX 77005-1892.

Table 1. Data Sets and Results.

Name of file	# of extreme points in:		New extreme points found by HyperClimb	TIME (CPU Seconds)	
	dim. $m + 1 = 5$	dim. $m = 4$		Naive	HyperClimb
05by0500F1I1	14	10	1	1.4279	0.0313
05by0500F1I2	14	11	2	1.4234	0.0365
05by0500F1I3	14	13	0	1.4249	0.3620
05by0500F1O1	14	7	5	1.4247	0.0363
05by0500F1O2	14	9	3	1.4262	0.0377
05by0500F2I1	38	19	5	2.6151	0.0392
05by0500F2I2	38	15	6	2.6200	0.0378
05by0500F2I3	38	25	3	2.6176	0.0343
05by0500F2O1	38	19	7	2.6019	0.0362
05by0500F2O2	38	25	5	2.6129	0.0438
05by0500F3I1	58	22	7	2.2542	0.0451
05by0500F3I2	58	36	6	2.2626	0.0395
05by0500F3I3	58	33	7	2.2422	0.0387
05by0500F3O1	58	32	7	2.2470	0.0362
05by0500F3O2	58	27	10	2.2416	0.0394
05by1000F1I1	34	21	5	5.1204	0.0591
05by1000F1I2	34	27	2	5.1290	0.0570
05by1000F1I3	34	20	8	5.1376	0.0778
05by1000F1O1	34	14	11	5.1245	0.0926
05by1000F1O2	34	24	5	5.1612	0.0808
05by1000F2I1	52	20	5	8.1071	0.0595
05by1000F2I2	52	26	7	8.1076	0.0657
05by1000F2I3	52	36	2	8.0777	0.0602
05by1000F2O1	52	34	6	8.0803	0.0697
05by1000F2O2	52	25	6	8.1095	0.0822
05by1000F3I1	73	42	9	9.3439	0.0739
05by1000F3I2	73	44	5	9.3275	0.0658
05by1000F3I3	73	26	13	9.3603	0.0783
05by1000F3O1	73	52	7	9.3372	0.0765
05by1000F3O2	73	23	15	9.3895	0.0927

Table 1. Data Sets and Results (Cont'd).

Name of file	# of extreme points in:		New extreme points found by HyperClimb	TIME (CPU Seconds)	
	dim. $m + 1 = 5$	dim. $m = 4$		Naive	Hyper-Climb
05by2500F1I1	87	46	12	46.7472	0.2221
05by2500F1I2	87	52	10	46.8225	0.1776
05by2500F1I3	87	43	13	46.7906	0.2153
05by2500F1O1	87	50	13	46.8426	0.2055
05by2500F1O2	87	49	15	46.9319	0.2291
05by2500F2I1	81	42	5	42.3322	0.1650
05by2500F2I2	81	46	9	42.2534	0.1911
05by2500F2I3	81	40	9	42.0879	0.1786
05by2500F2O1	81	43	9	42.2463	0.1692
05by2500F2O2	81	42	9	42.1752	0.1953
05by2500F3I1	81	45	9	38.8630	0.1666
05by2500F3I2	81	41	12	39.0496	0.2031
05by2500F3I3	81	43	13	38.8450	0.2042
05by2500F3O1	81	54	9	38.8062	0.1977
05by2500F3O2	81	40	11	38.8257	0.2092
05by5000F1I1	90	42	9	166.3457	0.3180
05by5000F1I2	90	39	11	168.9527	0.3990
05by5000F1I3	90	46	7	166.3295	0.3269
05by5000F1O1	90	53	9	165.2882	0.3881
05by5000F1O2	90	46	10	165.2055	0.3572
05by5000F2I1	74	38	8	145.4833	0.3028
05by5000F2I2	74	38	7	145.5456	0.3091
05by5000F2I3	74	46	7	146.0254	0.3152
05by5000F2O1	74	39	9	146.1863	0.3208
05by5000F2O2	74	36	14	147.4299	0.3773
05by5000F3I1	90	39	12	190.7843	0.3698
05by5000F3I2	90	35	20	191.4873	0.4385
05by5000F3I3	90	53	7	190.6809	0.3809
05by5000F3O1	90	58	10	190.9637	0.4668
05by5000F3O2	90	54	14	192.6963	0.4457

Table 1. Data Sets and Results (Cont'd).

Name of file	# of extreme points in:		New extreme points found by HyperClimb	TIME (CPU Seconds)	
	dim. $m + 1 = 5$	dim. $m = 4$		Naive	Hyper-Climb
05by10000F1I1	132	48	20	656.4589	0.8617
05by10000F1I2	132	55	20	652.9320	0.8179
05by10000F1I3	132	54	17	651.1085	0.7358
05by10000F1O1	132	75	18	661.1423	0.9441
05by10000F1O2	132	68	16	657.0817	0.7977
05by10000F2I1	135	49	19	626.0400	0.8588
05by10000F2I2	135	56	20	619.0960	0.8100
05by10000F2I3	135	59	16	640.6752	0.6784
05by10000F2O1	135	74	15	626.1016	0.8225
05by10000F2O2	135	68	12	621.2892	0.7220
05by10000F3I1	143	56	19	607.9314	0.8547
05by10000F3I2	143	63	20	604.1733	0.8011
05by10000F3I3	143	64	15	611.8874	0.6964
05by10000F3O1	143	73	17	604.6089	0.8732
05by10000F3O2	143	65	12	608.0389	0.7684