ON THE COMPLEXITY OF *MINOS* PACKAGE FOR LINEAR PROGRAMMING[1]

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 folowing area of continous 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:

$$\begin{aligned} &\min_{x,y} F(x) + c^T x + d^T y \\ &\text{subject to} \\ &f(x) + A_1 y = b_1, \\ &A_2 x + A_3 y = b_2, \\ &l_x \le x \le u_x, \\ &l_y \le y \le u_y, \end{aligned} \tag{1}$$

[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 nonlinearities 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 implemnts 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: (2)

$$\begin{aligned} &\tilde{f} + A_1 y = b_1, \\ &A_2 x + A_3 y = b_2, \\ &l_x \le x \le u_x, \\ &l_y \le y \le u_y. \end{aligned}$$

The objective in (2) is an augmented Lagrangian, λ_k is an estimation of the Lagrange multipliers associated to nonlinear constraints, and ρ is a penalty parameter. For solving (2) MINOS uses the reduced-gradient algorithm where $J(x_k)$ and A_i 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:

$$min\{c^T x: Ax \le b, x \ge 0\}, \quad x \in R^n, \quad b \in R^m. \tag{3}$$

If $X = \{x: Ax \le b, x \ge 0\}$, 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_s 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 clasified 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 coresponding 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 $(c+\lambda\overline{c})^T x$.

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 constraints 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 remarcably 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 convegent 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 competitives with and occasionally superior to the simplex method in practice (specially 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 numer 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

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

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 $D(m,n) \le n-m.$ Excepting the case $m-n \le 5$ and $n=3,$ the Hirsch's conjecture is still open. The best bounds on the diameter are $D(m,n) \le 2^{m-3} n$ given by Larman [1970], and $D(m,n) \le n^{1+\log m}$ of Kalai and Kleitman [1992]. However, the existence of such of 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_1AS_2, S_1b, c^TS_2). 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 distribuited independently, identically and symmetrically under rotations on $R^n-\{0\}$.

A number of average–case analyses of the simplex algorithm using boths 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

$$max\left\{c^Tx : Ax \le b, x \in R^n, b \in R^m\right\}, \tag{4}$$

where b is positive, the average number of pivot steps is $O\left(n^2 m^{1/(n-1)}\right)$ [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: $E_{m,n} \sim \sqrt{\ln(m)}\,n^{3/2}$.
- Uniform distribution on the full unit ball of R^n: $E_{m,n} \sim n^2 m^{1/(n+1)}$.
- Uniform distribution on the unit sphere of R^n: $E_{m,n} \sim n^2 m^{1/(n-1)}$.

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

$$max\left\{c^Tx : Ax \le b, x \ge 0, x \in R^n, b \in R^m\right\} \tag{5}$$

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 polinomial, 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:

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

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

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

$$\left(\frac{2}{n}+\frac{2}{m+1}\right)^{-1} \le min\left\{n, m+1, (m+n+1)/4\right\}/2,$$

for problems of type $max\left\{c^T x : Ax \le b, x \ge 0, x \in R^n, b \in R^m\right\}$,

$$\left(\frac{2}{n}+\frac{2}{m-n+1}\right)^{-1} \le min\left\{n, m-n+1, (m+1)/4\right\}/2,$$

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

$$\left(\frac{2}{n-m}+\frac{2}{m+1}\right)^{-1} \le min\left\{n-m, m+1, (n+1)/4\right\}/2,$$

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

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 $max_x T(x)$ with the average case $avg_r T(r)$ as $max_x avg_r T(x+er)$, where r is a perturbation. They consider the maximum over $\overline{A}$ and $\overline{y}$ of the expected running time of the simplex algorithm on problems of the form:

$$max\left\{z^T x : (\overline{A}+G)x \le (\overline{y}+h)\right\},$$

where $\overline{A}$ and $\overline{y}$ are arbitrary and G and h are matrix and vector of independently chosen Gaussian random variables of mean 0 and standard deviation $\sigma(max_i \left\|(\overline{y}_i, \overline{a}_i)\right\|)$. 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 $\overline{A}$. 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 programms 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 LPAKO primal simplex

[Park, *et all*, 1998] and GLPK [Makhorin, 2002] on a different number of Netlib problems shows that the total number of iterations corresponding to these packages can be approximated by the function:

$$T \approx 2^{\alpha} (m+n)^{\beta}, \tag{6}$$

where the parameters α and β , as well as the number of problems considered are given as:

Package	**# problems**	α	β
LPAKO	55	-2.38199	1.078104
CPLEX 1.1	38	-1.175938	0.992153
PD (Vanderbei)	69	-1.03561	1.05151
GLPK	94	-1.459651	1.0205

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^{α} 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:

$$T \approx K^{\alpha} (m+n)^{\beta} \ , \qquad (7)$$

where α and β 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 refering 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 1. **Characteristics of the problems and the number of iterations with MINOS.**

Name	m	n	**iter**	**Name**	m	n	**iter**
25fv47	822	1571	7025	pcb3000	3961	6810	10869
adlittle	57	97	100	pds-2	2954	7535	3074
afiro	28	32	9	perold	626	1376	4095
agg	489	163	103	pilot	1442	3652	16273
agg2	517	302	152	pilot4	411	1000	1412
agg3	517	302	172	pilotja	941	1988	6669
band	126	129	26	pilotnov	976	2172	2088
bandm	306	472	476	pilotwe	723	2789	4656
beaconfd	174	262	45	r05	5191	9500	1710
blend	75	83	79	recipe	92	180	27
bnl1	644	1175	1566	refine	30	33	26
bnl2	2325	3489	4972	rosen1	521	1024	2901
boeing1	352	384	517	rosen2	1033	2048	6059
boeing2	167	143	195	rosen3	2057	4096	12585
brandy	221	249	321	rosen7	265	512	711
capri	272	353	254	rosen8	521	1024	1700
cetsud	1907	3347	2567	rosen9	1033	2048	3084
ch	3853	5062	23306	rosen10	2057	4096	6730
co5	5879	7993	34355	sc105	106	103	27
cre-a	3517	4067	4015	sc205	206	203	52
cre-c	3069	3678	4337	sc50a	51	48	14
cq5	5150	7530	50423	sc50b	51	48	16
czprob	930	3523	1502	scagr25	472	500	499
cycle	1904	2857	2779	scagr7	130	140	100
d2q06c	2172	5167	45644	scfxm1	331	457	384
d6cube	416	6184	94360	scfxm2	661	914	733
degen2	445	534	1254	scfxm3	991	1371	1150
degen3	1504	1818	6918	scrs8	491	1169	769
e226	224	282	488	scsd1	78	760	425
etamacro	401	688	571	scsd6	148	1350	1142
fffff800	525	854	488	scsd8	398	2750	3028

finnis	498	614	486	sctap1	301	480	243
fit1p	628	1677	843	sctap2	1091	1880	742
fit1d	25	1026	2411	sctap3	1481	2480	888
forplan	162	421	323	seba	516	1028	294
ganges	1310	1681	662	share1b	118	225	209
ge	10340	11098	14449	share2b	97	79	103
gfrd-pnc	617	1092	656	shell	537	1775	323
greenbea	2393	5405	26850	ship04l	403	2118	266
greenbeb	2393	5404	12774	ship04s	403	1458	180
grow7	141	301	111	ship08l	779	4283	438
grow15	301	645	285	ship08s	779	2387	256
grow22	441	946	412	ship12l	1152	5427	837
israel	175	142	282	ship12s	1152	2763	414
kb2	44	41	51	sierra	1228	2036	1080
ken-07	2427	3602	1826	stair	357	467	512
ken-11	14695	21349	15343	standata	360	1075	101
lotfi	154	308	200	standmps	468	1075	247
maros	847	1443	2225	stocfor1	118	111	56
maros-r7	3137	9408	2520	stocfor2	2158	2031	1925
modszk1	688	1620	1025	tuff	334	587	557
nesm	663	2923	3472	tora4	1179	5454	831
optanyrf	529	1077	3225	tsp42	43	861	283
p0201	134	201	225	vbnl5	2966	4130	5139
p0282	242	282	296	willett	185	494	390
p0291	253	291	91	woodlp	245	2594	755
p0248	177	548	429	woodw	1099	8405	3772
p05	5091	9500	1638	work1	45	78	45
p2756	756	2756	797	vtp	199	203	163
pcb1000	1566	2428	3255				

m = number of constraints,
n = number of variables,
iter = number of iterations (MINOS, simplex primal).

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

Table 2. **The values of α and β .**

K	α	β
2	-3.02155	1.18766
3	-1.9063	1.18766
5	-1.3013	1.18766
7	-1.0763	1.18766
8	-1.00718	1.18766

Therefore, it seems that the best suited formula for T is:

$$T \approx 8^{\alpha}(m+n)^{\beta}\ , \tag{8}$$

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 α and β corresponding to some number of LP problems from table 1.

Table 3. **The values of α and β .**

# problems	α	β
50	-1.264688	1.294397
60	-1.100808	1.241885
70	-1.057952	1.230226
80	-1.142604	1.258767
90	-1.232220	1.280509
100	-1.172837	1.252718
119	-1.007185	1.187660

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

Adler, I., (1983) The expected number of pivots needed to solve parametric linear programs and the efficiency of the self–dual simplex method. Technical Report, University of California at Berkeley, May 1983.

Amenta, N., Ziegler, G., (1999) Deformed products and maximal shadows of polytopes. In B. Chazelle, J.E. Goodman and R. Pollack, (Eds.), *Advances in Discrete and Computational Geometry*, number 223 in Contemporary Mathematics, AMS, 1999, pp.57–90.

Andrei, N., (2003) On complexity of simplex method, Technical Report, Research Institute for Informatics, Bucharest, 2003.

Avis, D., Chvátal, V., (1978) Notes on Bland's pivoting rule. Polyhedral Combinatorics, volume 8 of Mathematical Programming Study, 1978, pp.24–34.

Bixby, R.E., (1992) Implementing the simplex method. The initial basis. ORSA Journal on Computing, vol.4, pp.267-284, 1992.

Bixby, R.E., (2002) Solving real–world linear programs: a decade and more of progress. Operations research, 2002 INFORMS vol.50, 2002, pp.3–15.

Borgwardt, K.H., (1977) Untersuchungen zur Asymptotik der mittleren Schrittzahl von Simplex-Verfahren in der Linearen Optimierung. Diss, Univ. Kaiserslautern, 1977.

Borgwardt, K.H., (1980) The simplex method: a probabilistic analysis. Springer–Verlag, 1980.

Borgwardt, K.H., (1999) A sharp upper bound for the expected number of shadow-vertices in LP-polyhedra under orthogonal projection on two-dimensional planes. Math. Oper. Res., vol.22, 1999, pp.925-984.

Borgwardt, K.H., Damm, R., Donig, R., Joas, G., (1993) Empirical studies on the average efficiency of simplex-variants under rotation-symmetry. ORSA J. Comput., vol.5, 1993, pp.249-260.

Borgwardt, K.H., Huhn, P., (1999) A lower bound on the average number of pivot–steps for solving linear programs – valid for all variants of the simplex algorithm. Mathematical Methods of Operations Research, vol.49, 1999, pp.175–210.

Dantzig, G.B., (1963) Linear programming and extensions. Princeton University Press, Princeton, 1963.

Davidon, W.C., (1959) Variable metric methods for minimization. Argonne National Labroatory, Argonne, 1959.

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.

Gay, D., (1985) Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Newsletter, vol.13, pp.10-12,

1985.

Goldfarb, D., Reid, J.K., (1977) A practicable steepest-edge simplex algorithm. Math. Programming, vol.12, 1977, pp.361-371.

Goldfarb, D., Sit, W.T., (1979) Worst case behavior of the steepest edge simplex method. Discrete Applied Mathematics, vol.1, 1979, pp.277–285.

Haimovich, M., (1983) The simplex algorithm is very good: On the expected number of pivot steps and related properties of random linear programs. Technical Report, Columbia University, April 1983.

Ho, J.K., Loute, E., (1983) Computational experience with advanced implementation of decomposition algorithms for linear programming. Math. Programming, vol.27, 1983, pp.283-290.

Jeroslow, R.G., (1973) The simplex algorithm with the pivot rule of maximizing improvement criterion. Discrete Mathematics, vol.4, 1973, pp.367–377.

Klee, V., (1974) Polytope pairs and their relationship to linear programming. Acta Mathematica, vol.133, 1974, pp.11-25.

Klee, V., Minty, G.J., (1972) How good is the simplex algorithm? In Shisha, O., (Ed.) Inequalities III, Academic Press, 1972, pp.159–175.

Kalai, G., (1992) A subexponential randomized simplex algorithm. Proc. 24th Ann. ACM Symposium on Theory of Computing. Victoria, B.C. Canada, May 1992, pp.475–482.

Kalai, G., Kleitman, D.J., (1992) A quasi-polynomial bound for diameter of graphs of polyhedra. Bulletin of the AMS, vol.24, 1992, pp.315-316.

Karmarkar, N., (1984) A new polynomial time algorithm for linear programming. Combinatorica, vol.4, 1984, pp.373–395.

Khachiyan, L.G., (1979) A polynomial algorithm in linear programming. Doklady Akademia Nauk SSSR, 1979, pp.1093–1096.

Küfer, K., (1996) An improved asymptotic analysis of the number of pivot steps required by the simplex algorithm. Z. Oper. Res., vol.44, 1996, pp.147-170.

Kuhn, H.W., Quandt, R.E., (1963) An experimental study of the simplex method. in Metropolis et al. (Eds.) Proc. 15th Symposium on Appl. Math., AMS, 1963, pp.107-124.

Larman, D.G., (1970) Paths on polytopes. Proceedings of the London Mathematical Society, vol.20, 1970, pp.161-178.

Marsten, R., (1974) An algorithm for large set partitioning problems. Management Sciences, vol.20, 1974, pp.774-787.

Matousek, J., Sharir, M., Welzl, E., (1996) A subexponential bound for linear programming. Algorithmica, vol.16, 1996, pp.498–516.

Makhorin, A., (2002) GLPK. GNU linear programming kit. Version 3.0.8, User's guide, May 2002.

McMullen, P., (1970) The maximum numbers of faces of a convex polytope. Mathematika, vol.17, 1970, pp.179–184.

Megiddo, N., (1986) Improved asymptotic analysis of the average number of steps performed by the self–dual simplex algorithm. Mathematical Programming, vol.35, 1986, pp.140–172.

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

optimization. Mathematical Programming, vol.14, 1978, pp.41-72.

Murtagh, B.A., Saunders, M.A., (1982) A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints, Mathematical Programming Study 16, 1982, pp.84-117.

Murtagh, B.A., Saunders, M.A., (1995) MINOS 5.4, User's Guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, February 1995.

Murty, K.G., (1980) Computational complexity of parametric linear programming. Mathematical Programming, vol.19, 1980, pp.213–219.

Park, S., Kim, W., Park, C.K., Lim, S.M., (1998) Software of simplex method for linear programming. Korean Management Science Review, vol.15, 1998, pp.49-62.

Robinson, S.M., (1972) A quadratically convergent algorithm for general nonlinear programming problems. Mathematical Programming, vol.3, 1972, pp.145-156.

Rosen, J.B., Kreuser, J., (1972) A gradient projection algorithm for nonlinear constraints. in *Numerical Methods for Nonlinear Optimization*, (F.A. Lootsma Ed.), Academic Press, 1972, pp.297-300.

Shamir, R., (1987) The efficiency o f the simplex method: A survey. Management Science, vol.33, 1987, pp.301-334.

Smale, S., (1983) On the average number of steps in the simplex method of linear programming. Mathematical Programming, vol.27, 1983, pp.241–262.

Spielman, D.A., Teng, S.H., (2003) Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. http://www–math.mit.edu/~spielman/simplex, October 2003.

Todd, M.J., (2002) The many facets of linear programming. Mathematical Programming, vol.91, 2002, pp. 417-436.

Vanderebei, R.J., (1996) Linear Programming. Foundations and extensions. Kluwer Academic Publishers, 1996.

Wolfe, P., (1962) The reduced-gradient method. RAND Corporation, 1962.

Wolfe, P., Cutler, L., (1963) Experiments in linear programming. in Graves and Wolfe (Eds.), *Recent Advances in Mathematical Programming*, McGraw Hill, New York, 1963, pp.177-200.