GRIPS Research Report Series I-98-0008

# Toward Efficient and Stable Computation for Large-Scale Data Envelopment Analysis

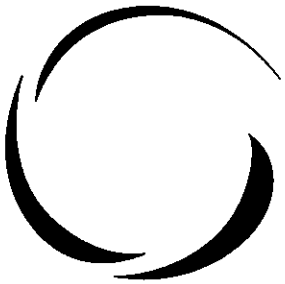
Kaoru Tone

National Graduate Institute for Policy Studies
Urawa, Saitama 338-8570, Japan

tel: 81-48-858-6096
fax: 81-48-852-0499
e-mail: tone@grips.ac.jp

January 15, 1999

Research supported by Grant-in-Aid for Scientific Research(C),
the Ministry of Education, Science, Sports and Culture,
Japan



GRIPS

NATIONAL GRADUATE INSTITUTE
FOR POLICY STUDIES

# Toward Efficient and Stable Computation

# for

# Large-scale Data Envelopment Analysis

**Kaoru Tone**

National Graduate Institute for Policy Studies
Urawa, Saitama 338-8570, Japan

tel: 81-48-858-6096
fax: 81-48-852-0499
e-mail: tone@grips.ac.jp

January 15, 1999

Research supported by Grant-in-Aid for Scientific Research (C), the Ministry of Education, Science, Sports and Culture, Japan.

# Toward Efficient and Stable Computation for Large-scale Data Envelopment Analysis

Kaoru TONE*

**Abstract**

In this paper we will propose several techniques for improving computations in Data Envelopment Analysis (DEA) and demonstrate their validity for large-scale problems using numerical experiments. Since DEA stems from mathematical programming approaches to efficiency analysis and specifically utilizes linear programming (LP) for evaluating the relative efficiency of decision making units (DMUs), the speeding up of the LP routine is an important factor for efficient DEA computation. However, there is another route of improvement, such as by incorporating the special characteristics of the relationship between efficient and inefficient DMUs in DEA. For example, in most DEA models, e.g. CCR, BCC and Additive, among others, it is demonstrated that no inefficient DMUs appear in the reference set of any DMUs. In the LP terminology, these inefficient DMUs are never included in the optimal basis for any DMUs. Therefore, knowledge on inefficient DMUs narrows the candidates of the incoming variables to the basis and hence subsequently reduces the load of computation. In this paper, these two directions, i.e. improvement of LP and utilization of the special characteristics of DEA efficiency structure, will be examined.

*National Graduate Institute for Policy Studies, Urawa Saitama 338-8570, Japan. e-mail:tone@grips.ac.jp. This research is supported by Grant-in-Aid for Scientific Research (C), the Ministry of Education, Science, Sports and Culture, Japan.

# 1 Introduction

Data Envelopment Analysis (Charnes, Cooper and Rhodes [4]) is an innovative method for evaluating the relative performance of enterprises by means of numerical data and many applications have been carried out in various fields of organizational activities, including governments, schools, hospitals, financial institutions, manufacturers and so forth. Some of recent studies include several thousands decision making units (DMUs) as object of comparisons. In dealing with such large-scale applications there arise computational problems with respect to the amount of time required for completion and the quality of results obtained.

In this paper, we will try to improve some of the computational aspects of DEA. Since DEA is a linear programming (LP) formulation for efficiency evaluation, several techniques developed for LP are also valuable for DEA. In Section 2, the role of multiple pricing and the product form of inverse will be discussed. Another route for improvement exists in the use of special characteristics of DEA formulation. For example, in representative DEA models, e.g. CCR, BCC and Additive, it is demonstrated that no inefficient DMUs appear in the reference set to any DMUs. In the LP terminology these inefficient DMUs are never included in the optimal basis for any DMU. Therefore, knowledge on inefficient DMUs narrows the candidates of the incoming variables to the basis and hence reduces subsequently the load of reduced cost computation. We will propose a two-stage strategy for this purpose in Section 3. Then numerical experiments will be introduced in Section 4. The experiments demonstrate the effectiveness of the proposed strategies, especially for large-scale problems. See also Ali [1, 2] and Ali and Seiford [3] for another aspects of DEA computation.

# 2 Multiple Pricing and Product Form of Inverse

In this section the effect of *multiple pricing* for choosing a nonbasic column and *product form of inverse* are discussed.

Before going into the details of computation, we will briefly describe the DEA models as formulated by linear programs. Let the number of DMUs, inputs and outputs be $n$, $m$ and $s$, respectively. The input (output) of DMU$_j$

$(j = 1, \ldots, n)$ is denoted by the vector $(\boldsymbol{x}_j)$ $((\boldsymbol{y}_j))$. Throughout this paper we assume that $\boldsymbol{x}_j$ and $\boldsymbol{y}_j$ $(j = 1, \ldots, n)$ are *semipositive*, i.e. nonnegative and nonzero. Input matrix $X \in R^{m \times n}$ and output matrix $Y \in R^{s \times n}$ are composed of the set of vectors $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ and $(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)$, respectively. We solve the following LP once for each $\mathrm{DMU}_o$ $(o = 1, 2, \ldots, n)$ and hence $n$ LPs in total.

$$\begin{aligned} \min \quad & \theta & (1)\\ \text{subject to} \quad & \theta \boldsymbol{x}_o - X\boldsymbol{\lambda} \geq \mathbf{0} & (2)\\ & Y\boldsymbol{\lambda} \geq \boldsymbol{y}_o & (3)\\ & L \leq \boldsymbol{e}\boldsymbol{\lambda} \leq U & (4)\\ & \boldsymbol{\lambda} \geq \mathbf{0}, & (5) \end{aligned}$$

where $\theta$ and $\boldsymbol{\lambda}$ are variables to be determined optimally, $L$ and $U$ are the lower and upper bounds to the sum of the elements of $\boldsymbol{\lambda}$ and $\boldsymbol{e}$ is the row vector with all elements equal to one. Usually, DEA computation consists of two phases. In Phase I, we solve the above LP and find the optimal objective value $\theta^*$ which is the score of the $\mathrm{DMU}_o$. Then we maximize the sum of *input excesses* and *output shortfalls* while keeping the objective value at $\theta^*$. This process, called Phase II, is formulated as follows:

$$\begin{aligned} \max \quad & \boldsymbol{e}\boldsymbol{s}^- + \boldsymbol{e}\boldsymbol{s}^+ & (6)\\ \text{subject to} \quad & \boldsymbol{s}^- = \theta^* \boldsymbol{x}_o - X\boldsymbol{\lambda} & (7)\\ & \boldsymbol{s}^+ = Y\boldsymbol{\lambda} - \boldsymbol{y}_o & (8)\\ & L \leq \boldsymbol{e}\boldsymbol{\lambda} \leq U & (9)\\ & \boldsymbol{\lambda} \geq \mathbf{0}, \boldsymbol{s}^- \geq \mathbf{0}, \boldsymbol{s}^+ \geq \mathbf{0} & (10) \end{aligned}$$

where $\boldsymbol{s}^-$ and $\boldsymbol{s}^+$ are vectors representing input excesses and output shortfalls, respectively. If we try to solve this two-phase process using commercial LP codes, it will be necessary to compromise the two objective functions above into one. This will be usually done by a function such as

$$\min \quad \theta - \varepsilon(\boldsymbol{e}\boldsymbol{s}^- + \boldsymbol{e}\boldsymbol{s}^+) \qquad (11)$$

where $\varepsilon$ is a small positive number. However, this causes another problem, i.e. selection of a reasonable $\varepsilon$. As Ali and Seiford [3] reported, this formula is unstable in the sense that the resulting solution loses numerical accuracy

and robustness. This is one of the reasons why we need specially designed software for DEA. It should contain functions such as computations of the efficiency score ($\theta^*$), the reference set ($\boldsymbol{\lambda}^*$), slacks ($s^{-*}, s^{+*}$), the optimal dual values (the weights), the projection onto efficient frontier and statistics on the problem and results. In what follows we will discuss subjects valuable in designing DEA software based on our experiences.

## 2.1 Multiple Pricing

In the LP formulation (1)-(5) of DEA the number of rows ($m + s + 2$) is considerably smaller than that of columns ($n$). Thus, the pricing out of all nonbasic variables at each iteration incurs high costs. The multiple pricing (see [7, 6] for example) works efficiently in our case. This strategy goes as follows.

1. Choose $p$ (say $p = 5$) candidate columns with $p$ most negative reduced cost.

2. Update the entering columns and find the leaving basic variable for each column.

3. Choose as the entering column the candidate with the largest negative value of improvement in the objective function.

The above process is repeated until the remaining columns do not have a sufficiently large negative reduced cost to enter the basis. If there is no candidate, we choose new $p$ candidate columns with the negative reduced cost.

The effect of multiple pricing can be observed in the numerical experiments in Section 4.

## 2.2 Product Form of Inverse

One of the characteristics of LP formulation (1)-(5) is that its coefficient matrices $X$ and $Y$ are dense. So, we cannot make use of the sparse matrix techniques developed for usual LP problems. However, even in the dense matrix case, the product form of inverse technique is valuable in keeping the numerical stability of the simplex iterations. We will reinvert the basis after the number of simplex iterations exceeds a certain limit.

# 3 A Two-Stage Strategy for DEA Computation

The "efficiency" and "inefficiency" of the $DMU_o$ is dfined as follows:

**Definition 1 (Efficiency and Inefficiency)** *The $DMU_o$ is efficient if, for every optimal solution $(\theta^*, \boldsymbol{\lambda}^*, \boldsymbol{s}^{-*}, \boldsymbol{s}^{+*})$ to (1)-(10), it holds*

$$\theta^* = 1, \ \boldsymbol{s}^{-*} = \mathbf{0}, \ \boldsymbol{s}^{+*} = \mathbf{0}.$$

*Otherwise, the $DMU_o$ is inefficient.*

Thus, "inefficiency" means either $\theta^* < 1$ or $\theta^* = 1$ and $\boldsymbol{s}^{-*} \neq \mathbf{0}$ and/or $\boldsymbol{s}^{+*} \neq \mathbf{0}$ for some $\boldsymbol{s}^{-*}$ and/or $\boldsymbol{s}^{+*}$.

**Definition 2 (Reference set)** *A reference set to $DMU_o$ is defined by*

$$E_o = \{j | \lambda_j^* > 0\} \ (j = 1, 2, \ldots, n). \tag{12}$$

Using $E_o$ we can express $\theta^* \boldsymbol{x}_o$ and $\boldsymbol{y}_o$ by

$$\theta^* \boldsymbol{x}_o = \sum_{j \in E_o} \lambda_j^* \boldsymbol{x}_j + \boldsymbol{s}^{-*} \tag{13}$$

$$\boldsymbol{y}_o = \sum_{j \in E_o} \lambda_j^* \boldsymbol{y}_j - \boldsymbol{s}^{+*}. \tag{14}$$

In the DEA models, the cases $(L = 0, U = \infty)$ and $(L = 1, U = 1)$ are called CCR (Charnes, Cooper and Rhodes [4]) and BCC (Banker, Charnes and Cooper [5]) models, respectively. These two are representative DEA models, in that the former corresponds to the *constant* returns-to-scale situation and the latter to the *variable* returns-to-scale one. With respect to them, we have the following propositions.

**Proposition 1** *For the CCR and BCC models, the DMUs in the reference set to $DMU_o$ are efficient.* (See Appendix for a proof.)

This proposition can be rewritten in LP terminology as follows.

**Proposition 2** *In the CCR and BCC models, we can choose an optimal basis for any DMU which consists of columns corresponding to efficient DMUs and slack variables ($\boldsymbol{s}^-$ and/or $\boldsymbol{s}^+$).* (See Appendix for a proof.)

Proposition 2 has a significant role in DEA computation. If we can find, by some means, that a certain DMU is inefficient, then the corresponding column will not be included in an optimal basis for any DMU and so we need not calculate its reduced cost, i.e., we can neglect this DMU in the simplex method. By experience, we know that in large-scale problems the majority of DMUs are inefficient, and in fact account for more than 90% of all DMUs. Hence, knowledge on inefficiency reduces the amount of DEA computation significantly. The following two-stage strategy aims at realizing this idea.

## 3.1 The First Stage

In the first stage, we will solve $n$ LPs corresponding to $n$ DMUs within a certain limit of simplex iterations. This number may be set, for example, at $2\times$(the number of constraints of the LP). So we do not pursue an optimal solution at this stage. Eventually we will finish this stage in one of the following three cases.

**Case 1** As soon as the objective value $\theta$ of $\mathrm{DMU}_o$ is found to be less than 1, we stop the simplex iteration even before the prefixed number of iterations and record the set of indices of the basis. Furthermore, if the basis is optimal then we record the fact using a one bit logical variable.

**Case 2** If $\mathrm{DMU}_o$ is found to be efficient within the iteration, we record the set of indices of the basis and one bit of logical variable indicating optimality.

**Case 3** If we cannot decide whether $\mathrm{DMU}_o$ is efficient or not within the given number of iterations, we record the set of indices of the last basis and stop the procedure dealing with $\mathrm{DMU}_o$.

The first stage will end up with the information on whether $\mathrm{DMU}_o$ is inefficient ([Case 1]), efficient ([Case 2]) or unknown ([Case 3]). However, since the number of simplex iterations is limited, the first stage is free from the occurrence of a large number of degenerate pivots, which is often observed when the numbers of inputs and outputs are large.

### 3.2 The Second Stage

Following the first stage we solve the $n$ LPs again using the results of the first stage. The second stage strategy is summarized as follows.

1. We start the iteration for each DMU from the last basis of the DMU obtained in the first stage.

2. The DMUs which are found to be inefficient in the first stage should not be candidates for inclusion in the optimal basis and hence we do not calculate their reduced cost.

3. For DMUs whose basis are found to be optimal in the first stage we calculate the optimal solution and the simplex multipliers directly from the set of basis.

For large-scale problems, the effect of Strategy 2 is remarkable, as will be demonstrated by the numerical experiments in the next section. If we solve LP (1)-(5) directly for each DMU, the number of candidates for incoming basis variables is large and therefore we need to calculate reduced costs of many nonbasic variables. This may also give rise to occurrence of a large number of degenerate pivot sequences. Our two-stage strategy will be useful to prevent such undesirable iterations.

## 4 Numerical Experiments

In this section numerical experiments on the proposed methods will be demonstrated. In all cases, we apply the methods to the CCR model, i.e., the problem (1)-(10) with $L = 0$ and $U = \infty$. The codes are written using Visual Basic for Applications (VBA) in Excel 97 for Windows 98 (a trademark of Microsoft Inc.).

### 4.1 The Methods Compared

We compared the following three methods using numerical experiments.

**Standard** Simplex method without multiple pricing, i.e. computing the reduced costs of all nonbasic variables and choosing the most negative one.

**Multiple** Simplex method with multiple pricing.

**Two-stage** Simplex method with two-stage strategy and multiple pricing.

In all cases we use the product form of inverse formula for basis change and conduct reinversion at every 2×(the number of constraints). Proposition 2 and the discussions following it enable us to omit the calculation of reduced cost for the column after the corresponding DMU is found to be inefficient. We apply this policy throughout experiments. In the two-stage method the upper bound of iterations in the first stage is set to 2×(the number of constraints).

## 4.2 Test Problems

Two kinds of test problems are chosen. One is "real world" and the other a "randomly generated" data set. We apply the notation W(R)$n$-$m$-$s$-($p$%) to denote a real World (Random) problem with $n$ DMUs, $m$ inputs and $s$ outputs, including about $p$% efficient DMUs. As an example, Table 1 shows statistics of the problem W8000-6-2.

Table 1: Statistics of Problem W8000-6-2

| | INPUT | | | | | | OUTPUT | |
|---|---|---|---|---|---|---|---|---|
| | Input1 | Input2 | Input3 | Input4 | Input5 | Input6 | Output1 | Output2 |
| Max | 9501 | 915 | 1993 | 99 | 682 | 551 | 27028 | 1815 |
| Min | 0 | 0 | 20 | 0 | 0 | 0 | 0 | 0 |
| Average | 16.7 | 41.1 | 174.8 | 4.1 | 38.5 | 29.5 | 1509 | 142.3 |
| Std. D. | 44 | 78 | 173 | 5.4 | 34.9 | 31.9 | 2050 | 149 |

## 4.3 Results of the Experiments

### (1) For real world problems

First we exhibit CPU time for W$n$-6-2 with $n$=500, 1000, 2000, 4000 and 8000 in Table 2. In this case the numbers of inputs ($m = 6$) and outputs ($s = 2$) are fixed. However, the density ($p$) of efficient DMUs are decreasing with respect to $n$ (the number of DMUs) as listed in the table. Percentages in the parenthesis designate the relative CPU times of 'Multiple' and 'Two-stage' to 'Standard'. The results show that the effect of the multiple pricing

Table 2: Comparisons of CPU Seconds for Real World Problems*

| W$n$-$m$-$s$-($p$%)** | Standard | Multiple | Two-stage |
|---|---|---|---|
| W500-6-2-(7.8%) | 29(100%) | 24(83%) | 25(86%) |
| W1000-6-2-(5.8%) | 104(100%) | 83(80%) | 76(73%) |
| W2000-6-2-(2.6%) | 337(100%) | 285(80%) | 245(73%) |
| W4000-6-2-(1.6%) | 1666(100%) | 1133(68%) | 935(56%) |
| W8000-6-2-(1.0%) | 6897(100%) | 4701(68%) | 3618(52%) |

* A PC with 400MHz CPU and 128MB main memory was used for this experiment.
** $n, m, s$= Numbers of DMUs, inputs and outputs, and $p$ = percentage of efficient DMUs.

becomes evident as the number of DMUs grows beyond 1000 and that the two-stage strategy works well for large-scale problems.

**(2) For randomly generated problems**

Table 3 shows the results of experiments with random problems R$n$-10-10-($p$%) where $p$ ranges about 5%, 10%, 15% and 20%.

We can see that the superiority of the two-stage strategy becomes apparent as the number of DMUs grows and the density of efficient DMUs decreases.

## 4.4 Some Observations

**(1) Sensitivity of the first-stage iteration**

In the experiments in Tables 2 and 3 we set the upper bound of the simplex iterations in the first stage to 2×(the number of constraints). In order to check the sensitivity of this bound, we set the bound to 0.5, 1, 1.5, 2 and 2.5×(the number of constraints) and applied them to problems R2000-10-10-(5%) and R2000-10-10-(9%). The results are exhibited in Table 4, where "First-stage rate" means the percentage of the number of inefficient DMUs found at the first-stage. As a matter of course, this number grows as the upper bound of iterations goes up. As far as this sample concerned, CPU times are not so sensitive to the upper bound between $M$ and $2M$ ($M$ =

Table 3: Comparisons of CPU Seconds for Random Problems*

| R$n$-$m$-$s$-($p$%)** | Standard | Multiple | Two-stage |
|---|---|---|---|
| R500-10-10-(6%) | 85(100%) | 85(100%) | 89(105%) |
| -(12%) | 110(100%) | 114(104%) | 107(97%) |
| -(16%) | 122(100%) | 146(120%) | 138(113%) |
| -(20%) | 132(100%) | 170(129%) | 148(112%) |
| R1000-10-10-(5%) | 336(100%) | 318(95%) | 276(82%) |
| -(10%) | 434(100%) | 419(97%) | 343(79%) |
| -(14%) | 491(100%) | 503(102%) | 423(86%) |
| -(19%) | 555(100%) | 590(106%) | 482(87%) |
| R2000-10-10-(5%) | 1506(100%) | 1187(79%) | 932(62%) |
| -(9%) | 1865(100%) | 1625(87%) | 1220(65%) |
| -(12%) | 2251(100%) | 1943(86%) | 1515(67%) |
| -(16%) | 2396(100%) | 2238(93%) | 1694(71%) |

* A PC with 300MHz CPU and 32MB main memory was used for this experiment.
** $n, m, s$= Numbers of DMUs, inputs and outputs, and $p$ = percentage of efficient DMUs.

the number of constraints). This may correspond to the empirical results that most LP problems are solved in $M$ to $3M$ iterations (Gass[6], p. 303). It is also remarkable that 96 to 100 % of inefficient DMUs were identified at the first-stage for the upper bound beyond $M$. However, notice that this identification does not always mean the discovery of the optimal efficiency score of the DMU.

### (2) Influence of the rate of efficient DMUs

The results of Table 3 exhibit the fact that the merit of 'Two-stage' strategy decreases as the rate of efficient DMUs grows up. In the most extreme case, if most DMUs are efficient, this strategy will lose all its merits. However, such cases are likely to be exceptional.

Table 4: Sensitivity of First-Stage Iterations

| Upper bound | $0.5M^*$ | $M$ | $1.5M$ | $2M$ | $2.5M$ |
|---|---|---|---|---|---|
| R2000-10-10-(5%) | 1121** | 949 | 926 | 932 | 942 |
| First-stage rate | 72% | 97% | 100% | 100% | 100% |
| R2000-10-10-(9%) | 1445 | 1229 | 1235 | 1220 | 1306 |
| First-stage rate | 71% | 96% | 99% | 99% | 99% |

* $M$ = the number of constraints. ** CPU seconds.

### (3) Number of inputs and outputs

We experimented to investigate how increases in numbers of inputs and outputs affect the behavior of codes on random problems. Table 5 exhibits the results. Percentages in the parenthesis designate the relative CPU time of 'Multiple' and 'Two-stage' to 'Standard'. From the table it is observed that as the numbers of inputs ($m$) and outputs ($s$) increase the superiority of 'Multiple' and 'Two-stage' over 'Standard' decreases. This may reflect the fact that the workload of basis maintenance becomes heavy as the numbers of inputs and outputs increase and, in contrast, the cost of reduced cost computations loses its weight comparatively. However, the ratios of 'Two-stage' to 'Multiple' stay almost constant at 80%.

Table 5: Effect of Inputs and Outputs Numbers*

| R2000-$m$-$s$-($p$%) | Standard | Multiple | Two-stage |
|---|---|---|---|
| R2000-5-5-(4%) | 807**(100%) | 555(69%) | 444(55%) |
| R2000-10-10-(5%) | 1506(100%) | 1187(79%) | 932(62%) |
| R2000-15-15-(5%) | 2367(100%) | 2072(88%) | 1679(71%) |

* A PC with 300MHz CPU and 32MB main memory was used for this experiment.
** CPU seconds.

# 5 Conclusions

In this paper we discussed several issues aiming at accelerating DEA computation for large-scale problems. Numerical experiments demonstrated their usefulness, and especially the two-stage strategy utilizes efficiently the special structure of DEA problems. This process can be applied not only for CCR, BCC and Additive models but also for the cases with $(L = 0, U = 1)$ (not increasing returns-to-scale model), and $(L = 1, U = \infty)$ (not decreasing returns-to-scale model) as well. Also, we can apply this strategy to the allocative models.

Future research includes finding faster methods for identifying inefficient DMUs in the first stage. With respect to this point, interior point methods and greedy algorithms may deserve consideration.

As has been widely recognized, DEA needs specially designed codes in order to take full advantage of this method. We hope that our proposed methods will contribute to this purpose.

# References


[1] Ali, A.I., "Data Envelopment Analysis: Computational Issues," *Computers, Environment, and Urban Systems,* 14, 157-165 (1990).

[2] Ali, A.I., "Computational Aspects of DEA," in *Data Envelopemnt Analysis*, eds Charnes, Cooper, Lewin and Seiford, Kluwer Academic Publishers, 1994.

[3] Ali, A.I. and Seiford L.M., "Computational Accuracy and Infinitesimals in Data Envelopment Analysis," *INFOR*, 31(4), 290-297 (1993).

[4] Charnes, A., W.W. Cooper and E. Rhodes, "Measuring the Efficiency of Decision Making Units," *European Journal of Operational Research,* 2, 429-444 (1978).

[5] Banker, R.D., A. Charnes and W.W. Cooper, "Some Models for Estimating Technical and Scale Inefficiency in Data Envelopment Analysis," *Management Science,* 30, 1078-1092 (1984).

[6] Gass, S.I., *Linear Programming,* 5th ed., McGraw-Hill, 1985.

[7] Murtagh, B.A., *Advanced Linear Programming: Computation and Practice*, McGraw-Hill, New York, 1981.

# Appendix

## Proof of Proposition 1

We will prove the proposition in the case of the CCR model. The BCC case can be proved in a similar way.

Assume that the DMU $(\boldsymbol{x}_o, \boldsymbol{y}_o)$ has an optimal solution $(\theta^*, \boldsymbol{\lambda}^*, \boldsymbol{s}^{-*}, \boldsymbol{s}^{+*})$ to (1)-(10) and an *inefficient* DMU $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ is a member of the reference set to $(\boldsymbol{x}_o, \boldsymbol{y}_o)$ with $\lambda_1^* > 0$. Further assume that the inefficient $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ has an optimal solution $(\theta_1^*, \boldsymbol{\mu}^*, \boldsymbol{t}^{-*}, \boldsymbol{t}^{+*})$ and can be expressed as

$$\theta_1^* \boldsymbol{x}_1 = \sum_{k \in E_1} \mu_k^* \boldsymbol{x}_k + \boldsymbol{t}^{-*} \tag{15}$$

$$\boldsymbol{y}_1 = \sum_{k \in E_1} \mu_k^* \boldsymbol{y}_k - \boldsymbol{t}^{+*}, \tag{16}$$

where $E_1$ is a reference set to $(\boldsymbol{x}_1, \boldsymbol{y}_1)$.

From (15), we have

$$\boldsymbol{x}_1 = \sum_{k \in E_1} \mu_k^* \boldsymbol{x}_k + \boldsymbol{t}^{-*} + (1 - \theta_1^*)\boldsymbol{x}_1. \tag{17}$$

By substituting right sides of (17) and (16) into (13) and (14), we obtain

$$\theta^* \boldsymbol{x}_o = \lambda_1^* \sum_{k \in E_1} \mu_k^* \boldsymbol{x}_k + \sum_{j \in E_o - \{1\}} \lambda_j^* \boldsymbol{x}_j + \boldsymbol{s}^{-*} + \lambda_1^* (\boldsymbol{t}^{-*} + (1 - \theta_1^*)\boldsymbol{x}_1) \tag{18}$$

$$\boldsymbol{y}_0 = \lambda_1^* \sum_{k \in E_1} \mu_k^* \boldsymbol{y}_k + \sum_{j \in E_o - \{1\}} \lambda_j^* \boldsymbol{y}_j - \boldsymbol{s}^{+*} - \lambda_1^* \boldsymbol{t}^{+*}. \tag{19}$$

By arranging the last two expressions and using the notation $\pi_j^*$ for coefficient of $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, we have

$$\theta^* \boldsymbol{x}_o = \sum_j \pi_j^* \boldsymbol{x}_j + \boldsymbol{s}^{-*} + \lambda_1^* (\boldsymbol{t}^{-*} + (1 - \theta_1^*)\boldsymbol{x}_1) \tag{20}$$

$$\boldsymbol{y}_0 = \sum_j \pi_j^* \boldsymbol{y}_j - \boldsymbol{s}^{+*} - \lambda_1^* \boldsymbol{t}^{+*}. \tag{21}$$

Since by assumption $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ is inefficient, we have either (i) $\theta_1^* < 1$ or (ii) $\theta_1^* = 1$ and $(\boldsymbol{t}^{-*}, \boldsymbol{t}^{+*})$ is semipositive.

In case (i), since $1 - \theta_1^* > 0$, $\boldsymbol{e}\boldsymbol{x}_1 > 0$ and $\lambda_1^* > 0$, it holds that

$$\begin{aligned} \boldsymbol{e}\boldsymbol{s}^{-*} &< \boldsymbol{e}\boldsymbol{s}^{-*} + \lambda_1^*(\boldsymbol{e}\boldsymbol{t}^{-*} + (1-\theta_1^*)\boldsymbol{e}\boldsymbol{x}_1) \quad (22) \\ \boldsymbol{e}\boldsymbol{s}^{+*} &\leq \boldsymbol{e}\boldsymbol{s}^{+*} + \lambda_1^*\boldsymbol{e}\boldsymbol{t}^{+*}. \quad (23) \end{aligned}$$

Thus, we have a larger sum of slacks for Phase II. This contradicts the maximality of $(\boldsymbol{s}^{-*}, \boldsymbol{s}^{+*})$.

In case (ii), similar inequalities hold as below.

$$\begin{aligned} \boldsymbol{e}\boldsymbol{s}^{-*} &\leq \boldsymbol{e}\boldsymbol{s}^{-*} + \lambda_1^*\boldsymbol{e}\boldsymbol{t}^{-*} \quad (24) \\ \boldsymbol{e}\boldsymbol{s}^{+*} &\leq \boldsymbol{e}\boldsymbol{s}^{+*} + \lambda_1^*\boldsymbol{e}\boldsymbol{t}^{+*}. \quad (25) \end{aligned}$$

Since in this case $(\boldsymbol{t}^{-*}, \boldsymbol{t}^{+*})$ is semipositive (nonnegative and nonzero), we have a larger sum of slacks for Phase II. This contradicts the maximality of $(\boldsymbol{s}^{-*}, \boldsymbol{s}^{+*})$.

By the above reasoning, every DMU in the reference set must be efficient.

## Proof of Proposition 2

By Proposition 1, an optimal basis to (1)-(10) consists of columns corresponding to (i) efficient DMUs with nonnegative basic solution, (ii) inefficient DMUs with the value zero in the basic solution and/or (iii) slack variables. We can remove the case (ii) from the basis by imposing a large penalty to the columns without changing the objective function value. Notice that we can start the simplex iteration from an initial feasible basis to (1)-(5) which consists of the columns corresponding to $\mathrm{DMU}_o$, $\theta$ and slack variables. Thus it is sufficient to deal with inefficient columns only for removing them from the basis.