Operational Research. An International Journal manuscript No.
(will be inserted by the editor)

Computational complexity of the Exterior Point Simplex Algorithm

Sophia Voulgaropoulou · Nikolaos Samaras · Angelo Sifaleras

Received: date / Accepted: date

Abstract In this paper, we investigate the computational behavior of the Exterior Point Simplex Algorithm. Up until now, there has been a major difference observed between the theoretical worst case complexity and practical performance of simplex-type algorithms. Computational tests have been carried out on randomly generated sparse linear problems and on a small set of benchmark problems. Specifically, 6,780 linear problems were randomly generated, in order to formulate a respectable amount of experiments. Our study consists of the measurement of the number of iterations that the Exterior Point Simplex Algorithm needs for the solution of the above mentioned problems and benchmark dataset. Our purpose is to formulate representative regression models for these measurements, which would play a significant role for the evaluation of an algorithm's efficiency. For this examination, specific characteristics, such as the number of constraints and variables, the sparsity and bit length, and the condition of matrix A, of each linear problem, were taken into account. What drew our attention was that the formulated model for the randomly generated problems reveal a linear relation among these characteristics.

Keywords Exterior Point Simplex Algorithm · Regression Analysis · Computational Complexity · Linear Relation

Mathematics Subject Classification (2000) 90C27 · 65K05 · 90C05 · 90B10 · 91A90

S. Voulgaropoulou
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
E-mail: svoulgaropoulou@uom.edu.gr

N. Samaras
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
Tel.: +30-2310-891866 / Fax: +30-2310-891879
E-mail: samaras@uom.gr

A. Sifaleras
Department of Applied Informatics
University of Macedonia, School of Information Sciences
156 Egnatias Str., 54636 Thessaloniki, Greece
Tel.: +30-2310-891884 / Fax: +30-2310-891881
E-mail: sifalera@uom.gr

Please cite this paper as:

Voulgaropoulou S., Samaras N., and Sifaleras A., "Computational complexity of the Exterior Point Simplex Algorithm", Operational Research. An International Journal, Springer Berlin / Heidelberg, Vol. 19, No. 2, pp, 297-316, 2019.

The final publication is available at Springer via http://dx.doi.org/10.1007/s12351-017-0291-z

1 Introduction

It is commonly held that a significant number of real world problems can be formulated as Linear Programming problems (LPs). Linear Programming (LP) designates the optimization (by minimizing or maximizing) of a linear function, within a certain domain, which is defined by a set of linear constraints and is perhaps the most important and well-studied optimization problem. Linear programming techniques are used to solve a wide range of real-life problems, such as the optimal operation of transportation networks and the minimization of production costs in manufacturing facilities or agricultural holdings. An interesting study on the latter is available in [28]. One of the most important algorithms in the field of LP is the Simplex Algorithm (SA), which was originally developed by George Dantzig [9,10]. SA has fascinated researchers since its first introduction in 1949, due to its practical performance, which was usually better in real-world problems than in theoretical worst case analysis. SA has been an algorithm with many applications in industry, economy and other fields and it is considered among the most important algorithms in computational theory.

The computational behavior of SA is inextricably linked to the pivoting rules used for its implementation. Different pivoting rules can radically affect the overall efficiency of the algorithm. In 1982, Borgwardt [8] proved that by using specific pivoting rules for the solution of a class of practical problems, SA may have seemingly polynomial complexity in practice, although its complexity is not polynomial in theory. The main disadvantage of the SA and other simplex-type algorithms is that the total number of iterations and thus, the time needed to find a solution cannot be predicted. In more detail, as the dimension n increases, the computational time increases exponentially. Since Klee - Minty [18] had given an example of SA's worst case behavior being exponential when applying specific pivoting rules, there have been a plethora of attempts to find more efficient methods for solving LPs. It still remains an open question whether there exists a strongly polynomial algorithm for LPs, i.e., one whose running time depends only on m and n, where m and n are the number of constraints and decision variables, respectively. The ellipsoid or Russian algorithm, invented by Khachiyan [17] in 1979, enriched the theoretical study of the field but proved to be of poor practical performance in real-world problems. Karmarkar [16] succeeded in creating the first interior point algorithm, which performs surprisingly well when it comes to large problems. In 1991, Paparrizos [22] introduced a new simplex-type algorithm, the exterior point algorithm, in order to solve the assignment problem. Later on, he extended his research by developing a general Exterior Point Simplex Algorithm (EPSA) for linear programming problems [23], which proved to be faster than SA on specifically structured LPs [14,26,27]. A recent survey on exterior point simplex algorithms, can be found in [24]. It is quite interesting that one of the common features among all simplex-type algorithms is that they all follow simplex-type paths, which lead to an optimal solution. What makes EPSA different from the other simplex-type algorithms is that its basic solutions are not feasible. In terms of geometry, EPSA uses two paths in order to find the optimal solution, one infeasible and one feasible. The following table (Table 1) presents the computational complexities of the Simplex ([7,8,18]), Ellipsoid ([17]) and Interior Point algorithms ([16]), respectively. In this case, let n be the number of variables in an LP problem and L, the number of bits necessary to represent the input data.

Table 1: Complexity of linear programming algorithms

	Complexity	
Algorithm	Worst case	Average case
Simplex	$O(2^n)$	$O(n^2)$
Ellipsoid	$O(n^4L)$	$O(n^4L)$
Interior point	$O(n^{3.5}L)$	$O(n^{3.5}L)$

Complexity analysis is a central area of research in theoretical computer science. There are three different approaches to analyzing algorithms; worst case analysis, average case analysis and experimental analysis. In classical complexity analysis, a theoretical study of algorithms is conducted, taking into consideration the problem dimensions. On one hand, worst and average case analyses describe CPU time as a function of problem dimension parameters only. Aspects, such as computer specifications, programming style, programming language and operating system are not taken into consideration. On the other hand, experimental analysis evaluates the real running time, the number of iterations performed and the solution quality on the selected dataset. Obviously, these methods are complementary.

In general, it is difficult to convert an asymptotic complexity into real time prediction. Therefore, it was a challenge for us to develop a regression model that could predict the actual performance of the EPSA. In this paper, we examine the most representative regression models for EPSA, as these were formulated after thorough regression analysis. We used a regression model framework with three parts: *(i)* generation of the Training dataset (TDS), *(ii)* execution of the algorithm and *(iii)* validation of the regression model. Regression analysis is an important statistical method, which can be of great significance while evaluating the computational behavior of an algorithm. Our study is conducted on a dataset of randomly generated sparse LPs and on a sub-set of benchmark LPs. Following this section, in section 2, we describe our implementation of EPSA and in section 3, our dataset is presented. Sections 4 and 5 contain our regression models, along with a detailed description of our findings. Finally, our conclusions are presented in section 6.

2 Algorithm Description

We are concerned with the following linear programming problem (LP.1) in the standard form:

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax = b \\ & x \geq 0 \end{aligned} \tag{LP.1}$$

where $A \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^m$, $c, x \in \mathbb{R}^n$, T denotes transposition and $\text{rank}(A) = m$, $1 \leq m < n$. Partitioning the matrix A as $A = (B,N)$ and with a corresponding partitioning and ordering of $x^T = [x_B \ \ x_N]$ and $c^T = [c_B \ \ c_N]$ (LP.1) is written as:

$$\begin{aligned} \min\ & c_B^T x_B + c_N^T x_N \\ \text{s.t. } & A_B x_B + A_N x_N = b \\ & x_B, x_N \geq 0 \end{aligned}$$

Here A_B is an $m \times m$ non-singular submatrix of A, called basic matrix (or basis), whereas A_N is an $m \times (n\text{-}m)$ submatrix of A, called non-basic matrix. The columns of A which belong to B are called basic and the remaining ones, non-basic. The solution $x_B = B^{-1}b$, $x_N = 0$ is called a basic solution. This solution is feasible iff $x_B \geq 0$. Otherwise, it is called infeasible. The solution of dual problem which corresponds to the basis B, is given by $(s_N)^T = (c_N)^T - w^T A_N$, $(s_B)^T = 0$, where $w^T = (c_B)^T A_B^{-1}$ are the Simplex multipliers and s_N are the dual slack variables. This solution is dual feasible iff $s_N \geq 0$. The ith row of the coefficient matrix A is denoted by $A_{i.}$ and the jth column by $A_{.j}$. The basis inverse $(A_B)^{-1}$ is maintained in some factorized form. At every iteration, its factors have to be updated. There are many techniques for updating the

invertible representation ate the basis matrix. The simplest updating scheme is the Product Form of the Inverse (PFI). The current basis inverse $(A_{\overline{B}})^{-1}$ can be updated from the previous inverse $(A_B)^{-1}$, using the relation $(A_{\overline{B}})^{-1} = (A_B E)^{-1} = E^{-1}(A_B)^{-1}$, where E^{-1} is the inverse of the so-called eta-matrix. The matrix E^{-1} is computed using the following relation

$$E^{-1} = \begin{bmatrix} 1 & & -\frac{h_{1r}}{h_{rr}} & & \\ & \ddots & \vdots & & \\ & & \frac{1}{h_{rr}} & & \\ & & \vdots & \ddots & \\ & & -\frac{h_{mr}}{h_{rr}} & & 1 \end{bmatrix} \tag{1}$$

where h_{rr} is the pivot element and $h_j = (A_B)^{-1} A_{.j}$.

EPSA generates solutions that are not feasible. In every iteration, EPSA generates two paths to the optimal solution. One path is infeasible (exterior), and the other is feasible. Using this movement, EPSA does not need to proceed by visiting one edge after another along the polyhedron $P = \{x \mid Ax \leq b, x \geq 0\}$. A formal description of the EPSA is given below.

2.1 EPSA algorithm

Step 0 (Initialization)

Start with a feasible partition *(B,N)*. Compute $(A_B)^{-1}$ and vectors x_B, w and s_N. Find the sets $P = \{j \in N : s_j < 0\}$ and $Q = \{j \in N : s_j \geq 0\}$. Compute s_0 using the relation

$$s_0 = \sum_{j \in P} s_j$$

Also, compute the vector direction d_B from

$$d_B = -\sum_{j \in P} h_j$$

with $h_j = (A_B)^{-1} A_{.j}$.

Step 1 (Test of termination)

i. (Optimality test): If $P = \varnothing$, STOP. Problem (LP.1) is optimal.

ii. (Choice of leaving variable): If $d_B \geq 0$, STOP. If $s_0 = 0$, problem (LP.1) is optimal. If $s_0 < 0$, problem (LP.1) is unbounded. Otherwise, choose the leaving variable $x_k = x_{B[r]}$ using the minimum ratio test:

$$\alpha = \frac{x_{B[r]}}{-d_{B[r]}} = min \left\{ \frac{x_{B[i]}}{-d_{B[i]}} : d_{B[i]} < 0 \right\}$$

If $\alpha = \min\{\varnothing\} = +\infty$, the problem (LP.1) is unbounded.

Step 2 (Choice of entering variable)

Compute the row vectors $H_{rP} = (A_B)^{-1}_{r.} A_P$ and $H_{rQ} = (A_B)^{-1}_{r.} A_Q$, where $(A_B)^{-1}_{r.}$ denotes the rth row of the basis inverse $(A_B)^{-1}$. Compute the ratios ϑ_1 and ϑ_2, using the relations:

$$\vartheta_1 = \frac{-s_p}{H_{rp}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} > 0 \wedge j \in P\right\}$$

$$\vartheta_2 = \frac{-s_q}{H_{rq}} = min\left\{\frac{-s_j}{H_{rj}} : H_{rj} < 0 \wedge j \in Q\right\}$$

Determine the indexes t_1 and t_2 so that $P(t_1) = p$ and $Q(t_2) = q$. If $\vartheta_1 \leq \vartheta_2$, set $l = p$. Otherwise, set $l = q$. The non-basic variable x_l enters the basis.

Step 3 (Pivoting)

Set $B[r] = l$. If $\vartheta_1 \leq \vartheta_2$, set $P \leftarrow P \setminus \{l\}$ and $Q \leftarrow Q \cup \{k\}$. Otherwise, set $Q(t_2) = k$. Using the new partition *(B,N)*, where N = *(P,Q)*, update the basis inverse $(A_{\overline{B}})^{-1} = E^{-1}(A_B)^{-1}$ and the vectors x_B, w and s_N. Also, update $d_{\overline{B}}$ by $d_{\overline{B}} = E^{-1}d_B$, where E^{-1} is computed by (1) and compute s_0. Go to Step 1.

In order to solve general LPs, we used the Two Phases method. This method for EPSA was initially presented in [31]. The problem of Phase I is constructed by the following procedure. First an artificial variable $x_{n+1} \geq 0$ is added to the problem (LP.2). The coefficients of x_{n+1} are given by the relation $g = -A_B e$, where $e \in \mathbb{R}^m$ is a column vector of ones.

The artificial problem which is solved in Phase I has the form

$$\begin{aligned} \min\ & x_{n+1} \\ \text{s.t. } & Ax + gx_{n+1} = b \\ & x, x_{n+1} \geq 0 \end{aligned} \tag{LP.2}$$

The first iteration in Phase I inserts the artificial variable x_{n+1} into the basis. The leaving variable is selected by

$$x_k = x_{B[r]} = min\{x_{B[i]} : i = 1, 2, ..., m\}$$

Now, the new partition is $B[r] = n+1$ and $N[n+1] = k$. Obviously, the corresponding basic solution is feasible, since $x_{B[r]} = -b_r > 0$, $x_{B[i]} = b_i - b_r \geq 0$, $i \neq r$ and $x_j = 0, j \in N$. In Phase II, the original problem (LP.1) is solved.

In order to solve general LPs, we used EPSA in both Phases. Specifically, EPSA is applied to the problem (LP.2) of Phase I. EPSA exits Phase I if (i) the artificial variable x_{n+1} leaves the basis and at the same time, direction d_B crosses the feasible region, or (ii) direction d_B does not cross the feasible region after x_{n+1} leaves the basis. In this case, EPSA must reach (LP.2) to optimality in order to obtain a feasible solution for the problem (LP.1). Using the following relation, EPSA checks if the current direction d_B crosses the feasible region.

$$\beta = max\left\{\frac{x_{B[i]}}{-d_{B[i]}} : x_{B[i]} < 0\right\} \leq \alpha = min\{\frac{x_{B[i]}}{-d_{B[i]}} : d_{B[i]} < 0\Big\}$$

where $1 \leq i \leq m$.

3 Dataset of Sparse Linear Problems

3.1 Characteristics of LPs

For the purpose of our computational study, 6,780 different sparse linear problems were randomly generated. Nowadays sparse linear problems arise in a plethora of applications, therefore we

consider it very important to examine such LPs. The expected number of feasible vertices of a random linear problem is less or equal to 2^m, where m is the dimension of the constraint set [6]. Moreover, 60 benchmark LPs were examined (Table 2), coming from the NETLIB library [4] and Mészáros LP collection [1].

Table 2: Characteristics of the MPS benchmark files

Name	m	n	density	nnz	L	cond(A)	niter
adlittle	56	97	0.0705	383	7,056	9.3747E+02	158
afiro	27	32	0.0961	83	1,095	9.5044E+16	22
agg	488	163	0.0303	2,410	88,832	4.9751E+16	135
agg2	516	302	0.0275	4,284	168,155	1.0680E+25	224
agg3	516	302	0.0276	4,300	168,171	7.7242E+32	229
aircraft	3,754	7,517	0.0007	20,267	28,330,724	2.0809E+04	3,504
beaconfd	173	262	0.0745	3,375	52,616	1.4610E+04	306
blend	74	83	0.0799	491	7,167	4.9210E+17	89
brandy	220	249	0.0392	2,148	59,001	INF	400
cari	400	1,200	0.3183	152,800	636,613	3.1292E+00	1,658
cr42	905	1,513	0.0048	6,614	1,399,865	6.8812E+02	1,842
degen2	444	534	0.0168	3,978	243,841	8.7744E+16	1,278
e226	223	282	0.0410	2,578	69,812	2.1027E+35	645
farm	7	17	0.3445	41	411	4.9293E+02	2
fffff800	524	854	0.0139	6,227	464,488	5.3703E+21	756
israel	174	142	0.0918	2,269	33,749	3.7416E+16	331
jendrec1	2,109	4,228	0.0100	89,608	9,287,885	1.3477E+03	6,858
lotfi	153	308	0.0229	1,078	50,578	4.1496E+07	373
nemscem	651	1,712	0.0034	3,840	1,125,700	4.4628E+01	514
nsic1	451	463	0.0137	2,853	245,545	1.9515E+22	404
nsic2	465	463	0.0140	3,015	247,175	2.9912E+18	522
nsir1	4,407	5,717	0.0055	138,955	27,224,988	4.5551E+20	4,078
nsir2	4,453	5,717	0.0059	150,599	27,126,852	3.1966E+20	7,886
p0201	133	334	0.0463	2,056	52,294	2.8742E+02	423
p0291	252	543	0.0167	2,283	143,616	1.4445E+02	102
p0040	23	63	0.0918	133	2,567	6.0275E+03	38
p2756	755	3,511	0.0037	9,692	2,718,092	1.4161E+04	1,211
problem	12	46	0.1558	86	1,232	1.0082E+01	14
rosen2	1,032	2,048	0.0220	46,504	2,251,347	9.5635E+01	3,885
rosen7	264	512	0.0575	7,770	159,181	7.3814E+01	691
rosen8	520	1,024	0.0292	15,538	580,394	1.2165E+02	1,570
rosen10	2,056	4,096	0.0074	62,136	8,613,209	2.3561E+02	3,923
sc50a	50	48	0.0542	130	2,746	4.7169E+01	40
sc50b	50	48	0.0492	118	2,694	7.4264E+01	44
sc105	105	103	0.0259	280	11,518	1.1079E+02	101
sc205	205	203	0.0132	551	42,971	6.3909E+02	249
scagr7	129	140	0.0233	420	20,072	1.0272E+04	130
scagr25	471	500	0.0066	1,554	242,644	2.7013E+09	641
scfxm1	330	457	0.0172	2,589	157,088	1.8342E+18	629
scfxm2	660	914	0.0086	5,183	615,824	1.7107E+18	1,412
scfxm3	990	1,371	0.0057	7,777	1,376,179	1.2020E+18	2,234
scrs8	490	1,169	0.0056	3,182	584,955	1.5015E+17	1,165
scsd1	77	760	0.0408	2,388	63,338	2.1212E+01	837
scsd6	147	1,350	0.0217	4,316	207,014	8.8483E+01	1,848
scsd8	397	2,750	0.0079	8,584	1,108,874	9.9320E+02	4,416
sctap1	300	480	0.0118	1,692	150,945	7.3158E+16	675
sctap2	1,090	1,880	0.0033	6,714	2,076,689	2.8883E+17	2,146
sctap3	1,480	2,480	0.0024	8,874	3,706,126	6.3612E+17	2,327
share1b	117	225	0.0437	1,151	32,368	1.3814E+05	356
share2b	96	79	0.0915	694	10,378	1.5057E+18	258
ship04l	402	2,118	0.0074	6,332	879,180	INF	782
ship04s	402	1,458	0.0074	4,352	605,233	INF	434
ship08l	778	4,283	0.0038	12,802	3,389,137	INF	1,643
ship08s	778	2,387	0.0038	7,114	1,889,137	INF	678

Table 2: Characteristics of the MPS benchmark files

Name	**m**	**n**	**density**	**nnz**	**L**	**cond(A)**	**niter**
ship12l	1,151	5,247	0.0026	16,170	6,309,794	INF	978
ship12s	1,151	2,763	0.0026	8,178	3,212,891	INF	509
stocfor1	117	111	0.0344	447	14,627	6.3562E+05	85
stocfor2	2,157	2,031	0.0019	8,343	4,411,089	1.0662E+06	1,072
sws	14,310	12,465	0.0005	93,015	178,605,200	3.3301E+18	2,066
zed	116	43	0.1137	567	7,494	5.6907E+05	135

The randomly generated sparse linear problems are subject to inequality constraints and they are optimal, meaning that the algorithm reaches an optimal solution after a specific number of iterations. These LPs have been created using a generator, that was specifically designed to generate random optimal LP instances [25]. The planes of the constraints are tangent on a sphere, so that its center is feasible. Also, these problems have a closed feasible region that is a closed polyhedron. We used an LP generator that takes as input m (number of constraints), n (number of decision variables), density (density of the non zero entries of A, $0< density \leq 1$) and seed (the seed number for the random number generator).

The ranges of the values that were used in matrices A, b, and c, in order to create the linear problems of our study are shown in Table 3 below.

Table 3: Value ranges of LPs

A	[10 400]
b	[10 100]
c	[-300 900]

In more detail, the lower and upper bounds of values in matrix A, b and c are 10, 10, -300 and 400, 100, 900, respectively. The entries of A, b and c were randomly generated with Matlab *sprand* function.

As explained above, the generator takes the value of density as input. The density was formed by using Matlab *rand* function with an upper limit of 30%, due to the fact that we are especially interested in examining the algorithm's behavior when dealing with sparse LPs. This is the reason why sparsity is one of the LP characteristics examined, as shown in the list below.

- m: number of linear constraints
- n: number of variables
- *sparsity*: problem's sparsity
- *nnz*: number of non-zero elements
- L: data length (bit length)
- *cond(A)*: condition of matrix A

Not only the first two characteristics (which are well-known LP attributes), but also the rest have a significant contribution to the algorithm's performance and overall efficiency. The respective values of the characteristics above were randomly formed, except for the benchmark LPs. Some useful statistical data about the above mentioned characteristics are presented in Table (4).

Table 4: Characteristics of linear problems

	m	n	*sparsity* (in %)	*nnz*	L	*cond(A)*
min. value	1,122	1,088	71.7%	127,970	3,478,842	24.9788
max. value	12,470	12,301	97.47%	27,051,872	311,324,768	573,956,480

The quantities of m and n were formed by using Matlab *rand* function, within the bounds shown in Table (4) above. The number of non-zero elements indicates the elements within matrix

A which are not zero. Data length L is the number of bits which are required in order to represent integer data of A, b and c. Regarding *cond(A)*, its number is close to 1 when the data within matrix A is of good condition. The condition number can be used to predict how ill-conditioning affects the computed solution of a LP.

We should keep in mind that the above mentioned values are uniformly distributed, according to the *sprand* and *rand* functions of Matlab.

3.2 Computing environment

Our algorithm was implemented in MATLAB programming language, using the MathWorks MATLAB programming environment (version R2014a) [2]. MATLAB was our preferred option, due to its inherent capability for matrix and vector operations, regarding sparse matrices and large linear problems. Our code is designed to take advantage of Matlab's sparse matrix functions. Apart from the programming language though, one should always take into account the hardware and software characteristics of a specific computing environment, since they can prove to have a great impact on the performance of an algorithm [20]. Therefore, the computing environment in which our 6,780 LPs were generated and solved is fully described in the following table (Table 5).

Table 5: Characteristics of the computing environment

CPU	Intel® Xeon™, 3.00 GHz (2 processors)
RAM size	12,288 Mb
L2 Cache size	2×1,024 Kb
L1 Cache size	2×16 Kb
Operating System	Windows 7 Professional, SP1, 64-bit
MATLAB version	8.3.0.532 R2014a

4 Regression Models

This section includes a full presentation and explanation of the process we followed in order to create the regression models for our algorithm. It also describes each regression model separately and provides numerical and graphical representation of the results. The most important thing to notice is the linear relation of the dependent with the independent variables in the model for the randomly generated sparse LPs. Nevertheless, before we continue with analyzing our results, we will first provide some fundamental pieces of information regarding Regression Analysis and its methods and characteristics. We will also present the metrics we used in order to evaluate the best regression model for our study and then proceed with the details on our models and the model validation part.

This study focuses on the required number of iterations of the Exterior Point Simplex Algorithm and not the CPU time. The problem with using regression analysis to study CPU time is that the memory hierarchy (caches, virtual memory system), creates a piece-wise cost function for time, with abrupt changes when the algorithm outgrows a level in the hierarchy. The location of the breaks, and the magnitude of the changes, are highly platform dependent. Therefore, measurements on one platform are not useful for predicting costs on another platform [21].

4.1 Regression Analysis

Regression Analysis (RA) is a statistical method for the prediction of the value of a "dependent" variable using one or more "independent" variables. Simple Regression examines the relation between one dependent and one independent variable, whereas Multiple Regression examines the relation between one dependent and a set of independent variables. RA allows us to study and evaluate the effect of the independent variables on the dependent ones. One of the main goals of RA is the formulation of specific models which not only determine the relation between the independent and dependent variables, but can also be used in order to predict the value of the dependent variables when the values of the independent are given. During our study, regression proved to be the most useful and effective method in order to generate specific "regression models" for our LPs. Before we move on with the full description of our results, it would be interesting to provide with some further details on the nature of RA and the importance of the several metrics used for the evaluation of our models.

To begin with, the general model of a linear regression is of the following form:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_r X_r + \varepsilon \tag{2}$$

As we can see, this mathematical equation determines that Y is linearly related to the independent variables $X_1 \ldots X_r$, while β_0, β_1 $\ldots$, β_r are the so-called regression coefficients. If this equation was to be geometrically represented on a 2-dimension Cartesian coordinate system, then this would be a straight line with intercept of β_0 and slope of β_1. It is worth mentioning that ε is the residual of the regression equation and it is very important for the determination of a statistical model as a good-fitting one or not. Residuals are the distance between the observed value and our estimation. Small values of residuals indicate a good fit of a statistical model, while big values of residuals indicate a bad or non-fit to the examined data. Especially in statistical practice, researchers prefer to use and analyze graphic plots which are based on residuals, and more specifically, the standardized ones. The value of a standardized residual comes from the division of the residual's value by its standard deviation.

A more thorough presentation of all aspects of RA theory exceeds the purpose of our study, therefore, we will focus on the actual process we followed, as well as the statistical environment we used, in order to create and evaluate our models. We will also describe the metrics which were studied and taken into consideration during the evaluation of several models until we reach the final ones. After that, we will thoroughly present the regression model created for the proposed dataset. Until the resulting model was formed, several other models had been tested and evaluated (approx. 90-100 in total). The models tested were not only linear, but also exponential and logarithmic. They also included squares, pairwise products, and quantities such as $n\log n$. The model selected was found to be the best fit for the sampled dataset of randomly generated LPs. However, the different characteristics among the several models can be subject to further comparative analysis in the future.

4.1.1 Statistical Environments

For the purpose of our study, one of the most widely known statistical environments was used, namely Minitab. Minitab is a general statistics software [3], originally developed by Barbara F. Ryan, Thomas A. Ryan, Jr. and Brian L. Joiner in 1972, as a light version of OMNITAB, a statistical analysis program, in the National Institute of Standards and Technology (NIST) [5].

Our models were generated and tested in Minitab, using the "Best Subsets" model selection method [15]. This is an automated process for identifying the best-fitting regression model for a given dataset, taking all available independent variables into account. The general scope is to

select a subset of independent variables that best satisfies specific statistical metrics, which are explained further in the next subsection.

4.1.2 Explanation of Metrics

In this section, we provide a brief explanation about the metrics we examined in order to select the best-fitting models for our study. All of them are well-known metrics, commonly used in the statistical research field for evaluation and model performance testing.

- R-squared (*R-Sq*): Coefficient of determination. It defines how well a statistical model fits the examined data, therefore the bigger its value is, the better fitting the model has. Although this coefficient is interesting in any regression model, it is characterized by two main disadvantages. First of all, the R-squared value increases each time we add a new parameter in the examined model. This is one of the reasons why the R-squared alone cannot guarantee the good fit of a model. Secondly, in case of a large number of parameters and higher order polynomials in the examined model, this particular metric is affected by the random noise of the data. This problem is known as over-fitting and it produces misleadingly high values for *R-Sq*, making the model less suitable to be used for predictions [29].
- Adjusted R-squared (*R-Sq(adj)*): Adjusted coefficient of determination. It is extremely useful during the comparison of models with different number of predictors, due to the fact that it is adjusted according to the number of predictors in a model. This means that it increases only if a new predictor improves the model more than it was expected by chance and decreases when a predictor improves the model less than expected by chance. It is worth mentioning that, its value is always lower than the R-squared value [29].
- Predicted R-squared (*R-Sq(pred)*): The predicted R-squared indicates how well a regression model predicts responses for new observations. This metric can help us identify if a model is not capable of providing valid predictions for new observations, although it seems to fit the original data. Similarly to adjusted R-squared, predicted R-squared is always lower than R-squared and could be also negative. One of the most important benefits of predicted R-squared is that it can prevent us from over-fitting a model. This is based on the fact that, it is impossible to predict random noise. Therefore, in case of an over-fitted model, the value of predicted R-squared would drop. More specifically, if the predicted R-squared of a model is much lower than the regular R-squared this means that, the model is most probably over-fitted and cannot be used for predictions [19].
- Standard error of regression (S): S measures the units of the response variable and represents the standard distance between data values and the estimated regression line. The lower the value of S, the better the model predicts the response. When comparing different models, the model with the lowest S value reflects the best fit [15].

Apart from these metrics, which are shown for every model in the following analysis, there are additional parameters that should be explained before we present the details of each regression model.

- Standard error of the coefficient (*SE Coef*): The standard error of the coefficient is the standard deviation of the estimate of a coefficient in a regression model. It measures how precisely the model estimates the coefficient's unknown value and it is always positive. The smaller the standard error, the more precise the estimate. Now, if we divide the coefficient by the respective standard error, we will calculate a specific t-value (T). This value is also known as t-statistic and measures the likelihood that the actual value of the parameter is not zero. The larger the absolute value of T, the less possible that the real value of the parameter

is zero. If the probability value (P) related to the t-statistic is less than the determined level of significance we conclude that, the coefficient is significantly different from zero [11].
- Degrees of freedom (DF): The number of values in the final calculation of a statistic that are free to vary. The notion of this metric was initially introduced by Gosset [30], while the specific naming degrees of freedom was used by Fisher some years later [13].
- Sum of Squares (SS): This metric indicates the deviation from the mean and is calculated as the sum of the squares of the differences from the mean [19].
- Mean Squares (MS): The value of mean squares is calculated by dividing the respective sum of squares by the degrees of freedom. This metric is an estimate of the population variance. In a regression model, the mean squares are used to determine whether the parameters of the model are significant [19].

Moreover, we should also pay attention to the F and the corresponding P values of each model. The importance of these metrics derives from the fact that, although R-squared provides an estimate of the strength of the relationship between a regression model and the dependent variable, it does not provide any formal hypothesis test for this relationship. The F-test is the metric which determines if this relationship is statistically significant or not. Therefore, if the P value for the overall F-test is less than the applied significance level, then the specific regression model has statistically significant predictive capability [11]. The significance level in the current study has been set to 5%.

In the following section, we will present the regression model for randomly generated LPs. The respective probability plot will be presented after the model equation and statistical details. This P-P plot is a normal probability-probability (P-P) plot based on the standardized residuals. In this study, the X axis represents the Observed Cumulative Probability (Observed Cum Prob), which is based on the percentiles in the frequency distribution of the residuals. The Y axis, which represents the Expected Cumulative Probability (Expected Cum Prob), is based on the Standardized Residual (Z-score) and on the computation of the cumulative density from the normal distribution. If the residuals are normally distributed, then the values should fall exactly on the diagonal line. In our study, the systematic deviation from the diagonal line, as shown in the (P-P) plot of the following section, indicate a positive skewness of the distribution. This means that the right side tail of the curve, if this was depicted in a histogram, is longer than the left side tail and the mean is greater than the mode. Skewness is actually the asymmetry of a distribution and can be quantified to measure the extent to which this distribution is distorted and how much it differs from a normal distribution. This matter can be subject to further analysis in the future. As mentioned earlier in this paper, an extensive study of several possible regression models was conducted, so that we can reach a conclusion about the best suitable model for the sampled dataset. It would be useful for the reader to get an idea of the differences spotted among the resulted models. For this reason, the best exponential and logarithmic models are included in this study as well.

4.2 Regression model for randomly generated LPs

In this subsection, the regression equation (Eq. 3) shows that the relation between the number of iterations and the number of constraints, m and variables, n is positive, indicating that an increase of the problem dimensions is followed by a corresponding increase in the number of iterations.

The positive relation of the number of iterations with the problem sparsity reflects once again possible difficulties that the algorithm may face during the computation of specific mathematical equations in very sparse problems. Moreover, the condition of matrix A *cond(A)* is positively

related to the number of iterations. This means that if the condition of our data (namely of matrix A) gets worse (i.e., the corresponding value increases), then the corresponding number of iterations will increase as well. The regression equation of the corresponding regression model is the following (Eq. 3).

$$niter = -5,935.063 + 0.148m + 0.699n + 6,453.502sparsity + 1.150\text{e}{-6}cond(A) \quad (3)$$

As shown in the following table (Table 6) a full description of the model's statistics is provided. All independent variables are statistically significant ($P < 0.05$). The coefficient of determination (not only *R-Sq*, but also *R-Sq(adj)*, is approximately equal to 96% and the F-test is of significance $P < 0.001$, which means (as previously stated) that the model is suitable for the overall description and explanation of the variability of the whole dataset. Finally, the Durbin-Watson statistic [12] is approximately equal to 1.94, which is really close to 2 and indicates that there is no auto-correlation within the sample.

Table 6: Statistical values of Regression model

Predictor	Coef	SE Coef	T	P	
Constant	-5,935.063	56.482	-105.080	0.000	
m	0.148	0.003	48.887	0.000	
n	0.699	0.003	229.803	0.000	
sparsity	6,453.502	67.330	95.849	0.000	
cond(A)	1.150e−6	0.000	2.021	0.043	
R-Sq = 96.0%	R-Sq(adj) = 95.9%	S = 383.634			
R-Sq(pred) = 95.93%					
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	4	23,640,513,358.335	5,910,128,339.584	40,157.160	0.000
Residual Error	6,778	997,551,872.102	147,174.959		
Total	6,780	24,638,065,230.437			
Durbin-Watson	1.936				

A graphical representation of the standardized residuals, regarding the number of iterations, is provided through the corresponding normal probability plot below (Fig. 1).

Following, we will provide specific details of an exponential (Eq. 4) and a logarithmic (Eq. 5) model, which were found to be the best out of the rest models examined for the total dataset (Tables 7 and 8).

$$niter = -6,830 + 0.804n + 2,805exp(sparsity) \quad (4)$$

$$niter = -24,624 + 1,282log(m) + 6,654log(n) + 12,380log(sparsity) + 91.4log(condA) \quad (5)$$

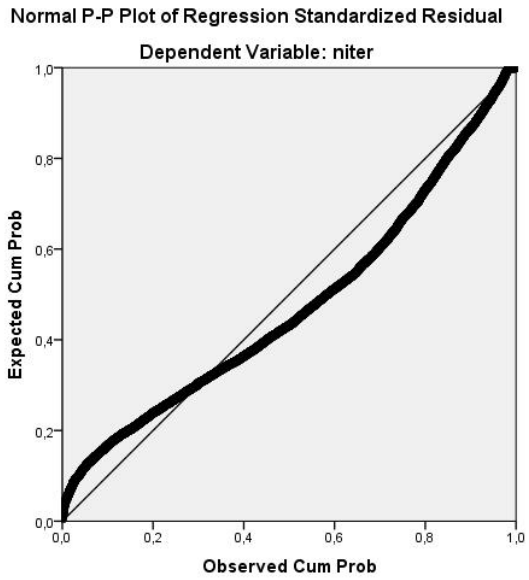

Fig. 1: Normal Probability plot of iterations standardized residuals

Table 7: Statistical values of exponential Regression model

Predictor	Coef	SE Coef	T	P	
Constant	-6,830.37	77.34	-88.32	0.000	
n	0.803768	0.002466	325.89	0.000	
exp(sparsity)	2,804.57	33.54	83.62	0.000	
R-Sq = 94.6%	R-Sq(adj) = 94.6%	S = 441.373			
R-Sq(pred) = 94.63%					
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	2	23,317,254,689	11,658,627,345	59846.20	0.000
Residual Error	6,780	1,320,810,541	194,810		
Total	6,782	24,638,065,230			
Durbin-Watson	1.542				

Table 8: Statistical values of logarithmic Regression model

Predictor	Coef	SE Coef	T	P	
Constant	-24,623.8	150.3	-163.81	0.000	
log(m)	1,281.69	55.03	23.29	0.000	
log(n)	6,653.55	55.14	120.66	0.000	
log(sparsity)	12,380.3	218.9	56.57	0.000	
log(cond(A))	91.402	5.216	17.52	0.000	
R-Sq = 88.4%	R-Sq(adj) = 88.4%	S = 648.856			
R-Sq(pred) = 88.40%					
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	4	21,784,430,694	5,446,107,674	12,935.69	0.000
Residual Error	6,778	2,853,634,536	421,014		
Total	6,782	24,638,065,230			
Durbin-Watson	1.839				

4.3 Benchmark LPs

This subsection includes the analysis in the 60 benchmark LPs from various collections. The regression model created for these LPs is shown in Equation 6. As expected, the performance of our algorithm drops, since the problems examined are degenerate. It was quite impressive that during the analysis seven LPs were automatically excluded from the analysis, since their condition number reached infinity. These problems could not, eventually, participate in the computation of the regression model. The corresponding results are shown below (Table 9) and the respective regression model is presented in Equation 6.

$$niter = 539 + 0.0296 sparsity \times nnz + 0.177 n \times sparsity \tag{6}$$

Table 9: Statistical values of Regression model - Benchmark

Predictor	Coef	SE Coef	T	P	
Constant	539.0	171.9	3.14	0.003	
sparsity × nnz	0.029562	0.005815	5.08	0.000	
n × sparsity	0.17708	0.08942	1.98	0.053	
R-Sq = 61.9%	R-Sq(adj) = 60.3%	S = 1064.60			
R-Sq(pred) = 23.71%					
Analysis of Variance					
Source	DF	SS	MS	F	P
Regression	2	91,940,699	45,970,349	40.56	0.000
Residual Error	50	56,669,020	1,133,380		
Total	52	148,609,719			
Durbin-Watson	1.892				

As shown, *R-Sq(adj)* reaches 60.3%, however *R-Sq(pred)* shows that the particular model may not be so suitable for prediction purposes. This is reasonable, if we take into consideration

the structure and unique characteristics of benchmark problems, which are degenerate and ill-conditioned. As a result, it may be difficult for simplex-type algorithms to handle such problems and especially to have results that would form a satisfying regression model, which could be used for predictive purposes, as well.

5 Validation of Regression Models

In the previous section, the regression model which corresponds to the dataset of the randomly generated LPs was thoroughly presented. This section aims to provide a further validation of this model in order to ensure its accuracy and prediction ability. For this purpose, additional LPs were generated, following the same process that was applied during the creation of the initial dataset. In terms of statistical analysis, our initial LPs are the training dataset for our models, while the additional LPs, which are mentioned here, are the validation dataset, which confirms the regression model we created. The value ranges regarding the number of constraints m, the number of variables n, the problem density, as well as the type of constraints are the same to the ones applied in the initial dataset. During the creation of our validation dataset, we examined the same characteristics as the ones studied in the initial dataset. Nevertheless, the reader should not be confused by the fact that the same characteristics have been maintained. The additional LPs that were created during the validation step of our process are totally independent from the initial dataset.

The number of the additional LPs that were generated, covers approximately 10% of the total dataset (TDS). The number of LPs during validation is presented in the following table (Table 10).

Table 10: Validation Dataset

	Number of LPs	Percentage against TDS
Total	647	9.54

The validation process included the following steps:

1. Regression Models: The regression equations of the resulted models are given
2. Value Replacement: The independent variables (m, n, etc.) within our models are replaced by the corresponding observed values from the validation dataset
3. Calculation: The estimated values of iterations are calculated, based on our regression models
4. Comparison: The estimated values are compared to the observed ones
5. Deviation computation: The deviation between the estimated and the observed value is calculated

For the purpose of this computation, we used the following ratio (Eq. 7):

$$DeviationRatio = (EstimatedValue - ObservedValue)/ObservedValue \tag{7}$$

This process resulted in having a complete view of the efficiency of our models, both for representation and prediction of the number of iterations. Table 11 below, shows the maximum, minimum and average deviation for each problem category. These results are based on the absolute value of the calculation explained above (Eq. 7).

Table 11: Deviation from Observed values

	Number of Iterations		
	max	min	average
Total	112.80%	0.02%	9.81%

The regression model created for the number of iterations seems to be quite accurate and the actual average differences between the estimated and observed values are remarkably small. Apart from the abovementioned analysis, though, we also examined how these instances are distrubuted and how they affect the average practical performance of our algorithm. EPSA performance seems to falls behind in extreme instances like the ones shown in the box plot on the right side of Fig. 2 below. This is also depicted on the respective histogram, which indicates right skewed instances. This right or positive skewness is characterized by a "tail" of the instances to the right. This means, that there are many instances in the validation dataset, for which the number of iterations is relatively small, while in few instances the number of iterations increase significantly. The existence of a specific pattern of characteristics in these extreme instances, which may eventually affect the practical performance of EPSA, could be subject to further analysis.

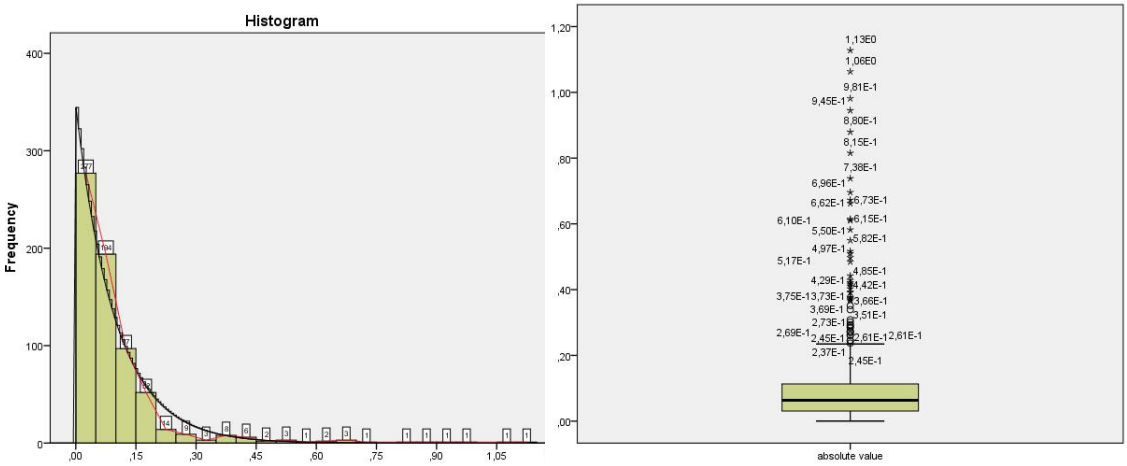

Fig. 2: Deviation Histogram & Outliers - Validation

6 Conclusions

With the above mentioned results considered, we would like to conclude this paper, by pointing out the linear relation between the independent and dependent variables in the regression model of the randomly generated LPs.

Apart from the above, it is clear that there are many aspects, which can be improved and offer a great chance for future study. Applying the above methodology to non-feasible algorithms, as well, could provide some useful results about their practical performance in real-world applications. Furthermore, it was noticed that our algorithm tends to have a slightly worse than its usually good performance, when it comes to extreme conditions, as explained in the previous section. Therefore, it remains to be examined how can we improve this aspect of EPSA, in order to take advantage of its capabilities and achieve the best possible outcome for any linear problem given.

References

1. Linear programming test problems. http://www.sztaki.hu/~meszaros/public_ftp/lptestset/misc. (Last checked on 2016-10-05)
2. Mathworks MATLAB. http://www.mathworks.com/products/matlab. (Last checked on 2016-10-05)
3. Minitab. http://www.minitab.com/en-us/academic. (Last checked on 2016-10-05)
4. NETLIB, netlib benchmark lps. http://www.netlib.org/benchmark. (Last checked on 2016-10-05)
5. NIST, national institute of standards and technology. http://www.nist.gov. (Last checked on 2016-10-05)
6. Berenguer, E., Smith, L.: The expected number of extreme points of a random linear program. Mathematical Programming **35**, 129–134 (1986)
7. Borgwardt, K.: The average number of pivot steps required by the simplex method is polynomial. Zeitschrift für Operations Research **26**(1), 157–177 (1982)
8. Borgwardt, K.: Some distribution independent results about the asymptotic order of the average number of pivot steps in the simplex method. Mathematics of Operations Research **7**(3), 441–462 (1982)
9. Dantzig, G.: Programming of interdependent activities: II, mathematical model. Econometrica **3–4**, 200–211 (1949)
10. Dantzig, G.: Linear Programming and Extensions. NJ: Princeton, Princeton University Press (1963)
11. Draper, N., Smith, H.: Applied Regression Analysis, 3rd edn. John Wiley and Sons (1998)
12. Durbin, J., Watson, G.: Testing for serial correlation in least squares regression I. Biometrika **37**(3,4), 409–428 (1950)
13. Fisher, R.: On the interpretation of χ^2 from contingency tables and the calculation of P. Journal of the Royal Statistical Society **85**(1), 87–94 (1922)
14. Glavelis, T., Samaras, N.: An experimental investigation of a primal-dual exterior point simplex algorithm. Optimization: A Journal of Mathematical Programming and Operations Research **62**(8), 1143–1152 (2013)
15. Hosmer, D., Jovanovic, B., Lemeshow, S.: Best subsets logistic regression. Biometrics **45**(4), 1265–1270 (1989)
16. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica **4**(4), 373–395 (1984)
17. Khachiyan, L.: A polynomial algorithm in linear programming. Soviet Mathematics Doklady **20**, 191–194 (1979)
18. Klee, V., Minty, G.: How good is the simplex algorithm? Shisha O (Eds), Inequalities III (1972)
19. Kutner, M., Neter, J., Nachtsheim, C., Wasserman, W.: Applied Linear Statistical Models, 5th edn. McGraw-Hill (2004)
20. Maros, I., Khaliq, M.: Advances in design and implementation of optimization software. Technical report (1999)
21. McGeoch, C., Sanders, P., Fleischer, R., Cohen, P.R., Precup, D.: Using finite experiments to study asymptotic performance. In: Experimental algorithmics, pp. 93–126. Springer (2002)
22. Paparrizos, K.: An infeasible exterior point simple algorithm for assignment problems. Mathematical Programming **51**, 45–54 (1991)
23. Paparrizos, K.: An exterior point simplex algorithm for (general) linear programming problems. Annals of Operations Research **47**, 497–508 (1993)
24. Paparrizos, K., Samaras, N., Sifaleras, A.: Exterior point simplex-type algorithms for linear and network optimization problems. Annals of Operations Research **229**(1), 607–633 (2015)
25. Paparrizos, K., Samaras, N., Stephanides, G.: A method for generating random optimal linear problems and a comparative computational study. In: Proceedings of the 13th National Conference of the Hellenic Operational Research Society, pp. 785–794. Piraeus, Greece (2000). (in Greek)
26. Paparrizos, K., Samaras, N., Stephanides, N.: An efficient simplex type algorithm for sparse and dense linear problems. European Journal of Operational Research **148**(2), 323–334 (2003)
27. Paparrizos, K., Samaras, N., Stephanides, N.: A new efficient primal dual simplex algorithm. Computers and Operations Research **30**(9), 1383–1339 (2003)
28. Prišenk, J., Turk, J., Čr. Rozman, Borec, A., Zrakić, M., Pažek, K.: Advantages of combining linear programming and weighted goal programming for agriculture application. Operational Research **14**(2), 253–260 (2014)
29. Rao, C.: Coefficient of Determination, Linear Statistical Inference and its Applications, 2nd edn. New York: Wiley (1973)
30. Student: The probable error of a mean. Biometrika **6**(1), 1–25 (1908)
31. Triantafyllidis, C., Samaras, N.: Three nearly scaling invariant versions of an exterior point algorithm for linear programming. Optimization: A Journal of Mathematical Programming and Operations Research **64**(10), 2136–2181 (2015)