Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 18** : 1–9.
ISSN-0037-8712
doi:10.5269/bspm.82737

Preventing Degeneracy: The Complexity and Limitations of Lexicographic Rule and Bland's Rule *

Asmaa Ait Brik and Seddik Abdelalim

ABSTRACT: Degeneracy and cycling present two of the most significant theoretical and practical hurdles in Linear Programming (LP). They happen when the simplex algorithm gets stuck, wasting time without getting close to an answer. As artificial intelligence (IA) grow more complex, especially in areas like neural network verification and safety certification, these mathematical roadblocks become more then just academic curiosities. The represent real challenges that can prevent AI from finding optimal solution or guaranteeing safety. This paper provides a comprehensive comparative analysis of two foundational classical approaches: Lexicographic and Bland's smallest index rules. Think of these as different traffic control systems designed to prevent infinite loops in the mathematical journey toward the best solution while the methods provides the rigorous mathematical guaranties needed for high stacks AI applications, our analysis reveals a trade-off: they are like having a perfect but incredibly slow GPS it will get you there, but the journey might take forever. The computational overhead makes them impractical for today's massive high-dimensional AI problems. This work lays the ground work for exploring more modern, efficient methods that can handle these mathematical challenges without the heavy computational cost-essentially paving the way for traffic control systems that work well even in today's AI superhighways.

Keywords: Linear Programming, simplex algorithm, degeneracy, cycling and stalling, Bland's smallest index rule, lexicographic pivot rule.

Contents

1. Introduction

Imagine you are navigating through a complex maze of possibilities, trying to find the best path forward. Now imagine getting stuck in a loop, going in circles without making progress. That is essentially without happens with degeneracy in the mathematical world of Linear Programming (LP), a critical tool that is becoming increasingly important for Artificial Intelligence.

Linear programming remains the bedrock of mathematical optimization, powering complex decision making in sectors ranging from industrial logistics to the cutting edge of artificial intelligence. In modern AI, LP solvers are indispensable for tasks like Neural Network verification and certifying adversarial robustness, where they ensure that autonomous systems remain within safe operational bounds.

The efficiency of the simplex method, originally developed by Dantzig and Thapa [3], depends heavily on the geometric structure of the constraint polytope, a field deeply explored in the foundations of convex analysis by Rockefellar [8].

However, a significant computational challenge arises when multiple bases correspond to a single vertex, a state known as degeneracy [1,9].

Mathematically, a basic feasible solution is degenerate if a least one basis variable equals zero. This structural redundancy leads to stalling, where the algorithm performs basis changes without improving

* The project is partially supported by LAM2A.
2020 *Mathematics Subject Classification:* 90C05, 90C31.
Submitted March 16, 2026. Published June 22, 2026.

the objective value, and in the worst-case scenario, it results in cycling, an infinite loop of degenerate pivots that prevents the algorithm from ever reaching an optimal solution.

This phenomenon is a common occurrence in the highly structured datasets used in AI and large-scale networks.

This paper provides a foundational study of the two most famous classical methods designed to prevent cycling:

- The lexicographic rule utilizes symbolic perturbation to guarantee a strictly decreasing sequence in ordered spaces, a technique frequently applied in function analysis and optimization theory.
- On the other hand, Bland's rule provides a combinatorial approach based on the smallest index of variables to break ties. This is like always choosing the next step in alphabetical order to ensure you never repeat the same sequence.

While these methods are theoretically robust, they impose a heavy computational cost of certainty. As highlighted in the works of Luenberger and Ye And Minoux [6,7], the lexicographic approach increases the cost per iteration, while Bland's rule often leads to a prohibitive number of pivots in real-world applications.

The Lexicographic rule increases the cost per iteration to $O(m^2)$, while Bland's rule, despite its simple $O(n)$ selection logic, frequently triggers a massive increase in total number of iterations. It is like having a perfect but incredibly slow GPS, it will get you there, but the journey might take forever.

Crucially, the reliability of these AI system is fundamentally tied to the solver's ability to handle the geometric intricacies of the constraints polytope, particularly the phenomenon of degeneracy. While these methods are historically significant and provide the theoretical guarantee of finiteness required for formel AI verification, they carry a substantial computational burden.

The primary objective of this work is to establish a rigorous understanding of these classical pillars as a necessary precursor to more advance optimization studies. By analyzing the limitations and the cost of certainty associated with Lexicographic and Bland's rules, drawing on the seminal works Bazaraa and al. [1,2] and Dantzig and Thapa [3], this paper sets the stage for a subsequent investigation into modern, more effective methods designed to prevent stalling and cycling with significantly lower computational overhead.

As LP becomes increasingly integrated into AI for tasks such as neural network verification and safety certification, the need for robust anti-cycling mechanisms has never been more critical. This work serves as a necessary theoretical precursor to the study of modern, low-overhead methods designed to mitigate stalling and cycling in massive datasets.

Notation

All mathematical notations used in this paper, are presented in the following list with their description

$\mathbf{z}$: The objective function of the linear program.
$\mathbf{c}$: An n-vector formed by the coefficients of the objective function.
$\mathbf{x}$: An n-vector formed by the variables of the linear program.
$\mathbf{A}$: An $m \times n$ matrix formed by the constants of the linear program.
$\bar{\mathbf{a}}_\mathbf{j}$: The j-th column of A. $\mathbf{B}$: The basis of a feasible basic solution, which is an invertible matrix.
$\mathbf{N}$: The non basic matrix, which is a submatrix of A.
$\mathbf{b}$: An m-vector representing the right-hand side of the linear program.
$\mathbf{m}$: The number of constraints in the linear program.
$\mathbf{n}$: The number of variables in the linear programming.
$\mathbf{x_B}$: The components of x that represent a feasible basic solution.
$\mathbf{x_N}$: The components of x that represent a non basic solution.
$\mathbf{c_B}$: The vector of the coefficients in the objective function corresponding to the basic feasible solution variables.
$\mathbf{c_N}$: The vector of the coefficients in the objective function corresponding to the non basic variables.
$\mathbf{I}$: The set of indices corresponding to the basic feasible solution.
$\mathbf{J}$: The set of indices corresponding to the non basic variables.

k : The index of the entering variable in the simplex algorithm.
r : The index of the leaving variable in the simplex algorithm.

2. Lexicographic Rule (Selecting an Exiting Variable)

The purpose of studying this method is to understand how imposing a total ordering on solutions guarantees convergence by eliminating geometric ambiguity.
While this approach establishes the theoretical foundations for managing stagnation, it presents significant computability challenges due to computational burden proportional to the problem's dimension, paving the way for modern optimizations based on the structure of redundant bases.

Consider the following linear programming problem

$$(P)\begin{cases} \text{Minimize } z = c^\top.x, \\ A.x = b \\ x \geq 0. \end{cases}$$

where $A = (a_1, a_2, \ldots, a_n)$ is an $m \times n$ matrix of rank m, $c^\top = (c_1, \ldots, c_n)$, $x = (x_1, \ldots, x_n)^\top$ are vectors of dimension n and $b = (b_1, \ldots, b_m)^\top$ is a vector of dimension m.

Note that n represent the number of variables and m represents the number of constraints.

Definition 1 *Any regular square submatrix of order m extracted from A is called a* ***basis*** *denoted by B. We can always write A in the form $A = [B, N]$, where N is the submatrix formed by the columns of A that are not in the basis.Similarly, we can partition x into $[x_B^\top, x_N^\top]$ and $c^\top$ into $[c_B, c_N]$.*

Since the simplex method is usually started with the initial basis as the identity matrix (corresponding to slack and/or artificial variables), we shall assume that the first m columns of A form the identity.

Suppose that we have a basic feasible solution $\hat{x}_B = B^{-1}b = \hat{b}$ whose objective value $\hat{z}_0$ is given by $\hat{z}_0 = c_B^T \hat{x}_B$.

Now, we have

$$Ax = b = Bx_B + Nx_N.$$

Multiplying this equation by B^{-1} , we get

$$B^{-1}b = x_B + B^{-1}Nx_N$$

This implies,

$$\begin{aligned} x_B &= B^{-1}b - B^{-1}Nx_N \\ &= \hat{b} - \sum_{j \in J} B^{-1}a_j x_j \end{aligned}$$

Therefore, we have

$$x_B = \hat{b} - \sum_{j \in J} \overline{a}_j x_j$$

where $\overline{a}_j = (\overline{a}_{j1}, \overline{a}_{j2}, \ldots, \overline{a}_{jm}) = B^{-1}a_j$ is an m-dimensional column vector and J is the current set of indices corresponding to the nonbasic variables.

Now given a basic feasible solution with basis B, suppose that the nonbasic variable x_k is chosen to enter the basis.
The index r of the variable x_{B_r} leaving the basis. let the tableau of simplex represented by the augmented matrix $(B^{-1}, \hat{b}_r)$ before pivoting. Then, using Gaussian elimination the i-th row of augmented matrix $(B^{-1}, \hat{b}_r)$ is given by

$$\left(\overline{a}_{i1} - \frac{\overline{a}_{r1}}{\overline{a}_{rk}}\overline{a}_{ik}, \overline{a}_{i2} - \frac{\overline{a}_{r2}}{\overline{a}_{rk}}\overline{a}_{ik}, \ldots, \overline{a}_{im} - \frac{\overline{a}_{rm}}{\overline{a}_{rk}}\overline{a}_{ik}, \hat{b}_i - \frac{\hat{b}_r}{\overline{a}_{rk}}\overline{a}_{ik}\right) \qquad \text{for } i \neq r$$

and for $i = r$, we have

$$\left(\frac{\hat{b}_r}{\overline{a}_{rk}}, \frac{\overline{a}_{r1}}{\overline{a}_{rk}}, \ldots, \frac{\overline{a}_{rm}}{\overline{a}_{rk}}\right)$$

The lexicographic rule begins with the standard ration test used for selecting the leaving variable in the traditional simplex method. However, in this approach, we do not merely consider the scalar $\hat{b}_r$; instead, we evaluate the entire augmented row vector.

Let

$$I_0 = \left\{ r \Big/ \frac{\hat{b}_r}{\overline{a}_{rk}} = \min_{1 \le i \le m} \left\{ \frac{\hat{b}_i}{\overline{a}_{ik}}, \overline{a}_{ik} > 0 \right\} \right\}$$

If I_0 is a singleton, then x_{B_r} leaves the basis.

Otherwise, if $|I_0| > 1$, we move to the next index (1). refers to the first column of the initial identity matrix, ensuring that the selection of the leaving variable is deterministic and prevent cycling by strictly increasing the lexicographic value of the basis). We choose the row $r \in I_0$ such that

$$I_1 = \left\{ r \Big/ \frac{\overline{a}_{r1}}{\overline{a}_{rk}} = \min_{i \in I_0} \left\{ \frac{\overline{a}_{i1}}{\overline{a}_{ik}}, \overline{a}_{ik} > 0 \right\} \right\}$$

If I_1 is a singleton, then x_{B_r} leaves the basis, and if a tie still exists at the index 1, you move to index 2 , index 2, ... , until the tie is broken

$$I_j = \left\{ r \Big/ \frac{\overline{a}_{rj}}{\overline{a}_{rk}} = \min_{i \in I_{j-1}} \left\{ \frac{\overline{a}_{i1}}{\overline{a}_{ik}}, \overline{a}_{ik} > 0 \right\} \right\}$$

Eventually, for some $j \le m$, I_j will be a singleton. If $I_j = \{r\}$, then x_{B_r} leaves the basis.

If this test gives a unique index, then the corresponding variable leaves the basis. In case of a tie, we try to break it by replacing the right-hand-side in the minimum ratio calculation by the first column y_1 and by only using the rows corresponding to the tie.

If the tie is still not broken, the second column is used in the minimum ratio test, and so forth. When or before column m is reached, the tie must be broken, for if this were not the case, we would have two rows of the matrix B^{-1} that are proportional. This is impossible, however, in view of linear independence of the rows of B^{-1}.

Now, lets showing that none of the previous bases visited by the simplex method are repeated. In view of the finite number of bases, this automatically guarantees termination in a finite number of iterations.

For that, consider the following notion of a lexicographically positive vector.

Definition 2 *[1] A vector $u \in \mathbb{R}^n$ is said to be **Lexicographically larger** (or **smaller**) than another vector $v \in \mathbb{R}^n$ if $u \neq v$ and the first nonzero component of $u - v$ is positive (or negative, respectively).*

Symbolically, we write

$$u \succ v \qquad or \qquad u \prec v.$$

Example 1 *For example, let*

$$u = (0, 2, -3, 1),\ v = (1, 4, -1, 10), \qquad and \qquad w = (0, 0, 5, -6)$$

the vectors

$$v - u = (1, 2, 2, -1) \qquad and \qquad u - w = (0, 2, -8, 7)$$

are lexicographically positive vectors, whereas

$$u - v = (-1, -2, -2, 1) \qquad and \qquad w - v = (0, -2, 8, -7)$$

are not.

1. **Lexicographical Positivity of Tableau Rows**

   The primary objective is to ensure that every row of the key submatrix $(\mathbf{B}^{-1}\mathbf{b}, \mathbf{B}^{-1})$ in the tableau remains **lexicographically positive** ($\succ \mathbf{0}$) after every pivot. Lexicographical positivity is maintained even when degeneracy occurs ($\hat{b}_i = 0$).

   - **Pivot Row (r):** The row trivially remains non-negative after normalization because the pivot element $\bar{a}_{rk}$ is positive.
   - **Other Rows ($i \neq r$):** Analysis of the row operations (based on partitioning the index sets $I_0, I_1, \dots$) ensures that even if the right-hand side element $\hat{b}_i$ becomes zero, the subsequent elements in the B^{-1} part of the row guarantee the row maintains $\succ 0$ positivity.

   Conclusion: Each row of the extended vector $(\hat{b}, B^{-1})$ is $\succ 0$ at every iteration, maintaining a form of "lexicographical feasibility."

2. **Strict Lexicographical Improvement of the Objective Function**

   The lexicographical rule ensures a strict, non-repeatable improvement in the objective function at every step.

   - **Row 0 Comparison:** The difference between the objective row vector before and after the pivot is shown to be proportional to the extended pivot row vector

   $$(c_B B^{-1} b, c_B B^{-1}) - (c_{B'} B'^{-1} b, c_{B'} B'^{-1}) = \frac{c_k - z_k}{\hat{a}_{rk}} \cdot (\hat{b}_r, \bar{a}_r)$$

   - **Strict Decrease:** Since the factor $\frac{c_k - z_k}{\bar{a}_{rk}}$ is **strictly positive** ($c_k - z_k > 0$ and $\bar{a}_{rk} > 0$), and the pivot row vector $(\hat{b}_r, \bar{a}_r)$ has been established as **lexicographically positive** ($\succ 0$), it follows that

   $$\text{(Old Row 0)} \succ \text{(New Row 0)}$$

   The objective value is therefore **strictly decreasing** in the lexicographical sense at every step.

3. **Proof of Finite Convergence**

   The strict lexicographical decrease of the objective row vector guarantees the algorithm never revisits the same basis.

   - **By Contradiction:** Assume a cycle of bases $B_j, \dots, B_t$ is generated such that $B_t = B_j$.
   - **Impossibility:** Summing the strict lexicographical decreases over the cycle would require the net change to be strictly lexicographically positive ($\succ 0$), leading to the contradiction $0 \succ 0$.

   Conclusion: This contradiction proves that the bases generated are **distinct**. Since there is only a finite number of possible bases, the procedure is guaranteed to converge in a finite number of steps.

3. Time Complexity of Lexicographic Rule

The algorithm provided is the Simplex Method combined with the Lexicographic Rule (implemented in the lexico function). The Lexicographic Rule is crucial because it ensures the algorithm does not cycle during degeneracy, thereby guaranteeing finite convergence.

Here is a detailed breakdown of the time complexity, considering the size of the problem; m constraints (rows) and n original non-basic variables (columns). The full tableau size is $(m+1)(m+n+1)$.

Cost Per Iteration

The cost per iteration in the main simplex loop is dominated by the Pivot Step and, potentially, the Lexicographic Rule check.

The cost of each iteration is determined by the most expensive operation: the Pivot step or the Lexicographic tie-breaking routine.

1. **Standard Pivot Cost (Gauss-Jordan)** The majority of the work involves updating the $m + 1$ rows of the tableau, each of size $n + m + 1$.

$$\text{Cost per Pivot} = O(m \times (n + m)) = O(mn + m^2)$$

2. **Lexicographic Rule Cost (Lexico function)** The lexico function is called only when a tie occurs. It performs a comparison that can iterate over all $n + m$ non-RHS columns of the tableau.

   - Let $T \leq m$ be the number of tied rows.
   - The algorithm iterates over $j = 0$ to $n + m - 1$ columns.
   - In the worst case, the loop runs $n + m$ times, with $O(T)$ operations inside the loop.

$$\text{Lexicographic Cost (Worst-Case)} = O((n + m)m)$$

   The overall **Cost Per Iteration** is dominated by these operations: $O(mn + m^2)$.

Number of Iterations

The Lexicographic Rule's primary function is to provide a theoretical bound on the number of iterations by preventing cycling.

1. **Worst-Case Complexity (Theoretical)** The Lexicographic Rule guarantees that a distinct basis is visited at every step, even under degeneracy, proving **finite termination**.

   - The number of iterations is bounded by the total number of possible Basic Feasible Solutions (vertices): $\binom{n+m}{m}$.
   - **Worst-Case Iterations:** The bound remains **exponential**.

$$\text{Worst-Case Iterations} \approx O(2^{n+m})$$

2. **Average-Case Complexity (Practical)** In practice, the Simplex Method is highly efficient, and the Lexicographic Rule adds minimal overhead.

   - **Empirical Observation:** The number of iterations typically scales polynomially.
   - **Average-Case Iterations:** $O(m)$ or $O(m + n)$.

Overall Time Complexity

- **Worst-Case Overall:**

$$O(\text{Iterations} \times \text{Cost per Pivot}) = O(2^{n+m\cdot(mn+m^2)})$$

- **Average-Case Overall (Practical):**

$$O(\text{poly}(m, n) \times \text{Cost per Pivot}) \approx O(\text{poly}(m, n) \cdot mn)$$

4. Bland's Rule

Robert Bland suggested a simple rule to prevent cycling in the Simplex algorithm, which simultaneously restricts the choice of both the entering and leaving variables.

The purpose of studying this method is to analyze how the application of a rigid selection rule breaks repetitive sequences to ensure the algorithm's finiteness.
This approach highlights a critical trade-off in computability: while the simplicity of combinatorial selection is easy to implement, it risks significantly lengthening the path to the optimum.
This creates a bottleneck in large-scale data processing that current research attempts to mitigate by isolating the structural components of the problem to streamline the pivoting process.

The rule first requires that all variables be ordered ($e.g. x_1, x_2, \ldots, x_n$).

The selection process is then based on choosing the lowest-indexed variable from the eligible candidates

- **Entering Variable** : Of all nonbasic variables that can improve the objective function($i.e.$,have $c_j - z_j > 0$).
- **Leaving Variable** : In the minimum ratio test, if there is a tie for the variable to leave the basis (indicating a degenerate pivot), the variable with the smallest index among the tied candidates is chosen to exit.

In the absence of degeneracy (when the minimum ratio test has a unique leaving variable), the objective function strictly decreases, naturally preventing a return to a previous basis. Other methods, like the lexicographic rule, guarantee finite convergence by ensuring a lexicographically monotone decrease in a related vector.

If a variable x_r enters the basis during a sequence of degenerate pivots, x_r cannot leave the basis until another variable with a higher index than r (which was nonbasic when x_r entered) has also entered the basis.

This feature prevents cycling because in a cycle, every variable that enters must also eventually leave. If a cycle were to occur, there would have to be a highest-indexed variable that enters and then leaves, which directly contradicts the established monotone feature.

The text outlines a proof by contradiction

- Assume a degenerate sequence leads to a pivot where x_p enters and the initial entering variable x_q leaves, with all nonbasic variables x_j with $j > q$ remaining nonbasic.
- By analyzing the conditions at this new pivot using the Simplex tableau representation, it's shown that another basic variable, x_r, must exist such that $r < q$ and x_r is also eligible to leave the basis (it's zero and ties in the minimum ratio test).
- Since $r < q$, Bland's rule would select the lower indexed x_r to leave, not x_q.
- This contradiction proves that x_q cannot leave the basis until a variable with an index higher than q has entered.

For more detailed explanation refer to [1,9]

5. Complexity Analysis of the Simplex Algorithm with Bland's Rule

Cost Per Pivot

The computational complexity of a single pivot operation is dominated by the row operations required to update the entire tableau.

- **Pivot Row Normalization**

  The pivot row (with $n + m + 1$ elements) is divided by the pivot element.

  $$\text{Complexity:} \quad O(n + m)$$

- **Operations (Elimination)**

  The pivot column is cleared by performing a row operation on the remaining m constraint rows and 1 objective row (m rows total, excluding the pivot row). Each operation involves $n + m + 1$ elements.

$$\text{Complexity:} \quad m \times O(n+m) = O(mn + m^2)$$

- **Overall Cost Per Pivot**

  The total cost per pivot is the sum of these steps

$$\text{Cost per Pivot} = O(n+m) + O(mn + m^2) = O(m(n+m))$$

  In cases where $n \gg m$, this simplifies to $O(mn)$.

Number of Iterations

- **Worst-Case Complexity**

  The general Simplex algorithm is susceptible to cycling under degeneracy, but even with anti-cycling rules like Bland's, the theoretical worst-case remains exponential. The Klee-Minty cube examples demonstrate that the algorithm can visit an exponential number of vertices.

  – **General Simplex:** The maximum number of vertices bases is $\binom{n+m}{m}$.

  – **Worst-Case Iterations:**

$$\text{Worst Case Iterations} = O\left(2^m\right) \quad \text{or} \quad O\left(\binom{n+m}{m}\right)$$

  Bland's Rule **guarantees finite termination** (by preventing cycling) but **does not improve** this exponential worst-case bound.

- **Average Case Complexity**

  In the real world, the simplex algorithm is actually incredibly fast. Even though it is hard to prove why with just pen and paper, years of testing show that as problems get bigger, the time it takes to solve them only grows a little bit. IT usually finds the answer almost as fast as you can feed it the data.

$$\text{Average Case Iterations} \approx O(m) \quad \text{to} \quad O(m \cdot log(n))$$

  This leads to an efficient average total complexity

$$\text{Average Total Complexity} \approx O(m \cdot log(n) \cdot m(n+m)) \approx O(m^2 n log(n)).$$

Conclusion

This study has explored the mathematical mechanics and complexity trade-off two classical anti-cycling rules which are well-established pillars in linear programming literature. Our analysis reveals that while both successfully resolve the theoretical issue of cycling, they incur a substantial computational cost.

The first approach introduces a computationally expensive vector-based tie-breaking system, whereas the second, despite its simplicity, often leads the simplex algorithm along inefficient and circuitous paths to optimality.

In a modern context-ranging from large-scale optimization to emerging AI safety applications involving millions of variables these inefficiencies become clear bottlenecks, as their high per-iteration complexity and tendency to induce stalling render them poorly suited for industry-scale use.

Consequently, recent research has shifted away from merely treating the symptoms of cycling to addressing the internal geometry of redundant bases, allowing for an effective reduction in problem size. By isolating the specific variables responsible for stagnation, these emerging techniques aim to stabilize the algorithm while optimizing computational complexity, paving the way for a more fluid resolution of large-scale system.

Therefore, this work lays the necessary foundation for shifting toward these modern anticycling strategies, ensuring that linear programming remains a scalable and reliable tool for the next generation of autonomous intelligence systems.

Acknowledgments

Special thanks to editors and anonymous referees for their recommendations that helped us to improve the content of our paper.

References

1. Bazaraa, M. S., Jarvis, J. J., & Sherali, H. D., *Linear Programming and Network Flows*, (4th edition). John Wiley & Sons, (2010).
2. Bazaraa, M. S., Sherali, H. D., & Shetty, C. M., *Nonlinear Programming: Theory and Algorithms*, (3rd ed.). Wiley-Interscience, (2006).
3. Dantzig, G. B., & Thapa, M. N., *Linear Programming 1: Introduction*. Springer-Verlag, (1997).
4. Dodge, Y., & Melfi, G., *Optimisation appliquée*. Springer, (2008).
5. Lay, D. C., *Linear Algebra and Its Applications*. Addison-Wesley, (1994).
6. Luenberger, D. G., & Ye, Y., *Linear and Nonlinear Programming*, (3rd ed.). Springer, (2008).
7. Minoux, M., *Programmation mathématique : Théorie et algorithmes*. Tec & Doc, (2008).
8. Rockafellar, R. T., *Convex Analysis*. Princeton University Press, (1970).
9. Vanderbei, R. J., *Linear Programming: Foundations and Extensions*, (4th ed.). Springer, (2014).complexity

Asmaa AIT BRIK,
LAM2A, Faculty of sciences Ain Chock (FSAC),
University Hassan II of Casablanca,
Casablanca, Morocco.
E-mail address: aitbrikasmaa15@gmail.com

and

Seddik ABDELALIM,
LAM2A, Faculty of sciences Ain Chock (FSAC),
University Hassan II of Casablanca,
Casablanca, Morocco.
E-mail address: seddikabd@hotmail.com