A Combinatorial Active Set Algorithm for Linear and Quadratic Programming

Andrew J. Miller

A Combinatorial Active Set Algorithm for Linear and Quadratic Programming

Andrew J. Miller *

University of Wisconsin-Madison
1513 University Avenue, ME 3242
Madison, WI, USA 53706-1572
ajmiller5@wisc.edu

Abstract. We propose an algorithm for linear programming, which we call the Sequential Projection algorithm. This new approach is a primal improvement algorithm that keeps both a feasible point and an active set, which uniquely define an improving direction. Like the simplex method, the complexity of this algorithm need *not* depend explicitly on the size of the numbers of the problem instance. Unlike the simplex method, however, our approach is not an edge-following algorithm, and the active set need not form a row basis of the constraint matrix. Moreover, the algorithm has a number of desirable properties that ensure that it is not susceptible to the simple pathological examples (e.g., the Klee-Minty problems) that are known to cause the simplex method to perform an exponential number of iterations.

We also show how to randomize the algorithm so that it runs in an expected time that is on the order of $mn^{2\log n}$ for most LP instances. This bound is strongly subexponential in the size of the problem instance (i.e., it does not depend on the size of the data, and it can be bounded by a function that grows more slowly than 2^m, where m is the number of constraints in the problem). Moreover, to the best of our knowledge, this is the fastest known randomized algorithm for linear programming whose running time does not depend on the size of the numbers defining the problem instance.

Many of our results generalize in a straightforward manner to mathematical programs that maximize a concave quadratic objective function over linear constraints (i.e., quadratic programs), and we discuss these extensions as well.

Key words: Linear Programming, Quadratic Programming, Polyhedral Combinatorics, Asymptotic Complexity, Randomized Algorithm

* This research is being supported in part by NSF grant CMMI-0521953 and AFOSR grant FA9550-07-1-0389

1 Introduction

We consider a linear program (LP) of the form

$$\max_{x \in \mathbb{R}^n} \; c^T x \tag{1}$$

$$\text{s.t.} \; Ax \leq b. \tag{2}$$

Here that we assume that $c \in \mathbb{Z}^n$, $b \in \mathbb{Z}^m$, and $A \in \mathbb{Z}^m \times \mathbb{Z}^n$. Note that we also assume that all bounds are incorporated in the constraint matrix. For simplicity, we let $N := \{1, ..., n\}$ and $M := \{1, ..., m\}$.

Linear programming is one of the fundamental problems encountered in the fields of operations research, mathematics, computer science, and many others. One of the many reason why LP is so important is because the use of LP models, algorithms, and theory is foundational in the development of so many methodologies and results for integer programming and combinatorial optimization problems. The contributions made to our understanding of LP problems and of how to solve them over the years have been myriad, and one cannot possibly hope to justice to all of the in a full paper, let alone the introduction of an extended abstract. Nevertheless, we discuss some references to situate our work. The famous simplex method was first introduced by Dantzig (see e.g. [4, 5]); for many years it was widely believed to have a worst case complexity that was polynomial. This conjecture was proven false by [17], where the pivoting rule considered was the largest coefficient rule. Subsequently a number of other pivoting rules were shown to yield an exponential number of pivots (see e.g. [1, 3, 11, 22, 28]). The worst case complexity of LP was unresolved until Khachiyan [16] proposed the ellipsoid algorithm, which is a geometric (rather than combinatorial) algorithm (other references for the ellipsoid algorithm, and for the numerous implications of its discovery in various areas of optimization, include [2, 10, 12]). While theoretically efficient, however, the ellipsoid has not proven practical for solving LP problems.

The first family of polynomial time algorithms that also proved practically efficient for solving LP problems are interior point methods (e.g., [15, 20, 21, 27]). Like the ellipsoid algorithm, these algorithms are *weakly* polynomial, i.e., the number of operations performed depends on the amount of space required to encode the problem. Unlike the ellipsoid algorithm, these algorithms have proven to be computationally practical, on many instances rivalling or even outperforming the simplex method. For other approaches to solving LP, see (among a host of others) [6, 7, 8, 24].

The similarities between *quadratic programs* (i.e., problems with linear constraints and a *concave quadratic* objective functions) have long been recognized. For example, it has been known since [26] that the simplex method can be extended to quadratic programming (QP), and most commercial LP/MIP solvers now have a simplex algorithm implemented for QPs as well. Such algorithms move from basis to basis, where the Karush-Kuhn-Tucker conditions form part of the constraint matrix. Numerous interior point algorithms for QP have also been proposed; see, e.g., [25, 27].

In this paper we propose a new algorithm for LP, which we call the Sequential Projection algorithm. Like interior point methods, this algorithm does not necessarily follow edges of the feasible polyhedron to the optimal solution. Unlike them, however, it does seek to move within faces of the polyhedron towards optimality. Like the simplex algorithm, it solves systems of linear equations to calculate an improving direction. Unlike the simplex method, however, the system need not define a basis of the constraint matrix, and the algorithm is not confined to move only between extreme points.

In Section 2 we will provide some theoretical background and discuss standard assumptions we make. In Section 3 we prove some fundamental results necessary to prove correctness of the algorithm and define some important properties, including correctness and finiteness proofs and asymptotic worst-case running times. While we have not yet been able to obtain an interesting general bound for the deterministic algorithm, it is easy to show that a simple randomization of the algorithm leads to an expected worst-case running time that is always strongly subexponential. Moreover, to the best of our knowledge, this expected running time is faster than that known for any randomized simplex method, including those with known strongly subexponential complexity (see, e.g., [9, 13, 14, 18]). Section 4 contains this expected worst-case analysis. Finally, in Section 5, we discuss how many of our results (including the correctness, and perhaps even the complexity, of the randomized algorithm) generalize easily to QP problems.

2 Background theory and assumptions

Given our formulation, the dual of LP (DLP) is given by

$$\begin{aligned} \min_{\mu \in \mathbb{R}^m} \quad & b^T \mu \\ \text{s.t.} \quad & A^T \mu = c; \mu \geq 0. \end{aligned}$$

We define the row vectors

$$A_i = [a_{i1} \ a_{i2} \ \ldots \ a_{in}], i \in M,$$

so that the i^{th} constraint of (2) can be written as $A_i x \leq b_i$. We refer to this constraint as (A_i, b_i), or simply as constraint i. Let $P = \{x \in \mathbb{R}^n : Ax \leq b\}$; this is the feasible region, which is a polyhedron. For each constraint $i \in M$, define $H_i = \{x \in \mathbb{R}^n : A_i x = b_i\}$, the hyperplane defined by constraint i, and $F_i = H_i \cap P$, the face of the polyhedron P defined by i. For any set of indices $i_1, ..., i_{h'} \in M$, $h' \leq n$, we use $H_{i_1,...,i_{h'}}$ to denote $\bigcap_{h=1}^{h'} H_{i_h}$; we will likewise let $F_{i_1,...,i_{h'}} := \bigcap_{h=1}^{h'} F_{i_h}$, and let $P_{i_1,...,i_{h'}} = \{x \in \mathbb{R}^n : A_{i_h} x \leq b_{i_h}, h = 1, ..., h'\}$.

We may make a number of fairly general assumptions for the purpose of simplifying the presentation.

Assumption 1 *(**Full dimensionality**) $dim(P) = n$.*

Among other things, this assumption implies that the representation of each of the inequalities of (2) is unique (subject to scaling), and that there exists an *interior point* $\bar{x}$ in P; i.e., a point $\bar{x} \in P$ such that

$$\sum_{j \in N} A_{ij} x_j < b_i, \forall i \in M.$$

Assumption 2 *(**Upper bound**) We are given an upper bound c_0 on the value of $c^T x$, so that $c^T x \leq c_0$ is valid inequality for LP.*

Note that this assumption can be seen essentially as assuming that the dual LP is full dimensional, and any dual feasible solution can provide the bound assumed.

Assumption 3 ***(Primal nondegeneracy)*** *All extreme points of P are defined by only n linear inequalities of the system (2).*

Note that this does *not* imply that (2) contains no redundant constraints. It does imply that all of the faces

$$F_i = \{x \in P : \sum_{j \in N} A_{ij} x_j = b_i\}$$

are either facets of P (i.e., have dimension $n-1$) or are empty (i.e., have dimension -1).

Assumption 4 ***(Dual nondegeneracy)*** *For all extreme points $\bar{\mu}$ of $DP := \{\mu \in \mathbb{R}^m_+ : A^T\mu = c\}$, let $\mathcal{I}_0 := \{i \in M : \bar{\mu}_i = 0\}$. Then $|\mathcal{I}_0| = m - n$.*

Note that this assumption implies that there are no faces of P that are parallel to $H_0 := \{x \in \mathbb{R}^n : c^T x = c_0$.

The last two assumptions are not restrictive, as any LP can be made primal and dual nondegenerate by perturbing the right hand sides and objective coefficients, resp., by small amounts. The first two assumptions are not restrictive, either; they can be relaxed by applying the algorithm defined in this extended abstract to a modified version of the primal (resp., dual) LP in which any feasible solution (if one exists) to the original problem is optimal to the modified one.

We will use $||x||$ to denote the standard Euclidean norm of $x \in \mathbb{R}^n$. If $\hat{x} \in \mathbb{R}^n$ and $H_{i_1,...,i_{h'}}$ is the intersection of the hyperplanes defined by the inequalities $i_1, ..., i_{h'}$, $h' < n$, we define the projection of $\hat{x}$ onto $H_{i_1,...,i_{h'}}$

$$\begin{aligned} \text{proj}_{H_{i_1,\ldots,i_h}}(\hat{x}) &= \text{argmin}_{x \in H_{i_1,\ldots,i_h}}(||x - \hat{x}||) \\ &= \begin{array}{ll} \text{argmin}_{x \in \mathbb{R}^n} & (x_j - \hat{x}_j)^2 \\ \text{s.t.} & A_{i_h} x = b_{i_h}, h = 1, ..., h'. \end{array} \end{aligned}$$

We can solve the quadratic problem thus defined by applying the Karush-Kuhn-Tucker conditions to define the following equality system, which has $n + h' = \mathcal{O}(n)$ variables and the same number of constraints:

$$A_{i_h} x = b_{i_h}, h = 1, ..., h';\ 2(x_j - \hat{x}_j) + \sum_{h=1}^{h'} A_{i_h j}\mu_h = 0, j = 1, ..., n;\ x \in \mathbb{R}^n, \mu \in \mathbb{R}^{h'}.$$

Since we will need to solve systems of equations like this repeatedly, our results depend on the definition of what it means to be able to perform the operations needed to do so in strongly polynomial time. Our definition includes the following assumption.

Assumption 5 (Strong polynomiality of basic arithmetical operations) *Given a system of p linear independent equalities in $\mathbb{R}^p$ $Dx = d$, where $D \in \mathbb{Z}^{p \times p}$ and $d \in \mathbb{Z}^p$, the unique solution $x = D^{-1}d$ can be found in time $\mathcal{G}(p)$, which is some polynomial that does* not *depend on the size of the entries of D or d.*

3 The Sequential Projection algorithm

3.1 Key theorems

Theorem 1. *Let $\hat{x}$ be a point in P. Let $\bar{x} = proj_{H_0}(\hat{x})$. If $\bar{x} \in P$, then it is an optimal solution to LP. Otherwise, for all inequalities $i \in M$ such that $A_i\bar{x} > b_i$, define*

$$\lambda_i := \frac{b_i - A_i\hat{x}}{A_i\bar{x} - A_i\hat{x}}$$

(Clearly at least one such inequality must exist if $\bar{x} \notin P$.) Choose

$$k = argmin_{i \in M: A_i\bar{x} > b_i}\{\lambda_i\},$$

and define

$$\hat{x}^k = \lambda_k\bar{x} + (1 - \lambda_k)\hat{x}.$$

If $\hat{x}$ is an interior point of P, then $\hat{x}^k \in F_k$. If $\hat{x} \in F_{i_1,...,i_{h'}}$ for some subset of inequalities $i_1, ..., i_{h'} \in M$, $h' < n$, then $\hat{x}^k \in F_{i_1,...,i_{h'}} \cap F_k$.

For the next two theorems we will need the following definition.

Definition 1. *For any $i \in M$ such that $F_i \neq \emptyset$ and any $\bar{x} \in F_i$, then $d \in \mathbb{Z}^n$ is an **improving direction** for $\bar{x}$ in F_i if there exists some $\epsilon > 0$ such that, for all $\alpha : 0 < \alpha < \epsilon$, 1) $c^T d > 0$; and 2) $\bar{x} + \alpha d \in F_i$.*

Note that the terms of the definition also imply that if d is an improving direction for $\bar{x}$ in F_i, then there exists some $\beta > 0$ for which $\bar{x} + \beta d \in H_0 \cap H_i$.

Theorem 2. *Let $\mathcal{A} = \{i_1, ..., i_k\} \subset M$ be any set of inequalities with $1 \leq k \leq n$. If $1 \leq k < n$, then $H_0 \cap H_{\mathcal{A}} \neq \emptyset$.*

Proof: This follows from the assumption of dual nondegeneracy. □

Theorem 3. *Let $\mathcal{A} = \{i_1, ..., i_n\} \subset M$ be any set of n inequalities. $H_{\mathcal{A}} = \hat{x}$ is the unique solution to $B^{\mathcal{A}}x = b^{\mathcal{A}}$, where*

$$B = \begin{bmatrix} A_{i_1} \\ \vdots \\ A_{i_n} \end{bmatrix} = \begin{bmatrix} A_{i_1 1} & A_{i_1 2} & \dots & A_{i_1 n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{i_n 1} & A_{i_n 2} & \dots & A_{i_n n} \end{bmatrix} \text{ and } b^{\mathcal{A}} = \begin{bmatrix} b_{i_1} \\ b_{i_2} \\ \vdots \\ b_{i_n} \end{bmatrix}.$$

Moreover, let $\hat{\mu} \in \mathbb{R}^m$ be defined by

$$\hat{\mu}_i = \begin{cases} \left[B^{\mathcal{A}^{-T}}c\right]_i = \sum_{j=1}^{n}[B^{\mathcal{A}^{-T}}]_{ij}c_j, & \text{if } i \in \mathcal{A} \\ 0, & \text{if } i \notin \mathcal{A}. \end{cases}$$

If $\hat{x} \in P$, the following statements are true:

1. *$\hat{x}$ is an optimal primal solution and $\hat{\mu}$ is an optimal dual solution if and only if $\hat{\mu}_i \geq 0, \forall i \in \mathcal{A}$.*

2. *If $\hat{x}$ is* not *optimal, then there exists an improving direction for $\hat{x}$ in $F_{\mathcal{A}^+}$, where $\mathcal{A}^+ := \{i \in \mathcal{A} : \hat{\mu}_i \geq 0\}$.*
3. *If $\hat{x}$ is* not *optimal, then let $\mathcal{A}^- = \{i : \mu_i < 0\}$. If $2 \leq |\mathcal{A}^-| \leq n-2$, then each of the directions*

$$proj_{H_{\mathcal{A}\setminus i'} \cap H_0}(\hat{x}) - \hat{x}, i' \in \mathcal{A}^-$$

is an improving direction for $\hat{x}$ in each of the faces $F_i, i \in \mathcal{A} \setminus i'$.

Proof: Statement 1 follows from LP complementary slackness. For statement 2, observe that if there exist any $i \in \mathcal{A} : \hat{\mu}_i < 0$, then

$$\max_{x \in P}\{c^T x : A_i x = b_i, i \in \mathcal{A} : \hat{\mu}_i > 0\} > \max_{x \in P}\{c^T x : A_i x = b_i, i \in \mathcal{A}\},$$

and the result follows. Similarly for 3, under the given conditions let

$$x^{i'} = \text{argmax}_{x \in P}\{c^T x : A_i x = b_i, i \in \mathcal{A} \setminus i'\}.$$

Then we have that $c^T x^{i'} = \max_{x \in P}\{c^T x : A_i x = b_i, i \in \mathcal{A} \setminus i'\} > c^T \hat{x}$, and the result follows from the fact that $x' \in F_i$ for each $i \in \mathcal{A} \setminus i'$.

3.2 The framework of the algorithm

Algorithm 1 illustrates the overall algorithmic framework.

Algorithm 1 The Sequential Projection Algorithm

Given: A system of inequalities (A, b) with $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$; an objective function $c \in \mathbb{Z}^n$ and upper bound $c_0 \in \mathbb{R}$; and an interior point $\hat{x} \in P$.
Define $\bar{x} = \text{proj}_{H_0}(\hat{x})$.
If $\bar{x} \in P$ **then**
  Terminate; $\bar{x}$ is optimal.
Else for all $i \in M : A_i \bar{x} > b_i$
  Compute

$$\lambda_i := \frac{b_i - A_i \hat{x}}{A_i \bar{x} - A_i \hat{x}}.$$

Choose $i_{11} = \text{argmin}_{i \in M : A_i \bar{x} > b_i}\{\lambda_i\}$.
Define $\hat{x}^{11} = \lambda_{i_1} \bar{x} + (1 - \lambda_{i_1})\hat{x}$.
Initialize $h = 1$, $\mathcal{P}(h,1) = \mathcal{A}(h,1) = \{i_{h1}\}$, and $k = 1$.
While $\bar{x} \notin P$ **do**

  Process $\mathcal{A}(h,k)$, and $\hat{x}^{hk}$ and either 1) **find** an optimal solution $\bar{x} \in F_{\mathcal{A}(h,k)}$ or 2) **find** a number $k(h+1) : 1 \leq k(h+1) < n$, a set of inequalities $\mathcal{A}(h+1, k(h+1)) \subset M$ such that $|\mathcal{A}(h+1, k(h+1))| = k(h+1)$, and a point $\hat{x}^{h+1,k(h+1)} \in F_{\mathcal{A}(h+1,k(h+1))}$.

End-do

The algorithm is clearly correct if the following two conditions hold:

1. The **Process** and **Find** (**PF**) procedure in the **While** loop, each time it is called, correctly outputs either 1) an optimal solution $\bar{x} \in F_{\mathcal{A}(h,k)}$, or 2) an integer $k(h+1) < n$, a set of inequalities $\mathcal{A}(h+1, k(h+1)) \subseteq M$ such that $|\mathcal{A}(h+1, k(h+1))| = k(h+1)$, and a feasible point $\hat{x}^{h+1,k(h+1)} \in F_{\mathcal{A}(h+1,k(h+1))}$.
2. The algorithm terminates (that is, the **While** loop does not repeat forever).

Determining the complexity of the work done outside the **PF** step is straightforward. The total work before the **While** loop involves solving a system of equations ($\mathcal{G}(n)$) and a number of simple vector arithmetic operations. Thus, to show that the SP algorithm is correct, we need to show that the **PR** procedure 1) it correctly executes the necessary tasks; 2) repeats only a finite number of times before finding the optimal solution. We specify a procedure that meets these conditions in Algorithm 3.2.

Note that we let $k(h+1)$ denote the size of the active set at the beginning of the $h+1^{st}$ execution of the **PF** procedure. Thus $k(h+1) = |\{i \in \mathcal{A}(h,n) : \mu_i^k \geq 0\}| = |\mathcal{A}^-(h,n)|$, and $i^{h+1,k(h+1)}$ denotes the first inequality added to the active set during the $h+1^{st}$ execution of that procedure.

Algorithm 2 Process and Find Procedure

Given: A set of inequalities $\mathcal{A}(h,k) = \{i_{h11}, i_{h12}, ..., i_{h1k}\} \subseteq P$ such that $1 \leq k \leq n-1$, and a point $\hat{x}^{hk} \in F_{\mathcal{A}(h,k)}$.
While $k < n$ **do**
Define $\bar{x}^{kh} := \text{proj}_{H_0 \cap H_{\mathcal{A}(h,k)}}(\hat{x}^{hk})$.
If $\bar{x}^{kh} \in P$ **then**
$\bar{x}^{hk}$ is optimal; **set** $\bar{x} = \bar{x}^{hk}$ and **exit** the procedure.
Else for all $i \in M$ such that $A_i\bar{x} > b_i$
Compute

$$\lambda_i := \frac{b_i - A_i\hat{x}^{hk}}{A_i\bar{x}^{hk} - A_i\hat{x}^{hk}}.$$

Choose $i_{h,k+1} = \text{argmin}_{i \in M: A_i\bar{x} > b_i}\{\lambda_i\}$.
Define $\hat{x}^{h,k+1} = \lambda_{i_{h,k+1}}\bar{x}^{hk} + (1 - \lambda_{i_{h,k+1}})\hat{x}^{hk}$.
Set $\mathcal{A}(h,k+1) = \mathcal{A}(h,k) \cap i_{h,k+1}$, and **increment** k.
end-do
Define B to be the $n \times n$ matrix

$$B = \begin{bmatrix} A_{i_h 1} \\ \vdots \\ A_{i_{h,n}} \end{bmatrix} = \begin{bmatrix} A_{i_{h1}1} & A_{i_{h1}2} & \dots & A_{i_{h1}n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{i_{hn}1} & A_{i_{hn}2} & \dots & A_{i_{hn}n} \end{bmatrix}.$$

Define $\mu^h = B^{-T}c$.
If $\mu_j^h \geq 0, j = 1, ..., n$, **then** $\hat{x}^{hn}$ is optimal; **set** $\bar{x} = \hat{x}^{hn}$ and **exit** the procedure;
Else reset $k = |\mathcal{A}^+(h,n)|$, **set** $\mathcal{A}(h+1,k) = \mathcal{A}^+(h,n)$, and
set $i_{h+1,1}, ..., i_{h+1,...,k}$ to be the inequalities of $\mathcal{A}(h+1,k)$.
Exit the procedure.

Also note that each time we exit the **While** loop in the **PF** procedure, at that point $k = n$ and $\hat{x}^{hn} = B^{-1}b^h$, where

$$b^h = \begin{bmatrix} b_{i_{h1}} \\ b_{i_{h2}} \\ \vdots \\ b_{i_{hn}} \end{bmatrix}.$$

It is not difficult to see that this procedure, and therefore the algorithm, always maintains a feasible point $\hat{x}^{hk}$ and an active set $\mathcal{A}(h,k) = \{i_{h_1}, ..., i_{h_k}\}$ such that $A_i x = b_i$ for every $i \in \mathcal{A}(h,k)$. Moreover, it is also not difficult to see that the algorithm does not terminate incorrectly (i.e., without having found an optimal solution). Therefore, to prove correctness it suffices to show that the SP algorithm terminates.

Lemma 1. *During each execution of the **PF** procedure, the total of times the **While** loop is performed is less than n.*

The next lemma specifies conditions under which we can be sure that a given inequality will never again enter the active set. To do so, we consider

$$P_{\mathcal{A}(h',k')} = \{x \in \mathbb{R}^n : A_i x \leq b_i, i \in \mathcal{A}(h',k')\},$$

the feasible polyhedron with respect to the set of active constraints during the k'^{th} time that the inner loop is performed within the h'^{th} call to the **PF** procedure.

Lemma 2. *Let h' be any iteration of the SP algorithm, and let $k' = k(h')$. Consider any $i_* \in M \setminus \mathcal{A}(h',k')$. If $P_{\mathcal{A}(h',k')} \cap H_{i_*} \cap H_0 = \emptyset$, then either*

1. *$P_{\mathcal{A}(h',k'),i_*} \cap H_0 = \emptyset$, in which case $k'+1 = n$ and the point $H_{\mathcal{A}(h',k'),i_*}$ is either optimal or infeasible; or*
2. *$i_* \notin \mathcal{A}(h,k)$ will be true for any $h \geq h'$ and any $k : 1 \leq k \leq n$.*

Proof: Dual nondegeneracy implies that $P_{\mathcal{A}(h',k')} \cap H_0 \neq \emptyset$. The fact that $P_{\mathcal{A}(h',k')} \cap H_{i_*} \cap H_0 = \emptyset$ thus implies one of two things. *Either*

1. $P_{\mathcal{A}(h',k')} \cap \{x \in A_{i_*} x \leq b_{i_*}\} \cap H_0 = P_{\mathcal{A}(h',k'),i_*} \cap H_0 = \emptyset$.
   In this case $P_{\mathcal{A}(h',k'),i_*}$ will be a cone pointed at $H_{\mathcal{A}(h',k'),i_*}$ that does not extend forever in an improving direction (i.e., toward H_0). Therefore $k'+1 = n$ must be true, and the point $H_{\mathcal{A}(h',n-1),i_*}$ is either the optimal solution or infeasible.
   Or
2. $A_{i_*} x \leq b_{i_*}$ is implied by the constraints of $\mathcal{A}(h',k')$ within H_0. In this case, since $A_{i_*}\hat{x}^{hk} < b_{i_*}$ is also true, $\hat{x}^{hk}$ will never again lie in F_{i_*} for any $h \geq h'$, $1 \leq k \leq n$. □

It is not difficult to see that we can extend this Lemma to apply to faces that do not define facets of P.

Lemma 3. *Let h' and $k' < n$ be the values of h and k at any point at which the algorithm executes the first command of the **While** loop of the **PR** procedure. Consider any set $\mathcal{A}^* = \{i_1, ..., i_{k_*}\} \in M \setminus \mathcal{A}(h',k')$ with the property that $k^* = |\mathcal{A}^*| \leq n - k'$. If $P_{\mathcal{A}(h',k')} \cap H_{\mathcal{A}^*} \cap H_0 = \emptyset$, then either*

1. *$k^* - k' = n$, $P_{\mathcal{A}(h',k')} \cap P_{\mathcal{A}^*} \cap H_0 = \emptyset$, and the point $H_{\mathcal{A}(h',n),\mathcal{A}^*}$ is either the optimal solution or infeasible;*
   or

2. $|\mathcal{A}(h,k) \cap \mathcal{A}^*| \leq k_* - 1$ *will always be true for any* $h \geq h'$ *and any* $k : 1 \leq k \leq n$. *Moreover, if for all* $i \in \mathcal{A}^*$ *there exists some number* $h(i) > h'$ *such that* $i \in \mathcal{A}(h(i), n)$, *then we must have that* $|\mathcal{A}(h,k) \cap \mathcal{A}(h',k')| \leq k' - 1$ *for all* $h \geq h^*$ *and* $k : 1 \leq k \leq n$, *where*

$$h^* = \max_{i \in \mathcal{A}^*} [\min\{h : i \in \mathcal{A}(h,n)\}] .$$

In words, the lemma says that if $P_{\mathcal{A}(h',k') \cap \mathcal{A}^*}$ is a cone that has no intersection with H_0, then all of the inequalities of $\mathcal{A}^*$ cannot simultaneously be a part of the active set. Moreover, if all of the inequalities do enter the active set at some point in the algorithm (not simultaneously but *in any order*), the hyperplane $H_{\mathcal{A}(h',k')}$ will itself itself never again intersect the active set.

The theorem next will help to prove that the algorithm terminates, and will be useful in proving complexity bounds later.

Theorem 4. *Let* h' *and* $h' + 1$ *be any two consecutive calls of the* ***PF*** *procedure. Let* $\mathcal{A}^+(h') := \{i \in \mathcal{A}(h', n) : \mu_i^h \geq 0\}$ *and* $k_+ := \mathcal{A}^+(h')$; *also let* $\mathcal{A}^-(h', n) := \{i \in \mathcal{A}(h', n) : \mu_i^h < 0\}$ *be nonempty, and let* $k_- := |\mathcal{A}^-(h', n)| > 0$. *Then*

1. $|\mathcal{A}(h,k) \cap \mathcal{A}^-(h', n)| \leq k_- - 1$ *will always be true for any* $h \geq h'$ *and any* $k : 1 \leq k \leq n$.
2. *If in addition* $\mathcal{A}^+(h') \subseteq \mathcal{A}(h,k)$, *then* $|\mathcal{A}(h,k) \cap \mathcal{A}^-| \leq k_- - 2$ *will be true.*
3. *If for all* $i \in \mathcal{A}^-(h', n)$ *there exists some number* $h(i) > h'$ *such that* $i \in \mathcal{A}(h(i), n)$, *then we must have that* $|\mathcal{A}(h,k) \cap \mathcal{A}^+(h')| \leq k_+ - 1$ *for all* $h \geq h^*$ *and* $k : 1 \leq k \leq n$, *where*

$$h^* = \max_{i \in \mathcal{A}^-(h',n)} [\min\{h : i \in \mathcal{A}(h,n)\}] .$$

Proof: Statements 1 and 3 are immediate from Lemma 3. Statement 2 follows from the fact that $P_{\mathcal{A}^-(h',n)} \cap H_{\mathcal{A}^+(h') \setminus i_{h'n}}$ is a $k'_- - 1$-dimensional simplex for which the facets all intersect at the point $\bar{x}^{h',n-1}$, which is infeasible only with respect to inequality $i_{h',n}$. □

There are $2m^{\frac{n}{2}}$ possible distinct active sets of size n (one for each extreme point, see, e.g. [19, 23]) that the algorithm can define. Theorem 4 implies that each one will be defined in the algorithm at most once. We have therefore proven the following theorem.

Theorem 5. *The SP algorithm correctly solves LP in finite time.*

3.3 Complexity

Lemma 4. *There can be at most* m *calls to the* ***PF*** *procedure in which* $\mathcal{A}^-(h,n) = 1$.

Proof: Immediate from Theorem 4. □

Using Theorem 4, we can generalize this lemma to obtain a bound on the total number of iterations in which $\mathcal{A}^-(h,n) \leq g(m,n)$, for some nondecreasing function $g(m,n)$.

Lemma 5. *The number of calls to the* ***PF*** *procedure in which* $|\mathcal{A}^-(h,n)| \leq g(m,n)$ *is* $2\binom{m}{g(m,n)}$, *where* $g(m,n)$ *is any nondecreasing function of* m *and* n *that satisfies* $1 \leq g(m,n) < n$.

Corollary 1. *Let $f(m,n)$ be a function bounding the number of consecutive iterations of the **PF** procedure in which $\mathcal{A}^-(h,n) > g(m,n)$. Then the SP algorithm runs in $\mathcal{O}\left(f(m,n)m^{g(m,n)}n\mathcal{G}(n)\right)$ time.*

Therefore the SP algorithm will run in strongly polynomial time if there exist some functions $f(m,n)$ and $g(m,n)$ as defined above such that $f(m,n)$ is a polynomial in m and n, and $g(m,n)$ is bounded by some constant for all m and n.

3.4 Worst-case complexity

In this section we analyze how fast the function $f(m,n)$ defined in Corollary 5 can grow if $g(m,n) = K$ for any integer K. So let $\alpha_{ik}, i,k = K,...,n-K$ denote the maximum number of consecutive iterations in which $k \leq \mathcal{A}^-(h,n) \leq n-k$. Then we can show that, for $i' = K,...,n-K$,

$$\begin{aligned}\alpha_{n-i',i'} &\leq \lceil\frac{m-n+i}{i}\rceil + \lceil\frac{m-n+i}{i}\frac{i-2}{i}\rceil + \lceil\frac{m-n+i}{i}\frac{i-2}{i}\frac{i-4}{i}\rceil \\ &\quad ... + \lceil\frac{m-n+i}{i}\frac{(i-2)(i-4)...4}{i^{2i-4}}\rceil + \lceil\frac{m-n+i}{i}\frac{(i-2)(i-4)...4\cdot 2}{i^{2i-2}}\rceil \\ &\leq (\lceil\frac{m-n}{i'}\rceil+1)(\sum_{i=0}^{\frac{i'-1}{2}}\frac{i'^{2i}}{2^{2i}i'^{2i}}) \;=\; (\lceil\frac{m-n}{i'}\rceil+1)(\sum_{i=0}^{\frac{i'-1}{2}}\frac{1}{2^{2i}}) \;\leq\; 2(\lceil\frac{m-n}{i'}\rceil+1).\end{aligned}$$

Also, for any $i = K,...,n-K-1$ and $k = K,...,n-i-1$, we can show that

$$\alpha_{i,k} \leq \alpha_{i,k+1} + (m-n+i)\alpha_{i+1,k}.$$

Since the number of sets of inequalities of size k that leave the active set is bounded by $\frac{n-i-k-1}{2}$, we have that

$$\alpha_{i,k} = \mathcal{O}((m-n)^{\lfloor\frac{n-i-k-1}{2}\rfloor}),\ i = \lceil\frac{n}{2}\rceil,...,n-2K-1, k = K,...,n-2i-1.$$

Finally, since the total number of active sets in which $\mathcal{A}^+(h,n) = n-k$ is limited by $(m-n)^{\lfloor\frac{m-n-k-1}{2}\rfloor}$, we also have

$$\alpha_{i,k} = \mathcal{O}((m-n)^{\lfloor\frac{m-n-k-1}{2}\rfloor}),\ i = \lceil\frac{n}{2}\rceil,...,n-2K-1, k = K,...,n-2i-1.$$

Theorem 6. *For any integer $K : 2 \leq K \leq \lfloor\frac{n}{2}\rfloor$, the maximum number of consecutive iterations of the SP algorithm in which $\mathcal{A}^-(h,n) \leq K$ must be less than both $(m-n)^{\lfloor\frac{n-2K-1}{2}\rfloor}$ and $(m-n)^{\lfloor\frac{m-n-2K-1}{2}\rfloor}$.*

Note: Figure out how to get this down to m - n - 2K. The value of K that minimizes $K(n-2K-1)$ is $K = \frac{n-1}{2}$. Therefore we have

Theorem 7. *If $2n \leq m$, the SP algorithm runs in*

$$\min_{K:2\leq K\leq\lfloor\frac{n-1}{2}\rfloor} \mathcal{O}\left(m^K(m-n)^{\lfloor\frac{n-2K-1}{2}\rfloor}n\mathcal{G}(n)\right)$$

time. If $2n > m$*, the SP algorithm runs in*

$$\min_{K:2\leq K\leq \min\{\lfloor\frac{n-1}{2}\rfloor, 2n-m\}} \mathcal{O}\left(m^K (m-n)^{\lfloor\frac{m-n-2K-1}{2}\rfloor} n\mathcal{G}(n)\right)$$

time.

Corollary 2. *If* $n \leq 4\sqrt{m}$*, the SP algorithm runs in*

$$\mathcal{O}\left(m^{\sqrt{m}}(m-4\sqrt{m})^{\sqrt{m}}\mathcal{G}(\sqrt{m})\right) = \mathcal{O}\left((m-2\sqrt{m})^{2\sqrt{m}}\mathcal{G}(\sqrt{m})\right)$$

If $m - n \leq 3\sqrt{m}$*, the SP algorithm runs in*

$$\mathcal{O}\left(m^{\frac{3\sqrt{m}}{2}}\mathcal{G}(n)\right)$$

time.

Proof: Take $K = \sqrt{m}$. □

This clearly leaves something to be desired; the only cases we have successfully analyzed do not occur often in practice, and we do not know whether the algorithm has subexponential complexity when $4\sqrt{m} \leq n \leq m - 3\sqrt{m}$. While we have not yet been able to resolve this issue for the SP algorithm, we discuss below a *randomized* version of the algorithm that does solve the problem in subexponential time...or faster.

4 Randomizing the algorithm

In this section we define the Randomized Sequential Projection (RSP) algorithm (Algorithm 4). Except for those steps explicitly described, this algorithm is identical to the SP algorithm (Algorithm 1). Note that the only difference between the RSP and SP algorithms occurs when $\mathcal{A}^-(h', n) = \mathcal{A}(h', n') \setminus \mathcal{A}(h'+1, k(h'+1))$ is bigger than K during some iteration of the **While** loop. Since $K > 1$ must be true, Theorem 3 guarantees the following:

Lemma 6. *If* $K < n - k(h+1)$ *at any stage of the RSP algorithm, then* every $i \in \mathcal{A}(h, n)$ *contains an improving direction from* $\tilde{x}^{hn}$*.*

Thus every facet in the active set contains an improving direction from $\tilde{x}^{hn}$ which can be easily found using the proj() operator. Given this, from arguments in Section 3 we have

Theorem 8. *The RSP algorithm solves LP in finite time.*

4.1 Expected worst-case complexity

We now examine the expected worse case running time of the RSP algorithm, where the expectation is taken only over the internal choices of the algorithm in the **While** loop that calls the **PF** procedure. For this analysis, first note that, at any iteration h in which a recursive call is made, we can define the n subproblems

$$v^i = \text{argmax}\{c^T x : x \in F_i\}, i \in \mathcal{A}(h, n).$$

Lemma 6 tells us that $c^T v^i > c^T \tilde{x}^{h'+1,k(h'+1)}, \forall i \in \mathcal{A}(h', n)$. Then the random choice made before the recursive call to the RSP algorithm can be viewed as the choice of which of these subproblems to process. Now order these subproblems so that $c^T v^1 > ... > c^T v^n$. Since each v^i is chosen with probability $\frac{1}{n}$, we have

Algorithm 3 Randomized Sequential Projection Algorithm

Given: A system of inequalities (A, b) with $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$; an objective function $c \in \mathbb{Z}^n$ and upper bound $c_0 \in \mathbb{R}$; and an interior point $\hat{x} \in P$, and a threshold integer value $K \geq 1$.

$\vdots \qquad \vdots \qquad \vdots$

Initialize $h = 1$, $\mathcal{P}(h, 1) = \mathcal{A}(h, 1) = \{i_{h1}\}$, and $k = 1$.
While $\bar{x} \notin P$ **do**

Process $\mathcal{A}(h, k)$, and $\hat{x}^{hk}$ and either
1) **find** an optimal solution $\bar{x} \in F_{\mathcal{A}(h,k)}$ or
2) **find** a number $k(h+1) : 1 \leq k(h+1) < n$, a set of inequalities $\mathcal{A}(h+1, k(h+1)) \subseteq M$ such that $|\mathcal{A}(h+1, k(h+1))| = k(h+1)$, and a point $\tilde{x}^{h+1,k(h+1)} \in F_{\mathcal{A}(h+1,k(h+1))}$.
If $K < n - k(h+1)$ **then**
Choose randomly, with uniform probability, some $i' \in \mathcal{A}(h, n)$.
Starting from $\tilde{x}^{h+1,k(h+1)}$ and **using** threshold value K,
solve $\max\{c^T x : x \in P \cap H_{i'}\}$ recursively.
Define $\hat{x}^{h+1,k(h+1)} = \text{argmax}\{c^T x : x \in P \cap H_{i'}\}$.
Reset $k = |\{i \in M : \mu_i > 0 \text{ at } \text{argmax}\{c^T x : x \in P \cap H_{i'}\}\}|$.
Reset $\mathcal{A}(h+1, k(h+1)) = \{i \in M : \mu_i > 0 \text{ at } \text{argmax}\{c^T x : x \in P \cap H_{i'}\}\}$.
Return to the start of the **While** loop.
Else set $\hat{x}^{h+1,k(h+1)} = \tilde{x}^{h+1,k(h+1)}$.
Return to the start of the **While** loop.

End-do

Proposition 1. *After any recursive call to the RSP algorithm:*

- *for an instance of LP with n variables and $m \geq n + k$ constraints, the number of inequalities of $\mathcal{A}(h', n)$ that never again enters the active set will be at least j with probability $\frac{1}{n}$, for all $j = 1, ..., n$.*
- *for an instance of LP with n variables and $m < n + k$ constraints, the number of inequalities of $\mathcal{A}(h', n)$ that never again enters the active set will be at least j with probability $\frac{1}{n}$, for all $j = 1, ..., m - n - k$.*

The number of iterations in which $\mathcal{A}^-(h, n) \leq K$ at any point of the RSP algorithm is $\mathcal{O}(m^K)$. Now let $\mathcal{R}(m, n, K)$ be the maximum number of expected total iterations of the RSP algorithm in which $\mathcal{A}^-(h, n) > K$ for an LP instance of n variables and m constraints. We now consider the calculation of $\mathcal{R}(m, n)$. It is clear that

$$\mathcal{R}(m, n, K) = 0, \text{ if } n \leq K \text{ or if } m - n \leq K.$$

Using Proposition 1, we can show that

$$\mathcal{R}(m, K+1, K) = 1 + \frac{1}{K+1} \sum_{j=1}^{K+1} \mathcal{R}(m - j, K+1, K),$$

and therefore

$$\mathcal{R}(2K+2, K+1, K) = 1$$
$$\mathcal{R}(m, K+1, K) \leq \frac{2(m-K-1)}{K+2}, \forall m \geq 2K+3$$

Moreover, we also have that

$$\begin{aligned}\mathcal{R}(m, m-K-1, K) &= 1 + \mathcal{R}(m-1, m-K-2, K) \\ &= m-K-1\end{aligned}$$

The general form of the recursion when $K < n < m-K$ becomes

$$\mathcal{R}(m,n,K) = 1 + \mathcal{R}(m-1,n-1,K) + \frac{1}{n}\sum_{j=1}^{\min\{n,m-n-K-1\}} \mathcal{R}(m-j,n,K)$$

Assume for simplicity that $m \geq 2n+K$. Then we obtain

$$\mathcal{R}(m,n,K) = 1 + \mathcal{R}(m-1,n-1,K) + \frac{1}{n}\sum_{i=1}^{n} \mathcal{R}(m-i,n,K).$$

We can use this to show

$$\begin{aligned}\mathcal{R}(m,n,K) &\leq \mathcal{R}(m-1,n-1,K) + \frac{1}{2}\mathcal{R}(m-1,n,K) + \frac{1}{2}\mathcal{R}(m-n,n,K) \\ &\leq \mathcal{R}(m-2,n-2,K) + \frac{1}{2}\mathcal{R}(m-2,n-1,K) + \frac{1}{2}\mathcal{R}(m-1-n,n-1,K) \\ &\quad + \frac{1}{2}\mathcal{R}(m-2,n-1,K) + \frac{1}{4}\mathcal{R}(m-2,n,K) + \frac{1}{4}\mathcal{R}(m-1-n,n,K) \\ &\quad + \frac{1}{2}\mathcal{R}(m-n,n,K) \\ &\leq \mathcal{R}(m-2,n-2,K) + \mathcal{R}(m-2,n-1,K) + \frac{1}{4}\mathcal{R}(m-2,n,K) \\ &\quad + \frac{1}{2}\mathcal{R}(m-n,n-1,K) + \frac{3}{4}\mathcal{R}(m-n,n,K),\end{aligned}$$

where the last inequality follows because $\mathcal{R}(m',n',K) \leq \mathcal{R}(m',n'+1,K)$ is always true. Continuing the expansion,

$$\begin{aligned}\mathcal{R}(m,n,K) &\leq \mathcal{R}(m-3,n-3,K) + \frac{1}{2}\mathcal{R}(m-3,n-2,K) + \frac{1}{2}\mathcal{R}(m-2-n,n-2,K) \\ &\quad + \mathcal{R}(m-3,n-2,K) + \frac{1}{2}\mathcal{R}(m-3,n-1,K) + \frac{1}{2}\mathcal{R}(m-2-n,n-1,K) \\ &\quad + \frac{1}{4}\mathcal{R}(m-3,n-1,K) + \frac{1}{8}\mathcal{R}(m-3,n,K) + \frac{1}{8}\mathcal{R}(m-2-n,n,K) \\ &\quad + \frac{1}{2}\mathcal{R}(m-n,n-1,K) + \frac{3}{4}\mathcal{R}(m-n,n,K) \\ &\leq \mathcal{R}(m-3,n-3,K) + \frac{3}{2}\mathcal{R}(m-3,n-2,K) \\ &\quad + \frac{3}{4}\mathcal{R}(m-3,n-1,K) + \frac{1}{8}\mathcal{R}(m-3,n,K) \\ &\quad + \frac{1}{2}\mathcal{R}(m-n,n-2,K) + \mathcal{R}(m-n,n-1,K) + \frac{7}{8}\mathcal{R}(m-n,n,K),\end{aligned}$$

and eventually we arrive at

$$\begin{aligned}\mathcal{R}(m,n,K) &\leq \sum_{i=0}^{n-K-1} \binom{n}{i} \frac{1}{2^n} \max\{\mathcal{R}(m-n,n-i,K), m-K\} \\ &\quad + \sum_{i=0}^{K-1} \binom{n}{i} \frac{1}{2^n}(m-n-K-i) \\ &\leq n\binom{n}{i}\frac{1}{2^n} \max\left\{m-k, \max_{i=0,\ldots,n-K-1}\{\mathcal{R}(m-n,n-i,K)\}\right\}\end{aligned}$$

After a lot of headache, we can show that

$$\begin{aligned}\mathcal{R}(m,n,K) &\leq n^{\log n} \max\left\{m-k, \max_{i=0,\ldots,n-K-1}\{\mathcal{R}(m-n,n-i,K)\}\right\} \\ &\leq (m-K)(n^{\log n})^{\frac{m-K}{n}}\end{aligned}$$

Theorem 9. *For any fixed K, the expected worst-case running time of the RSP algorithm is*

$$\mathcal{O}\left((m-K)(n^{\log n})^{\frac{m-K}{n}} n\mathcal{G}(n)\right)$$

Note that this is subexponential if $n \geq \sqrt{m-K}$, which is one of the goals motivated in the last section. Moreover, if the expression $\frac{m-K}{n} \leq K$, we may be able to say even more. In particular, it may be reasonable to consider the case in which we fix $K = 2$, since *every* LP problem either has at most twice as many constraints as variables, or has a dual that does.

Theorem 10. *If we take $K = 2$, the expected worst-case running time of the RSP algorithm is*

$$\mathcal{O}\left(m^3 n^{2\log n}\mathcal{G}(n)\right)$$

Thus the dominant term in the expected worst-case complexity of the RSP algorithm grows only as fast as the log function. To the best of our knowledge, the RSP algorithm is the fastest known algorithm for LP whose running time does not depend on the data, in terms of expected worst-case running time.

The idea of choosing a facet randomly from among a set of facets, each of which is guaranteed to contain an improving direction, is related to various randomized rules proposed for the simplex algorithm discussed (e.g., [9, 13, 18]). While the resulting pivot rules yield subexponential worst-case expected running times for a simplex algorithm, to our knowledge all of these times are bounded from below by $\mathcal{O}\left(exp(\sqrt{n\log m})\right)$, which grows significantly faster than $2^{\log n \log n} = n^{\log n}$. The primary factor limiting randomized simplex algorithms from having expected worst case running times closer to that defined for RSH seems to be the fact that they are constrained to follow edges of the feasible polyhedron while RSH is not. Indeed, if the Hirsch conjecture (see, e.g., [4, 14]) turns out to be incorrect, then defining significantly faster randomized simplex algorithms than those already known may not even be possible.

5 Quadratic programming

Consider a QP problem of the form

$$\max_{x \in \mathbb{R}^n} \; c^T x + \frac{1}{2} x^T Q X \tag{3}$$

$$\text{s.t.} \;\; Ax \leq b, \tag{4}$$

where $Q \in \mathbb{R}^{n \times n}$ is a negative semi-definite matrix (and thus the objective function is concave). A remarkable feature of the SP algorithm is that it extends easily to this family of problems; we will call this extension the Quadratic SP (QSP) algorithm. The only significant alteration required is defining the analog of the proj$(\cdot)$ operator. Therefore, for any subset of constraints $\mathcal{A} \subset M$ and any point $x \in F_{\mathcal{A}}$, we define

$$\mathit{\Pi}_Q(\hat{x}) := \left\{ \begin{array}{c} \text{argmax } \frac{1}{2} x^T Q x + c^T x \\ \text{s.t. } A_i x = b_i, i \in \mathcal{A} \end{array} \right\}$$

From the KKT conditions we have that $\bar{x} = \mathit{\Pi}_Q(\hat{x})$ is defined by the unique solution $(\bar{x}, \bar{\mu})$ to the $n + |\mathcal{A}| \times n + |\mathcal{A}|$ system of equations

$$\begin{aligned} Qx + \sum_{i \in \mathcal{A}} A_{ij} \mu_i &= -c, \;\; \forall j \in N \\ A_i x &= b, \;\; \forall i \in \mathcal{A}. \end{aligned}$$

Thus, the $\mathit{\Pi}()$ operator finds the the point maximizing the objective function in the hyperplane defined by the constraints that are active at $\hat{x}$. Note that this maximizer may violate some or all of the constraints *not* active at $\hat{x}$.

The QSP algorithm can then be defined to operate identically to the SP algorithm, with the following two exceptions:

1. Rather than set $\bar{x} = \text{proj}_{H_0}(\hat{x})$, where $\hat{x}$ is an initial interior point, we initialize $\bar{x} = \text{argmax}_{x \in \mathbb{R}^n} \{\frac{1}{2} x^T Q x + c^T x\}$.
2. Rather than call the **Process and Find** procedure at each iteration, we call a slightly modified version, which we will the **QPF** procedure.

The **QPF** procedure, given in Algorithm 5, closely resembles the **PF** procedure of the SH heuristic; the main difference is that the **QPF** procedure defines $\bar{x}^{kh} := \mathit{\Pi}_Q(\hat{x}^{hk})$ whenever the **PF** would have defined $\bar{x}^{kh} := \text{proj}_{H_0 \cap \mathcal{A}(h,k)}(\hat{x}^{hk})$. Many of the results we have established for the SP algorithm extend immediately to the QSP algorithm. For example, the analog of Theorem 4 for the QSP algorithm is

Theorem 11. *Let $\bar{x}^{h'k'} = \mathit{\Pi}_Q(\hat{x}^{h'k'})$. If $\bar{x}^{h'k'} \in P$, then let $\mu^{h'}$ be the dual vector defined by solving for $\mathit{\Pi}_Q(\hat{x}^{h'k'})$. Then we have that*

$$\bar{x}^{h'k'} = \mathit{argmax}_x \{\frac{1}{2} x^T Q x + c^T x : x \in F_{\mathcal{A}^-(h',k')}\},$$

where $\mathcal{A}^-(h',k') = \{i \in \mathcal{A}(h',k') : \mu_i^{h'} < 0\}$. Moreover

1. *The solution $\bar{x}^{h'k'}$ is optimal to QP if and only if $\mathcal{A}^-(h',k') = \emptyset$.*
2. *If $\mathcal{A}^-(h',k') \neq \emptyset$, then $\mathcal{A}^-(h',k') \cap \mathcal{A}(h,k) \leq k-1$ for every $h \geq h'+1$.*

Since the number of possible active sets is finite, and since (by the preceding Theorem) none of them repeat, we have the following analog to Theorem 5:

Algorithm 4 Process and Find Procedure for QP (QPF procedure)

Given: An instance of QP defined by Q, c, A, b, a set of inequalities $\mathcal{A}(h,k) = \{i_{h11}, i_{h12}, ..., i_{h1k}\} \subseteq P$ such that $1 \leq k \leq n-1$, and a point $\hat{x}^{hk} \in F_{\mathcal{A}(h,k)}$.
While $\bar{x}^{kh} \notin P$ **do**
  Define $\bar{x}^{kh} := \Pi_Q(\hat{x}^{hk})$.
  If $\bar{x}^{kh} \notin P$ **then**
    Compute

$$\lambda_i := \frac{b_i - A_i\hat{x}^{hk}}{A_i\bar{x}^{hk} - A_i\hat{x}^{hk}}.$$

    Choose $i_{h,k+1} = \text{argmin}_{i \in M: A_i\bar{x} > b_i}\{\lambda_i\}$.
    Define $\hat{x}^{h,k+1} = \lambda_{i_{h,k+1}}\bar{x}^{hk} + (1 - \lambda_{i_{h,k+1}})\hat{x}^{hk}$.
    Set $\mathcal{A}(h,k+1) = \mathcal{A}(h,k) \cap i_{h,k+1}$, and **increment** k.
  End-if
End-do
Define $\mu^h \in \mathbb{R}^k$ to be the vector such that

$$Q\bar{x}^{hk} + \textstyle\sum_{i \in \mathcal{A}(h,k)} A_{ij}\mu_i^h = -c, \forall j \in N$$
$$A_i\bar{x}^{hk} = b_i, \forall i \in \mathcal{A}(h,k).$$

If $\mu_i^h \geq 0, j \in \mathcal{A}(h,k)$, **then**
  $\hat{x}^{hk}$ is optimal; **set** $\bar{x} = \hat{x}^{hk}$ and **exit** the procedure;
Else reset $k = |\mathcal{A}^+(h,k)|$, **set** $\mathcal{A}(h+1,k) = \mathcal{A}^+(h,n)$, and **set** $i_{h+1,1}, ..., i_{h+1,...,k}$ to be the inequalities of $\mathcal{A}(h+1,k)$.
Exit the procedure.

Theorem 12. *The QSP algorithm correctly solves QP in finite time.*

Moreover, we can define, almost trivially, a randomized version of the QSP algorithm (call this RQSP) in exactly the same way that we defined the RSP algorithm from SP. That is, if $\mathcal{A}^-(h',k')$ is ever bigger than some threshold K, we can randomly choose one of the inequalities in the active set (with uniform probability) and recursively solve the QP defined by projecting the current problem into the hyperplane defined by that inequality. Since the cardinality of the active set used to define $\mu^{h'}$ need not be n (as it must be in the SP and RSP algorithms), the complexity analysis of RQSP is more difficult than the analysis of RSP, and we have not yet been able to complete it. However, it seems likely that RQSP will have an expected worst case running time is subexponential; to the best of our knowledge, such a result would be the first known for QP.

6 Conclusion

It seems likely that the SP, RSP, QSP, and RQSP algorithms would perform well in practice. In particular, if it is possible to use information from the solution of one projection problem $\text{proj}_{H_0 \cap H_{\mathcal{A}(h,k)}}(\hat{x}^{hk})$ to enhance the solution of the next projection problem $\text{proj}_{H_0 \cap H_{\mathcal{A}(h,k+1)}}(\hat{x}^{h,k+1})$ (perhaps, for example, in a way that simplex codes update basis factorizations from one pivot to the next), our results would seem to suggest that these algorithms may even rival simplex methods in practical efficiency. We intend to explore this possibility in the near future.

Bibliography

[1] D. Avis and V. Chvatal. Notes on bland's pivoting rule. In M. L. Balinski and A. J. Hoffman, editors, *Polyhedral Combinatorics*, volume 8, pages 24–34. 1978.

[2] R.G. Bland, D. Goldfarb, and M.J. Todd. The ellipsoid method: A survey. *Operations Research*, 29:1039–1091, 1981.

[3] W. Cunningham. Theoretical properties of the network simplex method. *Mathematics of Operations Research*, 4:196–208, 1979.

[4] G.B. Dantig. *Linear Programming and Extensions*. Princeton University Press, Princeton, N.J., 1963.

[5] G.B. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. Wiley, New York, 1951.

[6] G.B. Dantzig. An ϵ-precise feasible solution to a linear program with a convexity constraint in $1/\epsilon^2$ iterations independent of problem size. Technical Report SOL 92-5, 1992.

[7] M. Epelman and R.M. Freund. Condition number complexity of an elementary algorithm for computing a reliable solution of a conic linear system. *Mathematical Programming*, 88:451–485, 2000.

[8] J.L. Goffin. The relaxation method for solving systems of linear inequalities. *Mathematics of Operations Research*, 5:388–414, 1980.

[9] Michael Goldwasser. A survey of linear programming in randomized subexponential time. *ACM SIGACT News*, 26(2):96104, 1995.

[10] M. Grötschel, L. Lovasz, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin, 1988.

[11] R. Jeroslow. The simplex algorithm with the pivot rule of maximizing criterion improvement. *Discrete Mathematics*, 4:367–377, 1973.

[12] D.B. Judin and A.S. Nemirovskii. Evaluation of the informational complexity of mathematical programming problems. *Matekon Transl. Russian and East European Math. Economics*, 13(2):3–24, 1977. Translated from *Ekonom Mat Metody* 12 (1976).

[13] G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Mathematical Programming*, 79:217233, 1997.

[14] G. Kalai and D.J. Kleitman. A quasi-polynomial bound for the diameter of graphs of polyhedra. *Bulletin of the American Mathematical Society*, 26(2):315316, 1992.

[15] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–397, 1984.

[16] L.G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[17] V. Klee and G.J. Minty. How good is the simplex algorithm? In 0. Shisha, editor, *Inequalities*, pages 159–175. Academic Press, New York, 1972.

[18] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4-5):498516, 1996.

[19] P. McMullen. The maximum number of faces of a convex polytope. *Mathematika*, 17:179–184, 1970.

[20] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J on Optimization*, 2:575–601, 1992.

[21] R.D.C. Monteiro and I. Adler. Interior path following primal-dual algorithms. part i: Linear programming. *Mathematical Programming*, 44:27–41, 1989.
[22] K.G. Murty. Computational complexity of parametric linear programming. *Mathematical Programming*, 19:213–219, 1980.
[23] R. Seidel. The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Computational Geometry*, 5:115–116, 1995.
[24] E. Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3):247–255, 1985.
[25] R.J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, third edition, 2008.
[26] P. Wolfe. The simplex method for quadratic programming. *Econometrica*, 27:382–398, 1959.
[27] S.J. Wright. *Primal-Dual Interior Point Methods*. SIAM, Philadelphia, PA, 1997.
[28] N. Zadeh. Theoretical efficiency and partial equivalence of minimum cost flow algorithms: A bad network problem for the simplex method. *Mathematical Programming*, 5:255–266, 1973.