ON THE COMPLEXITY OF
MINOS
PACKAGE FOR LINEAR PROGRAMMING

Neculai Andrei
Research Institute for Informatics
Center for Advanced Modeling and Optimization
Bucharest, Romania

Dedicated to Professor Michael A. Saunders on the occasion of his 60th birthday

Abstract
Using the numerical computational evidence of the MINOS package, for a number of linear programming problems of NETLIB collection, into an empirical study referring to the number of iterations, we conjecture its quasi-linear complexity. Empirically we conjecture that the number of iterations of MINOS, for solving linear programming problems with $m$ constraints and $n$ variables, is quasi-linear in $m + n$.

1. Introduction

The MINOS package of Murtagh and Saunders [1978, 1982, 1995] is one of the most respected packages dedicated for solving linear and nonlinear optimization problems. The system permits the solution of large-scale problems in the following area of continuous optimization: linear programming, unconstrained optimization, linearly constrained optimization and nonlinear constrained optimization. The algorithms implemented in MINOS are: the simplex method [Dantzig, 1963], the quasi-Newton method [Davidon, 1959], the reduced-gradient method [Wolfe, 1962], and the projected Lagrangian method [Robinson, 1972], [Rosen and Kreuser, 1972].

MINOS is designed for solving the large-scale optimization problems expressed in the following standard form:

$$\min_{x,y} F(x) + c^T x + d^T y$$

subject to

$$f(x) + A_1 y = b_1,$$
$$A_2 x + A_3 y = b_2,$$
$$l_x \leq x \leq u_x,$$
$$l_y \leq y \leq u_y,$$

\[1\] Special thanks are extended to the Romanian Academy who supported this research through Grant no. 168/2003.
where the vectors \( c, d, b_1, b_2, l_x, l_y, u_x, u_y \) and the matrices \( A_1, A_2, A_3 \) are constant, \( F(x) \) is a smooth scalar function, and \( f(x) \) is a smooth vector function. The components of \( x \) are called the nonlinear variables. The components of \( y \) the linear variables. As we can see the constraints of the problem are separated into nonlinear constraints and linear ones respectively. In this form (1) represents one of the most general continous optimization problem.

If in (1) the functions \( F(x) \) and \( f(x) \) are absent, then the problem is a linear program. MINOS solves linear programs by implementing the primal simplex algorithm using a sparse LU factorization of the basis with Markowitz ordering scheme and Bartels-Golub updates.

If in (1) the nonlinearityes are present only in the objective function by the term \( F(x) \), then the problem is a linearly constrained nonlinear program. For these problems MINOS implements a reduced-gradient algorithm in conjunction with a quasi-Newton algorithm. In this case the variables of the problem are partitioned in basic, nonbasic and superbasic variables. The superbasic variables represents a set of independent variables that are free to move in any direction in order to improve the value of the objective, or to reduce the sum of infeasibilities. The basic variables can be modified in order to satisfy the linear constraints of the problem. At a solution, the basic and superbasic variables lie between their bounds, while the nonbasic variables are equal to one of their bounds.

If the problem (1) contains nonlinear constraints, then MINOS implements a projected augmented Lagrangian algorithm, where a sequence of major iterations are performed, each one involving the solution of a linearly constrained subproblem. Each subproblem contains the original linear constraints and bounds on variables, as well as the linearized versions of the nonlinear constraints. Thus, the \( f(x) \) in (1) is replaced by its linear approximation at the current point:

\[
\tilde{f}(x, x_k) \equiv \tilde{f} = f(x_k) + J(x_k)(x - x_k),
\]

where \( x_k \) is the estimate of the nonlinear variables at the \( k \)-th iteration, and \( J(x_k) \) is the Jacobian of the function. The subproblem considered at each major iteration is:

\[
\min_{x,y} F(x) + c^T x + d^T y - \lambda_k^T (f - \tilde{f}) + \frac{1}{2} \rho (f - \tilde{f})^T (f - \tilde{f})
\]

subject to:

\[
\begin{align*}
\tilde{f} + A_1 y &= b_1, \\
A_2 x + A_3, y &= b_2, \\
l_x &\leq x \leq u_x, \\
l_y &\leq y \leq u_y.
\end{align*}
\]

The objective in (2) is an augmented Lagrangian, \( \lambda_k \) is an estimation of the Lagrange multipliers associated to nonlinear constraints, and \( \rho \) is a penalty parameter. For solving (2) MINOS uses the reduced-gradient algorithm where \( J(x_k) \) and \( A_j \) are treated as sparse matrices.
The complexity study of MINOS algorithm, i.e. the determination of a closed formula predicting the number of iterations necessary to be taken to get a solution with a prespecified accuracy, is not an easy task. In this paper we consider the complexity of MINOS for solving linear programming problems from an empirical viewpoint. The idea is to analyse the results of MINOS package, referring to the number of iterations for solving a number of linear programming problems, into the context of the essence of simplex algorithm. Section 2 contains some aspects on the complexity of simplex algorithm. Using 119 linear programming problems from Netlib collection, and taking into account the principle of the simplex algorithm, in section 3, we conjecture that the complexity of MINOS package is quasi–linear in $m + n$.

2. Some aspects on the complexity of simplex algorithm

As we know linear programming considers the optimization of a linear function over a feasible set defined by a finite number of inequalities:

$$\min \{ c^T x : Ax \leq b, x \geq 0 \}, \quad x \in \mathbb{R}^n, \quad b \in \mathbb{R}^m.$$  

If $X = \{ x : Ax \leq b, x \geq 0 \}$, the feasible region, has vertices and if (3) has optimal solutions, then there is a vertex which is the optimum for (3). Each vertex of $X$ has at least $n$ restrictions which are active. Each edge of $X$ has at least $n - 1$ active restrictions. The problem seems to be trivial, since we need to examine a finite number of vertices of the polyhedron $X$, but when we are faced with large-scale problems, (3) turns out to be very challenging.

In 1947 Dantzig articulated the simplex method for linear programming which was the first practical approach for solving these problems and which remains widely used today. Basically, the simplex method has two phases. The first phase determine an initial vertex $x_0 \in X$. The second one construct a sequence of vertices $x_0, x_1, \ldots, x_s \in X$, such that for $i = 0, 1, \ldots, s - 1$ the vertices $x_i$ and $x_{i+1}$ are adjacent, and $c^T x_i < c^T x_{i+1}$. If $x_i$ is optimal, or at $x_s$ it is evident that an optimal solution does not exist, then stop. It is quite clear that both phases are similar and there is a freedom „how to determine the successor vertex“, if there are more than one possibility. The rule of selecting the vertex along the iterations, known as the rule of pivoting, determine the variant of the simplex algorithm. It is known that the most difficult step is the pivoting rule. The most known pivoting rules can be classified in the following three categories:

A) Variants of simplex considering information on the shape of $X$ and the objective:

1. Rule of greatest improvement. Choose the edge which gives the maximal improvement in the objective function value.
2. Steepest edge of Goldfarb and Reid. Choose that incident improving edge with smallest angle to the gradient of the objective.
B) Variants of the simplex considering only the objective:
3. **Dantzig’s rule.** Take that edge corresponding to the most negative reduced cost.
4. **Parametric rule (shadow-vertex algorithm).** This is the parametric objective simplex algorithm by providing all optimal vertices to the family of objectives \((c + \lambda \tilde{c})^T x\).

C) Variants of the simplex using combinatorial rules:
5. **Bland’s rule.** The current vertex \( v \) satisfies exactly \( d \) constraints with equality, i.e. it lies at the intersection of \( d \) faces of the polytope. Any adjacent vertex \( w \) lies on \( d - 1 \) of these faces. Therefore, when moving from \( v \) to \( w \), exactly one constraint is loosened from tightness. Choose that vertex \( w \) so that the constraint that is loosened has least index.
6. **Rule of justice.** Consider that edges that had been active very rarely.
7. **Random pivot selection.** Select randomly one of the improving edges.

Thus, the simplex algorithm is defined by the **pivot rule** – the way which decides which vertex of the polyhedron is selected when there are many to choose from. There is no deterministic pivot rule under which the simplex algorithm is known to take a subexponential number of iterations. For every deterministic pivot rule it is possible to build up a family of polytopes on which the simplex algorithm takes an exponential number of iterations. (See the papers: [Klee and Minty, 1972], [Jeroslow, 1973], [Avis and Chvatal, 1978], [Goldfarb and Sit, 1979], [Murty, 1980], [Kalai, 1992], [Matousek, Sharir and Welzl, 1996]). A survey and a unified procedure for construction of these polytopes is given in [Amenta and Ziegler, 1999]. The numerical behavior of simplex algorithm is very intriguing, being remarkably efficient in practice, while having no polynomial time worst-case complexity.

In 1979, Khachiyan used the ellipsoid algorithm to linear programming and proved that it always is convergent in time polynomial in \( m \), \( n \) and \( L \) – number of bits needed to represent the problem in computer. However, with all the efforts, in practice, the ellipsoid algorithm was not competitive with the simplex method.

The interior–point method introduced in 1984 by Karmarkar, which was the origin of a real revolution in mathematical programming, proved to be polynomial in \( m \), \( n \) and \( L \) and was developed in numerous variants that are competitive with and occasionally superior to the simplex method in practice (especially for large–scale problems).

In spite of more than 60 years of attempts to unseat it, the simplex method remains the most popular method for solving linear programs. However, there are no satisfactory theoretical explanation of its excellent performance.

The complexity study of simplex algorithm, defined as the number of iterations (pivoting) as a function of the dimensions of the problem (\( m \) and \( n \)),
classify in two main directions: a controlled numerically one and a theoretical oriented study, respectively. An excellent survey of the efficiency of the simplex algorithm was given by Shamir [1987].

The computational experience accumulated along the five decades on the behavior of the simplex algorithm in practice, in different implementation formule, is vast. For the very beginning it has been noted that the primal simplex algorithm needs $2m$ up to $3m$ iterations to get a solution. Based on empirical experience with “thousands” of practical problems, Dantzig [1963, pp.160] conclude:

"For an $m$-equation problem with $m$ different variables in the final basic set, the number of iterations may run anywhere from $m$ as a minimum to $2m$ and rarely to $3m$. The number is usually less than $3m/2$ when there are less than 50 equations and 200 variables (to judge from informal empirical observations)."

This is an empirical result in which the numerical effort of the first phase is included. Dantzig follows:

"It has been conjectured that, by proper choice of the variables to enter the basic set, it is possible to pass from any basic feasible solution to any other in $m$ or less pivot steps, where each basic solution generated along the way must be feasible."

This is known as Hirsch’s conjecture, given by Hirsch in 1957, being independent of the techniques for pivot selection:

"in a convex region in $n−m$ dimensional space defined by $n$ halfspaces, is $m$ an upper bound for the minimum length chain of vertices joining two given vertices? "

Wolfe and Cutler [1963], Marsten [1974], Goldfarb and Reid [1977], Ho and Loute [1983] consider the behavior of simplex algorithm on a limited number of LP problems. Their conclusion is that the number of pivot steps is usually between $m$ and $4m$ when $n/m$ is about 3. Only very seldom does the number of iterations exceed $10m$. When $n/m$ increases, then the number of iterations seems to increase slowly [Shamir, 1987]. More recent numerical experiments given by Bixby [1992, 2002] with CPLEX on 90 Netlib [Gay, 1985] linear programming problems shows that for 72 of them (i.e. 80%), the number of total iterations was at most 3 times the number of rows. For unbalanced $m$ and $n$ problems the number of iterations is larger [Todd, 2002].

Using a probabilistic approach Kuhn and Quandt [1963], Avis and Chvátal [1978] and Dunham, Kelly and Tolle [1977] consider the analysis of the simplex algorithm when the problem data are randomly generated according to some predetermined distribution. Although the numerical experiments documented in the above mentionaed papers differ in many respects, Ron Shamir [1987] comes to the following overall conclusion:
"The average number of pivots appears to be a slightly superlinear function of one dimension of the problem, and a sublinear, slowly increasing function of the other dimension". With other words, "the smaller dimension $n$ (the dimension of the polyhedra) enters the mean value function of the number of iterations in a slightly superlinear way, and the larger dimension $m$ (the number of inequality constraints including the sign-one) has only a significantly sublinear influence"

Bordwardt, Damm, Donig and Joas [1993] tested seven variants of simplex algorithm with the above mentioned pivoting rules under three rotation-symmetric distributions: uniform distribution on $R^n$, uniform distribution on the full unit ball of $R^n$ and uniform distribution on the unit sphere of $R^n$, respectively. Their conclusion is that the best performance shows the rule of steepest ascent. This algorithm is slightly better than the rule of greatest improvement. These two variants show a very good performance when the current vertex is far away from the optimal one. The worse performance is associated with the combinatorial variants. Among them the best is the random choice, followed by the rule of justice and finally by the Bland’s rule.

The theoretical study of simplex’s complexity is based on the study of polyhedra on one hand, and on a probabilistic analysis on the other hand. Any explanation of simplex algorithm begins with the study of polytopes and polyhedra. McMullen [1970] showed that an $d-$dimensional polytope with $n-$facets has at least

$$\left\lfloor n - \frac{d+1}{2} \right\rfloor + \left\lfloor n - \frac{d+2}{2} \right\rfloor$$

vertices. Klee [1974] extended this result to unbounded polyhedra.

The success of the simplex method can be explained by the theoretical study of the diameter of the polyhedron corresponding to the constraints of the problem. This is the largest number of edges in a shortest path joining two of its vertices. Then, the best number of iterations for the worst linear programming problem is given by the largest diameter of the corresponding polyhedron. Considering $D(m,n)$ the largest diameter of a $m-$dimensional polyhedron with $n$ facets, Hirsch conjectured that $D(m,n) \leq n - m$. Excepting the case $m-n \leq 5$ and $n = 3$, the Hirsch’s conjecture is still open. The best bounds on the diameter are $D(m,n) \leq 2^{m-3}n$ given by Larman [1970], and $D(m,n) \leq n^{1+\log m}$ of Kalai and Kleitman [1992]. However, the existence of such a short path does not imply that the simplex algorithm will find it.

Klee and Minty gave the first example of a polytope for which the simplex
algorithm with standard pivoting rule take an exponential number of iterations. Kalai [1992] and Matousek, Sharir and Welzl [1996] proved that the best bound of the iterations number for the simplex with random pivot rule is \( \exp\left(K \sqrt{d \log(n)}\right) \), where \( K \) is a constant. To our knowledge, the best analysis of random pivot rules shows a \( n^{O(\sqrt{n})} \) time [Kalai, 1992].

The idea of probabilistic analysis is to study a deterministic variant of the simplex algorithm to solve random problem-instances. Two stochastic models are known: Sign-Invariance Model (SIM) and Rotation-Symmetry Model (RSM). A sign matrix is a diagonal matrix with +1 or −1 on the diagonal. For problem (3) with data \((A, b, c)\) the sign-invariance model consider another problem as: \((S, AS, Sb, c^T S)\). The rotational-symmetry model considers \(b = [1, 1, \ldots, 1]\) and the rows \(a_1, \ldots, a_m\) of the matrix \(A\) and \(c\) be distributed independently, identically and symmetrically under rotations on \( R^a \). A number of average–case analyses of the simplex algorithm using both models have been performed. For example, Borgwardt [1980] proved that the simplex algorithm with the shadow vertex pivot rule runs in expected polynomial time for polytopes whose constraints are drawn independently from spherically symmetric distributions. For the problem

\[
\max\{c^T x : Ax \leq b, x \in R^n, b \in R^m\},
\]

where \( b \) is positive, the average number of pivot steps is \( O(n^2 m^{1/(n-1)}) \) [Borgwardt, 1999]. Using different distributions for fixed \( n \) and \( m \to \infty \), the expected number of pivot steps \( E_{m,n}(S) \), where \( S \) is the number of shadow-vertices is as follows (see the papers of Borgwardt, [1977, 1987] and Küfer, [1996]).

- Gaussian distribution on \( R^n \) : \( E_{m,n} = \sqrt{\ln(m)} n^{3/2} \).
- Uniform distribution on the full unit ball of \( R^n \) : \( E_{m,n} = n^2 m^{1/(n+1)} \).
- Uniform distribution on the unit sphere of \( R^n \) : \( E_{m,n} = n^2 m^{1/(n-1)} \).

Using a SIM probabilistic model, and a different pivot rule, Smale [1983] proved bounds on the expected running time of Lemke’s self–dual parametric simplex algorithm on linear programming problems chosen from a spherically–symmetric distribution (Gaussian distribution centered at the origin). For the problem

\[
\max\{c^T x : Ax \leq b, x \geq 0, x \in R^n, b \in R^m\}
\]

for every fixed \( m \), the average number of pivots given by Smale is \( E_{m,n}(s) = C(n)\left(1 + \ln(m + 1)\right)^{n/(n+1)} \). (\( C(n) \) is an exponential function of \( n \).)
Smale’s bound is not polynomial, but when \( n \to \infty \), for fixed \( m \), it is better than that of Borgwardt. Megiddo [1986] improved the analysis of Smale.

Another model of random linear programming problems was studied independently by Haimovich [1983] and Adler [1983]. They consider the maximum subject to the matrices \( A \), of the expected time taken by parametric simplex algorithms to solve linear programs for which the directions of the inequalities are chosen at random. They proved that parametric simplex algorithm takes an expected linear number of steps to go from the vertex corresponding to the minimum the objective function to the vertex corresponding to the maximum of the objective function. Haimovich combined the pivot rule of Borgwardt with a more general version of Smale’s probabilistic model, and proved that the average number of pivots is linear and can be bounded by:

\[
\min \left\{ \frac{1}{2} n, \frac{1}{2} (m + m + 1), \frac{1}{8} (m + 1) \right\}, \text{ for problems of type (4)}
\]

\[
\min \left\{ \frac{1}{2} n, \frac{1}{2} (m + 1), \frac{1}{8} (m + n + 1) \right\}, \text{ for problems of type (5)}.
\]

In fact, Haimovich [1983] obtained some better bounds:

\[
\left( \frac{2}{n + \frac{2}{m+1}} \right)^{-1} \leq \min \left\{ n, m+1, (m + n + 1) / 4 \right\} / 2,
\]

for problems of type \( \max \left\{ c^T x : Ax \leq b, x \geq 0, x \in R^n, b \in R^m \right\} \),

\[
\left( \frac{2}{n} \frac{2}{m-n+1} \right)^{-1} \leq \min \left\{ n, m-n+1, (m+1) / 4 \right\} / 2,
\]

for problems of type \( \max \left\{ c^T x : Ax \leq b, x \in R^n, b \in R^m \right\} \) and

\[
\left( \frac{2}{n-m} \frac{2}{m+1} \right)^{-1} \leq \min \left\{ n-m, m+1, (n+1) / 4 \right\} / 2,
\]

for problems of type \( \max \left\{ c^T x : Ax = b, x \geq 0 \right\} \).

Recently, Spielman and Teng [2003] proposed an analysis, called smoothed analysis, based on Gaussian perturbations of inputs to algorithms, and measure the running times in terms of their input size and the standard deviation of the Gaussian perturbations. In fact, the smoothed analysis is an interpolation between worst case and average case analysis, i.e. combines the worst case \( \max T(x) \) with the average case \( \text{avg}_r T(x) \) as \( \max \text{avg}_r T(x + er) \), where \( r \) is a perturbation. They consider the maximum over \( \tilde{A} \) and \( \tilde{y} \) of the expected running time of the simplex algorithm on problems of the form:

\[
\max \left\{ z^T \left( \tilde{A} + G \right) x \leq \left( \tilde{y} + h \right) \right\},
\]

where \( \tilde{A} \) and \( \tilde{y} \) are arbitrary and \( G \) and \( h \) are matrix and vector of independently chosen Gaussian random variables of mean 0 and standard deviation \( \sigma \). In these circumstances they proved that a two–phase shadow–
vertex simplex method to solve such a linear program is polynomial in $1/\sigma$ and the dimensions of $\tilde{A}$. The smoothed analysis exploits the geometric properties of the condition number of a matrix.

All these efforts show a frustration and an inconsistency between the exponential worst-case behavior of the simplex method and its everyday practicality. More than this, the average-case analysis of the simplex algorithm, on different random linear programs hypothesis are very unlike, mainly because the hypothesis of randomness are not at all so present in real problems as it is supposed. Real large-scale linear programming problems always are sparse and in many cases the nonzero elements of the activity matrix have unity values. All these characteristics of real linear problems are not considered in any probabilistic hypothesis of the above analysis.

In these circumstances it is necessary to consider another approach in which the numerical evidence of the implementation of the simplex method is placed on the first place. To verbalize: using computational evidence how many iterations, on average, do we expect a package implementing the simplex method to take for solving a linear programming problem with $m$ constraints and $n$ variables? The idea is to try to find a closed formula for the number $T$ of iterations, as a function of the dimensions of the problem, using only the results of computational experiments with simplex packages, and taking into account the internal mechanisms of the simplex algorithm.

Considering the dynamics of the simplex algorithm (no matter which variant) we see that at each iteration the algorithm selects the corresponding primal nonbasic variable to enter the basis, and a leaving variable is chosen by means of ratio test. After pivoting the the coefficient of the variable that exited the basis is positive (assuming the nondegeneracy of the problem). Therefore, the whole effect of one pivoting in the simplex algorithm is to change the sign of one of the negative values from a list of $m+n$ values. In fact this is the essence of the simplex algorithm. Every variant of the simplex algorithm try to solve in an efficient manner the degeneracy and concentrates on this idea of sign changing.

Without lost of generality we can assume that out of $m+n$ values $(m+n)/2$ are negative. So, for nondegenerate linear programming problems we can expect, (i.e. we conjecture) that the simplex algorithm will take $(m+n)/2$ iterations, on average.

Numerical experiments of Vanderbei [1996] with a primal-dual simplex method on 69 Netlib linear programming problems show that the total number of iterations for this variant of simplex algorithm is:

$$T \approx 0.488(m+n)^{1.052}$$

which is very close to $(m+n)/2$. The conclusion given by Vanderbei is that for nondegenerate linear problems the number of iteration of any version of simplex method seems to be linear in $m+n$ [Vanderbei, 1996, pp.187].

More recent experiments by Andrei [2003] with LPACKO primal simplex
[Park, et all, 1998] and GLPK [Makhorin, 2002] on a different number of Nettlib problems shows that the total number of iterations corresponding to these packages can be approximated by the function:

\[ T \approx 2^\alpha (m + n)^\beta, \]

where the parameters \( \alpha \) and \( \beta \), as well as the number of problems considered are given as:

<table>
<thead>
<tr>
<th>Package</th>
<th># problems</th>
<th>( \alpha )</th>
<th>( \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPAKO</td>
<td>55</td>
<td>-2.38199</td>
<td>1.078104</td>
</tr>
<tr>
<td>CPLEX 1.1</td>
<td>38</td>
<td>-1.175938</td>
<td>0.992153</td>
</tr>
<tr>
<td>PD (Vanderbei)</td>
<td>69</td>
<td>-1.03561</td>
<td>1.05151</td>
</tr>
<tr>
<td>GLPK</td>
<td>94</td>
<td>-1.459651</td>
<td>1.0205</td>
</tr>
</tbody>
</table>

So, based on empirical evidence, it is strongly supported that the number of iterations for the primal simplex method is quasi-linear in \( (m + n) \). The coefficient \( 2^\alpha \) is a constant, the value of which is dependent by the degeneracy of the problems. From the above table it follows that the CPLEX package ranks first in this list. In the following we shall consider the analysis of MINOS package.

### 3. Complexity of MINOS package for Linear Programming

MINOS is able to solve large-scale linear programming problems using an advanced implementation of the primal simplex method\(^2\). In MINOS the simplex method is implemented in a very sophisticated manner, so the theoretical study of the complexity of MINOS is not easy. Therefore, we shall consider an empirical study of its complexity.

The problem considered here is as follows: using numerical evidence how many iterations, on average, do we expect the MINOS method to take for solving a linear programming problem with \( m \) constraints and \( n \) variables? The idea is to try to find a closed formula for the number \( T \) of iterations using only the results of computational experiments with MINOS.

As we know the simplex algorithm, as implemented in MINOS, selects a nonbasic variable to enter the basis, and the leaving variable is determined by a ratio test. After the pivot, the variable that exited now appears as a nonbasic variable with a positive reduced cost (in case of nondegeneracy). Hence, the idea of one pivot of the simplex algorithm in MINOS is to change the sign of one of the negative values from a list of \( m + n \) values into a positive one. In a more general

---

\(^2\)Many thanks to Professor Michael Saunders who gave me the modern MINOS 5.51 version library.
context we can expect, on the average, that the number of the iterations should be a fraction \( K \) of \( m + n \). Therefore, we can conjecture that \( T \) can be approximated by a function of the form:

\[
T \approx K^\alpha (m + n)^\beta, 
\]

where \( \alpha \) and \( \beta \) is a pair of real numbers. Our goal is to find the values of these parameters that best fits the data obtained from a set of empirical observations referring to the computational behavior of MINOS.

Table 1 shows the number of iterations of MINOS for solving a number of 119 test linear programming problems of NETLIB [Gay, 1985].

<table>
<thead>
<tr>
<th>Name</th>
<th>( m )</th>
<th>( n )</th>
<th>iter</th>
<th>Name</th>
<th>( m )</th>
<th>( n )</th>
<th>iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>25fv47</td>
<td>822</td>
<td>1571</td>
<td>7025</td>
<td>pcb3000</td>
<td>3961</td>
<td>6810</td>
<td>10869</td>
</tr>
<tr>
<td>adlittle</td>
<td>57</td>
<td>97</td>
<td>100</td>
<td>pds-2</td>
<td>2954</td>
<td>7535</td>
<td>3074</td>
</tr>
<tr>
<td>afiro</td>
<td>28</td>
<td>32</td>
<td>132</td>
<td>perold</td>
<td>626</td>
<td>1376</td>
<td>4095</td>
</tr>
<tr>
<td>agg</td>
<td>489</td>
<td>163</td>
<td>103</td>
<td>pilot</td>
<td>1442</td>
<td>3652</td>
<td>16273</td>
</tr>
<tr>
<td>agg2</td>
<td>517</td>
<td>302</td>
<td>152</td>
<td>pilot4</td>
<td>411</td>
<td>1000</td>
<td>1412</td>
</tr>
<tr>
<td>agg3</td>
<td>517</td>
<td>302</td>
<td>172</td>
<td>pilotja</td>
<td>941</td>
<td>1988</td>
<td>6669</td>
</tr>
<tr>
<td>band</td>
<td>126</td>
<td>129</td>
<td>26</td>
<td>pilotnov</td>
<td>976</td>
<td>2172</td>
<td>2088</td>
</tr>
<tr>
<td>bandm</td>
<td>306</td>
<td>472</td>
<td>476</td>
<td>pilotwe</td>
<td>723</td>
<td>2789</td>
<td>4656</td>
</tr>
<tr>
<td>beaconfl</td>
<td>174</td>
<td>262</td>
<td>45</td>
<td>r05</td>
<td>5191</td>
<td>9500</td>
<td>1710</td>
</tr>
<tr>
<td>blend</td>
<td>75</td>
<td>83</td>
<td>79</td>
<td>recipe</td>
<td>92</td>
<td>180</td>
<td>27</td>
</tr>
<tr>
<td>bnl1</td>
<td>644</td>
<td>1175</td>
<td>1566</td>
<td>refine</td>
<td>30</td>
<td>33</td>
<td>26</td>
</tr>
<tr>
<td>bnl2</td>
<td>2325</td>
<td>3489</td>
<td>4972</td>
<td>rosen1</td>
<td>521</td>
<td>1024</td>
<td>2901</td>
</tr>
<tr>
<td>boeing1</td>
<td>352</td>
<td>384</td>
<td>517</td>
<td>rosen2</td>
<td>1033</td>
<td>2048</td>
<td>6059</td>
</tr>
<tr>
<td>boeing2</td>
<td>167</td>
<td>143</td>
<td>195</td>
<td>rosen3</td>
<td>2057</td>
<td>4096</td>
<td>12585</td>
</tr>
<tr>
<td>brandy</td>
<td>221</td>
<td>249</td>
<td>321</td>
<td>rosen7</td>
<td>265</td>
<td>512</td>
<td>711</td>
</tr>
<tr>
<td>capri</td>
<td>272</td>
<td>353</td>
<td>254</td>
<td>rosen8</td>
<td>521</td>
<td>1024</td>
<td>1700</td>
</tr>
<tr>
<td>cetsud</td>
<td>1907</td>
<td>3347</td>
<td>2567</td>
<td>rosen9</td>
<td>1033</td>
<td>2048</td>
<td>3084</td>
</tr>
<tr>
<td>ch</td>
<td>3853</td>
<td>5062</td>
<td>23306</td>
<td>rosen10</td>
<td>2057</td>
<td>4096</td>
<td>6730</td>
</tr>
<tr>
<td>co5</td>
<td>5879</td>
<td>7993</td>
<td>34355</td>
<td>sc105</td>
<td>106</td>
<td>103</td>
<td>27</td>
</tr>
<tr>
<td>cre-a</td>
<td>3517</td>
<td>4067</td>
<td>4015</td>
<td>sc205</td>
<td>206</td>
<td>203</td>
<td>52</td>
</tr>
<tr>
<td>cre-c</td>
<td>3069</td>
<td>3678</td>
<td>4337</td>
<td>sc50a</td>
<td>51</td>
<td>48</td>
<td>14</td>
</tr>
<tr>
<td>cq5</td>
<td>5150</td>
<td>7530</td>
<td>50423</td>
<td>sc50b</td>
<td>51</td>
<td>48</td>
<td>16</td>
</tr>
<tr>
<td>czprob</td>
<td>930</td>
<td>3523</td>
<td>1502</td>
<td>scagr25</td>
<td>472</td>
<td>500</td>
<td>499</td>
</tr>
<tr>
<td>cycle</td>
<td>1904</td>
<td>2857</td>
<td>2779</td>
<td>scagr7</td>
<td>130</td>
<td>140</td>
<td>100</td>
</tr>
<tr>
<td>d2q06c</td>
<td>2172</td>
<td>5167</td>
<td>45644</td>
<td>scfxm1</td>
<td>331</td>
<td>457</td>
<td>384</td>
</tr>
<tr>
<td>d6cube</td>
<td>416</td>
<td>6184</td>
<td>94360</td>
<td>scfxm2</td>
<td>661</td>
<td>914</td>
<td>733</td>
</tr>
<tr>
<td>degen2</td>
<td>445</td>
<td>534</td>
<td>1254</td>
<td>scfxm3</td>
<td>991</td>
<td>1371</td>
<td>1150</td>
</tr>
<tr>
<td>degen3</td>
<td>1504</td>
<td>1818</td>
<td>6918</td>
<td>scrs8</td>
<td>491</td>
<td>1169</td>
<td>769</td>
</tr>
<tr>
<td>e226</td>
<td>224</td>
<td>282</td>
<td>488</td>
<td>scsd1</td>
<td>78</td>
<td>760</td>
<td>425</td>
</tr>
<tr>
<td>etamacro</td>
<td>401</td>
<td>688</td>
<td>571</td>
<td>scsd6</td>
<td>148</td>
<td>1350</td>
<td>1142</td>
</tr>
<tr>
<td>fffh800</td>
<td>525</td>
<td>854</td>
<td>488</td>
<td>scsd8</td>
<td>398</td>
<td>2750</td>
<td>3028</td>
</tr>
<tr>
<td>m = number of constraints,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n = number of variables,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>iter = number of iterations (MINOS, simplex primal).</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Considering this set of 119 LP problems and using $L^2$-regression technique, for different values of parameter $K$, the following values of $\alpha$ and $\beta$ have been obtained, table 2:

### Table 2. The values of $\alpha$ and $\beta$.

<table>
<thead>
<tr>
<th>$K$</th>
<th>$\alpha$</th>
<th>$\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-3.02155</td>
<td>1.18766</td>
</tr>
<tr>
<td>3</td>
<td>-1.9063</td>
<td>1.18766</td>
</tr>
<tr>
<td>5</td>
<td>-1.3013</td>
<td>1.18766</td>
</tr>
<tr>
<td>7</td>
<td>-1.0763</td>
<td>1.18766</td>
</tr>
<tr>
<td>8</td>
<td>-1.00718</td>
<td>1.18766</td>
</tr>
</tbody>
</table>
Therefore, it seems that the best suited formula for \( T \) is:

\[
T \approx 8^n (m + n)^\alpha, \quad (8)
\]

where \( \alpha = -1.007 \) and \( \beta = 1.187 \), showing that the number of iterations of MINOS for solving linear problems is quasi-linear in \( m + n \), with a relatively small coefficient.

Let us consider the behavior of the formula (8) for different numbers of LP problems. Table 3 shows the values of \( \alpha \) and \( \beta \) corresponding to some number of LP problems from table 1.

<table>
<thead>
<tr>
<th># problems</th>
<th>( \alpha )</th>
<th>( \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>-1.264688</td>
<td>1.294397</td>
</tr>
<tr>
<td>60</td>
<td>-1.100808</td>
<td>1.241885</td>
</tr>
<tr>
<td>70</td>
<td>-1.057952</td>
<td>1.230226</td>
</tr>
<tr>
<td>80</td>
<td>-1.142604</td>
<td>1.258767</td>
</tr>
<tr>
<td>90</td>
<td>-1.232220</td>
<td>1.280509</td>
</tr>
<tr>
<td>100</td>
<td>-1.172837</td>
<td>1.252718</td>
</tr>
<tr>
<td>119</td>
<td>-1.007185</td>
<td>1.187660</td>
</tr>
</tbody>
</table>

Therefore, we can conjecture that the number of iterations of MINOS package for solving linear programming problems is quasi–linear in \( m + n \), with a relatively small coefficient of proportionality. Clearly, this is an empirical result. But, the main aspect which must be emphasized here is that the complexity analysis of the MINOS is considered here into the direct connection with the anatomy of the simplex algorithm. This suggest us to consider the algebraic formula (8) as a prediction formula for the number of iterations.

4. Conclusion

Using an empirical approach we have established a closed formula giving the number of iterations of the MINOS package for solving linear programming problems. Basically, this is quasi–linear in \( m + n \). To emphasize this rezult more numerical experiments with MINOS package must be done.

The theoretical analysis of MINOS algorithm remains a very difficult undertaking. In this context it seems that the MINOS algorithm, like any other one, remains unknown. This suggests us that the only approach we have to consider is an empirical one. Clearly, in this case the analysis refers to the MINOS package, not to the MINOS algorithm. However, this is not a big problem because we are
most interested into the MINOS package, for solving real practical problems, than the MINOS algorithm.

References
Dunham, J.R., Kelly, D.G., Tolle, J.W., (1977) Some experimental results concerning the expected number of pivots for solving randomly generated linear programs. TR77-16, ORSA Department, University of North Carolina at Chapel Hill, 1977.


Murtagh, B.A., Saunders, M.A., (1978) Large-scale linearly constrained


