The face simplex method in the cutting plane framework

Beltran, Cesar

The face simplex method in the cutting plane framework

Cèsar Beltran *

April 11, 2003

Abstract

In order to maximize a piecewise affine concave function one can basically use the simplex method or an interior point method. As an alternative, we propose the *face* simplex method. The *vertex to vertex* scheme of the simplex method is replaced by a more general *face to face* scheme in the face simplex method. To improve the current iterate, in the face simplex method, one computes the steepest ascent *on the current face* of the objective function graph and then an exact linesearch determines next iterate. This new procedure can be used in the cutting plane framework as a substitute of the simplex method. As a preliminary numerical test, this new version of the cutting plane method is compared with three other methods: subgradient, Kelley cutting plane and ACCPM.

Key words. Nonsmooth optimization, cutting plane methods, Lagrangian relaxation, subgradient method, steepest ascent method.

1 Introduction

In smooth unconstrained optimization the gradient method and the Newton method represent the two basic optimization philosophies. If we wish to maximize the smooth function f_s, as is well know, in the gradient method one chooses the steepest direction and picks up the best point within this direction. At each iteration the gradient method uses only a slice of the graph of f_s, as the information source to compute the step length. On the other hand, the Newton method, first, constructs g_s, which is a second order model of f_s based at the current iterate and then moves to the maximizer of g_s. Therefore, the difference between the two methods is based on the level of information about f_s used at each iteration. As expected, low level of information methods such as the gradient method produce optimization algorithms with low computation burden but may suffer from slow convergence. With the high level of information methods, such as the Newton method, the situation is symmetrical: high rate of convergence but with a high computational burden. The choice among a low or a high level of information method is problem dependent.

*cesar.beltran@hec.unige.ch, Logilab, HEC, University of Geneva, Switzerland. This work has been partially supported by the Fonds National Suisse de la Recherche Scientifique, subsidy FNS 1214-057093,99, and by the Spanish government, MCYT subsidy dpi2002-03330

In nonsmooth unconstrained optimization we can also distinguish between low and high level of information methods. In the low level of information family we find the subgradient method and the steepest ascent method, the nonsmooth version of the gradient method. In the high level of information family, we find the cutting plane method. If we wish to maximize the nonsmooth function f_{ns}, the cutting plane method constructs and updates at each iteration g_{ns}, a model of f_{ns} based on first order information (cutting planes). Different types of cutting plane methods arise depending on how the method determines the next iterate based on g_{ns}. In Kelley's cutting plane method [Kel60, WG59] the next iterate is taken as the maximizer of g_{ns}. Somehow, we could say that Kelley's cutting plane method is the nonsmooth version of the Newton method.

A more sophisticated and effective version of the cutting plane method is the proximal bundle method [HUL96, Kiw88, Fra02], in which a quadratic penalty term is appended to the linear objective of the cutting plane method in order to stabilize its performance. The other improved version of the cutting plane method is the analytic center cutting plane method (ACCPM) [GV99a, GV99c, GV99b], in which the analytic center of the localization set (the hypograph of g_{ns} plus an hyperplane to bound the hypograph) gives the next iterate. In practice, cutting plane methods have a good convergence behavior, with the exception of Kelley's version. However, for large scale problems they may be computationally demanding due to the high amount of information managed at each iteration.

When applied to the maximization of the function f_{ns}, the subgradient method [Sho69, Pol69, Sho98], at each iteration, tries to improve the current iterate (in terms of the distance to the optimal set), by following the direction given by a subgradient of f_{ns}. If in theory, the subgradient method has a guaranteed convergence to the optimum, in practice it has no clear stopping criterion which ensures optimality. The other drawback of the subgradient method is that the subgradient direction may not be an ascending direction. The steepest ascent method [HUL96] overcomes this handicap of the subgradient method by computing the minimum norm subgradient, which indeed, is an ascending direction. Computing the minimum norm subgradient may not be easy (it depends on the knowledge of ∂f_{ns}), so the steepest ascent method has so far not been a very successful method. Nevertheless, some successful direction following heuristic methods [Erl78, BH02] or problem depending ascent methods [CC90] have been proposed.

The aim of this paper is to introduce the face simplex method and to use it in the cutting plane framework. The objective of the face simplex method is to maximize a (concave) piecewise affine function. Therefore, a direct use of the face simplex method is to maximize the piecewise affine function g_{ns} which arises in the cutting plane framework. The philosophy of the face simplex method is to keep as much as possible the computational lightness of the direction following methods while using the first order information utilized by the cutting plane methods. Employed as a direction following method, the face simplex method first computes the steepest ascent *on the current face* of the graph of g_{ns} (a concave piecewise affine function) and second, the next iterate is computed by an exact linesearch. Employed in the cutting plane framework, the face simplex method replaces the simplex method as the tool to maximize g_{ns}. However, at each iteration of the cutting plane method, the face simplex method is truncated before reaching a Kelley's point (a point which maximizes g_{ns}). Apparently the points generated in this truncated way remain in a region centered around the optimal set as

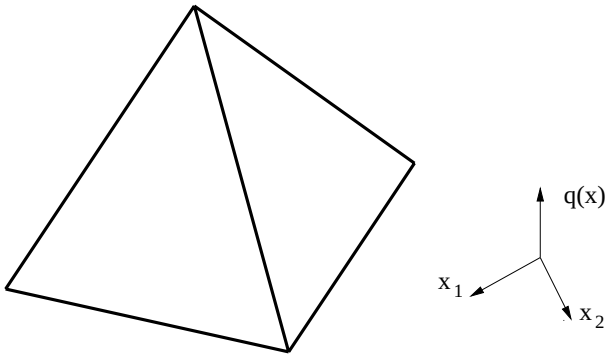

Figure 1: The pyramid function graph.

it happens with the bundle method and with ACCPM.

In the following example we introduce the face simplex method from an intuitive point of view. The algebra of the face simplex method is developed in section 3, together with its stopping criteria. In section 4, the face simplex method is proposed as a substitute of the simplex method within the cutting plane framework. Finally, section 5 presents the first computational experiences with the face simplex method in the cutting plane framework and its performance when compared to the subgradient method, Kelley cutting plane method and ACCPM.

Given $x \in \mathbb{R}^n$ and $z \in \mathbb{R}$, to lighten notation we will write (x, z) instead of $\binom{x}{z} \in \mathbb{R}^{n+1}$.

2 Example

To introduce the face simplex method we use $q(x)$, a pyramid shaped function. Let us define the following vectors: $v_1 = (-1,-1)', v_2 = (1,-1)', v_3 = (1,1)', v_4 = (-1,1)'$, the affine functions $q_1(x) = v_1'x = -x_1 - x_2$, $q_2(x) = v_2'x = x_1 - x_2$, $q_3(x) = v_3'x = x_1 + x_2$, $q_4(x) = v_4'x = -x_1 + x_2$ and the index set $I = \{1,2,3,4\}$. The pyramid function is defined as $q(x) : \mathbb{R}^2 \to \mathbb{R}$ with $q(x) = \min_{i \in I}\{q_i(x)\}$, whose graph is depicted in Figure 1. The pyramid problem is then:

$$\max_{x \in \mathbb{R}^2} q(x). \tag{1}$$

Different versions of the subgradient method ($x^{k+1} = x^k + \alpha^k s^k / \|s^k\|$ with $s^k \in \partial q(x^k)$) arise depending on the chosen step length. Thus the *basic* subgradient method takes $\alpha^k = c/(a+bk)$ [Sho69, Sho98]. One way to improve the very slow convergence of the basic subgradient method is to use the Polyak subgradient method [Pol69, Ber99] which takes $\alpha^k = \beta^k(\hat{q}^k - q(x^k))/\|s^k\|$, where $\hat{q}^k$ is an estimate of the optimum of $q(x)$ and $\beta^k \in]0,2[$. In the pyramid problem, by taking $\beta^k = 1/k$, $\hat{q}^k$ defined as in section 5.2 and with 50 iterations, we obtain a suboptimal solution: $q^{50} = -3.2 \cdot 10^{-3}$ (it is not difficult to see that the optimum is $q^* = 0$). The path that follow the Polyak subgradient iterates is depicted in Figure 2. Note that the Polyak subgradient method improves the basic subgradient method by incorporating dual information $(\hat{q}^k - q(x^k))/\|s^k\|)$ to compute more accurate step lengths. However, the Polyak subgradient method requires an estimate of the optimum to compute $\hat{q}^k$, which in general may not be easy to tune.

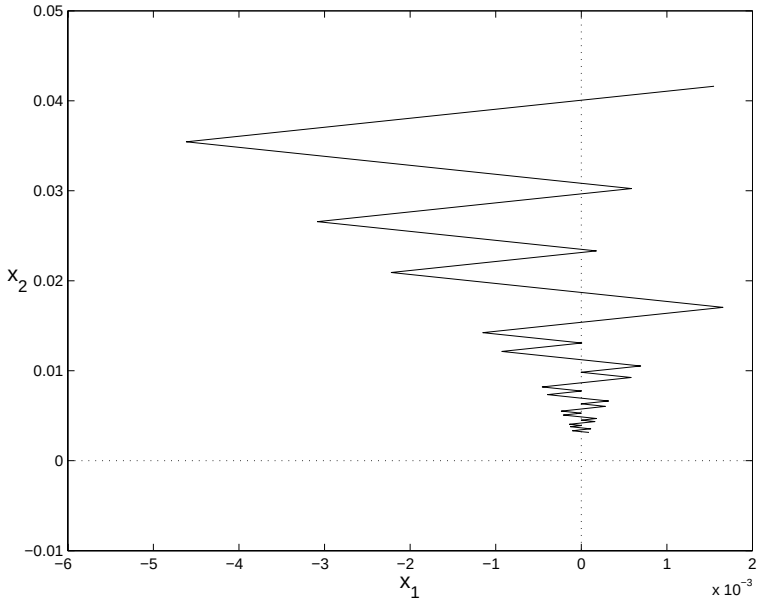

Figure 2: The Polyak subgradient path for the pyramid problem

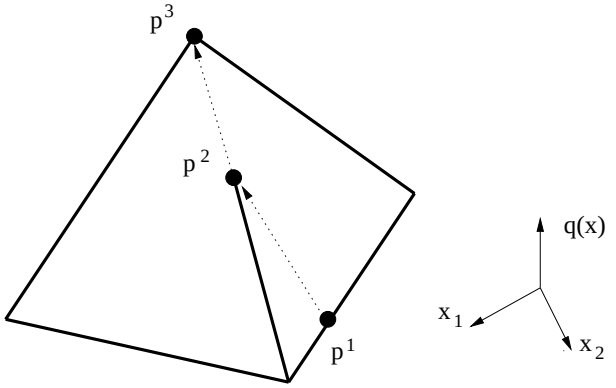

Figure 3: The face simplex method improves the subgradient method.

To end this example, we solve the pyramid problem by the face simplex method. The subgradient method works in the x-space ($\mathbb{R}^2$ for the pyramid problem), whereas the face simplex method works in the (x, z)-space ($\mathbb{R}^3$ for the pyramid problem), where $z = q(x)$. Let us consider G the graph of $q(x)$ whose shape is a pyramid and let $p^1 = (x^1, z^1) \in G$ (see Figure 3). From p^1, by following the steepest ascent on the current face of G we obtain p^2, the best point of G in this direction. Now the face associated to p^2 is an edge, therefore the steepest ascent on the current face follows this edge. Finally, the best point in the current ascent direction is p^3, the best point of the pyramid. If $p^3 = (x^3, z^3)$ then x^3 maximizes $q(x)$ and the optimum is $z^3 = q(x^3)$.

For the pyramid problem, we reach the optimum by using only two iterations of the face simplex method. The difference with the Polyak subgradient method is that we do not use estimated information about $q(x)$, but have an exact knowledge of it. The difference with the cutting plane method, is that now, at each iteration, we do not use all the available information about $q(x)$, we only use restricted information (an exact

linesearch).

3 The face simplex method

The aim of the face simplex method is to maximize a concave piecewise affine function defined by a finite family of known affine functions. To be more precise, first we define the index set $I = \{1, \ldots, m\}$, the family of affine functions $q_i : \mathbb{R}^n \to \mathbb{R}$ with $q_i(x) = s_i' x - b_i$ for all $i \in I$ and the piecewise affine function $q(x) = \min_{i \in I}\{q_i(x)\}$. Then the problem to be solved is

$$\max_{x \in \mathbb{R}^n} q(x) \tag{2}$$

whose optimum is assumed finite.

A general method to solve this problem is the steepest ascent method ([HUL96], Vol. 1, Chapter VIII). Given that a subgradient at a nonoptimal point x^k may not be an ascent, the steepest ascent ensures an ascending direction by taking the steepest ascent d^k, which can be computed as

$$d^k = \operatorname{argmin}\{\frac{1}{2}\|s\|^2 : s \in \partial q(x^k)\}. \tag{3}$$

Then an exact linesearch along d^k gives the step lengthα^k, which defines the next point as $x^{k+1} = x^k + \alpha^k d^k$. Unfortunately the steepest ascent method with exact linesearch may fail to converge ([HUL96], Vol. 1, VIII.2.2). To ensure convergence of the steepest ascent method, in the case of a piecewise affine function, one can use the 'step-to-nearest-edge' linesearch ([HUL96], Vol. 1, Theorem VIII.3.4.8). A second handicap of the steepest ascent method is its numerical instability, which may produce a large number of very small steps. In [NBR00] these difficulties are overcome by considering an outer approximation to the differential $\partial q(x^k)$.

As an alternative to the steepest ascent method, we propose the face simplex method. To give a first sketch of this method, we consider the graph of $q(x)$, $G = \{(x, z) : x \in \mathbb{R}^n, z = q(x)\} \subset \mathbb{R}^{n+1}$, a piecewise affine graph formed by the faces F_i, $(i = 1, \ldots, n_F)$ which may range from dimension 0 (vertices), dimension 1 (edges), up to dimension $n-1$ (facets), i.e. $G = \bigcup_{i=1}^{n_F} F_i$. We also define F^k as the face of minimal dimension that contains $p^k = (x^k, z^k)$, A^k as the smallest affine space containing F^k and V^k as the vector space parallel to A^k. p^k is said to be an optimal point of G, if and only if, x^k is an optimizer of $q(x)$.

The face simplex method ensures an ascending direction at p^k by taking the *face* steepest ascent, that is, the steepest ascent on F^k, or equivalently, the direction in V^k with the steepest slope. It is not difficult to see that the face steepest ascent and the steepest ascent are different concepts. To see that, one can imagine that the current iterate p^k is on an edge of G, i.e. the dimension of F^k is one. In such a case, we have only one choice for the face steepest ascent: it must be parallel to the one dimensional vector space V^k. On the other hand, the steepest ascent may be a different direction from the direction given by V^k, as can be seen in the example given in [HUL96], Vol. 1, VIII.2.2.

Furthermore, in the steepest ascent method one have to solve problem (3), which amounts to solving a quadratic programming problem if $\partial q(x^k)$ is polyhedral, as is

the case for a piecewise affine function. In contrast, with the face simplex method one can develop an explicit and easy to compute formula to solve the associated quadratic programming problem (see section 3.1). This can be done by computing the search direction on the graph of the objective function.

Note that if the current point p^k is a nonoptimal vertex of G, then F^k has dimension zero and it makes no sense to compute the face steepest ascent on it. In this case, the steepest adjacent edge to p^k will define the search direction for the face simplex method (*vertex steepest ascent*).

Face simplex algorithm (sketch)

Step 1 [Initialization.] Set $k = 1$ and initialize $p^1 = (x^1, z^1) \in G$.

Step 2 [Stopping criteria.] If p^k fulfills any of the stopping criteria, stop.

Step 3 [Direction: case 1.] If $\dim(V^k) > 0$ then compute d^k as the steepest direction within V^k.

Step 4 [Direction: case 2.] If $\dim(V^k) = 0$ then compute d^k as the parallel direction to the steepest adjacent edge to p^k.

Step 5 [Step length.] Compute α^k by exact linesearch along d^k.

Step 6 [Updating.] Compute next iterate $p^{k+1} = (x^{k+1}, z^{k+1})$, where $x^{k+1} = x^k + \alpha^k d^k$, $z^{k+1} = q(x^{k+1})$. Set $k \leftarrow k + 1$ and go back to step 2.

In the following sections we develop the algebra associated to the above face simplex method. We distinguish two cases:

3.1 First case: search direction on a face of positive dimension

In this section we wish to compute the face steepest ascent at a point of G which is neither an optimal point nor a vertex. As usual, if $q(x) = \min_{i \in I}\{q_i(x)\}$, then the hyperplane associated to the function $q_i(x) = s_i'x - b_i$ is said to be active at $p^k = (x^k, z^k)$ if $q_i(x^k) = q(x^k)$. In this case, $\partial q(x^k) \ni s_i$ is said to be active at x^k. The face F^k associated to p^k is defined by the intersection of all the active hyperplanes at p^k. If I^k is the index-set of active hyperplanes at p^k, i.e. $I^k = \{i : q_i(x^k) = q(x^k)\}$, then the smallest affine subspace that contains F^k is $A^k = \{p = (x, z) \in \mathbb{R}^{n+1} : z = q_i(x), i \in I^k\}$. Since

$$z = q_i(x) \Leftrightarrow z = s_i'x - b_i \Leftrightarrow b_i = (s_i', -1) \cdot \begin{pmatrix} x \\ z \end{pmatrix} \Leftrightarrow b_i = (s_i', -1) \cdot p,$$

the associated vector space parallel to A^k is $V^k = \{v \in \mathbb{R}^{n+1} : (s_i', -1) \cdot v = 0,\ i \in I^k\}$. Computing the face steepest ascent on F^k is equivalent to computing the steepest direction within V^k. For any $v^* = (v_1^*, \ldots, v_{n+1}^*)$ defining the steepest direction in V^k, the nonoptimality of p^k ensures that $v_{n+1}^* \neq 0$. Consequently, we can use $w^* = (1/v_{n+1}^*)v^* =: (w_1^*, \ldots, w_n^*, 1)$ to define the steepest direction in V^k. That is, in order to find the steepest direction within V^k, we can fix $v_{n+1} = 1$ and restrict our search within the affine subspace $W^k = \{w : (w, 1) \in V^k\} = \{w \in \mathbb{R}^n : s_i'w = 1,\ i \in I^k\}$.

The slope of $(w,1) \in V^k$ is $1/\|w\|$. Thus looking for the vector of maximum slope in V^k (the steepest ascent) is equivalent to looking for $w \in W^k$ of minimum norm. Therefore, the problem of finding the face steepest ascent d^k on F^k turns out to be equivalent to the problem

$$w^k = \text{argmin}\{\frac{1}{2}\|w\|^2 : w \in W^k\} = \text{argmin}\{\frac{1}{2}\|w\|^2 : s_i'w = 1,\ i \in I^k\}. \tag{4}$$

If we define matrix $S = (s_1, \ldots, s_m)$ where $m = |I^k|$ (without loss of generality we can assume that the active hyperplanes correspond to the first m hyperplanes) and vector $e' = (1, \ldots, 1) \in \mathbb{R}^m$, problem (4) can be recast as $w^k = \text{argmin}\{\frac{1}{2}\|w\|^2 : S'w = e\}$, whose first order optimality conditions are given by the linear system on (w, μ):

$$\begin{array}{rcl} w + S\mu & = & 0, \\ S'w - e & = & 0. \end{array} \tag{5}$$

From the first equation in (5) $w = -S\mu$, which substituted in the second equation gives $\mu^k = -(S'S)^{-1}e$. We can write $w^k = S(S'S)^{-1}e$.

So far we have assumed that $S'S$ has complete rank and therefore $(S'S)^{-1}$ exists. Suppose now that $S'S$ has incomplete rank. If $W^k \neq \emptyset$, problem (4) is well-posed and the optimality conditions (5) must have a solution independently of the rank of $S'S$. A solution of (5) can be computed as $\mu^k = -(S'S)^+e$, $w^k = S(S'S)^+e$, where $(S'S)^+$ denotes the pseudo-inverse matrix of $S'S$. (The pseudo-inverse of the $m \times n$ matrix A, denoted by A^+, is the unique $n \times m$ matrix such that $x = A^+b$ is the vector of minimum Euclidean length that minimizes $\|b - Ax\|_2$, see [GMW95], Sec. 2.2.5.6). The above discussion is summarized in the following proposition.

Proposition 3.1 *Let p^k be a non optimal point of G, F^k its associated face, $S = (s_1, \ldots, s_m)$ the matrix of active subgradients at p^k and $e' = (1, \ldots, 1)$. If F^k is not a vertex of G, then the face steepest ascent on F^k is given by $(w^k, 1)$, where $w^k = S(S'S)^{-1}e$ if $S'S$ has full rank and by $w^k = S(S'S)^+e$ otherwise.*

3.2 Second case: search direction at a non optimal vertex

In the second case we compute the vertex steepest ascent at a non optimal vertex p^k. To do this, we adapt the algebra of the simplex method to our case. Any point $(x, z) \in G$ satisfies

$$\begin{array}{rcl} z & \leq & q_1(x), \\ z & \leq & q_2(x), \\ & \ldots\ldots\ldots\ldots & , \\ z & \leq & q_m(x). \end{array} \tag{6}$$

Furthermore, considering that $q_i(x) = s_i'x - b_i$, there must exist a vector of slacks $y \geq 0$ such that

$$\begin{array}{rcl} b_1 & = & s_1'x - z - y_1, \\ b_2 & = & s_2'x - z - y_2, \\ & \ldots\ldots\ldots\ldots\ldots\ldots & , \\ b_m & = & s_m'x - z - y_m. \end{array} \tag{7}$$

Equivalently

$$\begin{pmatrix} s'_1 & -1 & -1 & & & \\ s'_2 & -1 & & -1 & & \\ \vdots & \vdots & & & \ddots & \\ s'_m & -1 & & & & -1 \end{pmatrix} \begin{pmatrix} x \\ z \\ y \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \tag{8}$$

For convenience we express the matrix of the above linear system as

$$A = \begin{pmatrix} V & -I_{n+1} & \\ W & & -I_l \end{pmatrix}, \tag{9}$$

where

$$V = \begin{pmatrix} s'_1 & -1 \\ \vdots & \vdots \\ s'_{n+1} & -1 \end{pmatrix}, \quad W = \begin{pmatrix} s'_{n+2} & -1 \\ \vdots & \vdots \\ s'_m & -1 \end{pmatrix}, \tag{10}$$

I_n is the $n \times n$ identity matrix and $l = m - (n+1)$.

As in the simplex method, we partition matrix A into B and N, i.e. $A = (B\ N)$ with

$$B = \begin{pmatrix} V & 0 \\ W & -I_l \end{pmatrix} \text{ and } N = \begin{pmatrix} -I_{n+1} \\ 0 \end{pmatrix}. \tag{11}$$

Next we present a simple result to be used later.

Lemma 3.1 *If V has full rank $n+1$ then:*

a) B also has full rank m.

b)

$$B^{-1} = \begin{pmatrix} V^{-1} & 0 \\ WV^{-1} & -I_l \end{pmatrix}. \tag{12}$$

c)

$$B^{-1}N = -\begin{pmatrix} V^{-1} \\ WV^{-1} \end{pmatrix}. \tag{13}$$

Proof. a) Given the triangular block structure of B we have that

$$|\det(B)| = |\det(V)| \cdot |\det(-I_l)| = |\det(V)|.$$

b) and c) have a direct proof. □

If the current point (x, z) is a vertex of G then the number of active hyperplanes (y_i=0) is at least $n+1$. We take $n+1$ of them, say the first $n+1$, to construct V and the other l hyperplanes to construct W. Even if some of the hyperplanes in W are active, from now on we will use the term active only for hyperplanes in V and the term non active for hyperplanes in W. In what follows we use the notation $v(i)$ to denote component i of vector v.

To continue our analysis we express the linear system (8) as

$$(B\ N)\begin{pmatrix} x \\ z \\ y_{_B} \\ y_{_N} \end{pmatrix} = b, \tag{14}$$

where $y_{_B} \geq 0$ are the slacks corresponding to the non active hyperplanes at the current point (*non active slacks*) and $y_{_N} = 0$ are the slacks corresponding to the active hyperplanes (*active slacks*). From (14)

$$B\begin{pmatrix} x \\ z \\ y_{_B} \end{pmatrix} + Ny_{_N} = b, \Rightarrow \begin{pmatrix} x \\ z \\ y_{_B} \end{pmatrix} = B^{-1}b - B^{-1}Ny_{_N}. \tag{15}$$

Now we define $c_{_B}$ as the $n+1$ canonical vector of $\mathbb{R}^m$, i.e. $c_{_B}(i) = 0$ for $i \neq n+1$ and $c_{_B}(n+1) = 1$. We also define $c_{_N}$ as 0_n, the zero vector of $\mathbb{R}^n$ and finally, we define the cost vector $c' = (c'_{_B}, c'_{_N})$. With this cost vector z can be expressed as:

$$z = c'\begin{pmatrix} x \\ z \\ y_{_B} \\ y_{_N} \end{pmatrix} = c'_{_B}\begin{pmatrix} x \\ z \\ y_{_B} \end{pmatrix}, \tag{16}$$

since $c_{_N} = 0_{n+1}$.

From (15) and (16), we obtain z as a function of $y_{_N}$:

$$z(y_{_N}) = c'_{_B}B^{-1}b - c'_{_B}B^{-1}Ny_{_N} \tag{17}$$

When moving from the current point $p^k = (x^k, z^k)$ (a vertex) to $p^{k+1} = (x^{k+1}, z^{k+1})$ at least one active hyperplane associated to $q_{i_0}(x)$ will become non active, i.e. $y_{_N}(i_0)$ will increase from 0 to a strictly positive value. The variation of z caused by this variation of $y_{_N}$ is studied in the following proposition.

Proposition 3.2 *Let $p^k = (x^k, z^k)$ be a vertex of G the graph of $q(x)$, V as in (10) and $y_{_N}$ the active slacks at p^k. Then*

$$z(y_{_N}) = z^k + \sum_{j=1}^{n+1} V^{-1}_{n+1,j}\ y_{_N}(j). \tag{18}$$

Proof. From (17)

$$z(y_{_N}) = c'_{_B}B^{-1}b - c'_{_B}B^{-1}Ny_{_N}$$

Now, considering that $z^k = z(0) = c'_{_B}B^{-1}b$ and lemma 3.1 c), we can write

$$\begin{aligned} z(y_{_N}) &= z^k - c_{_B}B^{-1}Ny_{_N} \\ &= z^k + (0'_n, 1, 0'_l)\begin{pmatrix} V^{-1} \\ WV^{-1} \end{pmatrix} y_{_N} \\ &= z^k + V^{-1}_{n+1,*} \cdot y_{_N} = z^k + \textstyle\sum_{j=1}^{n+1} V^{-1}_{n+1,j}\ y_{_N}(j). \end{aligned}$$

□

Next, with the elements of this section, we derive the search direction designed to improve the current point (vertex). To do so, let us define dz_j as $\frac{\partial z(y_N)}{\partial y_N(j)}(0)$, which is the partial derivative of $z(y_N)$ respect to $y_N(j)(0)$ at $y_N(j) = 0$ $(j = 1, \ldots, n+1)$. From proposition 3.2 we have that $dz_j = V^{-1}_{n+1,j}$, $(j = 1, \ldots, n+1)$ and considering that $y_N \geq 0$, it is clear that if $dz \leq 0$ we cannot improve the current value of z^k and therefore the current point p^k (a vertex) is optimal. Otherwise, there must exist $dz_j > 0$. Defining $J_1 = \{j : dz_j > 0\} \neq \emptyset$, we can improve z^k by increasing any $y_N(j_0)$, $j_0 \in J_1$. When, for a particular $j_0 \in J_1$, $y_N(j_0)$ is increased from 0 to a strictly positive value, y_B must remain nonnegative. y_B has two components: $y_{B1} = 0$ and $y_{B2} > 0$. When leaving the current point we must ensure that y_{B1} remains positive. From (15)

$$y_{B1}(y_N) = B^{-1}_{B1} b - (B^{-1}N)_{B1} y_N$$

where notation B^{-1}_{B1} denotes the rows of B^{-1} corresponding to y_{B1}. Given that $y_{B1}(0) = B^{-1}_{B1} b$ and by hypothesis $y_{B1}(0) = 0$, it turns out that $B^{-1}_{B1} b = 0$ and then

$$y_{B1}(y_N) = 0 - (B^{-1}N)_{B1} y_N$$

Finally, by applying lemma 3.1 c) we obtain:

$$y_{B1}(y_N) = (WV^{-1})_{B1}\ y_N .$$

Let $Dy_{B1} = (WV^{-1})_{B1}$, formed by the column vectors $Dy_{B1}(j)$, $j = 1, \ldots, n+1$. Given that $y_N(j_0) \geq 0$, to ensure $y_{B1} \geq 0$ we can only use positive columns of Dy_{B1}. The index-set of such columns is $J_2 = \{j : Dy_{B1}(j) \geq 0\}$.

The idea to derive the search direction is to increase *only* the active slack $y_N(j_0)$, which improves the most function $z(y_N)$ but restraining our choice of j_0 among the positive columns of Dy_{B1}, that is, j_0 must be in $J_1 \cap J_2$. To be more precise, let

$$dz_{j_0} = \arg \max_{j \in J_1 \cap J_2} \{dz_j\}$$

and rewrite (15) as

$$\begin{pmatrix} x(y_N) \\ z(y_N) \\ y_B(y_N) \end{pmatrix} = B^{-1}b - B^{-1}N y_N = \begin{pmatrix} x^k \\ z^k \\ y^k \end{pmatrix} + \begin{pmatrix} V^{-1} \\ WV^{-1} \end{pmatrix} y_N .$$

Taking into account that the new vector y'_N is equal to $(0, \ldots, y_N(j_0), \ldots, 0)$ it follows that

$$\begin{pmatrix} x(y_N) \\ z(y_N) \end{pmatrix} = \begin{pmatrix} x^k \\ z^k \end{pmatrix} + V^{-1}_{*j_0} \cdot y_N(j_0),$$

from where

$$x(y_N) = x^k + y_N(j_0) \cdot \begin{pmatrix} V^{-1}_{1j_0} \\ \vdots \\ V^{-1}_{nj_0} \end{pmatrix} = x^k + \alpha^k d^k .$$

Note that so far we have only computed the search direction

$$(d^k)' = (V^{-1}_{1j_0}, \ldots, V^{-1}_{nj_0}).$$

The step length $\alpha^k = y_N(j_0)$ will be calculated by exact line search along d^k. We summarize the above discussion in next proposition.

Proposition 3.3 *Let $p^k = (x^k, z^k)$ be a non optimal vertex of G the graph of $q(x)$. Then d^k, the vertex steepest ascent from p^k, can be computed as*

$$d^k = \begin{pmatrix} V^{-1}_{1j_0} \\ \vdots \\ V^{-1}_{nj_0} \end{pmatrix}, \text{ where } V = \begin{pmatrix} s'_1 & -1 \\ \vdots & \vdots \\ s'_{n+1} & -1 \end{pmatrix},$$

j_0 is such that $dz_{j_0} = \arg\max_{j \in J_1 \cap J_2}\{dz_j\}$, $dz_j = V^{-1}_{n+1,j}$, $J_1 = \{j : dz_j > 0\}$, $J_2 = \{j : Dy_{B1}(j) \geq 0\}$, $Dy_{B1} = (WV^{-1})_{B1}$ and y_{B1} are the null non active slacks.

Proof. At vertex p^k we have $n+1$ active hyperplanes with the associated null slacks $y_N = 0$. Above, in the current section, we have seen that we leave p^k by increasing $y_N(j_0)$ while the other n active hyperplanes remain active, that is, following d^k we move along the edge of G associated to $y_N(j_0)$. We have also seen that we choose j_0 such that

$$dz_{j_0} = \arg \max_{j \in J_1 \cap J_2} \{dz_j\}$$

where $dz_j = \partial z(0)/\partial y_N(j)$, i.e. we choose the edge with the steepest slope □

In this section we have seen how the computation of the vertex steepest ascent has been inspired by the algebra of the simplex method. However, an important difference with the simplex method must be pointed out. In the simplex method iterates move from vertex to vertex of graph G. In the face simplex method iterates move from a face to another face of graph G, since once the search direction has been computed, next iterate is computed by exact line search. For example, assuming that the simplex and the steepest ascent methods are in the same vertex, and also assuming that the two methods compute the same improving direction (the vertex steepest ascent), the simplex method stops at the next vertex, whereas the steepest ascent method stops at the best encountered point on G by following the improving direction.

3.3 Stopping criteria

The face simplex method uses three different criteria that guarantee optimality of the current point $p^k = (x^k, z^k)$.

Null subgradient stopping criterion: A null subgradient ensures optimality, that is, if $\partial q(x^k) \ni s^k = 0$ then x^k is optimal ([HUL00] Theorem D.2.2.1).

Vertex stopping criterion: The following stopping criterion is tested whenever the current point p^k is a vertex.

Proposition 3.4 *Let $p^k = (x^k, z^k)$ be a vertex of G, V as in (10). If $V^{-1}_{n+1,j} \leq 0$, $j = 1, \ldots, n+1$, then x is an optimal point of $q(x)$.*

Proof. This result is clear from Proposition 3.2, where we saw that

$$z(y_N) = z^k + \sum_{j=1}^{n+1} V^{-1}_{n+1,j}\, y_N(j),$$

that is, $z(y_N) \leq z^k$ for all $y_N \geq 0$ considering that by hypothesis $V^{-1}_{n+1,j} \leq 0$, $j = 1, \ldots, n+1$. □

Deficient Rank stopping criterion: The following stopping criterion is applied to any point $p^k = (x^k, z^k)$ of G.

Proposition 3.5 *Let $\{s_1, \ldots, s_m\}$ be the set of active subgradients, which define the active hyperplanes at the current point p^k,*

$$S = \begin{pmatrix} s_1 & \cdots & s_m \end{pmatrix}, \ \widetilde{S} = \begin{pmatrix} s_1 & \cdots & s_m \\ -1 & \cdots & -1 \end{pmatrix},$$

and $V^k = \{v \in \mathbb{R}^{n+1} : (s_i', -1) \cdot v = 0,\ i = 1, \ldots, m\} = \{v \in \mathbb{R}^{n+1} : \widetilde{S}'v = 0\}$ the vector space associated to x^k. If rank(S) < rank$(\widetilde{S})$, then x^k maximizes $q(x)$.

Proof. Let us suppose that $\text{rank}(\widetilde{S}) = m$. First let us see that $e_z \perp V^k$ where $e_z' = (0_n', 1)$. By hypothesis $\text{rank}(S) < m \ \Rightarrow \exists \hat{v} \neq 0 : S\hat{v} = 0$

$$\Rightarrow \begin{pmatrix} s_1 & \cdots & s_m \\ -1 & \cdots & -1 \end{pmatrix} \hat{v} = \begin{pmatrix} 0 \\ -\sum_{i=1}^{n+1} \hat{v}_i \end{pmatrix}$$

$\Rightarrow \widetilde{S}\hat{v} = Ke_z$, where $K = -\sum_{i=1}^{n+1} \hat{v}_i$

$\Rightarrow \widetilde{S}v = e_z$ where $v = \hat{v}/K$.

Note that K cannot be 0, otherwise $\widetilde{S}\hat{v} = 0$ with $\hat{v} \neq 0$ which contradicts the fact that $\widetilde{S}$ has full rank.

If $w \in V^k$, by definition of V^k we have $\widetilde{S}'w = 0$. Then $e_z'w = v'\widetilde{S}'w = v'0 = 0$ with $v \neq 0 \ \Rightarrow e_z \perp V^k$.

Considering that problem (2) is equivalent to the linear programming problem that maximizes $z = q(x)$ on the graph of $q(x)$, $e_z \perp V^k$ implies that p^k is optimal.

If $\text{rank}(\widetilde{S}) = p < m$ we can repeat the proof by defining S and $\widetilde{S}$ from p linearly independent columns of $\widetilde{S}$. □

3.4 The face simplex algorithm

Section 3 can be summarized in the following algorithm.

Face simplex algorithm

Step 1 [Initialization.] Set $k = 1$ and initialize $p^1 \in G$.

Step 2 [Null subgradient stopping criterion.] If p^k fulfills the null subgradient stopping criterion, p^k is optimal. Stop.

Step 3 [Deficient rank stopping criterion.] Form the matrix of active subgradients at p^k, $S = (s_1, \ldots, s_m)$ and $\widetilde{S}$:

$$\widetilde{S} = \begin{pmatrix} s_1 & \cdots & s_m \\ -1 & \cdots & -1 \end{pmatrix},$$

If If $\text{rank}(S) < \text{rank}(\widetilde{S})$, p^k is optimal. Stop.

Step 4 [Direction: case 1.] If $\text{rank}(\widetilde{S}) < n+1$ then compute the search direction $d^k = S(S'S)^{-1}e$ if S has full rank and by $d^k = S(S'S)^{+}e$ otherwise. Go to step 7.

Step 5 [Vertex stopping criterion.] If $\text{rank}(\widetilde{S}) \geq n+1$ define V as in (10), compute $dz_j = V^{-1}_{n+1,j}$, $j = 1, \ldots, n+1$ and define $J_1 = \{j : dz_j > 0\}$. If J_1 is empty, then p^k is optimal. Stop.

Step 6 [Direction: case 2.] Compute the search direction d^k: Define W as in (10) and as in section 3.2, let $Dy_{B1} = (WV^{-1})_{B1}$ be the rows of WV^{-1} associated to the null non active slacks ($y_{B1} = 0$). Dy_{B1} is formed by the column vectors $Dy_{B1}(j)$, $j = 1, \ldots, n+1$. Define $J_2 = \{j : Dy_{B1}(j) \geq 0\}$ and compute

$$dz_{j_0} = \arg \max_{j \in J_1 \cap J_2} \{dz_j\}.$$

Then

$$d^k = \begin{pmatrix} V^{-1}_{1 j_0} \\ \vdots \\ V^{-1}_{n j_0} \end{pmatrix}.$$

Step 7 [Step length.] Compute α^k by exact linesearch along d^k.

Step 8 [Updating.] Compute next iterate $p^{k+1} = (x^{k+1}, z^{k+1})$, where $x^{k+1} = x^k + \alpha^k d^k$, $z^{k+1} = q(x^{k+1})$. Set $k \leftarrow k+1$ and go back to step 2.

Note that in step 4, by using next lemma we can check $\text{rank}(S)$ to know whether $(S'S)^{-1}$ exists.

Lemma 3.2 *Let S a $n \times m$ matrix. S has full rank, if and only if, $S'S$ has full rank.*

Proof. First, let us see that if S has full rank, then $S'S$ also has full rank. Let us consider $v \neq 0$. Then, since S has full rank, $Sv \neq 0$. On the other hand, if $S'Sv = 0$ and defining $w = Sv$, we would have $S'w = 0$ with $w \neq 0$, which contradicts the fact that S' has full rank ($\text{rank}(S) = \text{rank}(S')$).

Second, let us see that if $S'S$ has full rank, then S also has full rank. $Sv = 0$ implies $S'Sv = 0$. Considering that $S'S$ has full rank, then v must be equal to 0, which proves that S also has full rank □

4 The face simplex cutting plane method

The Kelley cutting plane method is intended to maximize a concave function $q(x)$ by maximizing successive piecewise affine outer approximations $q^k(x) = \min_{i=1,\ldots,k}\{q_i(x)\}$ of $q(x)$, where each $q_i(x)$ $(i = 1, \ldots, k)$, is a supporting hyperplane of $q(x)$.

Kelley cutting plane algorithm

Step 1 [Initialization.] Initialize $k = 1$, $q^1(x)$, $\epsilon > 0$ and the compact domain D.

Step 2 [Best point computing.] Compute $x^k = \arg\max_{x \in D} q^k(x)$.

Step 3 [Stopping criterion.] If $|q(x^k) - q^k(x^k)|/|q(x^k)| < \epsilon$, then x^k is an ϵ-optimal point. Stop.

Step 4 [Outer approximation updating.] Compute $s^k \in \partial q(x^k)$ and $q(x^k)$. Define $q^{k+1}(x)$ by appending the hyperplane $z = q(x^k) + (s^k)'(x - x^k)$ to the definition of $q^k(x)$.

Step 5 [Counter updating.] $k \leftarrow k + 1$ and go back to step 2.

The poor performance of the Kelley's algorithm may be due to the fact that at step 2 it searches for a global maximum of $q^k(x)$ though $q^k(x)$ may be a good approximation of $q(x)$ *locally*. Therefore, even though the simplex method is very effective to compute such a global optimum, it may be advantageous a method that in a few iterations substantially improves our current value of $q(x^k)$ within a relatively small region around x^k. Somehow, we find this approach in the bundle method and in ACCPM. By truncating the face simplex method we pursue the same efect: x^{k+1} not too far from x^k with $q^k(x^{k+1}) > q^k(x^k)$. Therefore only a few iterations of the face simplex method (inner iterations) will be allowed at each cutting plane iteration (outer iteration). The number of face simplex iterations is limited by a constant N_{inner}. In what follows, we call the previous approach the face simplex cutting plane (FSCP) method.

Furthermore, in the FSCP method, the face simplex optimization process is also truncated whenever an improving point $\hat{x}$ is found at an inner iteration. To be more precise, let $\hat{q}^k = \max\{q(x^i) : i = 1, \ldots, k\}$ be the best objective computed so far, $\Delta q \in]1, 2[$ and $LB^k = \Delta q \cdot \hat{q}^k$. Then if $q^k(\hat{x}) > LB^k$, the face simplex method stops at $\hat{x}$, sets $x^{k+1} = \hat{x}$, and a new outer iteration of the cutting plane method is performed. All in all, at each iteration of the cutting plane method we do not compute an optimal point of the outer approximation to $q(x)$ (Kelley point), only an improving point.

5 Numerical example

5.1 The generalized binary knapsack problem

In this section we show the performance of the face simplex cutting plane (FSCP) method by solving the dual problem associated to the Lagrangian decomposition [GK87] of the

generalized binary knapsack problem ([AMO93], pag. 131). In the generalized binary knapsack problem we must choose among n objects. Object i has utility u_i, weight w_i and volume v_i. The objective is to maximize the total utility subject to weight and volume limitations, that is, we can carry no more than weight W and volume V.

The generalized binary knapsack problem can be formulated (minimization version) as follows:

$$\begin{aligned} \min \quad & -u'x && (19a)\\ \text{s.t.} \quad & w'x \leq W, && (19b)\\ & v'x \leq V, && (19c)\\ & x \in \{0,1\}^n. && (19d) \end{aligned}$$

To obtain our generalized binary knapsack (Lagrangian decomposition) dual problem, we first duplicate vector x into y and obtain the equivalent problem:

$$\begin{aligned} \min \quad & -u'x && (20a)\\ \text{s.t.} \quad & w'x \leq W, && (20b)\\ & v'y \leq V, && (20c)\\ & x = y, && (20d)\\ & x \in \{0,1\}^n,\ y \in \{0,1\}^n. && (20e) \end{aligned}$$

Second, we relax the identity constraint (20d):

$$\max_{\lambda \in \mathbb{R}^n} \left\{ \begin{array}{ll} \min & (-u' + \lambda')x - \lambda' y \\ \text{s.t.} & w'x \leq W, \\ & v'y \leq V, \\ & x \in \{0,1\}^n,\ y \in \{0,1\}^n \end{array} \right\}. \qquad (21)$$

The minimization problem in (21) defines the so called dual function $q(\lambda)$. Then, problem (21) can be rewritten as $\max_{\lambda \in \mathbb{R}^n} q(\lambda)$. Note that the minimization problem in (21) decomposes into two knapsack problems.

5.2 Computational test

In this test we solve ten random instances of problem (21). Vectors u, v and w are randomly generated: $u_i = \lfloor 2n\xi_i^u \rfloor + 1$, $v_i = \lfloor 2n\xi_i^v \rfloor + 1$ and $w_i = \lfloor 2n\xi_i^w \rfloor + 1$, where ξ_i^u, ξ_i^v and ξ_i^w are random variables uniformly distributed in the interval $[0, 1]$. The maximum volume and weight are defined as $V = \lfloor (1/n) \sum_{i=1,\ldots,n} v_i \rfloor + \max_{i=1,\ldots,n}\{v_i\}$ and $W = \lfloor (1/n) \sum_{i=1,\ldots,n} w_i \rfloor + \max_{i=1,\ldots,n}\{w_i\}$. Obviously, with this definition of V and W any object can be used since $v_i \leq V$ and $w_i \leq W$ for all i. In the following tables, label 10.0 stands for a generalized binary knapsack instance with 10 objects, case 0. That is we label the instances by *'Number of objects.case'*. Recall that n, the problem dimension, equals the number of objects.

Programs have been written in Matlab 6.1 [HJ00] and run in a PC (Pentium-IV Xeon PC, 2.4 GHz, with 2 Gb of RAM memory) under Linux operating system.

5.2.1 Description of the face simplex cutting plane iterations.

First, we give a thorough description of the face simplex cutting plane (FSCP) iterations. The tuning parameters for the FSCP method are (see section 4 for its definition): $N_{inner} = 10$ and $\Delta q = 1.001$. We set the starting point as $\lambda^0 = 0 \in \mathbb{R}^n$ and the ϵ used to stop the cutting plane algorithm is 10^{-6}.

The optimization statistics for the FSCP method can be found in Table 1. For example in problem 10.1, the FSCP method needs 31 outer (cutting plane) iterations and 37 inner (face simplex) iterations. The final number of different hyperplanes used to approximate the dual function is 23. At each outer iteration k the cutting plane method generates a hyperplane. If this new hyperplane already exists in the set of hyperplanes that generates $q^k(\lambda)$, the approximation to $q(\lambda)$, then it is rejected and therefore $q^{k+1}(\lambda)$ and $q^k(\lambda)$ are the same function. In problem 10.1, 31 hyperplanes have been generated but, among them, only 23 different ones have been stored (8 repeated hyperplanes have been rejected). By direction type we mean the number of hyperplanes used by the face simplex method to compute the search direction. For problem 10.1, the average direction type has been 3.2 and the maximum direction type has been 17. The method has stopped after detecting an active set of subgradients with deficient rank (deficient rank stopping criterion). In problem 10.0 the method stops at a point with an associated null subgradient (null subgradient stopping criterion).

In Table 1, column 'Direction type / Average', we observe that 2.7, the average number of hyperplanes used to compute the face steepest ascent, is very low compared to $(11 + 26 + 51 + 76 + 101)/5 = 54$, the average working space dimension. Therefore, not many vertices have been visited, since to visit a vertex requires a number of hyperplanes at least equal to the working space dimension. This can also be seen in column 'Direction type / Maximum', where only in instance 10.1 the FSCP method visits a vertex (at some iteration we use 17 hyperplanes, which define a vertex in the working space, that has dimension 11). For the other instances the maximum direction type is always lower than the working space dimension: we visit faces which are not vertices. In all these instances, the level of information used by the FSCP method to compute the direction search is low.

Also it is worth pointing out that in no case has the vertex stopping criterion been active (see 'Stopping cause' in Table 1). Then, to get more insight in this matter, we have solved the pyramid problem introduced in section 2 for various dimensions (we do not report results for this test). We have observed that even if all the instances have a unique optimizer that corresponds to the vertex of the pyramid, the optimum has been detected by the deficient rank stopping criterion and not by the vertex stopping criterion. Here we find another difference with the simplex method: we do not necessarily need a number of hyperplanes equal to the working space dimension to declare the optimality of a vertex.

5.2.2 Performance of the face simplex cutting plane method.

Second, we compare the face simplex cutting plane method with other three methods: the subgradient method, the Kelley cutting plane method and ACCPM. We are interested in the quality of the solution found (Table 3) and the performance of the

Table 1: The face simplex cutting plane method: description of the iterations

Problem	Iterations		Hyperplanes	Direction type		Stopping cause
	Outer	Inner		Average	Maximum	
10.0	11	10	11	1.0	1	Null subgradient
10.1	31	37	23	3.2	17	Deficient rank
25.0	26	30	23	1.8	8	Deficient rank
25.1	77	141	67	3.1	20	Deficient rank
50.0	37	54	35	2.1	10	Null subgradient
50.1	20	27	19	1.9	6	Null subgradient
75.0	83	241	79	4.2	26	Null subgradient
75.1	73	328	71	4.3	22	Null subgradient
100.0	97	385	91	4.6	32	Deficient rank
100.1	17	16	17	1.0	1	Null subgradient
Average	47	127	44	2.7	14	

face simplex cutting plane method relative to the other methods (Tables 4 and 5). In these Tables, FSCP stands for face simplex cutting plane method, SG for subgradient method, KCP for Kelley cutting plane method and ACCPM for analitic center cutting plane method. The number of iterations ratio and the CPU time ratio are computed relative to the FSCP method. For example, in Table 4, instance 10,0, the number of iterations ratio for the subgradient method is $37/11 = 3.36$.

In all the tested methods, we set $\lambda^0 = 0 \in \mathbb{R}^n$ as the starting point, the number of iterations is limited by 500 and the stopping criteriterion tolerance ϵ is set equal to 10^{-6}.

Within the subgradient method we use the Polyak step length defined by $\beta^k = \alpha^k(\hat{q}^k - q^k)/\|s^k\|^2$, where $\alpha^k = \alpha^0/k \quad (k > 0)$, $q^k = q(\lambda^k)$ and $s^k \in \partial q(\lambda^k)$. Based on [Ber99], $\hat{q}^k$, an approximation to the optimum q^*, is chosen as

$$\hat{q}^k = (1 + \delta^k) \cdot \max_{i=1,\ldots,k} \{q^i\},$$

where δ^k is updated at each iteration as

$$\delta^{k+1} = \begin{cases} \min\{\max\{\underline{\delta},\ \delta^k \cdot \triangle\delta\},\ \overline{\delta}\} & \text{if} \quad q^k > q^{k-1}, \\ \\ \min\{\max\{\underline{\delta},\ \delta^k / \triangle\delta\},\ \overline{\delta}\} & \text{if} \quad q^k \leq q^{k-1}, \end{cases} \tag{22}$$

which ensures $\delta^{k+1} \in [\underline{\delta}, \overline{\delta}]$ for all k. $\underline{\delta}$ and $\overline{\delta}$ are, respectively, a lower and upper bound to δ^k and $\triangle\delta$ is a constant factor. The best tuning we have found is: α^0 displayed in Table 2, $\triangle\delta = 1.1$, $\delta^0 = 0.10$, $\underline{\delta} = 0.01$ and $\overline{\delta} = 0.20$.

Regarding the stopping criterion, the subgradient method is stopped whenever the average variation of λ^k for the last 5 iterations is small enough, i.e. whenever

$$\frac{\sum_{i=0}^{4} \|\lambda^{k-i} - \lambda^{k-i-1}\|_\infty}{5} < 10^{-6}. \tag{23}$$

Table 2: α^0 for the subgradient method

Problem	10.0	10.1	25.0	25.1	50.0	50.1	75.0	75.1	100.0	100.1
α^0	5	10	5	10	5	10	10	10	10	5

Table 3: Best dual objective (suboptimal values are in bold type)

Problem	FSCP	SG	KCP	ACCPM
10,0	-55.000	-55.000	-55.000	-55.000
10,1	-52.000	**-52.029**	-52.000	-52.000
25,0	-198.000	**-198.014**	-198.000	-198.000
25,1	-137.667	**-137.781**	-137.667	-137.667
50,0	-454.000	-454.000	**-479.444**	-454.000
50,1	-396.000	-396.000	**-420.273**	-396.000
75,0	-1009.000	-1009.000	**-1105.168**	-1009.000
75,1	-947.000	-947.000	**-1051.231**	-947.000
100,0	-1073.200	**-1073.487**	**-1346.901**	-1073.200
100,1	-941.000	-941.000	**-1133.700**	-941.000
Average	-526.287	-526.331	-597.938	-526.287

An important difference between the subgradient method and the cutting plane methods is the tuning process. The subgradient method with Polyak step, as we have implemented it, depends on parameters α^0, δ^0, $\triangle\delta$, $\underline{\delta}$ and $\overline{\delta}$, which may need a demanding tuning as happened in this test. On the contrary, the cutting plane methods are based on an increasingly accurate knowledge of the optimized function, which allows these methods to autoadapt to each problem. That is, in the subgradient method, the tuning parameters are much more problem dependent than in the cutting plane methods.

Regarding the quality of the solution found (Table 3), the subgradient method either converges to the optimum (instances 10.0, 50.0, 50.1, 75.0, 75.1 and 100.1) or finds a suboptimal solution (instances 10.1, 25.0, 25.1 and 100.0). In the first situation, the subgradient method stops after encountering a null subgradient. Intuitively, we can imagine the graph of the corresponding dual function as a truncated pyramid. In the second situation, no null subgradient is encountered and, intuitivelly we can imagine the graph of the corresponding dual function as a pyramid or as a tent.

Regarding the number of iterations and the CPU time (Tables 4 and 5), we find the same situations: for the truncated pyramid shape the subgradient method needs about one hundred or less iterations to converge, whereas for the pyramid shape the subgradient method is not able to compute the optimum with 500 iterations. The FSCP method computes the optimum for the two pyramid shapes, but it can be appreciated that more iterations are needed for the truncated pyramid shape.

In this test, the subgradient method needs an average of 6.32 times the number of FSCP iterations to obtain solutions of worse quality. Also on average, the FSCP method performs 4.77 times faster than the subgradient method.

Table 4: Number of iterations

Problem	Iterations				Ratio vs. FSCP		
	FSCP	SG	KCP	ACCPM	SG	KCP	ACCPM
10,0	11	37	52	28	3.36	4.73	2.55
10,1	31	**500**	40	29	16.13	1.29	0.94
25,0	26	**500**	193	47	19.23	7.42	1.81
25,1	77	**500**	261	55	6.49	3.39	0.71
50,0	37	103	**500**	82	2.78	13.51	2.22
50,1	20	103	**500**	78	5.15	25.00	3.90
75,0	83	106	**500**	106	1.28	6.02	1.28
75,1	73	63	**500**	104	0.86	6.85	1.42
100,0	97	**500**	**500**	131	5.15	5.15	1.35
100,1	17	46	**500**	91	2.71	29.41	5.35
Average	47	246	355	75	6.32	10.28	2.15

Table 5: CPU time is seconds

Problem	CPU time				Ratio vs. FSCP		
	FSCP	SG	KCP	ACCPM	SG	KCP	ACCPM
10,0	0.1	0.2	3.7	0.3	2.00	37.00	3.00
10,1	0.4	4.5	3.0	0.5	11.25	7.50	1.25
25,0	1.6	15.2	18.7	1.8	9.50	11.69	1.13
25,1	2.4	16.1	26.2	2.3	6.71	10.92	0.96
50,0	4.0	12.8	93.5	10.1	3.20	23.38	2.53
50,1	2.2	11.3	93.9	9.7	5.14	42.68	4.41
75,0	20.4	26.3	161.4	28.5	1.29	7.91	1.40
75,1	18.2	15.4	165.8	27.4	0.85	9.11	1.51
100,0	41.4	212.1	257.6	59.8	5.12	6.22	1.44
100,1	7.1	19.0	251.8	40.7	2.68	35.46	5.73
Average	9.8	33.3	107.6	18.1	4.77	19.19	2.33

As usual, in the Kelley cutting plane method we adopt the warm start technique, i.e. at each iteration the simplex method is initialized by using the optimal basis from the previous iteration. We use MOSEK (Version 2.5.1.65(RC), WWW: http://www.mosek.com) as linear programming solver.

Results with the Kelley cutting plane method where very poor. If we limit the number of iterations by 500, only the first four problems are solved up to optimality (Table 3). Thus, for example, in problem 50,0, if we wish to attain optimality we would need 1050 iterations. In this test the Kelley cutting plane number of iterations averages onver ten times the FSCP number of iterations. The much better performance of the new method is even more clear considering the CPU time ratio (19.19).

One way to improve the Kelley cuting plane method is to use the analytic center cutting plane method (ACCPM). It is clear in this test that, given the polyhedrom that approximates the dual function, it may be far better to compute a centered point of the polyhedrom (the analytic center in the ACCPM case) than the Kelley point of the polyhedrom. Suprisingly enough, in this problem the FSCP method (a low level of information method) on average needs less than half the number of ACCPM iterations to converge (ACCPM being a high level of information method). As a hypothesis, the reason for that could be the good quality of the FSCP search direction in this problem, in the sense that at any point of the cutting plane polyhedrom the face simplex direction points towards or near to the optimal face (as the Newton direction does for a quadatic function). Nevertheless, more research is still needed on this subject. Regarding the CPU time, the FSCP methods performs an average of 2.33 times faster than ACCPM in this test.

6 Conclusions and extensions

From a theoretical point of view, we have developed a new method to maximize a piecewise affine concave function: the face simplex method.The *vertex to vertex* scheme of the simplex method is replaced by a *face to face* scheme where any point on the graph of the objective function can be an iterate. To improve the current iterate, the face simplex method performs a linesearch so that we can use large steps in contrast with the constrained 'vertex to vertex' simplex steps. Also, by using the deficient rank stopping criterion, we do not necessarily need a number of hyperplanes equal to the working space dimension to declare the optimality of a vertex.

The face simplex method fits well in the cutting plane framework as an alternative to the simplex method. We call it the face simplex cutting plane (FSCP) method. The poor performance of the Kelley's cutting plane method may be due to the fact that it searches for a global maximum of q^k, the outer approximation to the maximized function q. However, considering that q^k may only be a good approximation to q locally, we have found advantageous to improve the current value of $q^k(x^k)$ in a neighborhood of x^k. To attain this approach, we have limited the number of face simplex iterations at each cutting plane iteration. The optimal point computed at each iteration by the simplex method is replaced by a suboptimal point given by the (truncated) face simplex method. Apparently, these suboptimal points are better positioned to collect the relevant hyperplanes necessary to describe q around the optimal face.

From a practical point of view, we have compared the FSCP method to three well known methods. The first of them has been the subgradient method. In the FSCP method, the problem depending tuning of the subgradient method is avoided by an exact linesearch which uses the cutting plane information. In the numerical test carried out, the FSCP on average has performed 4.77 times faster than the subgradient method. The second compared method has been the Kelley cutting plane method. By computing suboptimal solutions not too far from the current point the FSCP method has dramatically improved the Kelley cutting plane method. In the numerical test carried out, the FSCP on average has performed 19.19 times faster than the Kelley cutting plane method. The third compared method has been ACCPM. In this test, the FSCP method has shown to be over two times faster than ACCPM (iterations and CPU time).

Among the possible extensions of this work, it remains first to study the convergence of the face simplex method and second, to perform large-scale intensive tests to establish its effectiveness.

Acknowledgments. We would like to thank the technical support from Logilab (HEC-University of Geneva), and specially the valuable remarks and suggestions of professors Jean-Philippe Vial and Claude Tadonki.

References

[AMO93] R. K. Ahuja, T. L. Magnati, and J. B. Orlin. *Network flows*. Prentice Hall, Inc, 1993.

[Ber99] Dimitri. P. Bertsekas. *Nonlinear Programming*. Ed. Athena Scientific, Belmont, Massachusetts, (USA), 2n edition, 1999.

[BH02] C. Beltran and F. J. Heredia. An effective line search for the subgradient method. Technical Report DR 2002/15, Departament d'estadística i investigació operativa, Universitat Politècnica de Catalunya, Barcelona, 2002.

[CC90] A. R. Conn and G. Cornuéjols. A projection method for the uncapacitated facility location problem. *Mathematical programming*, (46):373–398, 1990.

[Erl78] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Operations Research*, 26:992–1009, 1978.

[Fra02] A. Frangioni. Generalized bundle methods. *SIAM journal on optimization*, 13(1):117–156, 2002.

[GK87] M. Guignard and S. Kim. Lagrangean decomposition: a model yielding stronger Lagrangean bounds. *Mathematical Programming*, (39):215–228, 1987.

[GMW95] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization*. Academic Press Limited, London, 10th edition, 1995.

[GV99a] J.-L. Goffin and J.Ph. Vial. Convex nondifferentiable optimization: a survey focussed on the analytic center cutting plane method. Technical Report 99.02, Geneva University - HEC - Logilab, February 1999.

[GV99b] J.-L. Goffin and J.Ph. Vial. Shallow, deep and very deep cuts in the analytic center cutting plane method. *Mathematical programming*, (84):89–103, 1999.

[GV99c] J.-L. Goffin and J.Ph. Vial. A two-cut approach in the analytic center cutting plane method. *Mathematical methods of operations research*, (49):149–169, 1999.

[HJ00] D. J. Higham and Higham N. J. *Matlab guide*. Siam, 2000.

[HUL96] J. B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*, volume I and II. Springer-Verlag, Berlin, 1996.

[HUL00] J.-B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis*. Springer, 2000.

[Kel60] J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.

[Kiw88] K. C. Kiwiel. *A survey of bundle methods for nondifferentiable optimization*. In: Proceedings, XIII. International Symposium on mathematical programming, Tokyo, 1988.

[NBR00] P. Neame, N. Boland, and D. Ralph. An outer approximate subdifferential method for piecewise affine optimization. *Mathematical programming*, (Ser. A 87):57–86, 2000.

[Pol69] B. T. Poljak. Minimization of unsmooth functionals. *CMMP*, 9(3):14–29, 1969.

[Sho69] N. Z. Shor. The generalized gradient descent. *In: Trudy 1 Zimnei Skoly po Mat. Programmirovaniyu, 3:578-585*, 1969.

[Sho98] N. Z. Shor. *Nondifferentiable optimization and polynomial problems*. Nonconvex optimization and its applications. Kluwer academic publishers, 1998.

[WG59] Cheney W. and A. A. Goldstein. Newton's method for convex programming and chebyshev approximation. *Numerische Mathematik*, 1(5):253–268, 1959.