A fresh view on the sagitta method

Ángel Santos-Palomo [a,*] Pablo Guerrero-García [a]

[a] *Department of Applied Mathematics, University of Málaga, 29071 Málaga, Spain*

Abstract

A restructuration of the original sagitta method is accomplished by maintaining its initial phase, but making it up with a normal phase in which primal and/or dual iterations, successively or interminglely, are carried out. In a primal or dual iteration, the method searchs for the corresponding feasibility, but does not require that the other one has been achieved; moreover, once a feasibility is achieved, its maintenance is not mandatory. The restarting of the original sagitta method is eliminated and thus, with this new structure, multiple modifications are possible.

The new sagitta methods usually start without any iteration point and in its initial phase attempt to find a descent direction of the feasible region. Constraint indices are added to the foreactive set until this direction is found or until there is not a null-space descent direction. In this last case, both primal and dual iteration points are available from the beginning of the normal phase, for every iteration and if required. Thus, with a global viewpoint using both primal and dual information, each new sagitta method adopts a different order of primal and dual iterations. Such methods generalize the simplex methodology and can be implemented using reduced or projected gradient techniques. An illustrative sample of numerical and graphical results obtained for a subset of NETLIB problems is included.

Key words: linear programming, non-simplex active-set method
1991 MSC: 90C05, 90C60

* Corresponding author.
Email addresses: santos@ctima.uma.es (Ángel Santos-Palomo), pablito@ctima.uma.es (Pablo Guerrero-García).
URL: http://www.satd.uma.es/matap/personal/pablito/ (Pablo Guerrero-García).

1 Introduction

We consider the usual unsymmetric primal-dual pair of linear programs using a non-standard notation (we have deliberately exchanged the usual roles of b and c, x and y, n and m, and (P) and (D), as in e.g. [15, §2]):

$$\begin{array}{lll} (P) & \min \ell(x) \doteq c^T x \ , \ x \in \mathbb{R}^n & (D) \quad \max \mathcal{L}(y) \doteq b^T y \ , \ y \in \mathbb{R}^m \\ & \text{s.t} \ \ A^T x \geq b & \text{s.t} \ \ Ay = c \quad , \ y \geq \mathbf{0} \end{array}$$

where $A \in \mathbb{R}^{n \times m}$ with $m \geq n$. The condition $c \neq \mathbf{0}$ is added. (In this work the dimension of the null vector $\mathbf{0}$ depends on the context, and $\| \cdot \|$ denotes the euclidean vector norm.) The importance of this problem is sufficiently well-known.

The original sagitta method for solving the primal P presented by Santos-Palomo in [23] is a non-simplex active-set method that starts without any iteration point, selects successive foreactive or active sets (in [23] the foreactive-set term is used because initially the constraints in this set do not have to be active constraints) and, as long as possible, computes corresponding null-space descent directions, because the method attempts to find a descent direction of the primal feasible region. An iteration point is computed if, for the current foreactive set, there is not a corresponding null-space descent direction. Then, while violated constraints exist, a primal-feasibility search loop is carried out. When suitable strategies are used, the first primal feasible point obtained by the method is generally an optimal solution; but, if this does not occur, the process restarts. The method convergence is not theoretically guaranteed and it is not possible to rule out the cycling possibility. Nevertheless, in spite of this potential risk, no cycling has been observed so far using suitable addition/exchange strategies.

In previous works [23,10] we have set up both theoretical and practical background for this method. Two sparse techniques have been developed [25,26] that lead to interesting implementations (as a reduced or projected gradient method) of the sagitta method with encouraging computational results [10]. Details for both a dense and a sparse implementation of this method and the computational results obtained by solving several NETLIB problems can be found in [28].

In this paper we present a structural modification of this method which eliminates the restarting and includes different iteration types in an only loop. Thus, the initial phase of the original method is maintained, but once the system $A_{\mathcal{A}}\mu = c$ is compatible, for $A_{\mathcal{A}}$ being the current active-set matrix, and the multiplier vector $y_{\mathcal{A}}$ (solution of this system) can be computed, the method enters a loop in which primal and/or dual iterations, successively or

interminglely, are carried out. A key feature of this loop is that, if it is required, the method can have both a primal point $x^{(j)}$ and a dual point $y^{(j)}$ available. The primal point $x^{(j)}$ is solution of the, generally underdetermined, system $A_{\mathcal{A}}^T x = b_{\mathcal{A}}$, for $b_{\mathcal{A}}$ being the subvector of b with elements b_i for all $i \in \mathcal{A}_j$. The elements of the dual point $y^{(j)}$ are zeros barring those corresponding to the current multiplier vector $y_{\mathcal{A}}$. Hence, the method can adopt a global viewpoint using both primal and dual information, and it can alter the order of the primal and dual iterations. Therefore, with this algorithmic improvement, multiple sagitta methods arise with unexplored possibilities — such an exploration is not an aim of this paper— and with the challenge to be able to guarantee their convergence.

This paper is organized as follows. In section 2 we briefly describe the initial phase and the primal-feasibility search loop of the original sagitta method. In section 3 we establish an algorithm for a loop of successive well-defined dual iterations, whereas in sections 4 and 5 we provide several modifications of the original sagitta method, dispensing with detailed formulae, rules and numerical methods. Computational results obtained with a dense projected gradient implementation by solving a subset of NETLIB problems with both the original and the new methods are supplied in section 6 as an illustrative sample. Finally, a summary with conclusions is presented in section 7.

2 Original sagitta method

The original sagitta method is an active-set method because, trying to determine a subset $\mathcal{A}_*$ of the active set $\mathcal{A}(x^*)$ of active constraints at an optimal solution x^* for the primal P, it works with a sequence of candidate sets $\mathcal{A}_j$, $\mathcal{A}_j \subseteq \{1,\ 2,\ \ldots,\ m\}$. Any set $\mathcal{A}_j$ defined by the algorithm with $\mid \mathcal{A}_j \mid > 0$ is a set of constraint indices with the property that the normals of the constraints in $\mathcal{A}_j$ are linearly independent, i.e. the submatrix $A_{\mathcal{A}}$ whose columns are the columns a_i of A for all $i \in \mathcal{A}_j$ is of full column rank. But, as $\mathcal{A}_j$ is not required to contain n indices, this sagitta method is a *non-simplex active-set method* (see definition in [8, §8.5.4]). Note that, then, it needs to work with non-square matrices, so numerical methods are needed to solve under- and over-determined systems.

Next, in accordance with the aim of this paper, this original sagitta method is described briefly and without detailed computational formulae, addition/deletion rules nor implementation techniques. Since all *foreactive-set or active-set matrices* $A_{\mathcal{A}}$ are of full column rank, the wording of the algorithm is simplified using statements such as:

- Select $p \in \mathcal{C}$ to be added to $\mathcal{A}_j$; $\mathcal{A}_{j+1} \leftarrow \mathcal{A}_j \cup \{p\}$.

- Select $p \in \mathcal{C}$ to add to $\mathcal{A}_j$.

The use of the first statement requires that the selection rule of p is such that the property of $\mathcal{A}_{j+1}$ is maintained, whereas it is not so using the second one and, then, the algorithm has to specify later what to do whether $a_p \in \mathcal{R}(A_{\mathcal{A}})$ or whether $a_p \notin \mathcal{R}(A_{\mathcal{A}})$, where $\mathcal{R}(A_{\mathcal{A}})$ is the range-space of $A_{\mathcal{A}}$.

The **initial phase** of the sagitta method put forward in [23] can be theoretically thought of as an attempt to determine if the second of the alternative propositions of Farkas' Lemma holds.

Lemma 1 *(Farkas' Lemma) Let $A \in \mathbb{R}^{n\times m}$ and $c \in \mathbb{R}^n$, then exacly one of the following propositions must be true, or*

$$Ay = c \qquad \textit{for some} \qquad y \geq \mathbf{0}$$

or else there exists a vector $d \in \mathbb{R}^n$ verifying

$$A^T d \geq \mathbf{0} \qquad \textit{and} \qquad c^T d < 0. \tag{1}$$

Then, starting with the "arrow" $-c$ as initial descent direction and $\mathcal{A}_1 = \varnothing$ (which can be considered an usual *cold start*), our active-set method computes successive null-space descent directions $d^{(j)}$ by solving in the jth step the constrained underdetermined system

$$A_{\mathcal{A}}^T d = \mathbf{0} \qquad \text{subject to} \qquad c^T d < 0 \tag{2}$$

where the foreactive-set matrix $A_{\mathcal{A}}$, corresponding to the current foreactive set $\mathcal{A}_j$, is of full column rank. Since a solution of (1) is pursued, the contrary constraints to $d^{(j)}$ (i.e., those that $a_i^T d^{(j)} < 0$) are candidate constraints to be added to $\mathcal{A}_j$. If (2) has no solution then the corresponding system $A_{\mathcal{A}}\mu = c$ is compatible even though its solution $y_{\mathcal{A}}$ does not always satisfy the inequality $y \geq \mathbf{0}$ (i.e., dual feasibility).

Therefore, the original sagitta method with this initial phase, without the details of its primal-feasibility search loop, is the following:

Original Sagitta Method

$j \leftarrow 1;\ \mathcal{A}_1 \leftarrow \varnothing;\ d^{(1)} \leftarrow -c\ (\ c \neq \mathbf{0})$
While $\varnothing \neq \mathcal{C} \leftarrow \{\, i \notin \mathcal{A}_j \mid\ a_i^T d^{(j)} < 0 \,\}$ **Do**
  Select $\mathcal{P} \subseteq \mathcal{C}$ to be added to $\mathcal{A}_j$; $\mathcal{A}_{j+1} \leftarrow \mathcal{A}_j \cup \mathcal{P}$.
  If there is not a null-space descent direction **Then**
    Primal-Feasibility Search Loop

If the optimality conditions are satisfied **Then Stop**
Else Select $\mathcal{Q} \subseteq \mathcal{A}_j$ to delete; $\mathcal{A}_{j+1} \leftarrow \mathcal{A}_j \backslash \mathcal{Q}$; **Endif**
Endif
Determine a null-space descent direction $d^{(j+1)}$; $j \leftarrow j+1$.
Endwhile
Stop *(Problem P has no solution or the objective is unbounded below).*

Disregarding a restarting event, this initial phase of the sagitta method is a loop that ends up with a descent direction d which is also a *direction of the primal feasible region* (see definition in [15, p. 48], in [9, p. 82] or in [3, p. 58]) and, then, it is shown [23] that the primal problem P has no solution or the objective is unbounded below, or else it ends because there is not a corresponding null-space descent direction. Anyway, this first execution of the initial phase ends after at most n additions to the foreactive set.

The **primal-feasibility search loop** of the sagitta method is a loop that corresponds to an execution of successive primal iterations when there is not a null-space descent direction, searching for primal feasibility but, in this method, with such search conditioned to maintain dual feasibility if it is eventually achieved. Note that, to avoid clutter, we describe this loop with regard to primal points, where

$$r_i(x^{(j)}) = a_i^T x^{(j)} - b_i$$

and we get rid of the details of the corresponding primal and dual changes and the numerical methods used.

Primal-Feasibility Search Loop

Determine a solution $x^{(j)}$ of the system $A_{\mathcal{A}}^T x = b_{\mathcal{A}}$.
While $\varnothing \neq \mathcal{V}_P \leftarrow \{i \notin \mathcal{A}_j \mid \ r_i(x^{(j)}) < 0\}$ **Do**
Select $p \in \mathcal{V}_P$ to add to $\mathcal{A}_j$.
If $a_p \notin \mathcal{R}(A_{\mathcal{A}})$ **Then** $\mathcal{A}_{j+1} \leftarrow \mathcal{A}_j \cup \{p\}$;
Else
Compute the solution $\delta_{\mathcal{A}}$ of the system $A_{\mathcal{A}}\eta = a_p$.
If $\delta_{\mathcal{A}} \leq \mathbf{0}$ **Then Stop** *(Problem P has no feasible solution).*
Select $q \in \mathcal{A}_j$ to be exchanged for p; $\mathcal{A}_{j+1} \leftarrow \mathcal{A}_j \backslash \{q\} \cup \{p\}$.
Endif
$j \leftarrow j+1$.
Determine a solution $x^{(j)}$ of the new system $A_{\mathcal{A}}^T x = b_{\mathcal{A}}$.
Endwhile

We note that, in this primal iteration, an incoming index to the active set is selected first and, later and only if required, a leaving index. We also point out that a selection rule conditioned with an alternative statement is all we need for different types of primal iterations to be able to exist in the same loop. For

example, it can be desirable in a practical implementation that the leaving index is selected using different rules depending on whether dual feasibility has been already achieved or not.

Finally, a **restarting event** occurs in the original sagitta method if the latter loop ends because a primal feasible point is reached but the optimality condition $y \geq \mathbf{0}$ is not satisfied by the associated multiplier vector. Then, a single (or multiple) constraint deletion makes possible the method restart with a *warm start*: the foreactive set without the deleted constraint(s) and the corresponding solution of (2) as descent direction.

The key point is that a restarting event entails, in general, that the primal feasibility achieved is lost and that, possibly for several iterations, the corresponding system $A_{\mathcal{A}}\mu = c$ becomes incompatible. To rectify the latter we shall incorporate in the sagitta methodology the use of dual iterations which, in its search for dual feasibility, always maintain the compatibility of the current system $A_{\mathcal{A}}\mu = c$.

3 Dual-feasibility search loop

Once primal feasibility has been achieved, an usual dual iteration in an active-set method consists of selecting a leaving constraint, determining a, usually null-space, descent search direction and, finally, computing the next primal feasible point with the maximum step along the direction down to an incoming constraint (see [8, pp. 380–381]) in such a way that primal feasibility is maintained.

On the other hand, if a restarting event occurs in the original sagitta method, we note that a *constraint deletion* makes possible the method restart with an initial phase that works without any iteration point, but with a *descent direction* and an *addition rule* to select an incoming constraint from amongst the contrary constraints to such direction. Therefore, we have the essential elements of a dual iteration available and so, although we dispense with the maintenance of primal feasibility, we propose a modification including dual iterations whose main feature is the maintenance of the compatibility of the system $A_{\mathcal{A}}\mu = c$.

We describe such a dual iteration just as it often occurs, namely as the body of a dual-feasibility search loop; to avoid clutter again, its description is carried out with regard to dual points and we get rid of the details of the changes for the corresponding primal points and the numerical methods used.

Dual-Feasibility Search Loop

Compute the solution $y_{\mathcal{A}}$ of the compatible system $A_{\mathcal{A}}\mu = c$ and the corresponding dual point $y^{(j)}$.
While $\varnothing \neq \mathcal{V}_D \leftarrow \{i \in \mathcal{A}_j \mid \ y_i^{(j)} < 0\}$ **Do**
Select $q \in \mathcal{V}_D$, and such that $q = \mathcal{A}_j(k)$, to be deleted from $\mathcal{A}_j$.
Determine a null-space descent direction $\tilde{d}$ as if q had been already deleted.
If $\varnothing = \mathcal{C} \leftarrow \{i \notin \mathcal{A}_j \mid \ a_i^T \tilde{d} < 0\}$ **Then**
Stop *(Problem P has no solution or the objective is unbounded below)*
Endif
Select $p \in \mathcal{C}$ to be added to $\mathcal{A}_j$.
If $a_p \notin \mathcal{R}(A_{\mathcal{A}})$ **Then**

$$\mathcal{A}_{j+1} \leftarrow \mathcal{A}_j \cup \{p\};\ y^{New} \leftarrow \begin{bmatrix} y_{\mathcal{A}} \\ 0 \end{bmatrix}.$$

Else
Compute the solution $\delta_{\mathcal{A}}$ of the system $A_{\mathcal{A}}\eta = a_p$ and the corresponding vector $\delta^{(j)}$ (its elements are zeros barring those corresponding to $\delta_{\mathcal{A}}$).

$$\tau \leftarrow y_q^{(j)}/\delta_q^{(j)};\ y^{New} \leftarrow \begin{bmatrix} \bar{y}_{\mathcal{A}} \\ 0 \end{bmatrix} + \tau \begin{bmatrix} -\bar{\delta}_{\mathcal{A}} \\ 1 \end{bmatrix}$$ for $\bar{y}_{\mathcal{A}}$ and $\bar{\delta}_{\mathcal{A}}$ respectively the vectors $y_{\mathcal{A}}$ and $\delta_{\mathcal{A}}$ without their corresponding kth element.
$\mathcal{A}_{j+1} \leftarrow \mathcal{A}_j \backslash \{q\} \cup \{p\}$.
Endif
$j \leftarrow j + 1$; $y_{\mathcal{A}} \leftarrow y^{New}$ ($y^{(j)}$ is the updated dual point).
Endwhile

We point out that a leaving constraint is selected first and later an incoming one, in both an usual and our dual iteration; the difference is that sometimes in our dual iteration the leaving constraint is not deleted and that the maintenance of a previously achieved primal feasibility is not regarded as a priority. Moreover, as in a primal iteration, a selection rule conditioned with an alternative statement is all we need for different types of dual iterations to be able to exist in the same loop. For example, it can be desirable in a practical implementation that the incoming index is selected using different rules depending on whether primal feasibility has been already achieved or not.

Summing up, the main modification that we have sketched in this section makes possible that a modified sagitta method can adopt a global viewpoint using both primal and dual information and that it can alter the order between primal and dual iterations.

4 Immediate modification

An immediate modification of the original sagitta method consists of the restarting elimination and the proper inclusion of primal and dual iterations in an unique normal phase of the new method. The modified saggita method is described next, without detailing those items already described in previous sections: **initial phase**, **primal iteration** and **dual iteration**.

Modified Sagitta Method

Initial phase searching for an active set $\mathcal{A}_j$ such that $A_{\mathcal{A}}\mu = c$ is compatible.
Compute $x^{(j)}$ solution of $A_{\mathcal{A}}^T x = b_{\mathcal{A}}$, $y_{\mathcal{A}}$ solution of $A_{\mathcal{A}}\mu = c$ and the corresponding dual point $y^{(j)}$.
$\mathcal{V}_P \leftarrow \{i \notin \mathcal{A}_j \mid \ r_i(x^{(j)}) < 0\}$; $\mathcal{V}_D \leftarrow \{i \in \mathcal{A}_j \mid \ y_i^{(j)} < 0\}$.
While $\mathcal{V}_P \neq \varnothing$ or $\mathcal{V}_D \neq \varnothing$ **Do**
  If $\mathcal{V}_P \neq \varnothing$ **Then**
    If $\mathcal{V}_D \neq \varnothing$ **Then**
      Primal iteration without dual feasibility achieved.
    Else
      Primal iteration maintaining dual feasibility already achieved.
    Endif
  Else
    Dual iteration.
  Endif
  $\mathcal{V}_P \leftarrow \{i \notin \mathcal{A}_j \mid \ r_i(x^{(j)}) < 0\}$; $\mathcal{V}_D \leftarrow \{i \in \mathcal{A}_j \mid \ y_i^{(j)} < 0\}$.
Endwhile

Note that, depending on the dual feasibility status, two types of primal iteration are used. Hence we state that a leaving constraint can be selected with a conditioned rule, as it frequently occurs in a practical implementation of the methods (see [28]).

Clearly, this modification of the sagitta method is essentially structural; nevertheless, the key point is that the paths traversed by both, modified and original, sagitta methods can be different even if all formulae and selection rules were the same (see Figure 1). For this situation to happen it suffices that a dual iteration of the modified method ends up with a constraint addition and a primal infeasible point, thus this method goes on with a primal iteration; on the other hand, the restarted initial phase of the original sagitta method only modifies the descent direction (no vanishing) and stays in this phase until such direction is eventually vanished.

The formulae for the multiplier vector updating are the same in both a primal iteration and a dual iteration (see §3) and then, as we establish in the following

lemma, the variation of the objective value, after any iteration and throughout the normal phase, can be easily computed.

Lemma 2 *Let $\mathcal{A}_j$ be the current active-set in a primal or dual iteration throughout the normal phase of the modified sagitta method, $A_{\mathcal{A}}$ be the corresponding active-set matrix, and p and $q = \mathcal{A}_j(k)$ be the incoming and leaving indices, respectively. Then, the variation of the objective value is*

$$\ell(x^{(j+1)}) - \ell(x^{(j)}) = \begin{cases} 0 & \text{if an addition occurs,} \\ -\tau r_p(x^{(j)}) & \text{if an exchange occurs,} \end{cases}$$

where $\tau \leftarrow y_q^{(j)}/\delta_q^{(j)}$.

Note that complementary slackness holds in this normal phase and, when $|\ \mathcal{A}_j\ | < n$ and an exchange occurs, the objective value $\ell(x^{(j+1)})$ is the same for every primal point $x^{(j+1)}$ which can be computed as solution of the undetermined system

$$\begin{bmatrix} \bar{A}_{\mathcal{A}}^T \\ a_p^T \end{bmatrix} x = \begin{bmatrix} \bar{b}_{\mathcal{A}} \\ b_p \end{bmatrix},$$

where $\bar{A}_{\mathcal{A}}$ is the matrix $A_{\mathcal{A}}$ without its kth column and $\bar{b}_{\mathcal{A}}$ is the vector $b_{\mathcal{A}}$ without their corresponding kth element.

Classical rules in primal and dual iterations are, for example, the following:

Classical Rules in Primal Iteration

Choose incoming constraint:	$p = \arg\min \left\{ r_i(x^{(j)}) \mid r_i(x^{(j)}) < 0,\ i \notin \mathcal{A}_j \right\}$
Choose leaving constraint:	$q = \mathcal{A}_j(k) = \arg\min \left\{ \dfrac{y_i^{(j)}}{\delta_i^{(j)}} \mid \delta_i^{(j)} > 0,\ i \in \mathcal{A}_j \right\}$

Classical Rules in Dual Iteration

Choose leaving constraint:	$q = \mathcal{A}_j(k) = \arg\min \left\{ y_i^{(j)} \mid y_i^{(j)} < 0,\ i \in \mathcal{A}_j \right\}$
Choose incoming constraint:	$p = \arg\min \left\{ \dfrac{r_i(x^{(j)})}{-a_i^T d^{(j)}} \mid a_i^T d^{(j)} < 0,\ i \notin \mathcal{A}_j \right\}$

However, different rules can be devised to determine the incoming constraint, namely a normalized criterion in primal iteration and a most-obtuse-angle criterion in dual iteration:

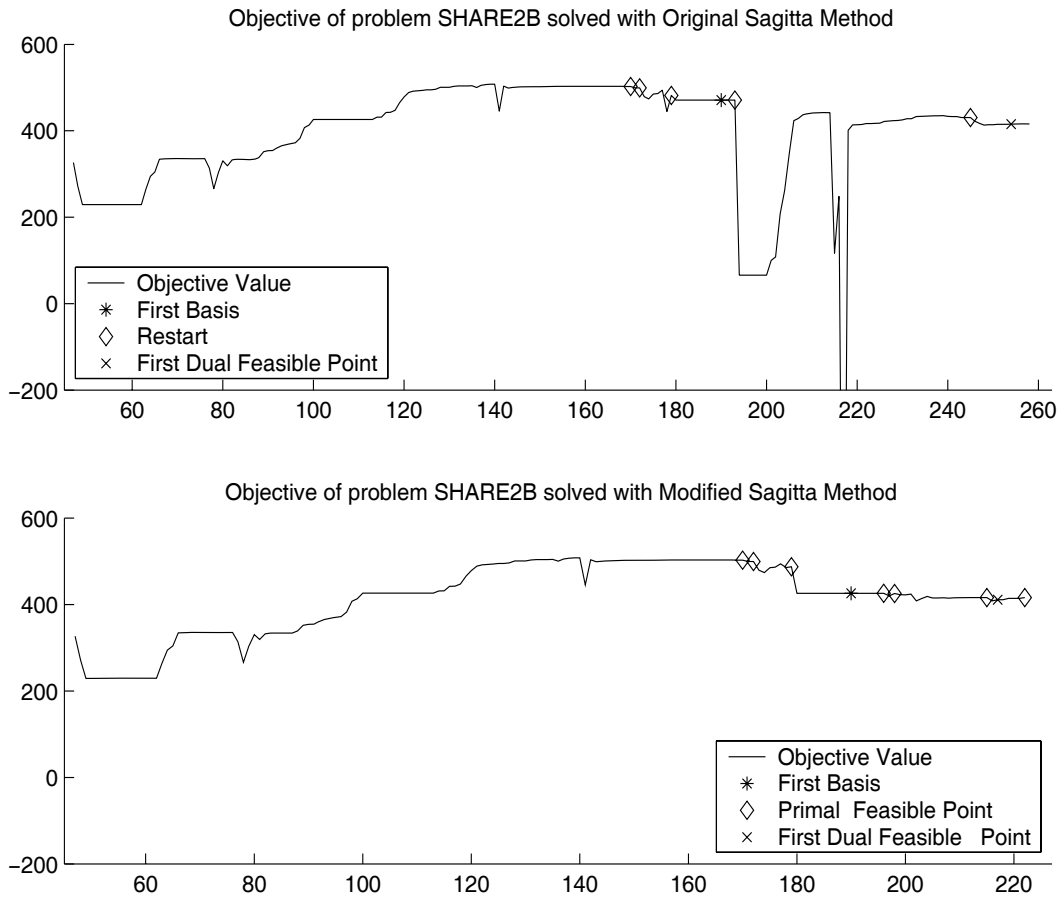

Fig. 1.

Sagitta Rules in Primal Iteration

Choose incoming constraint:	$p = \arg\min \left\{ \dfrac{r_i(x^{(j)})}{\parallel a_i \parallel} \mid r_i(x^{(j)}) < 0,\ i \notin \mathcal{A}_j \right\}$
Choose leaving constraint:	$q = \mathcal{A}_j(k) = \arg\min \left\{ \dfrac{y_i^{(j)}}{\delta_i^{(j)}} \mid \delta_i^{(j)} > 0,\ i \in \mathcal{A}_j \right\}$

Sagitta Rules in Dual Iteration

Choose leaving constraint:	$q = \mathcal{A}_j(k) = \arg\min \left\{ y_i^{(j)} \mid y_i^{(j)} < 0,\ i \in \mathcal{A}_j \right\}$
Choose incoming constraint:	$p = \arg\min \left\{ \dfrac{a_i^T d^{(j)}}{\parallel a_i \parallel} \mid a_i^T d^{(j)} < 0,\ i \notin \mathcal{A}_j \right\}$

Using these latter rules, the variations of the objective value in successive primal or dual iterations are non-monotonic as long as dual feasibility is not achieved, as can be seen in Figure 1. The independent variable is the iteration counter of the method and only the objective values corresponding to the normal phase are displayed in this graph of the objective function.

This non-monotonicity complicates an eventual theoretic proof that guarantees the convergence of both, original and modified, sagitta methods.

5 Other modifications

It is well-known that the usual way to assure a monotonic variation of the objective value in simplex methodology is to maintain a feasibility previously achieved. Then, to solve a given linear program, a frequently used resort is a two-phase algorithmic scheme, else formulating an auxiliary linear programming problem (usually called a Phase-I linear program) with additional (called artificial) variables and/or constraints —but in such a way that an obvious feasible point exists with respect to the Phase-I constraints— (see, for example, [6, §8.4],[8, §8.6.3,§8.6.4]), or else working with a non-monotonic Phase-I without artificial variables (see, for example, [14], [16,19] and the references therein or the criss-cross methods [32,5]). This friendly resort has been proposed to be used with different non-simplex active-set methods, else with a different Phase-I linear program (see, for example, [8, §7.9,§8.6], [17,18,20,27,10]) or else, working with a non-monotonic Phase-I without artificial variables [11,12].

Bearing in mind that a sagitta method works without artificial variables, other immediate modification of the sagitta method consists in firstly carrying out successive dual iterations until dual feasibility is achieved and, then, successive primal iterations maintaining the dual feasibility already achieved. Different methods can have such an algorithmic scheme, in accordance with the rules used, and we have called them *dual-then-primal* methods in [11]. Its normal phase is as follows:

Normal Phase of a Dual-then-Primal Sagitta Method

While $\mathcal{V}_P \neq \varnothing$ or $\mathcal{V}_D \neq \varnothing$ **Do**
  If $\mathcal{V}_D \neq \varnothing$ **Then**
    Dual iteration.
  Else
    Primal iteration with maintenance of dual feasibility already achieved.
  Endif
  $\mathcal{V}_P \leftarrow \{i \notin \mathcal{A}_j \mid \ r_i(x^{(j)}) < 0\}$; $\mathcal{V}_D \leftarrow \{i \in \mathcal{A}_j \mid \ y_i^{(j)} < 0\}$.
Endwhile

The use of a min-ratio rule is required to maintain feasibility, but numerical difficulties are usual and, as it is sufficiently well-known, an anticycling rule can be convenient in practice. Anyway, we prefer a normal phase with an unique loop to the (apparently equivalent) scheme with two successive feasibility loops because in this way the normal phase includes the possibility of a "restarting

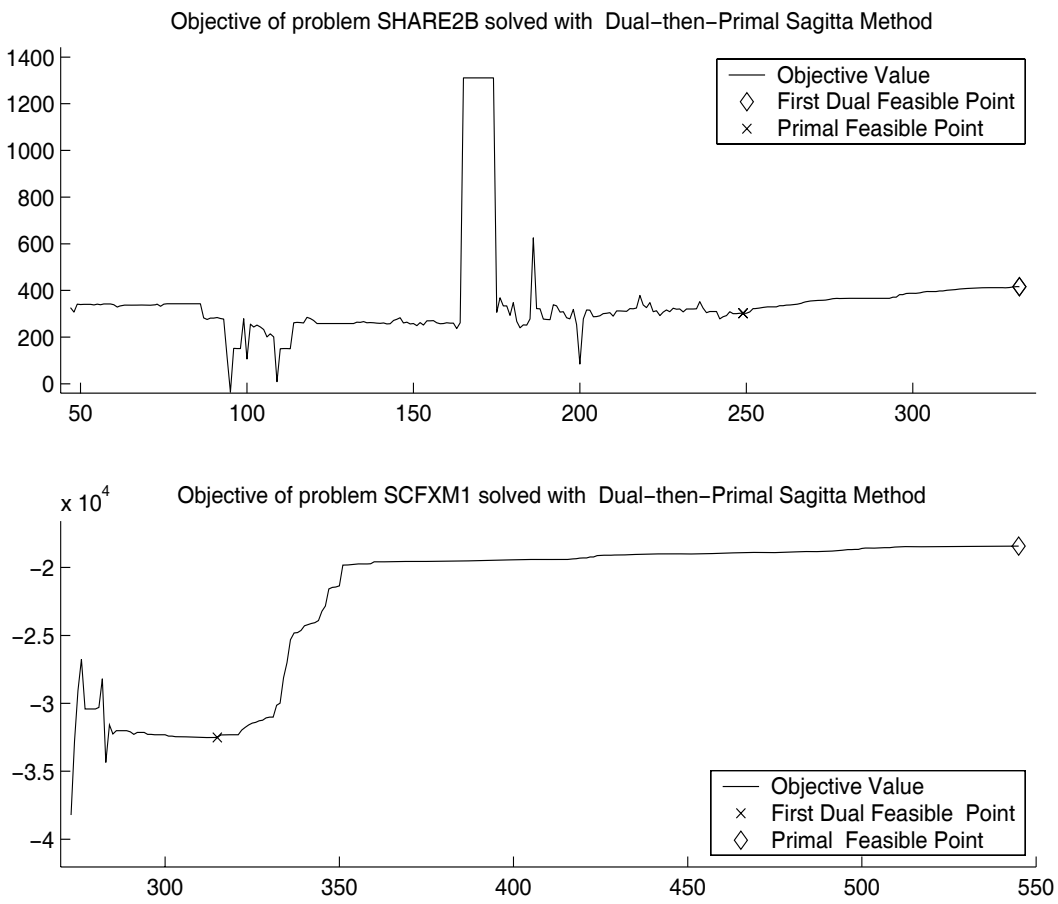

Fig. 2.

event" if, because of rounding errors, a primal iteration does not maintain the dual feasibility already achieved. A similar device is used in practical simplex implementations (see [31, §2.3.2]).

The objective functions displayed in Figure 2 illustrate the different performance, most likely because of problem specific features, of the successive blocks of dual and primal iterations when a simple rule (in this case the most-obtuse-angle rule) is used to choose the incoming constraint. Note that a dual-then-primal sagitta method works with the original linear program through the whole process and thus, in its first block of successive iterations, a global viewpoint can be adopted by combining the work of achieving feasibility with the work of achieving optimality. This often results in fewer iterations —even if the computational cost per iteration can be greater than in classical methods— and it is very likely that this combination avoids, at least sometimes, that the former block of successive dual iterations ends up with a highly-degenerated dual feasible point, hence also avoiding that a stalling or cycling event is possible in the later block of successive primal iterations while maintaining the dual feasibility already achieved. A more detailed analysis can be found in [29].

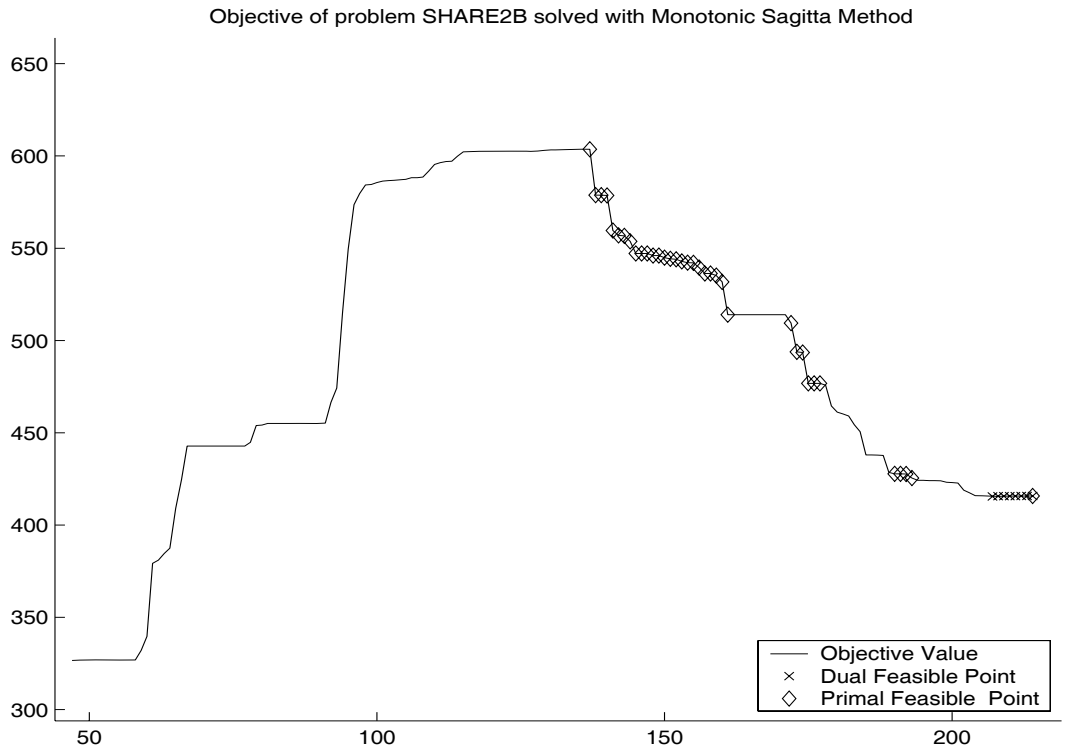

Fig. 3.

Other feature, with yet unexplored possibilities, of the sagitta methodology is that, without feasibility maintenance but a suitable dynamic change of the type of iteration bearing Lemma 2 in mind, a monotonic variation of the objective value [30] can be obtained. The challenge is to determine global strategies with piecewise monotonic variations of the objective value in the normal phase, in such a way that the convergence of the sagitta method is guaranteed. It seems that, apparently at least, it is more convenient to determine firstly a primal feasible point and then to carry out a loop of dual iterations with non-increasing variations of the objective value and without a mandatory primal feasibility maintenance. We show, for example, in Figure 3 the objective values obtained using a sagitta method with three stages corresponding to:

- A first loop of primal iterations with monotonic non-decreasing variations of the objective value.
- A second loop of dual iterations with monotonic non-increasing variations of the objective value. Note that, as it is observed in Figure 3, they can do without primal feasibility maintenance.
- Only if it is necessary and maintaining dual feasibility in such case, a final loop of primal iterations with monotonic non-decreasing variations of the objective value.

The sagitta method just described [30] has an algorithmic trend different from that of the monotonic build-up simplex variants of Anstreicher and Terlaky

[1]. The adaptations to the non-simplex case of either this monotonic build-up rule or the exterior point simplex algorithm [21,22] could provide other sagitta modifications of interest. Finally, we do not want to fail to notice that a proper use of the "local information" supplied by a primal feasible point ought to be considered too [24], and that the availability of different strategies for solving different type of problems can be advantageous.

6 Computational results

We have carried out a dense implementation of the original and the modified methods, using projected gradient techniques based on an orthogonal factorization (obtained with the classical Gram-Schmidt method with reorthogonalization [2, §2.4]) of the foreactive- or active-set matrix (see details in [28]).

The linear programs for which a restarting event occurs in the original sagitta method are rare. However, in order to test the method performance, we have selected the duals of 22 problems —in which neither BOUNDS nor RANGES sections occur— of the sufficiently well-known NETLIB library [7]. All test problems have been solved **without scaling nor preprocessing**; they have been read as linear programs in standard form and then dualized to obtain a primal problem P [10, §5.3]. Additionally, to be used as a reference, we supply the corresponding CPU times in seconds for the MATLAB code `linprog` of the Optimization Toolbox release 2.2, solving the same subset of NETLIB problems with the small and medium scale option that, in accordance with the documentation of the distributor, is a variation of the simplex method.

The computational results were obtained with an Intel Pentium III at 1.08GHz with 128MB RAM, using MATLAB release 13 and **interpreted** code, at least with regard to our source codes.

Code `linprog` has a large-scale option under which the interior-point compiled code LIPSOL is called instead. While we have developed two sparse techniques that lead to interesting implementations of the sagitta methods [25,26], a comparison of computational results using such large-scale option would not be fair because our source code is interpreted. Moreover, it is well-known [13, p. 10] that simplex methods outperform interior-point ones when the problem is not large enough.

In the tables below we shall use a first column labeled # to hold a number to recognize each NETLIB problem. This number was assigned to each NETLIB problem by Bixby in [4], according to the number of its nonzeros. The name of each NETLIB problem solved is also given in Tables 1 and 5.

TABLE 1. Computational results for the MATLAB *code* `linprog`.

#	*Name*	*n*	*m*	*Optimal value*	*Iter*	*Time*	*MinRes*
1	AFIRO	27	51	4.647531428571E+2	33	0.16	-3.4E-15
2	SC50B	50	78	6.999999999998E+1	48	0.44	-4.7E-13
3	SC50A	50	78	6.457507705856E+1	49	0.44	-4.6E-16
4	SC105	105	163	5.220206121128E+1	145	4.29	-6.3E-12
6	ADLITTLE	56	138	-2.254949631624E+5	116	1.32	-5.8E-11
7	SCAGR7	129	185	2.331389823918E+6	269	17.42	-1.7E-06
8	STOCFOR1	117	165	4.113197621944E+4	134	3.46	-5.1E-13
9	BLEND	74	114	3.081214983291E+1	105	5.32	-2.6E-09
10	SC205	205	317	5.220206121083E+1	306	34.77	-1.2E-11
12	SHARE2B	96	162	4.157322407409E+2	163	3.24	-4.5E-10
14	LOTFI	153	366	2.500931312371E+1	339	35.26	-2.2E-05
15	SHARE1B	117	253	7.658931857918E+4	241	8.63	-4.8E-12
17	SCORPION	388	466	-1.878124822738E+3	401	316.71	-2.2E-12
19	SCAGR25	471	671	(*)			
20	SCTAP1	300	660	-1.412250000000E+3	692	686.34	-1.1E-12
22	BRANDY	220	303	-1.518509896488E+3	754	585.40	-1.3E-09
23	ISRAEL	174	316	8.966448212766E+5	445	29.77	-2.1E-07
26	SCSD1	77	760	-8.666666674333E+0	109	4.72	-3.8E-15
28	AGG	488	615	(*)			
29	BANDM	305	472	(*)			
30	E226	223	472	(*)			
31	SCFXM1	330	600	(*)			

(*) Note: The problem is badly conditioned; the solution may not be reliable.

Table 1 sums up —along with the Bixby's number, the name, the number n of variables and the number m of constraints of each NETLIB problem— the computational results obtained using MATLAB code `linprog`. Total number of iterations and running time required to solve each problem are displayed in two columns labeled *Iter* and *Time*, along with two additional columns (labeled *Optimal value* and *MinRes*, respectively) with the computed optimal value of the objective and the minimum element of the residual vector at the optimal solution obtained. The displayed note issued by the code `linprog`,

which warns about its difficulties to solve five NETLIB problems of medium size, is incorporated in Table 1 as a footnote.

TABLE 2. Computational results for the Original Sagitta Method solving NETLIB *problems and using Most-Obtuse-Angle rule at start*

#	*Optimal value*	*Iter*	*Time*	*MinRes*	$\mid \mathcal{A}_* \mid$	*IPh*	*RS*	*%Itb*
1	4.647531428571E+2	23	0.06	-1.8E-15	20	7	0	0.0
2	7.000000000000E+1	67	0.22	0.0E+00	48	5	0	0.0
3	6.457507705856E+1	64	0.22	-1.2E-16	49	17	0	0.0
4	5.220206121171E+1	141	1.15	-1.9E-16	104	33	0	0.0
6	-2.254949631624E+5	153	0.93	-3.8E-12	56	39	0	41.8
7	2.331389824331E+6	188	2.64	-6.2E-12	129	113	3	16.5
8	4.113197621943E+4	127	0.82	-3.1E-13	117	98	0	8.7
9	3.081214984583E+1	127	0.66	0.0E+00	74	8	0	28.3
10	5.220206121171E+1	313	10.00	-6.5E-17	203	60	0	0.0
12	4.157322407415E+2	258	2.20	-2.8E-12	96	47	5	26.7
14	2.526470606237E+1	313	7.25	-9.4E-15	153	85	0	29.4
15	7.658931857919E+4	228	3.02	-9.8E-11	117	117	0	49.1
17	-1.878124822738E+3	383	18.84	-1.1E-12	336	260	1	0.0
19	1.475343306077E+7	757	95.19	-7.5E-12	448	419	0	0.0
20	-1.412250000000E+3	468	29.11	-1.1E-11	278	223	7	0.0
22	-1.518509896488E+3	489	13.79	-1.5E-13	170	159	0	0.0
23	8.966448218631E+5	401	10.66	-1.7E-11	174	171	0	25.9
26	-8.666666674333E+0	123	1.70	-1.3E-08	77	7	0	14.6
28	3.599176728654E+7	574	56.19	-2.6E-09	486	459	1	0.0
29	1.586280184509E+2	783	50.59	-1.9E-13	303	288	0	0.0
30	1.875192906631E+1	808	32.40	-2.4E-14	213	139	0	0.0
31	-1.841675902835E+4	558	38.12	-1.0E-12	320	273	0	0.0

TABLE 3. Computational results for the Modified Sagitta Method solving NETLIB *problems and using Most-Obtuse-Angle rule at initial phase*

#	*Optimal value*	*Iter*	*Time*	*MinRes*	$\mid \mathcal{A}_* \mid$	*IPh*	*%Itb*
1	4.647531428571E+2	23	0.06	-1.8E-15	20	7	0.0
2	7.000000000000E+1	67	0.17	0.0E+00	48	5	0.0
3	6.457507705856E+1	64	0.22	-1.2E-16	49	17	0.0
4	5.220206121171E+1	141	1.16	-1.9E-16	104	33	0.0
6	-2.254949631624E+5	153	1.54	-3.8E-12	56	39	41.8
7	2.331389824331E+6	188	2.52	-8.9E-13	129	113	16.5
8	4.113197621943E+4	127	0.77	-3.1E-13	117	98	8.7
9	3.081214984583E+1	127	1.27	0.0E+00	74	8	28.3
10	5.220206121171E+1	313	9.62	-6.5E-17	203	60	0.0
12	4.157322407414E+2	222	1.81	-3.0E-13	96	47	14.9
14	2.526470606237E+1	313	6.76	-9.4E-15	153	85	29.4
15	7.658931857919E+4	228	2.92	-9.8E-11	117	117	49.1
17	-1.878124822738E+3	383	16.48	-4.7E-13	336	260	0.0
19	1.475343306077E+7	757	85.58	-7.5E-12	448	419	0.0
20	-1.412250000000E+3	468	24.39	-1.5E-10	278	223	0.0
22	-1.518509896488E+3	489	13.08	-1.5E-13	170	159	0.0
23	8.966448218631E+5	401	10.32	-1.7E-11	174	171	25.9
26	-8.666666674333E+0	123	1.43	-1.3E-08	77	7	14.6
28	3.599176728661E+7	574	48.93	-6.2E-12	486	459	0.0
29	1.586280184509E+2	783	47.45	-1.9E-13	303	288	0.0
30	1.875192906631E+1	808	29.27	-2.4E-14	213	139	0.0
31	-1.841675902835E+4	558	33.67	-1.0E-12	320	273	0.0

Tables 2–4 sum up the computational results obtained for the original, modified and dual-then-primal sagitta method, using the same initial phase with the *most-obtuse-angle rule* to choose the incoming constraint. In these tables, apart from the five columns labeled #, *Optimal value*, *Iter*, *Time* and *MinRes* (displaying the Bixby's number and the corresponding computational results obtained with the respective method), other columns with additional numerical information of interest are included, on which we comment now:

TABLE 4. Computational results for Dual-then-Primal Sagitta Method solving NETLIB *problems and using Most-Obtuse-Angle rule at initial phase*

#	*Optimal value*	*Iter*	*Time*	*MinRes*	$\mid \mathcal{A}_* \mid$	*IPh*	*Itd*	*Itp*
1	4.647531428571E+2	23	0.11	-7.4E-32	20	7	0	16
2	7.000000000000E+1	64	0.27	0.0E+00	48	5	0	59
3	6.457507705856E+1	64	0.21	-5.1E-32	49	17	0	47
4	5.220206121171E+1	143	1.16	-6.2E-32	104	33	0	110
6	-2.254949631624E+5	163	1.04	-2.4E-13	56	39	12	112
7	2.331389824331E+6	200	2.59	-8.9E-13	129	113	51	36
8	4.113197621944E+4	127	0.71	-3.1E-13	117	98	0	29
9	3.081214984583E+1	127	0.77	0.0E+0	74	8	0	119
10	5.220206121171E+1	310	8.46	-3.5E-32	203	60	0	250
12	4.157322407414E+2	332	3.13	-2.3E-14	96	47	202	83
14	2.526470606188E+1	292	6.37	-5.0E-16	153	85	6	201
15	7.658931857919E+4	243	3.51	-9.8E-11	117	117	46	80
17	-1.878124822738E+3	383	18.07	-4.7E-13	335	260	24	99
19	1.475343306077E+7	678	76.40	-7.5E-12	449	419	108	151
20	-1.412250000000E+3	788	43.39	-6.0E-13	277	223	469	96
22	-1.518509896488E+3	732	19.71	-1.5E-13	170	159	387	186
23	8.966448218630E+5	401	10.44	-4.5E-13	174	171	0	230
26	-8.666666674333E+0	123	1.48	-1.3E-08	77	7	0	116
28	3.599176728658E+7	589	52.95	-7.6E-13	484	459	66	64
29	1.586280184501E+2	744	44.82	-1.9E-13	304	288	136	320
30	1.875192906637E+1	971	36.58	-2.1E-14	215	139	130	702
31	-1.841675902835E+4	545	33.51	-9.9E-14	320	273	42	230

- Column labeled $\mid \mathcal{A}_* \mid$, in Tables 2–4, shows the cardinal of the final active set $\mathcal{A}_*$, subset of the active set $\mathcal{A}(x^*)$ of active constraints at the computed optimal solution x^*. We can check that $\mid \mathcal{A}_* \mid < n$ for 13 out of the 22 problems solved, or in other words, that such problems are solved by sagitta methods working with basis deficiency throughout the whole process.
- Column labeled *IPh*, in Tables 2–4, shows the number of iterations performed in the initial phase, which coincides with the cardinal of the active set at the end of such phase and the beginning of the normal phase. Since

the initial phase has been the same for these three sagitta methods, such numbers coincide in the three tables. Note that only for problem SHARE1B the cardinal of the active set at the end of the initial phase is equal to n.

- Column labeled *RS*, in Table 2, shows the number of restarting events when the original sagitta method is used. The computational results obtained by the original and modified sagitta methods have to coincide if this number is zero, but the results could differ if it is nonzero. These differences are patent for problem SHARE2B, as we showed in Figure 1.
- Column labeled *%Itb*, in Tables 2–3, shows the percentage of square basis iterations, i.e. iterations with $\mid \mathcal{A}_j \mid = n$. It is worth noting that such percentage is zero or less than 50% for all problems solved.
- Columns labeled *Itd* and *Itp*, in Table 4, show the number of dual and primal iterations performed in the normal phase of the dual-then-primal sagitta method. Note that the same initial phase has been implemented for the three sagitta methods and that it obtains a dual feasible point for those problems, nine in total, in which the number in column *Itd* is zero. Note also that, when the initial phase does not end up with a dual feasible point, the number of dual iterations needed to obtain it varies greatly, ranging from relatively small for some problems like, for example, LOTFI and SCFXM1, to significantly large for other problems like, for example, SHARE2B or SCTAP1. The different performance of the successive blocks of dual and primal iterations, most likely because of problem specific features, is illustrated in Figure 2, which shows the variations of the objective value for SHARE2B and SCFXM1, respectively.

The numerical results in Tables 2 and 3 are nearly the same, because the modified sagitta method is, in essence, a structural modification of the original sagitta method. From now on, the modified sagitta method will be considered the definitive version of the sagitta method.

A systematic comparison of the methods based on the ratios of iterations and running time is displayed in Table 5. At a glance, we can observe that the MATLAB code `linprog` does not solve five problems of the set (see table footnote in Table 1) and that, moreover, there are differences in the quality of several of the optimal solutions obtained. Thus, comparing results for problems like SCAGR7, LOTFI and ISRAEL in columns labeled *MinRes*, code `linprog` commits a greater violation of the constraints. As a consequence, the third digit of the objective value computed at the optimal solution of problem LOTFI obtained is not correct.

TABLE 5. Ratios for method comparison

NETLIB *problem*		`Lp`/*Sag*		`Lp`/*DtPSag*		*Sag/DtPSag*	
#	*Name*	*Iter*	*Time*	*Iter*	*Time*	*Iter*	*Time*
1	AFIRO	1.43	2.67	1.43	1.45	1.00	0.55
2	SC50B	0.72	2.59	0.75	1.63	1.05	0.63
3	SC50A	0.77	2.00	0.77	2.10	1.00	1.05
4	SC105	1.03	3.70	1.01	3.70	0.99	1.00
6	ADLITTLE	0.76	0.86	0.71	1.27	0.94	1.48
7	SCAGR7	1.43	6.91	1.35	6.73	0.94	0.97
8	STOCFOR1	1.06	4.49	1.06	4.87	1.00	1.08
9	BLEND	0.83	4.19	0.83	6.91	1.00	1.65
10	SC205	0.98	3.61	0.99	4.11	1.01	1.14
12	SHARE2B	0.73	1.79	0.49	1.04	0.67	0.58
14	LOTFI	1.08	5.22	1.16	5.54	1.07	1.06
15	SHARE1B	1.06	2.96	0.99	2.46	0.94	0.83
17	SCORPION	1.05	19.22	1.05	17.53	1.00	0.91
19	SCAGR25	0.00	0.00	0.00	0.00	1.12	1.12
20	SCTAP1	1.48	28.14	0.88	15.82	0.59	0.56
22	BRANDY	1.54	44.76	1.03	29.70	0.67	0.66
23	ISRAEL	1.11	2.88	1.11	2.85	1.00	0.99
26	SCSD1	0.89	3.30	0.89	3.19	1.00	0.97
28	AGG	0.00	0.00	0.00	0.00	0.97	0.92
29	BANDM	0.00	0.00	0.00	0.00	1.05	1.06
30	E226	0.00	0.00	0.00	0.00	0.83	0.80
31	SCFXM1	0.00	0.00	0.00	0.00	1.02	1.00
Average		1.05	8.19	0.97	6.52	0.95	0.96

Table 5 displays ratios of iterations and time of MATLAB code `linprog` versus our MATLAB codes for modified and dual-then-primal sagitta methods (in columns headed `Lp`/*Sag* and `Lp`/*DtPSag*, respectively), and for both sagitta methods (in column headed *Sag/DtPSag*) too. The iteration results are reported to allow the checking that they are not significantly differents in average, although they are for some specific problems. Nevertheless, since the computational effort involved in a single iteration of the sagitta methods is far

less than that of `linprog`, the performance of the codes should be ranked properly by running times. The time ratios are quite impressive in general, but specially for problems of medium size like SCORPION, SCTAP1 and BRANDY. Summing up (but bearing in mind only those problems solved by `linprog`), the average time ratios (see bottom line in Table 5) are `Lp`/*Sag*= 8.19 and `Lp`/*DtPSag*= 6.52. Therefore, it is clearly established that, solving this subset of NETLIB problems, **both sagitta codes largely outperform linprog**. The comparison between sagitta codes with average time ratio *Sag/DtPSag*= 0.96 allows us to conclude that the modified sagitta is slightly better than the dual-then-primal method.

The average time ratios in Table 5 are more impressive than those obtained by Pan (see Table 6 in [18, p. 47]) for his compiled FORTRAN implementations of the revised two-phase simplex method versus his basis-deficiency-allowing variations of the simplex method. Note that his simplex code solves all the NETLIB problems included in our table, but Pan makes explicit in his paper [18, p. 50] that *the problems were first reduced in size by a preprocessor to remove redundant rows before executing* this code. The use of a preprocessor can change the method performance, but the performance variations for a sagitta method using a suitably adapted preprocessing procedure is a question to be analysed; nevertheless, we have already checked [10, §5] that a scaling of the rows and columns of the constraint matrix generally improves both the iterations and running time of the sagitta methods.

7 Summary

A set of modifications of the original sagitta method has been presented, based upon definitive and clearly established definitions of primal and dual iteration. These definitions allow us to consider the sagitta methodology as a generalization of the simplex methodology because, using a global viewpoint, a sagitta method:

- Starts usually without any iteration point and in its initial phase, bearing the objective in mind, attempts to find a descent direction of the feasible region or, alternatively, a solution of $Ay = c$.
- Works in its normal phase as an active-set method, generally with basis deficiency, i.e. with $\mid \mathcal{A}_j \mid < n$, although it can end up with a square basis, i.e. with $\mid \mathcal{A}_j \mid = n$. Every active-set $\mathcal{A}_j$ has the property that the normals of the constraints in $\mathcal{A}_j$ are linearly independent, and in this phase every system $A_{\mathcal{A}}\mu = c$ is compatible even if $\mid \mathcal{A}_j \mid < n$.
- Works without a previous feasibility achieved and, then, can perform primal and dual iterations, successively or interminglely; moreover, it can accomplish different primal or dual iterations, depending on the addition/exchange

rules used.

Our aim in this paper has been to specify this methodological modification, calling attention to its possibilities. We consider specially interesting the possible existence of efficient and monotonic convergent sagitta methods working without an usual "permanent maintenance" of feasibility, that is to say firstly in Phase-I and later in Phase-II. Furthermore, in view of the performance of the non-monotonic sagitta methods, we suspect, as Pan [16, p. 155] did,

> *that undue emphasis has been laid on such monotonicity for so long as might have confined ourselves from making further progress.*

Additionally, our opinion is that the computational results obtained with non-simplex active-set methods (or basis-deficiency-allowing simplex variations, see [18]) are nowadays impressive enough to be able to claim the suitability for code of these methods to be included in optimization program libraries. Moreover, even though the simplex methodology is strongly established, we are convinced of the insightful possibilities of the methodological modification presented.

References

[1] ANSTREICHER, K.M., and T. TERLAKY (1994). A monotonic build-up simplex algorithm for linear programming. *Operations Research* 42, pp. 556–561.

[2] BJÖRCK, Å. (1996). *Numerical Methods for Least Squares Problems*. SIAM Publications, Philadelphia, USA.

[3] BAZARAA, M.S., J.J. JARVIS, and H.D. SHERALI (1990). *Linear Programming and Network Flows*. Second Edition. John Wiley & Sons, New York.

[4] BIXBY, R.E. (1992). Implementing the simplex method: The initial basis. *ORSA J. Computing* 14(3), pp. 670–676.

[5] FUKUDA, K., and T. TERLAKY (1997). Criss-cross methods: A fresh view on pivot algorithms. *Mathematical Programming* 79, pp. 369–395.

[6] FLETCHER, R. (1987). *Practical Methods of Optimization*. Second Edition. John Wiley & Sons, Chichester.

[7] GAY, D.M. (1985). Electronic mail distribution of linear programming test problems. *COAL Newsletter* 13, 10–12.

[8] GILL, P.E., W. MURRAY, and M.H. WRIGHT (1991). *Numerical Linear Algebra and Optimization*, Vol. 1. Addison-Wesley Publishing, Redwood City, California.

[9] GOLDFARB, D., and M.J. TODD (1989). Linear Programming. Chap. II in: Nemhauser, G.L., et al. (eds.), *Optimization*, pp. 73–170.

[10] GUERRERO-GARCÍA, P. (2002) *Range-Space Methods for Sparse Linear Programs* (Spanish). Ph.D. thesis, Dept. App. Math., Univ. Málaga, Spain.

[11] GUERRERO-GARCÍA, P., and A. SANTOS-PALOMO (2003). A non-simplex active-set framework for basis-deficiency-allowing simplex variations. Presented at the 20th Biennial Conference on Numerical Analysis, Dundee (Scotland), June 2003. Submitted for publication.

[12] LI, W., P. GUERRERO-GARCÍA, and A. SANTOS-PALOMO (2004). A basis-deficiency-allowing primal Phase-I algorithm using the most-obtuse-angle column rule. In preparation.

[13] LUSTIG, I.J., R.E. MARSTEN, and D.F. SHANNO (1994). Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing* 6(1), 1–14.

[14] MAROS, I. (1986). A general Phase-I method in linear programming. *European Journal of Operational Research* 23, pp. 64–77.

[15] OSBORNE, M.R. (1985). *Finite Algorithms in Optimization and Data Analysis*. Wiley, Chichester.

[16] PAN, P.-Q. (1995). New non-monotone procedures for achieving dual feasibility. *Journal of Nanjing University* 12(2), pp. 155–162.

[17] PAN, P.-Q. (1998). A dual projective simplex method for linear programming. *Computers and Mathematics with Applications* 35(6), 119–135.

[18] PAN, P.-Q. (1998). A basis-deficiency-allowing variation of the simplex method for linear programming. *Computers and Mathematics with Applications* 36(3), pp. 33–53.

[19] PAN, P.-Q., and W. LI (2003). A non-monotone Phase-I method in linear programming. *Journal of Southeast University* (English Edition) 19(3), pp. 293–296.

[20] PAN, P.-Q., and Y. PAN (2001). A Phase-I approach for the generalized simplex algorithm. *Computers and Mathematics with Applications* 42, pp. 1455–1464.

[21] PAPARRIZOS, K. (2003). An exterior point simplex algorithm for (general) linear programming problems. *Annals of Operations Research* 32, pp. 497–508.

[22] PAPARRIZOS, K., N. SAMARAS, and G. STEPHANIDES (2003). An efficient simplex type algorithm for sparse and dense linear programs. *European Journal of Operational Research* 148, pp. 323–334.

[23] SANTOS-PALOMO, A. (2004). The sagitta method for solving linear programs. *European Journal of Operational Research* 157(3), pp. 527–539.

[24] SANTOS-PALOMO, A., and P. GUERRERO-GARCÍA (1998). The sagitta method for solving linear programs as a feasible point method. Presented at Optimization 98, Univ. Coimbra (Portugal), July 1998.

[25] SANTOS-PALOMO, A., and P. GUERRERO-GARCÍA (2001a). Solving a sequence of sparse compatible systems. Presented at the 19th Biennial Conference on Numerical Analysis, Dundee (Scotland), June 26–29, 2001. Submitted for publication.

[26] SANTOS-PALOMO, A., and P. GUERRERO-GARCÍA (2001b). Solving a sequence of sparse least squares problems. Technical report, Dept. App. Math., Univ. Málaga. Submitted for publication.

[27] SANTOS-PALOMO, A., and P. GUERRERO-GARCÍA (2001c). A non-simplex method for linear programs in standard form. Presented at the XXVI Congreso Nacional de Estadística e Investigación Operativa, Úbeda, Spain, November 6–9, 2001. Submitted for publication.

[28] SANTOS-PALOMO, A., and P. GUERRERO-GARCÍA (2004). Computational experiences wih dense and sparse implementations of the sagitta method. Submitted for publication.

[29] SANTOS-PALOMO, A., and P. GUERRERO-GARCÍA (2004). Sagitta method with guaranteed convergence. Submitted for publication.

[30] SANTOS-PALOMO, A., and P. GUERRERO-GARCÍA (2004). Monotonic sagitta methods. In preparation.

[31] SAUNDERS, M.A. (2003). Large-scale numerical optimization. Technical report, Dept. Management Science and Engineering, Stanford Univ., Stanford. Handouts of the MS&E 318 course.

[32] ZIONTS, S. (1969). The criss-cross method for solving linear programming problems. *Management Science* 15, pp. 426–445.