Localization and Cutting-Plane Methods

S. Boyd and L. Vandenberghe

April 9, 2018

Contents

In these notes we describe a class of methods for solving general convex and quasiconvex optimization problems, based on the use of *cutting-planes*, which are hyperplanes that separate the current point from the optimal points. These methods, called *cutting-plane methods* or *localization methods*, are quite different from interior-point methods, such as the barrier method or primal-dual interior-point method described in [**?**, §11]. Cutting-plane methods are usually less efficient for problems to which interior-point methods apply, but they have a number of advantages that can make them an attractive choice in certain situations.

- Cutting-plane methods do not require differentiability of the objective and constraint functions, and can directly handle quasiconvex as well as convex problems. Each iteration requires the computation of a subgradient of the objective or constraint functions.
- Cutting-plane methods can exploit certain types of structure in large and complex problems. A cutting-plane method that exploits structure can be faster than a general-purpose interior-point method for the same problem.
- Cutting-plane methods do not require evaluation of the objective and all the constraint functions at each iteration. (In contrast, interior-point methods require evaluating all the objective and constraint functions, as well as their first and second derivatives.) This can make cutting-plane methods useful for problems with a very large number of constraints.
- Cutting-plane methods can be used to decompose problems into smaller problems that can be solved sequentially or in parallel.

To apply these methods to nondifferentiable problems, you need to know about subgradients, which are described in a separate set of notes. More details of the analytic center cutting-plane method are given in another separate set of notes.

1 Cutting-planes

The goal of cutting-plane and localization methods is to find a point in a convex set $X \subseteq \mathbf{R}^n$, which we call the *target set*, or, in some cases, to determine that X is empty. In an optimization problem, the target set X can be taken as the set of optimal (or ϵ-suboptimal) points for the problem, and our goal is find an optimal (or ϵ-suboptimal) point for the optimization problem.

We do not have direct access to any description of the target set X (such as the objective and constraint functions in an underlying optimization problem) except through an *oracle*. When we query the oracle at a point $x \in \mathbf{R}^n$, the oracle returns the following information to us: it either tells us that $x \in X$ (in which case we are done), or it returns a separating hyperplane between x and X, *i.e.*, $a \neq 0$ and b such that

$$a^T z \leq b \text{ for } z \in X, \qquad a^T x \geq b.$$

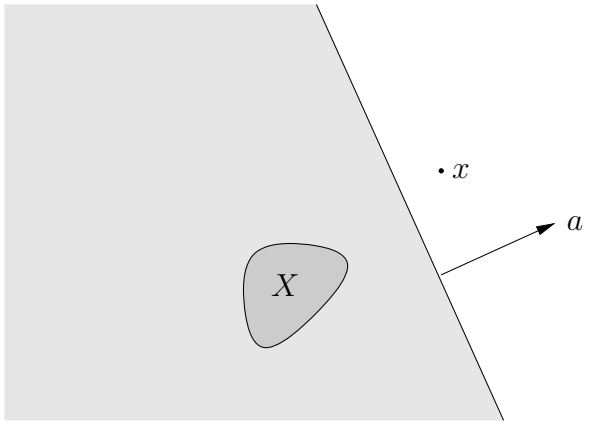

Figure 1: The inequality $a^T z \leq b$ defines a cutting-plane at the query point x, for the target set X, shown shaded. To find a point in the target set X we need only search in the lightly shaded halfspace; the unshaded halfspace $\{z \mid a^T z > b\}$ cannot contain any point in the target set.

This hyperplane is called a *cutting-plane*, or *cut*, since it 'cuts' or eliminates the halfspace $\{z \mid a^T z > b\}$ from our search; no such point could be in the target set X. This is illustrated in figure 1. We call the oracle that generates a cutting-plane at x (or the message that $x \in X$) a *cutting-plane oracle*. We can assume $\|a\|_2 = 1$, since dividing a and b by $\|a\|_2$ defines the same cutting-plane.

When the cutting-plane $a^T z = b$ contains the query point x, we refer to it as a *neutral cut* or *neutral cutting-plane*. When $a^T x > b$, which means that x lies in the interior of the halfspace that is being cut from consideration, the cutting-plane is called a *deep cut*. Figure 2 illustrates a neutral and deep cut. Intuition suggests that a deep cut is better, *i.e.*, more informative, than a neutral cut (with the same normal vector a), since it excludes a larger set of points from consideration.

2 Finding cutting-planes

In this section we show how to find cutting-planes for several standard convex optimization problems. We take the target set X to be the optimal set for the problem, so the oracle must either declare the point x optimal, or produce a hyperplane that separates x from the optimal set. It is straightforward to include equality constraints, so we leave them out to simplify the exposition.

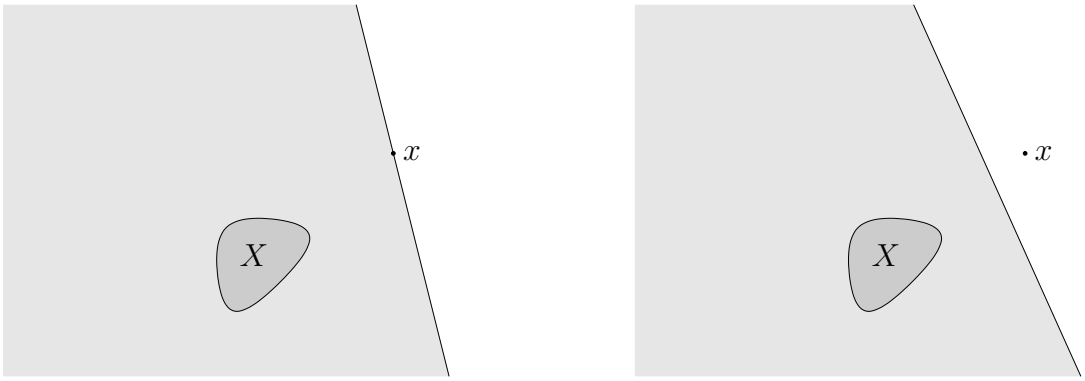

Figure 2: *Left:* a neutral cut for the point x and target set X. Here, the query point x is on the boundary of the excluded halfspace. *Right:* a deep cut for the point x and target set X.

2.1 Unconstrained minimization

We first consider the unconstrained optimization problem

$$\text{minimize} \quad f_0(x), \tag{1}$$

where f_0 is convex. To find a cutting-plane for this problem, at the point x, we proceed as follows. We find a subgradient $g \in \partial f(x)$. (If f is differentiable at x, then our only choice is $g = \nabla f(x)$.) If $g = 0$, then x is optimal, *i.e.*, in the target set X. So we assume that $g \neq 0$. Recall that for all z we have

$$f_0(z) \geq f_0(x) + g^T(z - x)$$

(indeed, this is the definition of a subgradient). Therefore if z satisfies $g^T(z-x) > 0$, then it also satisfies $f_0(z) > f_0(x)$, and so cannot be optimal (*i.e.*, in X). In other words, we have

$$g^T(z - x) \leq 0 \text{ for } z \in X,$$

and $g^T(z - x) = 0$ for $z = x$. This shows that

$$g^T(z - x) \leq 0$$

is a (neutral) cutting-plane for (1) at x.

This cutting-plane has a simple interpretation: in our search for an optimal point, we can remove the halfspace $\{z \mid g^T(z - x) > 0\}$ from consideration because all points in it have an objective value larger than the point x, and therefore cannot be optimal. This is illustrated in figure 3.

We can generate a deep cut if we know a number $\bar{f}$ that satisfies

$$f_0(x) > \bar{f} \geq f^\star,$$

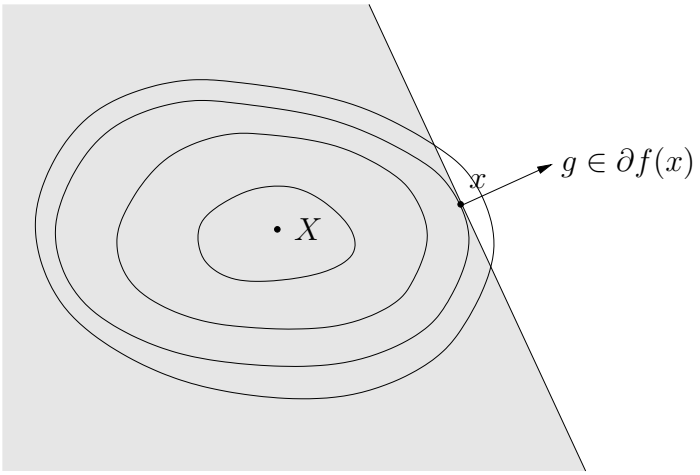

Figure 3: The curves show the level sets of a convex function f_0. In this example the optimal set X is a singleton, the minimizer of f_0. The hyperplane given by $g^T(z-x) = 0$ separates the point x (which lies on the hyperplane) from the optimal set X, hence defines a (neutral) cutting-plane. All points in the unshaded halfspace can be 'cut' since in that halfspace we have $f_0(z) \geq f_0(x)$.

where $f^\star = \inf_x f_0(x)$ is the optimal value of the problem (1). In this case we know that any optimal point $x^\star$ must satisfy

$$\bar{f} \geq f^\star \geq f_0(x) + g^T(x^\star - x),$$

so we have the deep cut

$$g^T(z-x) + f_0(x) - \bar{f} \leq 0.$$

When the problem (1) is quasiconvex (*i.e.*, when f_0 is quasiconvex), we can find a cutting-plane at x by finding a nonzero quasigradient of f_0 at x. Essentially by definition, the inequality $g^T(z-x) \leq 0$ is a cutting-plane when g is a nonzero quasigradient of f at x.

2.2 Feasibility problem

We consider the feasibility problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array}$$

where f_i are convex. Here the target set X is the feasible set.

To find a cutting-plane for this problem at the point x we proceed as follows. If x is feasible, *i.e.*, satisfies $f_i(x) \leq 0$ for $i = 1, \ldots, m$, then $x \in X$. Now suppose x is not feasible.

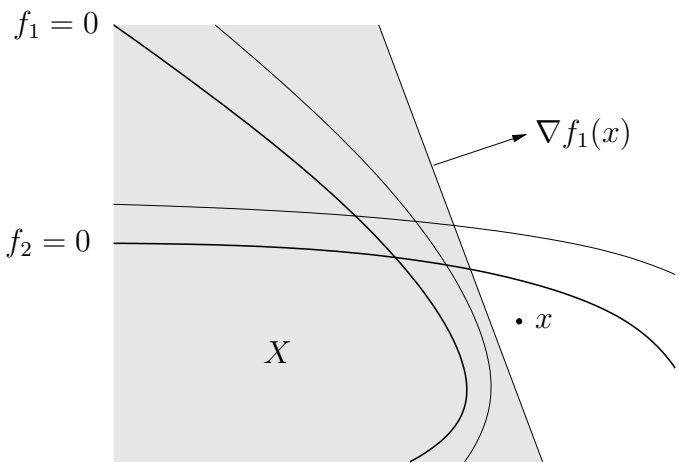

Figure 4: The curves show level sets of two convex functions f_1, f_2; the darker curves show the level sets $f_1 = 0$ and $f_2 = 0$. The feasible set X, defined by $f_1 \leq 0$, $f_2 \leq 0$, is at lower left. At the point x, the constraint $f_1(x) \leq 0$ is violated, and the hyperplane $f_1(x) + \nabla f_1(x)^T(z - x) = 0$ defines a deep cut.

This means that there is at least one index j for which $f_j(x) > 0$, *i.e.*, x violates the jth constraint. Let $g_j \in \partial f_j(x)$. From the inequality

$$f_j(z) \geq f_j(x) + g_j^T(z - x),$$

we conclude that if

$$f_j(x) + g_j^T(z - x) > 0,$$

then $f_j(z) > 0$, and so z also violates the jth constraint. It follows that any feasible z satisfies the inequality

$$f_j(x) + g_j^T(z - x) \leq 0,$$

which gives us the required cutting-plane. Since $f_j(x) > 0$, this is a deep cut.

Here we remove from consideration the halfspace defined by $f_j(x) + g_j(x)^T(z - x) > 0$ because all points in it violate the jth inequality, as x does, hence are infeasible. This is illustrated in figure 4.

Note that we can find a cutting-plane for *every* violated constraint, so if more than one constraint is violated at x, we can generate multiple cutting-planes that separate x from X. (We will see that some algorithms can make use of multiple cutting-planes at a given point.)

2.3 Inequality constrained problem

By combining the methods described above, we can find a cutting-plane for the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{2}$$

where $f_0, \ldots, f_m$ are convex. As above, the target set X is the optimal set.

Given the query point x, we first check for feasibility. If x is not feasible, then we can construct a cut as

$$f_j(x) + g_j^T(z - x) \leq 0, \tag{3}$$

where $f_j(x) > 0$ (*i.e.*, j is the index of any violated constraint) and $g_j \in \partial f_j(x)$. This defines a cutting-plane for the problem (2) since any optimal point must satisfy the jth inequality, and therefore the linear inequality (3). The cut (3) is called a *feasibility cut* for the problem (2), since we are cutting away a halfplane of points known to be infeasible (since they violate the jth constraint).

Now suppose that the query point x is feasible. Find a $g_0 \in \partial f_0(x)$. If $g_0 = 0$, then x is optimal and we are done. So we assume that $g_0 \neq 0$. In this case we can construct a cutting-plane as

$$g_0^T(z - x) \leq 0,$$

which we refer to as an *objective cut* for the problem (2). Here, we are cutting out the halfspace $\{z \mid g_0^T(z - x) > 0\}$ because we know that all such points have an objective value larger than x, hence cannot be optimal.

We can also find a deep objective cut, by keeping track of the best objective value f_{best}, among feasible points, found so far. In this case we can use the cutting-plane

$$g_0^T(z - x) + f_0(x) - f_{\text{best}} \leq 0,$$

since all other points have objective value at least f_{best}. (If x is the best feasible point found so far, then $f_{\text{best}} = f(x)$, and this reduces the neutral cut above.)

3 Localization algorithms

3.1 Basic cutting-plane and localization algorithm

We start with a set of initial linear inequalities

$$Cz \preceq d,$$

where $C \in \mathbf{R}^{q \times n}$, that are known to be satisfied by any point in the target set X. One common choice for this initial set of inequalities is the ℓ_∞-norm ball of radius R, *i.e.*,

$$-R \leq z_i \leq R, \quad i = 1, \ldots, n,$$

where R is chosen large enough to contain X. At this point we know nothing more than

$$X \subseteq \mathcal{P}_0 = \{z \mid Cz \preceq d\}.$$

Now suppose we have queried the oracle at points $x^{(1)}, \ldots, x^{(k)}$, none of which were announced by the oracle to be in the target set X. Then we have k cutting-planes

$$a_i^T z \leq b_i, \quad i = 1, \ldots, k,$$

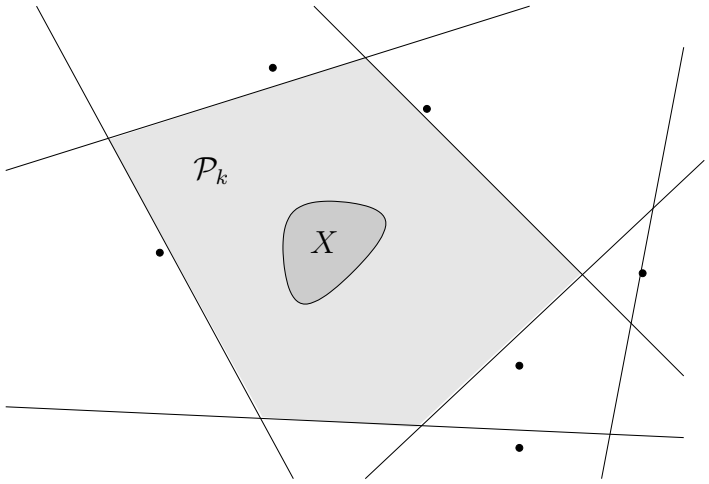

Figure 5: Points $x^{(1)}, \ldots, x^{(k)}$, shown as dots, and the associated cutting-planes, shown as lines. From these cutting-planes we conclude that the target set X (shown dark) lies inside the localization polyhedron $\mathcal{P}_k$, shown lightly shaded. We can limit our search for a point in X to $\mathcal{P}_k$.

that separate $x^{(k)}$ from X, respectively. Since every point in the target set must satisfy these inequalities, we know that

$$X \subseteq \mathcal{P}_k = \{z \mid Cz \preceq d,\ a_i^T z \leq b_i,\ i = 1, \ldots, k\}.$$

In other words, we have *localized* X to within the polyhedron $\mathcal{P}_k$. In our search for a point in X, we need only consider points in the *localization polyhedron* $\mathcal{P}_k$. This is illustrated in figure 5.

If $\mathcal{P}_k$ is empty, then we have a proof that the target set X is empty. If it is not, we choose a new point $x^{(k+1)}$ at which to query the cutting-plane oracle. (There is no reason to choose $x^{(k+1)}$ outside $\mathcal{P}_k$, since we know that all target points lie in $\mathcal{P}_k$.) If the cutting-plane oracle announces that $x^{(k+1)} \in X$, we are done. If not, the cutting-plane oracle returns a new cutting-plane, and we can update the localization polyhedron by adding the new inequality. This iteration gives the basic cutting-plane or localization algorithm:

Basic conceptual cutting-plane/localization algorithm

given an initial polyhedron $\mathcal{P}_0 = \{z \mid Cz \preceq d\}$ known to contain X.
$k := 0$.
repeat
  Choose a point $x^{(k+1)}$ in $\mathcal{P}_k$.
  Query the cutting-plane oracle at $x^{(k+1)}$.
  If the oracle determines that $x^{(k+1)} \in X$, quit.

Else, update $\mathcal{P}_k$ by adding the new cutting-plane: $\mathcal{P}_{k+1} := \mathcal{P}_k \cap \{z \mid a_{k+1}^T z \leq b_{k+1}\}$.
If $\mathcal{P}_{k+1} = \emptyset$, quit.
$k := k + 1$.

Provided we choose $x^{(k+1)}$ in the interior of $\mathcal{P}_k$, this algorithm generates a strictly decreasing sequence of polyhedra, which contain X:

$$\mathcal{P}_0 \supseteq \cdots \supseteq P_k \supseteq X.$$

These inclusions are strict since each query point $x^{(j+1)}$ is in the interior of $\mathcal{P}_j$, but either outside, or on the boundary of, $\mathcal{P}_{(j+1)}$.

3.2 Measuring uncertainty and progress

The polyhedron $\mathcal{P}_k$ summarizes what we know, after k calls to the cutting-plane oracle, about the possible location of target points. The size of $\mathcal{P}_k$ gives a measure of our ignorance or uncertainty about target points: if $\mathcal{P}_k$ is small, we have localized the target points to within a small set; if $\mathcal{P}_k$ is large, we still have much uncertainty about where the target points might be.

There are several useful scalar measures of the size of the localization set $\mathcal{P}_k$. Perhaps the most obvious is its diameter, *i.e.*, the diameter d of the smallest ball that contains $\mathcal{P}_k$. If this ball is $\{\hat{x} + u \mid \|u\|_2 \leq d/2\}$, we can say that the target set has been localized to within a distance $d/2$ of the point $\hat{x}$. Using this measure, we can judge the progress in a given iteration by the reduction in the diameter of $\mathcal{P}_k$. (Since $\mathcal{P}_{k+1} \subseteq \mathcal{P}_k$, the diameter always decreases.)

Another useful scalar measure of the size of the localization set is its volume. Using this measure, we can judge the progress in iteration k by the fractional decrease in volume, *i.e.*,

$$\frac{\mathbf{vol}(\mathcal{P}_{k+1})}{\mathbf{vol}(\mathcal{P}_k)}.$$

This volume ratio is affine-invariant: if the problem (and choice of query point) is transformed by an affine change of coordinates, the volume ratio does not change, since if $T \in \mathbf{R}^{n \times n}$ is nonsingular,

$$\frac{\mathbf{vol}(T\mathcal{P}_{k+1})}{\mathbf{vol}(T\mathcal{P}_k)} = \frac{\mathbf{vol}(\mathcal{P}_{k+1})}{\mathbf{vol}(\mathcal{P}_k)}.$$

(The diameter ratio does not have this property.)

The log of the volume of the uncertainty set, *i.e.*, $\log \mathbf{vol}(\mathcal{P}_k)$, has a nice interpretation as a measure of uncertainty. Up to a scale factor and an additive constant, $\log \mathbf{vol}(\mathcal{P}_k)$ gives the number of bits required to specify any point in the set with an accuracy ϵ. To see this, we first find a minimal cover of $\mathcal{P}_k$ with ϵ balls. This can be done with approximately

$$N = \mathbf{vol}(C)/(a_n \epsilon^n)$$

balls, where a_n is a constant that depends only on n. To specify one of these balls requires an index with $\lceil \log_2 N \rceil$ bits. This has form $c_n + \log \mathbf{vol}(C)$, where c_n depends on n and ϵ. Using this measure of uncertainty, the log of the ratio of the volume of $\mathcal{P}_k$ to the volume of $\mathcal{P}_{k+1}$ is exactly the decrease in uncertainty.

Volume arguments can be used to show convergence of (some) cutting-plane methods. In one standard method, we assume that the target set X contains a ball B_r of radius $r > 0$, and that $\mathcal{P}_0$ is contained in some ball B_R of radius R. We show that at each step of the cutting-plane method the volume of $\mathcal{P}_k$ is reduced at least by some factor $\gamma < 1$. If the algorithm has not terminated in k steps, then, we have

$$\mathbf{vol}(B_r) \leq \mathbf{vol}(\mathcal{P}_k) \leq \gamma^k \mathbf{vol}(\mathcal{P}_0) \leq \gamma^k \mathbf{vol}(B_R)$$

since $B_r \subseteq \mathcal{P}_k$ and since $B_R \supseteq \mathcal{P}_0$. It follows that

$$k \leq \frac{n \log(R/r)}{\log(1/\gamma)}.$$

3.3 Choosing the query point

The cutting-plane algorithm described above is only conceptual, since the critical step, *i.e.*, how we choose the next query point $x^{(k+1)}$ inside the current localization polyhedron $\mathcal{P}_k$, is not fully specified. Roughly speaking, our goal is to choose query points that result in small localization polyhedra. We need to choose $x^{(k+1)}$ so that $\mathcal{P}_{k+1}$ is as small as possible, or equivalently, the new cut removes as much as possible from the current polyhedron $\mathcal{P}_k$. The reduction in size (say, volume) of $\mathcal{P}_{k+1}$ compared to $\mathcal{P}_k$ gives a measure of how informative the cutting-plane for $x^{(k+1)}$ is.

When we query the oracle at the point $x^{(k+1)}$, we do not know which cutting-plane will be returned; we only know that $x^{(k+1)}$ will be in the excluded halfspace. The informativeness of the cut, *i.e.*, how much smaller $\mathcal{P}_{k+1}$ is than $\mathcal{P}_k$, depends on the direction a_{k+1} of the cut, which we do not know before querying the oracle. This is illustrated in figure 6, which shows a localization polyhedron $\mathcal{P}_k$ and a query point $x^{(k+1)}$, and two cutting-planes that could be returned by the oracle. One of them gives a large reduction in the size of the localization polyhedron, but the other gives only a small reduction in size.

Since we want our algorithm to work well no matter which cutting-plane is returned by the oracle, we should choose $x^{(k+1)}$ so that, no matter which cutting-plane is returned by the oracle, we obtain a good reduction in the size of our localization polyhedron. This suggests that we should choose $x^{(k+1)}$ to be deep inside the polyhedron $\mathcal{P}_k$, *i.e.*, it should be some kind of center of $\mathcal{P}_k$. This is illustrated in figure 7, which shows the same localization polyhedron $\mathcal{P}_k$ as in figure 6 with a more central query point $x^{(k+1)}$. For this choice of query point, we cut away a good portion of $\mathcal{P}_k$ no matter which cutting-plane is returned by the oracle.

If we measure the informativeness of the kth cut using the volume reduction ratio $\mathbf{vol}(\mathcal{P}_{k+1})/\mathbf{vol}(\mathcal{P}_k)$, we seek a point $x^{(k+1)}$ such that, no matter what cutting-plane is returned by the oracle, we obtain a certain guaranteed volume reduction. For a cutting-plane with normal vector a, the least informative is the neutral one, since a deep cut with the

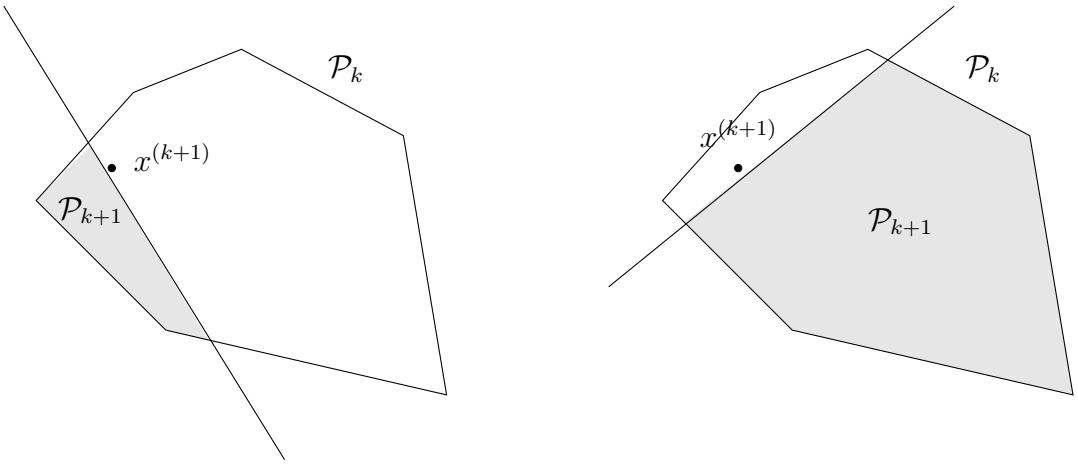

Figure 6: A localization polyhedron $\mathcal{P}_k$ and query point $x^{(k+1)}$, shown as a dot. Two possible scenarios are shown. *Left.* Here the cutting-plane returned by the oracle cuts a large amount from $\mathcal{P}_k$; the new polyhedron $\mathcal{P}_{k+1}$, shown shaded, is small. *Right.* Here the cutting-plane cuts only a very small part of $\mathcal{P}_k$; the new polyhedron $\mathcal{P}_{k+1}$, is not much smaller than $\mathcal{P}_k$.

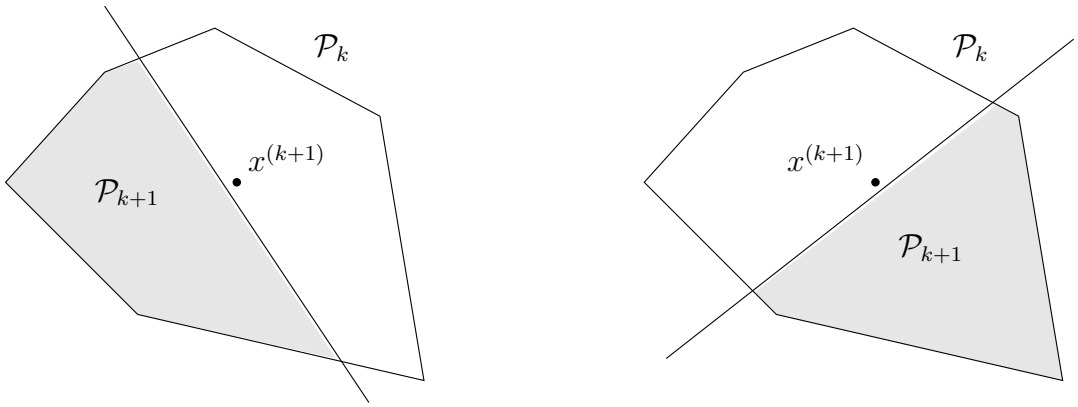

Figure 7: A localization polyhedron $\mathcal{P}_k$ and a more central query point x^{k+1} than in the example of figure 6. The same two scenarios, with different cutting-plane directions, are shown. In both cases we obtain a good reduction in the size of the localization polyhedron; even the worst possible cutting-plane at x would result in $\mathcal{P}_{k+1}$ substantially smaller than $\mathcal{P}_k$.

same normal vector leads to a smaller volume for $\mathcal{P}_{k+1}$. In a worst-case analysis, then, we can assume that the cuts are neutral, *i.e.*, have the form $a^T(z - x^{(k+1)}) \leq 0$. Let ρ denote the volume ratio, as a function of a,

$$\rho(a) = \frac{\mathbf{vol}(\mathcal{P}_k \cap \{z \mid a^T(z - x^{(k+1)}) \leq 0\})}{\mathbf{vol}(\mathcal{P}_k)}.$$

The function ρ is positive homogeneous, and satisfies

$$\rho(a) + \rho(-a) = 1.$$

To see this, note that a neutral cut with normal a divides $\mathcal{P}_k$ into two polyhedra,

$$\mathcal{P}_{k+1} = \mathcal{P}_k \cap \{z \mid a^T(z - x^{(k+1)}) \leq 0\}, \qquad \overline{\mathcal{P}}_{k+1} = \mathcal{P}_k \cap \{z \mid a^T(z - x^{(k+1)}) \geq 0\}.$$

The first is the new localization polyhedron, and the second is the polyhedron of points 'thrown out' by the cut. The sum of their volumes is the volume of $\mathcal{P}_k$. If the cut with normal $-a$ is considered, we get the same two polyhedra, with the new polyhedron and the polyhedron of 'thrown out' points switched.

From $\rho(a) + \rho(-a) = 1$, we see that the worst-case volume reduction satisfies

$$\sup_{a \neq 0} \rho(a) \geq 1/2.$$

This means that the worst-case volume reduction (over all possible cutting-planes that can be returned by the oracle) can never be better (smaller) than 1/2. The best possible (guaranteed) volume reduction we can have is 50% in each iteration.

4 Some specific cutting-plane methods

Many different choices for the query point have been proposed, which give different cutting-plane or localization algorithms. These include:

- The *center of gravity algorithm*, also called the *method of central sections*. The query point $x^{(k+1)}$ is chosen as the center of gravity of $\mathcal{P}_k$.
- *Maximum volume ellipsoid (MVE) cutting-plane method.* The query point $x^{(k+1)}$ is chosen as the center of the maximum volume ellipsoid contained in $\mathcal{P}_k$.
- *Chebyshev center cutting-plane method.* The query point $x^{(k+1)}$ is chosen as the Chebyshev center of $\mathcal{P}_k$, *i.e.*, the center of the largest Euclidean ball that lies in $\mathcal{P}_k$.
- *Analytic center cutting-plane method (ACCPM).* The query point $x^{(k+1)}$ is chosen as the analytic center of the inequalities defining $\mathcal{P}_k$.

Each of these methods has advantages and disadvantages, in terms of the computational effort required to determine the next query point, the theoretical complexity of the method, and the practical performance of the method.

4.1 Bisection method on R

We first describe a very important cutting-plane method: the *bisection method*. We consider the special case $n = 1$, *i.e.*, a one-dimensional search problem. We will describe the traditional setting in which the target set X is the singleton $\{x^*\}$, and the cutting-plane oracle always returns a neutral cut. The cutting-plane oracle, when queried with $x \in \mathbf{R}$, tells us either that $x^* \leq x$ or that $x^* \geq x$. In other words, the oracle tells us whether the point x^* we seek is to the left or right of the current point x.

The localization polyhedron $\mathcal{P}_k$ is an interval, which we denote $[l_k, u_k]$. In this case, there is an obvious choice for the next query point: we take $x^{(k+1)} = (l_k + u_k)/2$, the midpoint of the interval. The bisection algorithm is:

Bisection algorithm for one-dimensional search.

given an initial interval $[l, u]$ known to contain x^*; a required tolerance $r > 0$
repeat
  $x := (l + u)/2$.
  Query the oracle at x.
  If the oracle determines that $x^* \leq x$, $u := x$.
  If the oracle determines that $x^* \geq x$, $l := x$.
until $u - l \leq 2r$

In each iteration the localization interval is replaced by either its left or right half, *i.e.*, it is *bisected*. The volume reduction factor is the best it can be: it is always exactly 1/2. Let $2R = u_0 - l_0$ be the length of the initial interval (*i.e.*, $2R$ gives it diameter). The length of the localization interval after k iterations is then $2^{-k}2R$, so the bisection algorithm terminates after exactly

$$k = \lceil \log_2(R/r) \rceil \tag{4}$$

iterations. Since x^* is contained in the final interval, we are guaranteed that its midpoint (which would be the next iterate) is no more than a distance r from x^*. We can interpret R/r as the ratio of the initial to final uncertainty. The equation (4) shows that the bisection method requires exactly one iteration per bit of reduction in uncertainty.

It is straightforward to modify the bisection algorithm to handle the possibility of deep cuts, and to check whether the updated interval is empty (which implies that $X = \emptyset$). In this case, the number $\lceil \log_2(R/r) \rceil$ is an upper bound on the number of iterations required.

The bisection method can be used as a simple method for minimizing a differentiable convex function on $\mathbf{R}$, *i.e.*, carrying out a line search. The cutting-plane oracle only needs to determine the sign of $f'(x)$, which determines whether the minimizing set is to the left (if $f'(x) \geq 0$) or right (if $f'(x) \leq 0$) of the point x.

4.2 Center of gravity method

The center of gravity method, or CG algorithm, was one of the first localization methods proposed, by Newman [**?**] and Levin [**?**]. In this method we take the query point to be $x^{(k+1)} = \mathbf{cg}(\mathcal{P}_k)$, where the center of gravity of a set $C \subseteq \mathbf{R}^n$ is defined as

$$\mathbf{cg}(C) = \frac{\int_C z \, dz}{\int_C \, dz},$$

assuming C is bounded and has nonempty interior. The center of gravity is invariant under affine transformations, so the CG method is also affine-invariant.

The center of gravity turns out to be a very good point in terms of the worst-case volume reduction factor: we always have

$$\frac{\mathbf{vol}(\mathcal{P}_{k+1})}{\mathbf{vol}(\mathcal{P}_k)} \leq 1 - 1/e \approx 0.63.$$

In other words, the volume of the localization polyhedron is reduced by at least 37% at each step. Note that this guaranteed volume reduction is completely independent of all problem parameters, including the dimension n.

This guarantee comes from the following result: suppose $C \subseteq \mathbf{R}^n$ is convex, bounded, and has nonempty interior. Then for any nonzero $a \in \mathbf{R}^n$, we have

$$\mathbf{vol}\left(C \cap \{z \mid a^T(z - \mathbf{cg}(C)) \leq 0\}\right) \leq (1 - 1/e)\,\mathbf{vol}(C).$$

In other words, a plane passing though the center of gravity of a convex set divides its volume almost equally: the volume division inequity can be at most in the ratio $(1 - 1/e) : 1/e$, *i.e.*, about $1.72 : 1$.

In the CG algorithm we have

$$\mathbf{vol}(\mathcal{P}_k) \leq (1 - 1/e)^k \,\mathbf{vol}(\mathcal{P}_0) \approx 0.63^k \,\mathbf{vol}(\mathcal{P}_0).$$

Now suppose the initial polyhedron lies inside a Euclidean ball of radius R (*i.e.*, it has diameter $\leq 2R$), and the target set contains a Euclidean ball of radius r. Then we have

$$\mathbf{vol}(\mathcal{P}_0) \leq \alpha_n R^n,$$

where α_n is the volume of the unit Euclidean ball in $\mathbf{R}^n$. Since $X \subseteq \mathcal{P}_k$ for each k (assuming the algorithm has not yet terminated) we have

$$\alpha_n r^n \leq \mathbf{vol}(\mathcal{P}_k).$$

Putting these together we see that

$$\alpha_n r^n \leq (1 - 1/e)^k \alpha_n R^n,$$

so

$$k \leq \frac{n \log(R/r)}{-\log(1-1/e)} \approx 2.18 n \log(R/r).$$

We can express this using base-2 logarithms as

$$k \leq \frac{n(\log 2) \log_2(R/r)}{-\log(1-1/e)} \approx 1.51 n \log_2(R/r),$$

in order to compare this complexity estimate with the similar one for the bisection algorithm (4). We conclude that the CG algorithm requires at most $1.51n$ iterations per bit of uncertainty reduction. (Of course, the CG algorithm reduces to the bisection method when $n = 1$.)

Finally, we come to a very basic disadvantage of the CG algorithm: it is *extremely difficult* to compute the center of gravity of a polyhedron in $\mathbf{R}^n$, described by a set of linear inequalities. (It is possible to efficiently compute the center of gravity in very low dimensions, *e.g.*, $n = 2$ or $n = 3$, by triangulation.) This means that the CG algorithm, although interesting, is not a practical cutting-plane method.

Variants of the CG algorithm have been developed, in which an approximate center of gravity, which can be efficiently computed, is used in place of the center of gravity, but they are generally complicated. One recent and very interesting advance in this area is the so-called 'hit-and-run' algorithm, which is a randomized method for computing an approximation of the center of gravity of a convex set. This can be use to create a practical CG method; see, *e.g.*, [**?**].

4.3 MVE cutting-plane method

In the maximum volume inscribed ellipsoid method, due to Tarasov, Khachiyan, and Èrlikh [**?**], we take the next iterate to be the center of the maximum volume ellipsoid that lies in $\mathcal{P}_k$, which can be computed by solving a convex optimization problem [**?**, §8.4.2]. Since the maximum volume inscribed ellipsoid is affinely invariant, so is the resulting MVE cutting-plane method.

For the MVE cutting-plane method, there is also a bound on the volume reduction factor:

$$\frac{\mathbf{vol}(\mathcal{P}_{k+1})}{\mathbf{vol}(\mathcal{P}_k)} \leq 1 - 1/n.$$

In this case the volume reduction factor is dependent on n, and degrades (*i.e.*, increases) for increasing n. An analysis similar to the one given above for the CG algorithm shows that the number of iterations is no more than

$$k \leq \frac{n \log(R/r)}{-\log(1-1/n)} \approx n^2 \log(R/r),$$

provided the initial polyhedron lies in a ball of radius R, and the target set contains a ball of radius r.

4.4 Chebyshev center cutting-plane method

In the Chebyshev center cutting-plane method, due to Elzinga and Moore [**?**], the query point $x^{(k+1)}$ is taken to be the Chebyshev center of the current polyhedron $\mathcal{P}_k$, *i.e.*, the center of the largest Euclidean ball that lies inside it. This point can be computed by solving a linear program [**?**, §8.5.1]. Unlike the other methods described here, the Chebyshev center cutting-plane method is *not* affinely invariant. The Chebyshev center cutting-plane method can be strongly affected by problem scaling or affine transformations of coordinates.

4.5 Analytic center cutting-plane method

The analytic center cutting-plane method (ACCPM) uses as query point the analytic center of $\mathcal{P}_k$, *i.e.*, the solution of the problem

$$\text{minimize} \quad -\sum_{i=1}^{m_0} \log(d_i - c_i^T x) - \sum_{i=1}^{m_k} \log(b_i - a_i^T x),$$

with variable x, where

$$\mathcal{P}_k = \{z \mid c_i^T z \leq d_i,\ i = 1, \ldots, m_0,\ a_i^T z \leq b_i,\ i = 1, \ldots, m_k\}.$$

(We have an implicit constraint that $x \in \mathbf{int}\, \mathcal{P}_k$.) ACCPM was developed by Goffin and Vial [**?**] and analyzed by Nesterov [**?**] and Atkinson and Vaidya [**?**].

ACCPM seems to give a good trade-off in terms of simplicity and practical performance. It will be described in much more detail in a separate set of notes.

5 Extensions

In this section we describe several extensions and variations on cutting-plane methods.

5.1 Multiple cuts

One simple extension is to allow the oracle to return a set of linear inequalities for each query, instead of just one. When queried at $x^{(k)}$, the oracle returns a set of linear inequalities which are satisfied by every $z \in X$, and which (together) separate $x^{(k)}$ and X. Thus, the oracle can return $A_k \in \mathbf{R}^{p_k \times n}$ and $b_k \in \mathbf{R}^{p_k}$, where $A_k z \preceq b_k$ holds for every $z \in X$, and $A_k x^{(k)} \not\preceq b_k$. This means that at least one of the p_k linear inequalities must be a valid cutting-plane by itself. The inequalities which are not valid cuts by themselves are called *shallow cuts*.

It is straightforward to accommodate multiple cutting-planes in a cutting-plane method: at each iteration, we simply append the entire set of new inequalities returned by the oracle to our collection of valid linear inequalities for X. To give a simple example showing how multiple cuts can be obtained, consider the convex feasibility problem

$$\begin{array}{ll} \text{find} & x \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m. \end{array}$$

In §2 we showed how to construct a cutting-plane at x using any (one) violated constraint. We can obtain a set of multiple cuts at x by using information from *any* set of inequalities, provided at least one is violated. From the basic inequality

$$f_j(z) \geq f_j(x) + g_j^T(z - x),$$

where $g_j \in \partial f_j(x)$, we find that every $z \in X$ satisfies

$$f_j(x) + g_j^T(z - x) \leq 0.$$

If x violates the jth inequality, this is a deep cut. If x satisfies the jth inequality, this is a shallow cut, and can be used in a group of multiple cuts, as long as one neutral or deep cut is present. Common choices for the set of inequalities to use to form cuts are

- the most violated inequality, *i.e.*, $\operatorname{argmax}_j f_j(x)$,
- any violated inequality (*e.g.*, the first constraint found to be violated),
- all violated inequalities,
- all inequalities.

5.2 Dropping or pruning constraints

The computation required to find the new query point $x^{(k+1)}$ grows with the number of linear inequalities that describe $\mathcal{P}_k$. This number, in turn, increases by one at each iteration (for a single cut) or more (for multiple cuts). For this reason most practical cutting-plane implementations include a mechanism for dropping or pruning the set of linear inequalities as the algorithm progresses. In the conservative approach, constraints are dropped only when they are known to be redundant. In this case dropping constraints does not change $\mathcal{P}_k$, and the convergence analysis for the cutting-plane algorithm without pruning still holds. The progress, judged by volume reduction, is unchanged when we drop constraints that are redundant.

To check if a linear inequality $a_i^T z \leq b_i$ is redundant, *i.e.*, implied by the linear inequalities $a_j^T z \leq b_j$, $j = 1, \ldots, m$, we can solve the linear program

$$\begin{array}{ll} \text{maximize} & a_i^T z \\ \text{subject to} & a_j^T z \leq b_j, \quad j = 1, \ldots, m, \ j \neq i. \end{array}$$

The linear inequality is redundant if and only if the optimal value is (strictly) smaller than b_i. Solving a linear program to check redundancy of each inequality is usually too costly, and therefore not done.

In some cases there are other methods that can identify (some) redundant constraints, with less computational effort.

$$\mathcal{E} = \{Fu + g \mid \|u\|_2 \leq 1\} \tag{5}$$

is known to cover the current localization polyhedron $\mathcal{P}$. (In the MVE cutting-plane method, such an ellipsoid can be obtained by expanding the maximum volume ellipsoid inside $\mathcal{P}_k$ by a factor of n about its center.) If the maximum value of $a_i^T z$ over $\mathcal{E}$ is smaller than or equal to b_i, *i.e.*,

$$a_i^T g + \|F^T a_i\|_2 \leq b_i,$$

then the constraint $a_i^T z \leq b_i$ is redundant.

In practical approaches, constraints are dropped even when they are not redundant, or at least not known to be redundant. In this case the pruning can actually increase the size of the localization polyhedron. Heuristics are used to rank the linear inequalities in relevance, and the least relevant ones are dropped first. One method for ranking relevance is based on a covering ellipsoid (5). For each linear inequality we form the fraction

$$\frac{b_i - a_i^T g}{\|F^T a_i\|_2},$$

and then sort the inequalities by these factors, with the lowest numbers corresponding to the most relevant, and the largest numbers corresponding to the least relevant. Note that any inequality for which the fraction exceeds one is in fact redundant.

One common strategy is to keep a fixed number N of linear inequalities, by dropping as many constraints as needed, in each iteration, to keep the total number fixed at N. The number of inequalities kept (*i.e.*, N) is typically between $3n$ and $5n$. Dropping non-redundant constraints complicates the convergence analysis of a cutting-plane algorithm, sometimes considerably, so we will not consider pruning in our analysis. In practice, pruning often *improves* the performance, in terms of the number of iterations required to obtain a given accuracy, despite the increase in the localization polyhedron that occurs when non-redundant constraints are dropped. In terms of computational effort, which depends on the number of iterations as well as the number of inequalities in the localization polyhedron, the improvement due to pruning can be dramatic.

5.3 Nonlinear cuts

It is straightforward to extend the idea of cutting-planes, which are described by linear inequalities that hold for each point in X, to cutting-sets, which are described by more complex convex inequalities. For example, a quadratic cut has the form of a convex quadratic inequality that every $z \in X$ is known to satisfy. In this case the localization set is no longer a polyhedron. However, many of the same ideas work; for example, the analytic center of such a set is readily computed, so ACCPM can be extended to handle nonlinear cuts.

6 Epigraph cutting-plane method

For a convex optimization problem (as opposed to a quasiconvex problem), it is usually better to apply a cutting-plane method to the *epigraph form* of the problem, rather than directly to the problem.

We start with the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{6}$$

where $f_0, \ldots, f_m$ are convex. In the basic cutting-plane method outlined above, we take the variable to be x, and the target set X to be the set of optimal points. Cutting-planes are found using the methods described in §2.

Suppose instead we form the equivalent epigraph form problem

$$\begin{array}{ll} \text{minimize} & t \\ \text{subject to} & f_0(x) \leq t \\ & f_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{7}$$

with variables $x \in \mathbf{R}^n$ and $t \in \mathbf{R}$. We take the target set to be the set of optimal points for the epigraph problem (7), *i.e.*,

$$X = \{(x, f_0(x)) \mid x \text{ optimal for (6)}\}.$$

Let us show how to find a cutting-plane, in $\mathbf{R}^{n+1}$, for this version of the problem, at the query point (x, t). First suppose x is not feasible for the original problem, *e.g.*, the jth constraint is violated. Every feasible point satisfies

$$0 \geq f_j(z) \geq f_j(x) + g_j^T(z - x),$$

where $g_j \in \partial f_j(x)$, so we can use the cut

$$f_j(x) + g_j^T(z - x) \leq 0$$

(which doesn't involve the second variable). Now suppose that the query point x is feasible. Evaluate a subgradient $g \in \partial f_0(x)$. If $g = 0$, then x is optimal; otherwise, for any $(z, s) \in \mathbf{R}^{n+1}$ feasible for the problem (7), we have

$$s \geq f_0(z) \geq f_0(x) + g^T(z - x)$$

Since x is feasible, $f_0(x) \geq p^\star$, which is the optimal value of the second variable. Thus, we can construct *two* cutting-planes in (z, s):

$$f_0(x) + g^T(z - x) \leq s, \qquad s \leq f_0(x).$$

7 Lower bounds and stopping criteria

In this section we describe a general method for constructing a lower bound on the optimal value of a convex problem, assuming we have evaluated a subgradient of its objective and constraint functions at some points. The method involves solving a linear program using

data collected from the subgradient evaluations. This method can be used in cutting-plane methods to give a non-heuristic stopping criterion. In the analytic center cutting-plane method, a lower bound based on this one can be computed very cheaply at each iteration.

Consider a convex function f. Suppose we have evaluated f and a subgradient of f at points $x^{(1)}, \ldots, x^{(q)}$. We have, for all z,

$$f(z) \geq f(x^{(i)}) + g^{(i)T}(z - x^{(i)}), \quad i = 1, \ldots, q,$$

and so

$$f(z) \geq \hat{f}(z) = \max_{i=1,\ldots,q} \left(f(x^{(i)}) + g^{(i)T}(z - x^{(i)}) \right).$$

The function $\hat{f}$ is a convex piecewise-linear global underestimator of f.

Now suppose that we use a cutting-plane method to solve the problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Cx \preceq d, \end{array} \tag{8}$$

where f_i are convex. After k steps, we have evaluated the objective or constraint functions, along with a subgradient, k times (or more, if multiple cuts are used). As a result, we can form piecewise-linear approximations $\hat{f}_0, \ldots, \hat{f}_m$ of the objective and constraint functions. Now we form the problem

$$\begin{array}{ll} \text{minimize} & \hat{f}_0(x) \\ \text{subject to} & \hat{f}_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Cx \preceq d. \end{array} \tag{9}$$

Since the objective and constraint functions are piecewise-linear, this problem can be transformed to a linear program. Its optimal value is a lower bound on $p^\star$, the optimal value of the problem (8), since $\hat{f}_i(x) \leq f_i(x)$ for all x and $i = 0, \ldots, m$.

Computing this lower bound requires solving a linear program, and so is relatively expensive. In ACCPM, however, we can easily construct a lower bound on the problem (9), as a by-product of the analytic centering computation, which in turn gives a lower bound on the original problem (8).

Acknowledgments

Lin Xiao and Joëlle Skaf helped develop the material here.