A Simplex-Cosine Method for Solving Hard Linear Problems

FEDERICO TRIGOS[1], JUAN FRAUSTO-SOLIS[2] and RAFAEL RIVERA-LOPEZ[3]
[1] Division of Engineering and Sciences
ITESM, Campus Toluca
Av. Eduardo Monroy Cárdenas 2000, Col. San Antonio Buenavista, 50110, Toluca, Edo. de México
MÉXICO

[2,3] Department of Computer Science
ITESM, Campus Cuernavaca
Av. Paseo de la Reforma 182-A, 62289, Temixco, Morelos
MEXICO

Abstract: - In this paper a new method for linear problems resolution is presented. This method named Simplex Cosine Method (SCM) is based on the analysis of the angles between the gradient of the objective function and the gradient of each constraint; these angles are found by the application of the simple cosine function. As the traditional Simplex Method (SM), proposed by Dantzig, SCM selects one vertex of the feasible region as an initial point and applies a gaussian procedure, but SCM chooses this vertex using the smallest of those angles. The optimality condition analysis presented in the paper reveals that SCM start point is in the border of the optimal solution, so its number of iterations and its time for finding the optimum is lower than the classical simplex method. That situation is tested in this paper for a set of Klee-Minty problems showing substantial improvements in the computational performance of the algorithm.

Key-Words: -Linear programming, Simplex method, optimization.

1 Introduction

The Simplex Method (SM) developed by Dantzig in 1947 [1] marks the start of the modern era in optimization, because this method has made possible the resolution of many assignment problems, resulting on an improvement in the companies's efficiency. SM has been used in practically every human knowledge's area and has demonstrated to be very efficient in the resolution of linear problems. Even with the appearing of the interior point methods, proposed by Karmarkar in 1984 [2], the SM is still the most utilized method and its efficiency to resolve practical problems is undeniable.

SM applies a gaussian procedure in every chosen vertex in order to calculate the objective function and all the basic variables; SM selects in each iteration, the adjacent vertex allowing it to improve the value of the objective function. In SM the initial vertex is defined using a set of slack and artificial variables of the linear problem like the basic variables. Note that the definition of the initial vertex, as well as the rule that determines the form to select the adjacent vertex (pivot rule) affects the efficiency of the method. The reason of that is the length of the sequence of adjacent vertices until the optimal solution depends on a good selection of these two elements.

The SM requires an initial vertex that represents a feasible basic solution in order to assure the construction of a sequence of feasible basic solutions that allows to reach the optimal value when this one exists. The problem of finding a feasible initial point can be nontrivial and sometimes impossible for the classical simplex method (SM). In this case a common used approach is denominated the phase I of SM, where artificial variables are introduced to determine one first feasible basic solution. The solution of phase I is a feasible vertex that is used like starting point of phase II of the classical simplex method [1]. Alternatives approaches for the phase I were proposed by Wolfe [3] and Maros [4], and others try to improve the sequence of feasible basic solutions, defining different pivot rules ([5]-[7]) and, finally they are those very similar to SM [8].

In this paper SCM is presented; it is similar to SM but as we are going to see, it is more efficient. SCM has a new approach to determine the initial vertex based on the analysis of the angles between the gradient of the objective function and the gradients of the constraints. These angles are calculated through the theirs cosines; for this reason the method receives the name of simplex-cosine method or SCM.

2 Relation between the gradients of the constraints and the gradient of the objective function

The mathematical model of a linear problem is in its general form shown in (1).

$$\min z = f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \qquad (1)$$

subject to

$$\mathbf{a}_i^T\mathbf{x} \ (\geq,\leq) \ b_i \qquad i{:}1,\ldots,m$$
$$\mathbf{x} \geq \mathbf{0}$$
$$\mathbf{a}_i, \mathbf{c} \text{ and } \mathbf{x} \in R^n$$

Where **c** is the vector of costs, the vector **x** contains the decision variables, b_i is the resource limit for the constraint i, and $\mathbf{a}_i$ associates the use of each resource to each variable in the constraint i.

2.1 Optimality conditions of one solution

A form to define if a vector **x** produces a minimum for the problem (1) is using the concept of feasible direction. In [9] is defined that **d** is a feasible direction if the condition (2) is fulfilled, and applying the expansion of first order of the series of Taylor is defined the condition of feasible direction, showed in (3).

$$f(\mathbf{x} + \mathbf{d}) < f(\mathbf{x}) \qquad (2)$$
$$\nabla f(\mathbf{x})^T\mathbf{d} < 0 \qquad (3)$$

Since **x+d** produces a smaller value for the objective function that with **x**, **x** is not a minimum. **x+d** is also feasible, since it satisfies the constraints. In order that **x** represents a minimum of a linear problem, the condition of feasible direction is not due to fulfill, that is, **d** does must not exist.

Generalizing the notation of the constraints of the linear problem (1) with $\mathbf{g_i(x)\geq 0}$, the condition of feasible direction is defined as in (4).

$$g_i(\mathbf{x}+\mathbf{d}) = g_i(\mathbf{x}) + \nabla g_i(\mathbf{x})^T\mathbf{d} \geq 0 \qquad (4)$$

If $\mathbf{g_i(x)>0}$, any direction **d** can satisfy (3) and (4), reason why if **x** is a minimum, satisfy $\nabla f(\mathbf{x}^*)=0$. If $\mathbf{g_i(x)=0}$, then **d** must satisfy the inequality (5).

$$\nabla g_i(\mathbf{x})^T\mathbf{d} \geq 0 \qquad (5)$$

It is deduced that if **x** is a minimum, the vectors $\nabla f(\mathbf{x}^*)$ and $\nabla g_i(\mathbf{x})$ must lay for the same direction, that is, the condition (6) must be fulfilled, to $\mu^*_i \geq 0$ (Figure 1).

$$\nabla f(\mathbf{x}) = \mu_i \nabla g_i(\mathbf{x}) \qquad (6)$$

In order to group both cases ($g_i(\mathbf{x})>0$ y $g_i(\mathbf{x})=0$) is introduced the complementarity's condition (8) in (7).

$$\nabla f(\mathbf{x}^*) - \mu_i^* \nabla g_i(\mathbf{x}^*) = 0, \ \mu_i^* \geq 0 \qquad (7)$$
$$\mu_i^* g_i(\mathbf{x}^*) = 0 \qquad (8)$$

From this representation one classification of inequality constraints is derived: a constraint is active when its corresponding coefficient μ_i^* is positive, and is inactivate when μ_i^* is zero.

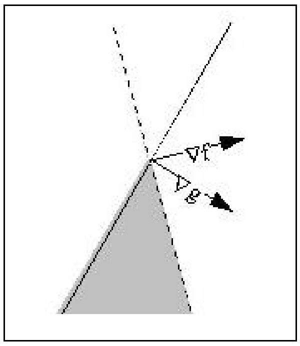

Fig 1. In order to reduce the space where **d** can exist, the vectors ∇f *y* ∇g must be parallel and in the same direction.

The relation (7) is interpreted geometrically as it follows: So that a value of **x** produces a minimum of (1) the gradient of the objective function must be a positive linear combination of the gradients of the active constraints, that is, this gradient remains within the convex cone represented by this active constraints (figure 2).

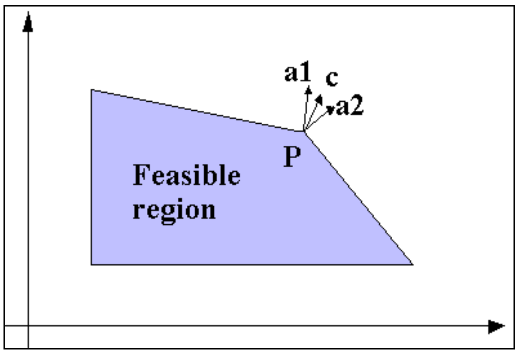

Fig. 2. The point **P** is optimum because the gradient of objective function **c** lies in the cone spanned by the gradients of the active constraints (**a1** and **a2**).

2.2 Angles between gradients

Of the optimality condition based on the gradients of the functions of the mathematical model of the linear problem, it is observed that the angles between the gradients of the constraints and the angle of the objective function can provide information about the constraints: particularly we can know if a constraint is or not an active one. This premise is used for constructing the simplex cosine method. In figure 3 the angles between the gradient of the objective function ∇f and the gradients of the constraints ∇g_i are shown. In this figure, point **x*** is the optimum. It is observed that, in general, the smaller angles of this figure are formed by the gradients of the active constraints ($\mathbf{g_1}$ and $\mathbf{g_2}$).

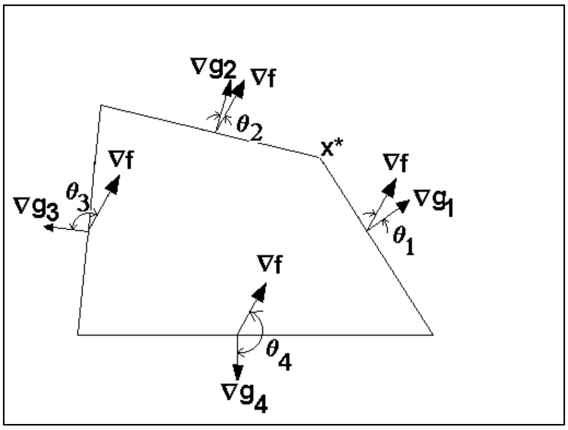

Fig. 3. The angles between the active constraints and the objective function are minors that those of the inactive constraints.

Because this observation cannot be generalized for all the linear problems, since it is not a guarantee that the constraints with the smaller angles are active in the optimum, is not possible to generate a procedure that uses only this information for determining an optimal value. Nevertheless, the vertex where the constraints form smallest angles can be used like starting point of the SM.

In order to determine the angle between two vectors the formula of the dot product of vectors used, that is shown in (9).

$$\nabla f(\mathbf{x})\cdot\nabla g_i(\mathbf{x})=|\nabla f(\mathbf{x})||\nabla g_i(\mathbf{x})|\cos\theta \qquad (9)$$

where θ is the angle between the vectors $\nabla f(\mathbf{x})$ and $\nabla g_i(\mathbf{x})$.

2.3 Slack variables

Another interesting observation is about the values that take the slack variables in the optimum, that is zero for an active constraint. This variables represent the nonbasic variables in the final tableau of the SM. For the problem represented by (10), the optimal solution of the objective function is z =21 while the value of the of decision variables are: x_1 = 3, x_2 = 1.5, s_3 = 2.5 and s_4 = 0.5. Where s_i represents the *i* slack variable used in the *i* constraint.

$$\max \quad z=5x_1+4x_2 \qquad (10)$$

Subject to

$$6x_1+4x_2 \le 24$$
$$x_1+2x_2 \le 6$$
$$-x_1+x_2 \le 1$$
$$x_2 \le 2$$

Figure 4 shows the result of the last iteration for the SM, where the nonbasic variables are s_1 and s_2; these are the slack variables of the active constraints in the optimum. We adopt a binary codification to represent the basic and nonbasic variables, as it is shown in the third row in figure 4. That means that the representation of the optimal solution can be done: 1) As it is usually done by the set of values of the variables (3, 1.5, 0, 0, 2.6 and 0.5) or well by 2) our binary representation (1, 1, 0, 0, 1 and 1) that we proposed in our Simplex-Genetic Method [10].

X_1	x_2	s_1	s_2	s_3	S_4	Variable
3.0	1.5	0	0	2.5	0.5	Value
1	1	0	0	1	1	Binary
141.34^0	128.65^0	4.96^0	24.77^0	96.34^0	51.34^0	Angle

Fig. 4. The slack variables are zero for the active constraints in the optimum

As it is observed in figure 4, the non-basic variables of the solution correspond the smaller angles between the gradients of the constraints and the gradient of the objective function. For the case of the original variables (x_1 and x_2) the angles are obtained from the constraints of non-negativity, and the slack variables are associated to each one of the constraints of the problem.

The simplex cosine method uses these two observations to obtain a point in which the SM can initiate its search of the optimum value.

3. The Simplex Cosine method

The main idea of this paper is to find one basic solution of the linear problem using the angles between the gradients of constraints and the gradient of the objective function, for using this solution as the initial vertex for the simplex procedure and reducing the number of iteration for reaching the optimum. This angles are used for determining the basic and non basic variables of this basic solution, constructing one binary string that represents this vertex. This string is processed to construct the actual value of the $\mathbf{B}^{-1}$ matrix and the revised simplex method (RSM) applies their simplex procedure [1].

In figure 5 the SCM algorithm is shown and in the next subsections, the principal aspects of this method are described.

The main components of the SCM are the schema for the representation of one solution using the information of the angles between the gradients and the transfer scheme from this binary representation to RSM.

procedure *Simplex_Cosine*;
begin
*{*** Cosine phase ***}*
q ¬ *Vector with the angles between each constraint and the objective function;*
s ¬ *Binary string with the representation of basic and non basic variables. The angles in* $\mathbf{q}$ *with the minor value represent one non basic variable* (s_i=0).
For n variables and m constraints, n minor angles represent the non basic variables.
*{***Transition phase***}*
Determine original $\mathbf{B}^{-1}$ ($\mathbf{B}^{-1}_0 = \mathbf{I}$);
I_{basic} ¬ $\{i | i \in s \wedge (s)_i = 1\}$;
Construct a matrix **T** of **A** such that $\{\mathbf{T}_j | j \in I_{basic}\}$;
Apply *i* row operations in matrix $[\mathbf{T}|\mathbf{B}^{-1}_0]$ such that $\mathbf{T}_i$=**I**;
$\mathbf{B}^{-1}_{better}$ ¬ $\mathbf{B}^{-1}_i$, the right side of matrix $[\mathbf{T}_i|\mathbf{B}^{-1}_i]$;
Actualize **C** such that $\{c_j = 0 | \mathbf{j} \in I_{basic}\}$;
Actualize **b**, $\mathbf{x}_B$ and $\mathbf{x}_{NB}$ with the *i* row operations;
*{***RSM phase***}*
optimality← *feasibility*← **true**;
while not *optimality*∧*feasibility* **do**
$\mathbf{R}_N \leftarrow \mathbf{C}_B\mathbf{B}^{-1}\mathbf{N} - \mathbf{C}_{NB}$;
if $\mathbf{R}_N > 0$ **then** *optimality*←**true**;
else begin
select *k* as the entering variable such that $(\mathbf{R}_N)_k$ is the minimum;
select *j* as the leaving variable as in (6);
if $\mathbf{x}_j$ not exist **then** *feasibility*←**false**;
else Update $\mathbf{x}_B$, $\mathbf{x}_{NB}$ and $\mathbf{B}^{-1}$;
end
end
if *optimality* **then** $z \leftarrow \mathbf{C}_B\mathbf{x}_B$;
end

Fig. 5. The Simplex Cosine Method.

3.1 Binary representation of solutions

Each string represents a basic solution of a LP. A LP with *n* variables *m* constraints has *m* basic and *n* non-basic variables. String length is *m*+*n* bits, where a bit 1 represents a basic variable and a bit 0 is a non-basic variable. All strings have *m* bits 1 and *n* bits 0. Figure 6 shows two examples of basic solutions for a LP with *n*=2 and *m*=4.

x_1	x_2	s_1	s_2	s_3	s_4
1	0	1	1	1	0

Basic variables: x_1, s_1, s_2, s_3
Non basic variables: x_2, s_4

x_1	x_2	s_1	s_2	s_3	s_4
1	0	0	1	1	1

Basic variables: x_1, s_2, s_3, s_4
Non basic variables: x_2, s_1

Fig. 6. Two examples of basic solution codified as binary string: bits 1 represent basic variables and bits 0 represent non-basic variables.

3.2 Determination of one initial solution

Using the problem represented in (10) the SCM will make the following calculations:

```
Gradient of the objective function:
   ∇f(x)= [ 5.00,  4.00]

Gradient of the non negativity constraints:
   x1 =[-1,  0] Cosine =-0.78 ( 141.34º)
   x2 =[ 0, -1] Cosine =-0.62 ( 128.65º)

Gradient of the constraints:
   ∇g1(x) = [ 6.00, 4.00] Cosine = 0.99 (4.96º)
   ∇g2(x) = [ 1.00, 2.00] Cosine = 0.90 (24.77º)
   ∇g3(x) = [-1.00, 1.00] Cosine =-0.11 (96.34º)
   ∇g4(x) = [ 0.00, 1.00] Cosine = 0.62 (51.34º)
```

The value between parenthesis is the angle corresponding to the cosine calculated. In this case the number of basic variables is four and the number of nonbasic variables is two; this is the reason the two smaller angles were taken from the previous calculations (the values associated to the constraints $\nabla g_1(\mathbf{x})$ and $\nabla g_2(\mathbf{x})$). This criterion allows to construct the binary string that represents a possible basic solution of the problem. The resulting codification to apply the angles is: 110011.

Before to apply the RSM, the solution is evaluated for determining if this represents a vertex of the feasible region. In this case the solution is feasible and optimal, producing the following values

```
Objective function = 21.00
Value of the variables
3.00 1.50 0.00 0.00 2.50 0.50
This is a feasible solution.
```

The solution of this phase can be feasible or not, optimal or not, so it is possible to apply four schemes:

1. Optimal Feasible Solution: This is the solution of the original problem.
2. Non-optimal feasible solution: To apply the revised simplex method (phase II).
3. Optimal infeasible solution: To apply the dual simplex method.
4. Non-optimal infeasible solution: To apply the simplex method(phase I and II) or some other method (simplex genetic method [10] or simplex annealing method [11]).

3.3 Transfer scheme from cosine phase to RSM

The solution produced by the cosine phase will be the initial solution for the RSM. The cosine solution (binary string *s* in the algorithm shows in figure 5) is transformed as the initial elements for RSM ($\mathbf{B}^{-1}_{\text{better}}$, $\mathbf{x}_B$, $\mathbf{x}_{NB}$, $\mathbf{C}$ and $\mathbf{b}$). The original $\mathbf{B}^{-1}_0$ is an identity matrix $\mathbf{I}$ (constructed for the coefficients of the slack variables in the restrictions). Matrix $\mathbf{T}$ is then constructed using several columns of the original matrix $\mathbf{A}$, this selection is based on the binary string: an $\mathbf{A}_j$ column is used in $\mathbf{T}$ if the site in binary string is 1. Matrix $\mathbf{T}$ is the actual $\mathbf{B}_{\text{better}}$. Then, reducing $\mathbf{T}$ to $\mathbf{I}$ and applying the same row operations in the original $\mathbf{B}^{-1}_0$, the actual $\mathbf{B}^{-1}_{\text{better}}$ is calculated. Formulas (11) and (12) show how $\mathbf{B}^{-1}_{\text{better}}$ can be calculated using this schema. Figure 7 shows an example of this schema.

$$\mathbf{B}_{\text{better}}\mathbf{x} = \mathbf{Ib} = \mathbf{B}^{-1}_0\mathbf{b} \qquad (11)$$
$$\mathbf{x} = \mathbf{B}^{-1}_{\text{better}}\mathbf{b} \qquad (12)$$

The value of new $\mathbf{C}$ is produced changing the c_j values by zeroes, if *j* is the index of a new basic variable. Vector $\mathbf{b}$ is actualized applying the same row operations used in the $\mathbf{T}$ reduction process. Finally, vectors $\mathbf{x}_B$ and $\mathbf{x}_{NB}$ are actualized. This schema avoids the direct calculus of $\mathbf{B}^{-1}_{\text{better}}$.

4 Experimentation and results

In order to prove this approach based on cosines, the cube of Klee-Minty was used (MK_d), a hard linear problem where the SM has an exponential behavior. This behavior was demonstrated by Klee and Minty in [12]. KM_d represented in (13) is a deformed product of one d-cube. The figure 8 shows a 3-dimensional KM_d using $\varepsilon < {}^1/_3$. KM_d has an exponential behavior for any deterministic pivot rule ([13]-[15]).

$\mathbf{B}_{\text{better}}$			$\mathbf{B}^{-1}_0$		
2	8	6	1	0	0
4	2	-2	0	1	0
3	-1	1	0	0	1

applying
row
operations

			$\mathbf{B}^{-1}_{\text{better}}$		
1	0	0	0	1/10	1/5
0	1	0	1/14	8/70	-1/5
0	0	1	1/14	-13/70	1/5

Fig. 7. An example of the transfer schema over the values of matrix $\mathbf{B}^{-1}$ and to calculate its new value that is passed to RSM.

$$\max x_d \qquad (13)$$
subject to
$$0 \le x_1 \le 1$$
$$ex_{j-1} \le x_j \le 1 - ex_{j-1}$$
$$\text{for } j{:}1,\ldots,d \text{ and } 0<e<{}^1/_2$$

The principal results obtained with the SCM run on a Intel Caleron with 800 Mhz and 64 MB en RAM and they are shown in figures 9, 10 and 11. Figure 9 shows an exponential behavior of RSM with the number of variables, and the same for SCM; however the increased rate in the later is much smaller that the former. It is clear that SCM has a substantial improvement over RSM for KM_d.

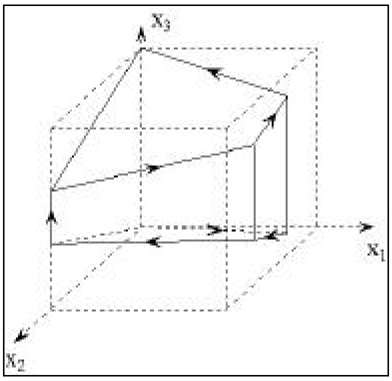

Fig. 8. The Klee-Minty cube for n=3 and ε < 1/3.

Of the obtained results (figure 10) it is observed that the SCM executes in average 66% less iterations than the SM, reason why is clear the advantage to use this method. Figure 11 shows the percentage saving for the SCM over the RSM in several dimensions of the Klee-Minty cube.

At the present time the method is becoming general to apply it to classic test problems, like those of the NETLIB [16], studying the impact of the use of the artificial variables in the development of the algorithm, but the behavior for this class of hard problems gives indications of its possible success in the resolution of these classic problems.

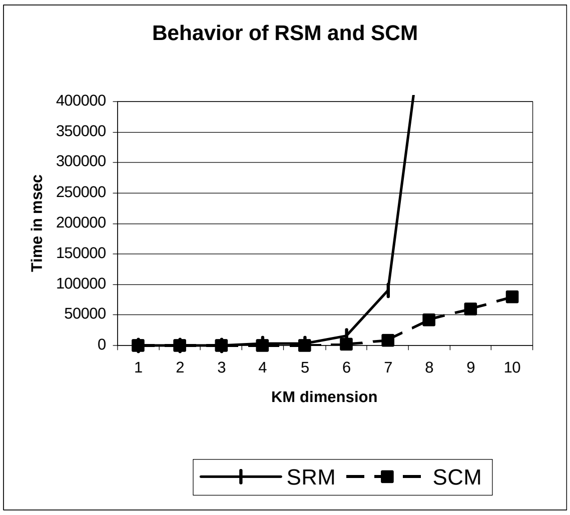

Fig. 9. Behavior of SCM and RSM.

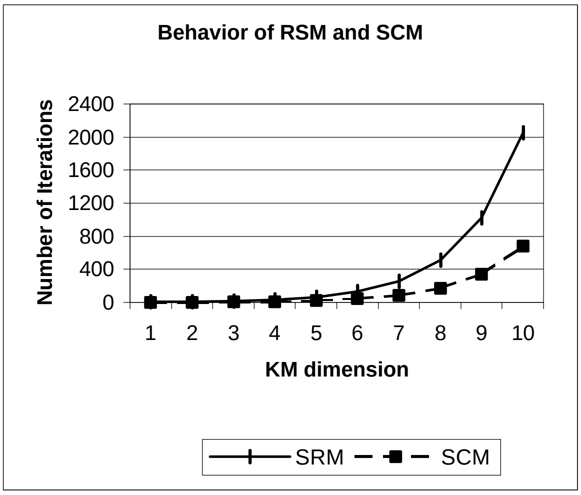

Fig. 10. The comparison between the RSM and SCM.

KMd dimension	Iterations			time in sec		
	RSM	SCM	% saving	RSM	SCM	% saving
1	4	2	50.00	0.06	0.05	16.67
2	8	3	62.50	0.17	0.09	47.06
3	16	6	62.50	0.50	0.11	78.00
4	32	11	65.63	2.69	0.43	84.01
5	64	22	65.63	2.91	0.44	84.88
6	128	43	66.41	15.32	1.60	89.56
7	256	86	66.41	90.47	8.13	91.01
8	512	171	66.60	603.58	41.47	93.13
9	1024	342	66.60	953.40	62.34	93.46
10	2048	683	66.65	8967.96	581.21	93.52

Fig. 11. % saving obtained between RSM and SCM.

References:

[1] G. B. Dantzig y M. N. Tapa. *Linear Programming, 1: Introduction,* Springer-Verlag, 1997.

[2] N. K. Karmarkar. A new polynomial time algorithm for linear programming, in *Combinatorica*, Vol. 4, 1984, pp. 373-395.

[3] P. Wolfe, The composite simplex algorithm, in *SIAM Review*, Vol. 7, 1965, pp. 42-54.

[4] I. Maros. A general phase-I method in linear programming, in *European Journal of Operational Research*, Vol. 23, 1986, pp. 64-77.

[5] T. Terlaky and S. Zhang, Pivot Rules for Linear Programming, in *Annals of Operations Research*, Vol. 46, 1993, pp. 203-233.

[6] K. H. Borgwardt, *The Simplex Method. A Probabilistic Analysis*, Vol. 1, Springer-Verlag, 1987.

[7] M. Goldwasser. A Survey of Linear Programming in Randomized Subexponential Time, in *SIGACT News*, Vol. 26, No. 2, 1995, pp. 96-104.

[8] M. J. Best and K. Ritter. *Linear Programming, Active Set Analysis and Computer Programs,* Prentice Hall, 1985.

[9] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming*, John Wiley and Sons, 1979.

[10] J. Frausto-Solis, R. Rivera-Lopez and F. Ramos-Quintana. A simplex Genetic Method for Solving the Klee Minty Cube, in *WSEAS Transactions on Systems*, Vol. 2, No. 1, April 2002, pp. 232-237.

[11] J. Frausto-Solis, R. Rivera-López, F. Ramos-Quintana and E.Morales. Fast hard Linear Problem resolution using Simulatead Annealing and Datzing's Rule, in *Proc. of IASTED Int. Conf. in Artificial Intelligence and Soft Computing (ASC 2001)*, May 21-24, 2001, Cancun, México.

[12] V. Klee and G. J. Minty, How good is the simplex algorithm?, in *Inequalities III*, Academic Press, 1972, pp. 159-175.

[13] T. Terlaky and S. Zhang, Pivot Rules for Linear Programming, in *Annals of Operations Research*, Vol. 46, 1993, pp. 203-233.

[14] N. Amenta and G. Ziegler, Deformed products and maximal shadows of polytopes, in *Advances in Discrete and Computational Geometry*, Amer. Math. Soc., Vol. 223, 1999, pp. 57-90.

[15] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, Dover, 1998.

[16] The Netlib Repository for linear problems at http://www.netlib.org/lp/data/