A Web-Based Decision Support System using Basis Update on Simplex Type Algorithms

Nikolaos Ploskas[1], Nikolaos Samaras[2], and Jason Papathanasiou[3]

[1] University of Macedonia, 156 Egnatia Str., Thessaloniki 54006, Greece, ploskas@uom.gr,
[2] University of Macedonia, 156 Egnatia Str., Thessaloniki 54006, Greece, samaras@uom.gr
[3] University of Macedonia, 156 Egnatia Str., Thessaloniki 54006, Greece, jasonp@uom.gr

Abstract. Linear Programming is a significant and well-studied optimization methodology. Simplex type algorithms have been widely used in Decision Support Systems. The computation of the basis inverse is a crucial step in simplex type algorithms. In this paper, we review and compare three basis update methods. We incorporate these methods on the exterior and the revised simplex algorithm in order to highlight the significance of the choice of the basis update method in simplex type algorithms and the reduction that can offer to the solution time. We perform a computational comparison in which the basis inverse is computed with three updating methods. Finally, we have implemented a web-based Decision Support System that assists decision makers in the selection of the algorithm and basis update method in order to solve their Linear Programming problems.

Key words: Linear Programming, Decision Support System, Exterior Point Simplex Algorithm, Revised Simplex Algorithm, Basis Inverse

1 Introduction

Web-based Decision Support Systems (DSS) are computerized information systems that supports decision-policy makers using a Web browser [20]. The three most widely used frameworks for implementing web-based DSS are communication-driven, knowledge-driven and document-driven DSS [3]. Communication-driven DSS bring together multiple decision makers using electronic communication technologies. Communication-driven DSS may also assist managers in collaborative decision-making processes (for more information, see [10]). Knowledge-driven DSS recomment actions to decision makers. Finally, document-driven DSS support managers organize and analyze large volumes of data.
Web-based DSS can be implemented using three different Web technologies [2]: (i) server-side technologies, (ii) client-side technologies, and (iii) distributed implementations. Server-side technologies include Java applications, ASP, PHP, CGI, and JSP, while client-side tecnologies include Java applets and ActiveX controls. Finally, distributed implementations are based on Java RMI, COM+,

and Enterprise Java Beans. Recently, DSS also utilize web services and messaging protocols like SOAP [3].
This paper presents a web-based knowledge-driven DSS that supports decision makers solving their Linear Programming problems. Operations research applications that the proposed DSS can be utilized include telecommunications, bio-informatics, revenue management, supply chain management, resource allocation, etc. A real application case study for the transition from fossil to renewable energy resources in the United States [11] adapted from Manne [15] is presented in Section 5, in which our application can be utilized in the decision making process. The user interface of the proposed DSS has been implemented using jsp, while the simplex type algorithms have been implemented with MATLAB.
The structure of the paper is as follows. Section 2 presents the background of our work. Section 3 includes the presentation of the three basis update methods, while in Section 4 the computational experiments are presented. In Section 5, a web-based DSS for the selection of the algorithm and basis update method is presented. Finally, the conclusions of this paper are outlined in Section 6.

2 Background

Linear Programming (LP) is the process of minimizing or maximizing a linear objective function $z = \sum_{j=1}^{n} c_j x_j$ to a number of linear equality and inequality constraints. The simplex algorithm is the most widely used method for solving Linear Programming problems (LPs). Consider the following LP (LP.1) in the standard form:

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

where $A \in \mathbb{R}^{m \times n}$, $(c, x) \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and T denotes transposition. We assume that A has full rank, $rank(A) = m, m < n$. Consequently, the linear system $Ax = b$ is consistent. The simplex algorithm searches for an optimal solution by moving from one feasible solution to another, along the edges of the feasible set. The dual problem associated with the (LP.1) is presented in (DP.1):

$$\begin{aligned} min \quad & b^T w \\ s.t. \quad & A^T w + s = c \\ & s \geq 0 \end{aligned} \tag{DP.1}$$

where $w \in \mathbb{R}^m$ and $s \in \mathbb{R}^n$.
Using a partition (B, N) (LP.1) can be written as follows:

$$\begin{aligned} min \quad & c_B^T x_B + c_N^T x_N \\ s.t. \quad & A_B x_B + A_N x_N = b \\ & x_B, x_N \geq 0 \end{aligned} \qquad (LP.2)$$

In the above problem, A_B is an $m \times m$ non-singular submatrix of A, called basic matrix or basis. The columns of A belonging to subset B are called basic and those belonging to N are called non basic. The solution of the linear problem $x_B = (A_B)^{-1} b, x_N = 0$ is called a basic solution. A solution $x = (x_B, x_N)$ is feasible iff $x \geq 0$; otherwise, the solution is infeasible. The solution of the (DP.1) is computed by the relation $s = c - A^T w$, where $w = (c_B)^T (A_B)^{-1}$ are the simplex multipliers and s are the dual slack variables. The basis A_B is dual feasible iff $s \geq 0$.

In each iteration, the simplex algorithm interchanges a column of matrix A_B with a column of matrix A_N and constructs a new basis $A_{\overline{B}}$. Any iteration of simplex type algorithms is relatively expensive. The total execution time of an iteration of simplex type algorithms is dictated by the computation of the basis inverse. This inverse, however, does not have to be computed from scratch during each iteration of the simplex algorithm. Simplex type algorithms maintain a factorization of basis and update this factorization in each iteration. This step must be carefully designed and implemented, because it is the most time-consuming step in simplex-type algorithms. Many updating methods have been proposed [1][4][6][7][16][21][22]. In Section 2 we present three well-known methods for the basis inverse.

Simplex type algorithms have been widely used in Decision Support Systems. Ghodsypour and O'Brien [9] have implemented a DSS for supplier selection using the revised simplex method. Lappi et al. [12] designed an information system for forest management planning systems which generate alternative treatment schedules for treatment units and select optimal schedule combinations using the simplex method. Venkataramanan and Bornstein [23] have proposed a network-based DSS for the assigning parking spaces utilizing primal network simplex algorithm. Lauer et al. [13] have implemented a DSS to schedule student computer lab attendants at a major university using the revised simplex algorithm. Lourenço et al. [14] have proposed a DSS for portfolio robustness evaluation using a solver based on the revised simplex algorithm.

In a previous paper [19], we reviewed and compared both the CPU- and GPU-based implementations of five updating methods, namely: (i) Gaussian Elimination, (ii) the built-in function inv of MATLAB, (iii) LU decomposition, (iv) Product Form of the Inverse (PFI) and (v) a Modification of the Product Form of the Inverse (MPFI); and incorporated them with the revised simplex algorithm. We have performed a computational study over dense randomly optimal generated LPs and concluded that MPFI, PFI and MATLAB's inv are the best serial basis update methods. In this paper, a computational study is performed in order to highlight the impact of the choice of the basis update methods on the exterior and the revised simplex algorithm. The Exterior Primal Simplex

Algorithm (EPSA) that we used has been proposed by Paparrizos et al. [18]. This algorithm outperforms the Revised Simplex Method (RSM) as the problem size increases and the density decreases. The RSM in the computational study has been proposed by Dantzig [5].

3 Basis Update Methods

In this section, three widely-used updating methods are presented: (i) the MATLAB's built-in method inv, (ii) the Product Form of the Inverse, and (iii) the Modification of the Product Form of the Inverse.

3.1 MATLAB's Built-in Function inv

MATLAB's built-in function inv can be used to compute the basis inverse in every step of the algorithms. MATLAB's inv uses LAPACK routines to compute the basis inverse and is already optimized. Hence, its execution time is smaller compared with other relevant methods that compute the explicit basis inverse. The time complexity of this method is $O(n^3)$.

3.2 Product Form of the Inverse

The new basis inverse can be updated, according to the PFI updating method [6], at any iteration using the equation (1):

$$(A_{\overline{B}})^{-1} = (A_B E)^{-1} = E^{-1}(A_B)^{-1} \tag{1}$$

where E^{-1} is the inverse of the eta-matrix and can be computed by (2).

$$E^{-1} = I - \frac{1}{h_{rl}}(h_l - e_l)\, e_l^T = \begin{bmatrix} 1 & & -h_{1l} & & \\ & \ddots & \vdots & & \\ & & 1/h_{rl} & & \\ & & \vdots & \ddots & \\ & & -h_{ml}/h_{rl} & & 1 \end{bmatrix} \tag{2}$$

If the current basis inverse is computed using regular multiplication, then the complexity of the PFI is $\Theta\left(m^3\right)$.

3.3 A Modification of Product Form of the Inverse

In MPFI updating scheme [4], the current basis inverse $(A_{\overline{B}})^{-1}$ can be computed from the previous inverse $(A_B)^{-1}$ using a simple outer product of two vectors and one matrix addition, as shown in the equation (3):

$$(A_{\overline{B}})^{-1} = \left(A_{\overline{B}_{r\cdot}}\right)^{-1} + v \otimes (A_{B_{r\cdot}})^{-1} \tag{3}$$

The updating scheme of the inverse is shown in (4).

$$(A_B)^{-1} : \left|b_{r1} \cdots b_{rr} \cdots b_{rm}\right|$$

$$(A_{\overline{B}})^{-1} = \begin{vmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ 0 & 0 & 0 \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mm} \end{vmatrix} + \begin{vmatrix} -\frac{h_{1l}}{h_{rl}} \\ \vdots \\ -\frac{1}{h_{rl}} \\ \vdots \\ -\frac{h_{ml}}{h_{rl}} \end{vmatrix} \tag{4}$$

The time complexity of this method is $\Theta(m^2)$.

4 Computational Study

A computational study of the aforementioned updating techniques on the EPSA and RSM is presented in this section. The computational comparison has been performed on a quad-processor Intel Core i7 3.4 GHz with 32 Gbyte of main memory and 8 cores, running under Microsoft Windows 7 64-bit. The algorithms have been implemented using MATLAB Professional R2013a.

The Netlib set of LPs [8] was used in this computational study, which is a well-known suite containing many read world LPs. 71% of the Netlib LPs are ill-conditioned [17], so numerical difficulties may occur. Table 1 presents some useful information about the test bed, which was used in the computational study. The first column includes the name of the problem, the second the number of constraints, the third the number of variables, the fourth the nonzero elements of matrix A and the fifth the objective value. For each instance, we averaged times over 10 runs.

In this computational study, we examine the significance of the choice of the basis updating technique on EPSA and RSM. Thus, we have executed these algorithms with the three different basis update methods presented in Section 2. Table 2 presents the results from the execution of EPSA and RSM with the aforementioned basis update methods over the Netlib set. All times in Table 2 are measured in seconds using the MATLAB's built-in command cputime. In Table 2, the following abbreviations are used: (i) INV - MATLAB's built-in function, (ii) PFI - Product Form of the Inverse, and (iii) MPFI - Modification of the Product Form of the Inverse. The results are graphically illustrated in Figure 1.

From the above results, we observe that: (i) the percentage of the time to compute the basis inverse to the total time is bigger in RSM than EPSA, and (ii) PFI is much faster than MPFI and MATLAB's inv over the Netlib set.

Table 1. Statistics of the Netlib LPs (optimal, Kennington and infeasible LPs)

Name	Constraints	Variables	Nonzeros A	Objective value
ADLITTLE	57	97	465	2.25E+05
AFIRO	28	32	88	-4.65E+02
AGG	489	163	2,541	-3.60E+07
AGG2	517	302	4,515	-2.02E+07
AGG3	517	302	4,531	1.03E+07
BANDM	306	472	2,659	-1.59E+02
BEACONFD	174	262	3,476	3.36E+04
BLEND	75	83	521	-3.08E+01
DEGEN2	445	534	4,449	-1.44E+03
E226	224	282	2,767	-1.88E+01
FFFFF800	525	854	6,235	5.56E+05
ISRAEL	175	142	2,358	-8.97E+05
LOTFI	154	308	1,086	-2.53E+01
SC50A	51	48	131	-6.46E+01
SC50B	51	48	119	-7.00E+01
SC105	106	103	281	-5.22E+01
SC205	206	203	552	-5.22E+01
SCAGR7	130	140	553	-2.33E+06
SCFXM1	331	457	2,612	1.84E+04
SCFXM2	661	914	5,229	3.67E+04
SCFXM3	991	1,371	7,846	5.49E+04
SCORPION	389	358	1708	1.88E+03
SCRS8	491	1,169	4,029	9.04E+02
SCTAP1	301	480	2,052	1.41E+03
SCTAP3	1,481	2,480	1,0734	1.42E+03
SHARE1B	118	225	1,182	-7.66E+04
SHARE2B	97	79	730	-4.16E+02
SHIP04L	403	2,118	8,450	1.79E+06
SHIP04S	403	1,458	5,810	1.80E+06
SHIP08L	779	4,283	17,085	1.91E+06
SHIP08S	779	2,387	9,501	1.92E+06
SHIP12L	1,152	5,427	21,597	1.47E+06
SHIP12S	1,152	2,763	10,941	1.49E+06
STOCFOR1	118	111	474	-4.11E+04

5 Decision Support System Analysis and Design

5.1 Object-Oriented Analysis

Figure 2 presents the decision making process that the decision-policy maker can perform using the DSS. Firstly, decision maker formulates the problem under examination as a linear programming problem. Then, the data acquisition, validation and verification phase follows, so, the decision maker may upload the input data to the DSS and select the algorithms and the basis update methods to be evaluated, in the next step. Then, the algorithms evaluation and execution

Table 2. Results of the Exterior and the Revised Simplex Algorithm Using Three Different Updating Methods over the Netlib Set)

Name	EPSA-INV		EPSA-PFI		EPSA-MPFI		RSM-INV		RSM-PFI		RSM-MPFI	
	Total	Inverse	Total	Inverse	Total	Inverse	Total	Inverse	Total	Inverse	Total	Inverse
ADLITTLE	0.1872	0.0653	0.1746	0.0362	0.1544	0.0377	0.0936	0.0468	0.0312	0.0020	0.0468	0.0312
AFIRO	0.0312	0.0123	0.0162	0.0042	0.0268	0.0050	0.0010	0.0001	0.0010	0.0001	0.0010	0.0001
AGG	0.1092	0.0468	0.0772	0.0179	0.0932	0.0190	0.3432	0.2496	0.0780	0.0312	0.0936	0.0468
AGG2	0.2808	0.0312	0.2317	0.0176	0.2642	0.0187	1.0140	0.8736	0.2184	0.0780	0.2496	0.0624
AGG3	0.3432	0.0468	0.1647	0.0265	0.3193	0.0276	1.5912	1.4040	0.2652	0.0936	0.2652	0.1248
BANDM	0.7176	0.1716	0.6816	0.0885	0.6943	0.0948	1.7160	1.2012	0.6552	0.1404	0.6552	0.1560
BEACONFD	0.0312	0.0153	0.0280	0.0094	0.0219	0.0108	0.0312	0.0230	0.0312	0.0020	0.0156	0.0050
BLEND	0.0936	0.0780	0.0681	0.0425	0.0782	0.0507	0.0780	0.0624	0.0312	0.0156	0.0312	0.0016
DEGEN2	5.2884	2.9172	3.1063	1.8425	3.4833	2.072	17.0509	15.4285	5.7408	3.8844	6.4428	4.5864
E226	0.4056	0.1716	0.2470	0.1101	0.2967	0.1256	1.2324	1.1544	0.2808	0.2028	0.2496	0.1404
FFFFF800	1.2792	0.2496	0.7301	0.1408	0.9131	0.1497	1.9032	1.0608	1.1232	0.2808	1.1232	0.2808
ISRAEL	0.1404	0.0468	0.1081	0.0238	0.1189	0.0266	0.5616	0.5460	0.1092	0.0624	0.0936	0.0624
LOTFI	0.2496	0.0312	0.1869	0.0171	0.2117	0.0203	0.3276	0.1716	0.1716	0.0468	0.1716	0.0468
SC50A	0.0936	0.0624	0.0616	0.0341	0.0622	0.0364	0.0936	0.0780	0.0624	0.0030	0.0312	0.005
SC50B	0.6396	0.4836	0.4236	0.1847	0.4633	0.2048	0.5304	0.4056	0.2184	0.1092	0.1872	0.0780
SC105	0.0312	0.0156	0.0243	0.0083	0.0234	0.0088	0.0468	0.0034	0.0312	0.0020	0.0156	0.0050
SC205	0.0156	0.0068	0.0094	0.0032	0.0127	0.0036	0.0312	0.0180	0.0156	0.0020	0.0000	0.0030
SCAGR7	0.0936	0.0312	0.0618	0.0156	0.0782	0.0182	0.0936	0.0560	0.0780	0.0156	0.0780	0.0156
SCFXM1	0.9360	0.1716	0.671	0.1076	0.8369	0.1211	1.5756	0.9204	0.7956	0.1560	0.7956	0.1716
SCFXM2	7.0356	1.482	4.6819	0.8760	5.1343	1.0299	13.7281	8.7985	6.2556	1.3416	6.5052	1.716
SCFXM3	25.8338	7.6440	18.7696	3.0041	19.1410	3.2197	54.9748	38.5478	21.4813	5.2260	23.1661	6.9576
SCORPION	0.7488	0.0624	0.5746	0.0396	0.7092	0.0455	0.9360	0.2340	0.7644	0.0624	0.7332	0.0780
SCRS8	6.4740	1.1856	4.8940	0.4658	5.0458	0.5091	9.8281	5.3664	5.2416	0.8112	5.3040	0.9048
SCTAP1	0.1716	0.0780	0.1251	0.0341	0.1374	0.0390	0.4680	0.3744	0.1716	0.1248	0.1872	0.0624
SCTAP3	3.7752	2.2776	1.7438	1.0635	1.8615	1.1210	104.7391	103.5223	7.0356	5.9124	7.2072	6.2088
SHARE1B	0.2340	0.0312	0.2096	0.0155	0.2195	0.0181	0.3120	0.1560	0.1872	0.0156	0.1560	0.0468
SHARE2B	0.0624	0.0241	0.0339	0.0158	0.0464	0.0178	0.0468	0.0468	0.0312	0.0212	0.0312	0.0212
SHIP04L	5.8812	0.0936	5.5653	0.0465	5.7380	0.0510	5.4600	0.0312	5.4444	0.0156	5.4756	0.0468
SHIP04S	2.2308	0.0156	1.9913	0.0070	2.1237	0.0084	1.9188	0.0312	1.9188	0.0312	1.9032	0.0312
SHIP08L	25.163	0.2340	24.0501	0.0932	24.4794	0.1004	26.177	2.5896	23.8214	0.2340	23.9462	0.2964
SHIP08S	4.2432	0.1092	4.0686	0.0410	4.1066	0.0436	4.3680	0.5304	3.8688	0.0780	3.8688	0.0936
SHIP12L	55.3648	0.1716	55.0681	0.0787	55.2934	0.0852	59.1244	5.6628	53.8359	0.3588	53.9763	0.3588
SHIP12S	7.6440	0.0936	7.5187	0.0334	7.6120	0.0357	8.4709	1.4196	7.1292	0.0780	7.1448	0.0624
STOCFOR1	0.0468	0.0250	0.4180	0.0101	0.4271	0.0114	0.0624	0.0156	0.0468	0.0030	0.0312	0.005
Average	**4.5846**	**0.5348**	**4.0231**	**0.2516**	**4.1244**	**0.2761**	**9.3803**	**5.6185**	**4.3286**	**0.5718**	**4.4172**	**0.6680**

step follows. The last step includes the presentation and the analysis of the results. Finally, the decision maker validates the results and if necessary provides feedback on the operation and the updated decision making process is performed again.

The sequence diagram in Figure 3 shows the whole interaction between the decision maker and the DSS in order to produce the report that will assist further in the decision making process. The decision-policy maker interacts with the initial screen of the DSS and uploads the input file, selects the algorithms (RSM and\or EPSA) and basis update methods (full inverse and\or PFI and\or MPFI) and presses the 'Report' button. Then, the system validates the input data and executes the RSM and EPSA algorithms for each basis update method, collects the total execution time, the time to perform the basis inverse, the number of iterations and the objective value, and presents these results in the report

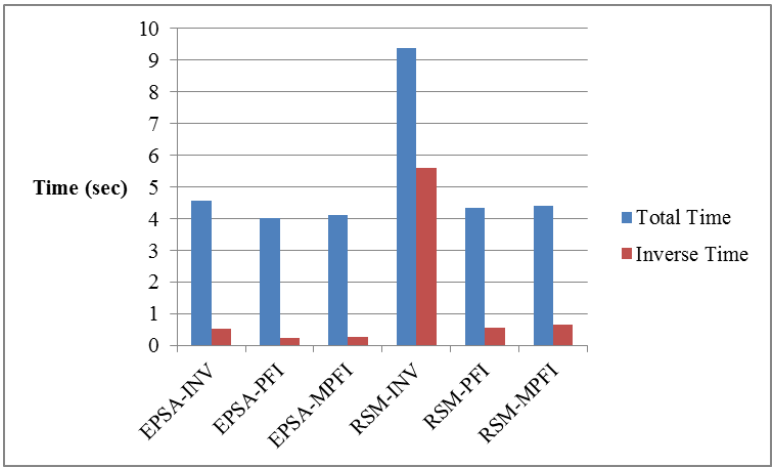

Fig. 1. Average Total and Inverse Time of the Exterior and Revised Simplex Algorithm Using Three Different Updating Methods over the Netlib Set

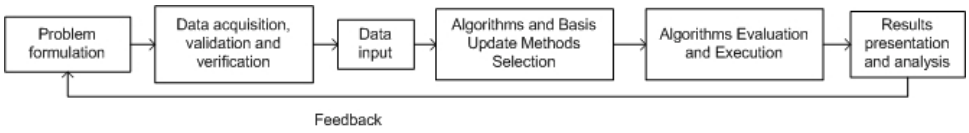

Fig. 2. Decision Making Process

screen. Finally, the decision maker can export the results as a pdf file.

Figure 4 presents the class diagram of the DSS. InitialScreen is a boundary class that includes three methods responding to decision maker's action events: i) upload input file, ii) select algorithms and basis update methods, and iii) press 'Report' button. SimplexAlgorithm is an abstract class that incorporates the common attributes and methods of RevisedSimplexAlgorithm and ExteriorPrimalSimplexAlgorithm. Matrix A contains the constraints coefficients, vector c the objective function coefficients, vector b the right-hand side values, vector Eqin the type of constraints (equality or inequality), and variable minMax the type of problem (minimization or maximization). Moreover, the SimplexAlgorithm class includes three methods that perform the basis update methods and an abstract method for the execution of the algorithms' logic. Finally, the RevisedSimplexAlgorithm and ExteriorPrimalSimplexAlgorithm classes override the abstract method executeAlgorithm of the SimplexAlgorithm and perform their steps for the solution of the linear programming problem.

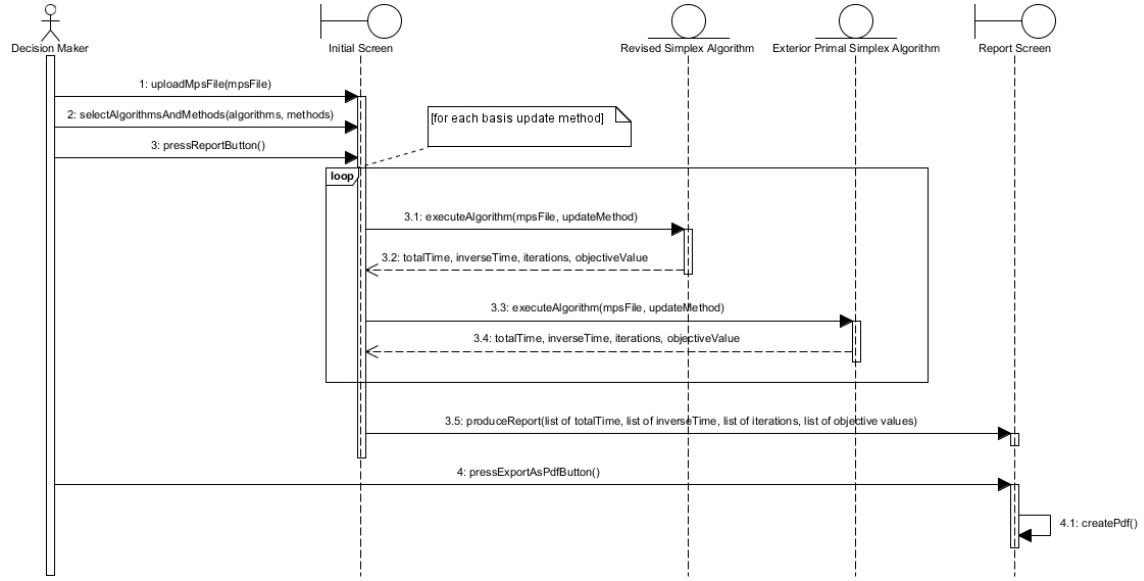

Fig. 3. Sequence Diagram

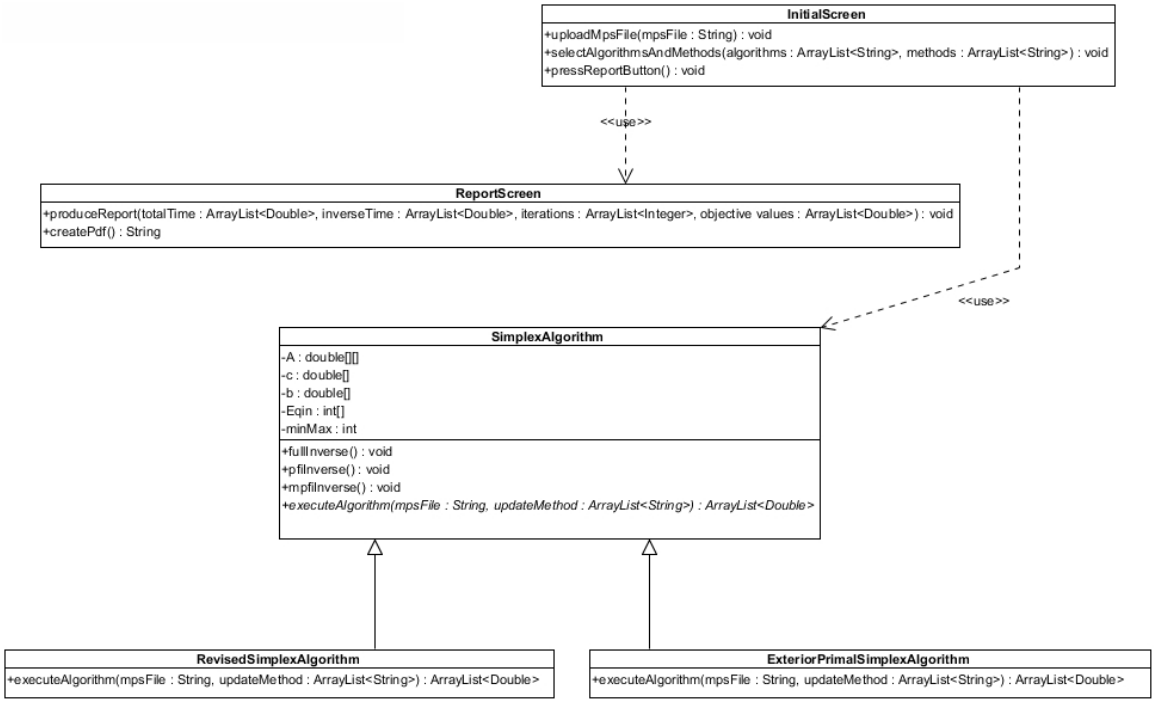

Fig. 4. Class Diagram

5.2 DSS Presentation

As stated in the previous section, all algorithms and update methods have been implemented using MATLAB. These algorithms have been converted to Java classes using the MATLAB Builder JA. The web interface of the DSS has been designed using jsp and multiple users can access it through a web browser.
The initial screen of the DSS is shown in Figure 5. The decision maker presses the 'Browse' button in order to upload the file containing the LP in mps format. In addition, the decision maker selects the algorithms and the basis update methods that will be included in the comparison. By pressing the 'Report' button, a screen with a report is presented, as shown in Figure 6. This screen includes the objective value and the iterations made by each algorithm. Furthermore, the total and the inverse time of each basis update method are presented both as numerical values and in a illustrative figure. Finally, the decision maker can export this report as a pdf file.
Many LPs of the Netlib set are real-world problems. Figures 5 and 6 present a case study for SCRS8, which is a problem of the Netlib set. SCRS8 is a technological assessment model for the transition from fossil to renewable energy resources in the United States [11] adapted from Manne [15]. SCRS8 includes $1,169$ variables with 491 constraints. From the results that are presented in 6, it is concluded that EPSA with the PFI updating method is the best choice for the solution of this problem.

The proposed web-based DSS offers important managerial implications. First, decision makers can formulate the problem under examination as a linear programming problem and get a thorough analysis. Problems that can be formulated as linear programming problems include telecommunications, bio-informatics, supply chain management, etc. Moreover, the decision-policy maker gets an overview of the type of the algorithm and basis update method that best suits to a specific problem.
Some limitations exist on the proposed DSS. Some problems cannot be formulated as linear programming problems. A second potential limitation of the proposed DSS is that it provides information only about the execution time of the basis update step; information about the other steps of the simplex type algorithms, like preconditioning techniques, scaling techniques and pivoting rules, can also be incorporated to the DSS.

6 Conclusions

The basis inverse is a crucial step in simplex-type algorithms. The time taken to perform the basis inverse dictates the total time of these algorithms. Hence, the basis update method must be carefully selected. In this paper, we performed a computational comparison of three basis update methods and incorporated them on EPSA and RSM. The results showed that PFI scheme is faster than MATLAB's inv and MPFI. Furthermore, the percentage of the time to compute

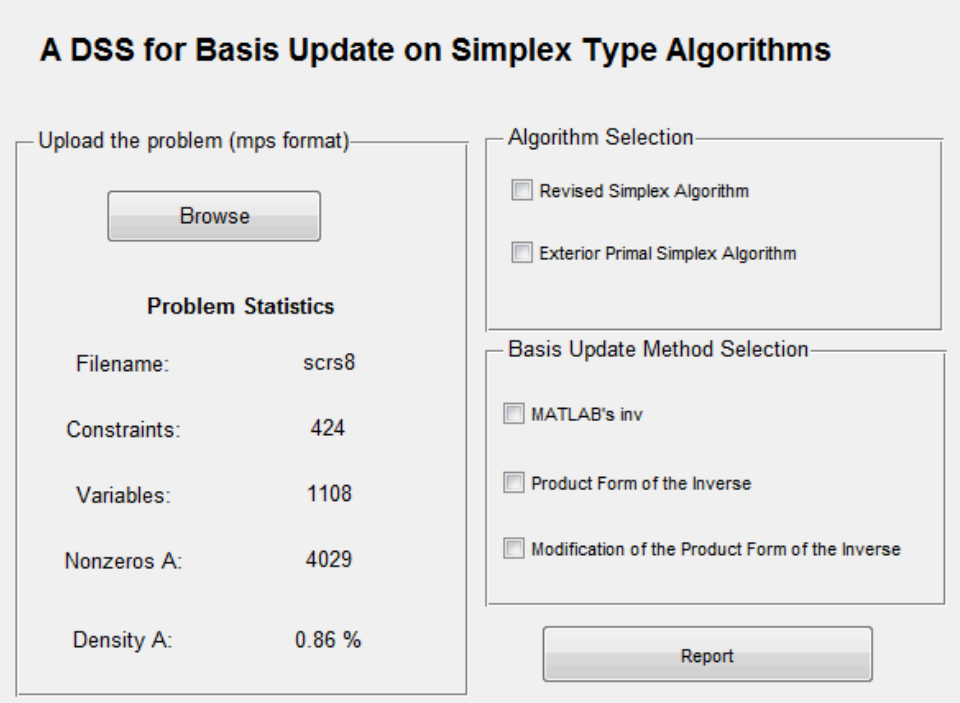

Fig. 5. Initial Screen of the web-based DSS

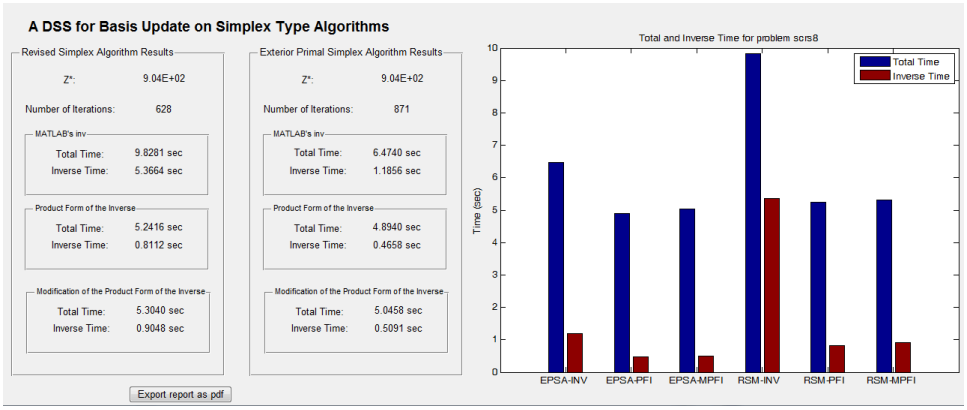

Fig. 6. Report of the Results

the basis inverse to the total time is bigger in the RSM than EPSA.
However, there are some instances that MPFI scheme is faster than PFI and the basis inverse time is bigger in EPSA than RSM. So, there isn't any best algorithm or best method for all instances. We have implemented a web-based DSS in order to assist decision-policy makers in the selection of the algorithm and basis update method to solve their LPs. All algorithms have been implemented using MATLAB and converted to Java classes using the MATLAB Builder JA. The web interface of the DSS has been designed using jsp and users can access it through a web browser from their PC/laptop or their smart device.
In future work, we plan to enhance the DSS with other options that will give decision maker the opportunity to control all the steps of simplex type algorithms. More specifically, we plan to add preconditioning techniques, scaling techniques and pivoting rules. Finally, we plan to present real application case studies that the proposed DSS can be utilized.

References

1. Bartels, R.H., Golub, G.H.: The Simplex Method of Linear Programming Using LU Decomposition. Communications of the ACM 12, 266–268 (1969)
2. Bhargava, H.K., Krishnan, R.: The World Wide Web: opportunities for operations research and management science. INFORMS Journal on Computing 10, 359–383 (1998)
3. Bhargava, H. K., Power, D. J., Sun, D.: Progress in Web-based decision support technologies. Decision Support Systems 43:4, 1083–1095 (2007)
4. Benhamadou, M.: On the Simplex Algorithm 'Revised Form'. Advances in Engineering Software 33, 769–777 (2002)
5. Dantzig, G.B.: Computational Algorithm of the Revised Simplex Method. RAND Report RM-1266. The RAND Corporation, Santa Monica, CA (1953)
6. Dantzig, G.B., Orchard-Hays, M.: The Product Form of the Inverse in the Simplex Method. Math. Comp. 8, 64–67 (1954)
7. Forrest, J.H., Tomlin, J.A.: Updated Triangular Factors of the Basis to Maintain Sparsity in the Product Form Simplex Method. Mathematical Programming 2, 263–278 (1972)
8. Gay, D.M.: Electronic Mail Distribution of Linear Programming Test Problems. Mathematical Programming Society COAL Newsletter 13, 10–12 (1985)
9. Ghodsypour, S.H., O'Brien, C.: A decision support system for supplier selection using an integrated analytic hierarchy process and linear programming. International Journal of Production Economics 56, 199–212 (1998)
10. Hernández, J., Zarate, P., Dargam, F., Delibašić, B., Liu, S., Ribeiro, R.: Decision Support Systems Collaborative Models and Approaches in Real Environments. Lecture Notes in Business Information Processing 121, Springer Berlin Heidelberg (2012)
11. Ho, J.K.: Nested decomposition of a dynamic energy model. Management Science 23, 1022–1026 (1977)
12. Lappi, J., Nuutinen, T., Siitonen, M.: A linear programming software for multilevel forest management planning. Management systems for a global economy with global resource concerns, 470–482 (1996)

13. Lauer, J., Jacobs, L.W., Brusco, M.J., Bechtold, S.E.: An interactive, optimization-based decision support system for scheduling part-time, computer lab attendants. Omega 22:6, 613–626 (1994)
14. Lourenço, J.C., Morton, A., Bana e Costa, C.A.: PROBEA multicriteria decision support system for portfolio robustness evaluation. Decision Support Systems 54, 534–550 (2012)
15. Manne, A.S.: Sufficient conditions for optimality in an infinite horizon development plan. Econometrica 38, 18–38 (1970)
16. Markowitz, H.: The Elimination Form of the Inverse and its Applications to Linear Programming. Management Science 3, 255–269 (1957)
17. Ordóñez, F., Freund, R.: Computational experience and the explanatory value of condition measures for linear optimization. SIAM J. Optimization 14:2, 307–333 (2003)
18. Paparrizos, K., Samaras, N., Stephanides, G.: An Efficient Simplex Type Algorithm for Sparse and Dense Linear Programs. European Journal of Operational Research 148:2, 323–334 (2003)
19. Ploskas, N., Samaras, N.: A Computational Comparison of Basis Updating Schemes for the Simplex Algorithm on a CPU-GPU System. Journal of Computational Science, Paper under review (2013)
20. Power, D. J., Kaparthi, S.: Building Web-based decision support systems. Studies in Informatics and Control 11:4, 291–302 (2002).
21. Reid, J.: A Sparsity-exploiting Variant of the Bartels-Golub Decomposition for Linear Programming Base. Mathematical Programming 24, 55–69 (1982)
22. Suhl, L.M., Suhl, U.H.: A Fast LU Update for Linear Programming. Annals of Operations Research 43:1, 33–47 (1993)
23. Venkataramanan, M.A., Bornstein, M.: A decision support system for parking space assignment. Mathematical and computer modelling 15:8, 71–76 (1991)