MS&E 213 / CS 269O : Chapter 7
Feasibility Problem and Cutting Plane Methods *

By Aaron Sidford (sidford@stanford.edu)

November 26, 2019

1 Motivation

In the last few lectures we discussed a tight connection between convex sets and convex functions. For a convex function $f : \mathbb{R}^n \to \mathbb{R}$ we also proved that for any $x_0 \in \mathbb{R}^n$ the set $\{x \in \mathbb{R}^n : f(x) < f(x_0)\}$ is convex and thus the set of points that have smaller function value than a given query point are convex. Moreover, we showed that convex sets always admit supporting hyperplanes. The particular theorem we will use regarding these hyperplanes is given by the following theorem.

Theorem 1 (Supporting Hyperplane Theorem)**.** *For any convex set $S \subseteq \mathbb{R}^n$ and $x_0 \notin S$ or $x_0 \in \partial S$ there exists a vector $g \in \mathbb{R}^n$ with $g \neq 0$ such that for all $x \in S$ we have $g^\top x \geq g^\top x_0$. In other words, the half-space $H_{\geq}(g, g^\top x_0) = \left\{y \in \mathbb{R}^n : g^\top y \geq g^\top x_0\right\}$ contains S.*

Consequently, for any convex function $f : \mathbb{R}^n \to \mathbb{R}$ and $x_0 \in \mathbb{R}^n$ there is always a vector g where not only is the minimizer in the halfspace induced by g_{x_0}, but all points of value less then $f(x_0)$ are in the halfspace induced by the g_{x_0} direction (though moving in the g_{x_0} direction may not itself yield progress on minimizing the objective function).

In this chapter we discuss algorithms for minimizing functions given access to these sorts of separation oracles on sets and level sets. Rather than provide a suite of algorithms tailored to the specifics of our assumptions for the problems we wish to solve, we provide a single general problem called the *feasibility problem* that encompasses numerous settings where one would wish to use a separation oracle. We show how to map various standard optimization problems to the feasibility problem and then provide efficient algorithms for the feasibility problem, broadly known as *cutting plane methods*.

2 The Feasibility Problem

The main problem we wish to solve in this chapter is a problem known as the *feasibility problem*. To define it formally, we need to introduce a little bit of notation regarding balls in various metric spaces.

Definition 2 (Balls)**.** For a given norm $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ the *ball of radius $r \in \mathbb{R}_{\geq 0}$ centered around $x_0 \in \mathbb{R}^n$* is given by

$$B_{\|\cdot\|}(r, x_0) \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n : \|x - x_0\| \leq r\} .$$

*These notes are a work in progress. They are not necessarily a subset or superset of the in-class material and there may also be occasional *TODO* comments which demarcate material I am thinking of adding in the future. These notes will converge to a superset of the class material that is TODO-free. Your feedback is welcome and highly encouraged. If anything is unclear, you find a bug or typo, or if you would find it particularly helpful for anything to be expanded upon, please do not hesitate to post a question on the discussion board or contact me directly at sidford@stanford.edu.

When the norm is omitted, it is assumed to be $\|\cdot\|_2$ and when x_0 is omitted it is assumed to be $\vec{0}$. When the norm is $\|\cdot\|_\infty$ we call the ball a *box*. Also, for simplicity we use $B_p(r, x_0) \stackrel{\text{def}}{=} B_{\|\cdot\|_p}(r, x_0)$.

Note that a ball of radius r around a point x_0 is simply the set of points at distance at most r from x_0 with respect to the $\|\cdot\|$. When $\|\cdot\| = \|\cdot\|_2$ the ball is a ball in the standard sense we typically think of.

Now the problem we wish to solve is the following:

Definition 3 (Feasibility Problem)**.** For $0 < r < R$ and $n \geq 1$ the (r, R, n)-*feasibility problem* is defined as follows. We are given an oracle that when queried with $x \in B_\infty(R)$ it outputs a vector $g_x \in \mathbb{R}^n$. The goal of the feasibility problem is to query the oracle and either compute some $x \in B_\infty(R)$ such that $g_x = \vec{0}$ or compute some points $x_1, ..., x_k \in B_\infty(R)$ such that

$$S \stackrel{\text{def}}{=} B_\infty(R) \cap_{i \in [k]} \text{half}(g_{x_i}, g_{x_i}^\top x_i)$$

does not contain any ball of radius r. We call an algorithm a $(\mathcal{T}_q, \mathcal{T}_t)$-*solution to the* (r, R, n)-*feasibility problem* if it achieves this goal with $O(\mathcal{T}_q)$ queries to the oracle and in $O(\mathcal{T}_t)$ total time.

The feasibility problem can be thought of as follows. We are given a box of radius R and we either want to find a good point (one where $g_x = \vec{0}$) or prove that the region of good points is fairly small (i.e. does not contain a box of radius r). We keep querying the oracle at various points in the hope of finding a good point. However, if instead we get vectors $g_x \neq 0$ ultimately we wish to show that the intersection of the half-spaces they induce is small.

Note that this problem says nothing about convexity or even the set of ϵ-optimal points. It is just about reasoning about intersections of half-spaces given an oracle that produces them. The generality of this formulation allows us to use algorithms that solve the feasibility problem to solve a broad range of optimization problems (possibly even beyond those which are convex).

Also, note that the choice of norms for the initial ball, i.e. $B_\infty(R)$, and the ball we wish to prove are not contained in the intersection of the half-spaces, $B_\infty(r, x)$ for unknown x, is somewhat arbitrary. As we will later see, changing the norm likely only affects logarithmic factors in bounds for solutions to the feasibility problem.

3 Reducing to the Feasibility Problem

Here we discuss how we can use algorithms that solve the feasibility problem to solve a broad class of problems in convex optimization. The idea of this section is to convey the flavor of how algorithms for the feasibility problem can be mapped to various common optimization settings.

3.1 Unconstrained Minimization

Here we consider the problem of unconstrained function minimization, i.e. $\min_{x \in \mathbb{R}^n} f(x)$. We need to introduce an oracle for how to access f that makes sense in the context of the feasibility problem. The oracle we assume is precisely the one that we know exists whenever level sets of the function f are convex. We overload terminology here and call this a separation oracle where it will be clear from context that the oracle is a separation oracle for a *function* (rather than a set).

Definition 4 (Separation Oracle (for a Function))**.** For a function $f : \mathbb{R}^n \to \mathbb{R}$ a *separation oracle* is a oracle that when queried at a point $x \in \mathbb{R}^n$ outputs a vector $g \in \mathbb{R}^n$, such that if $g = \vec{0}$ then $f(x) = f_*$ and if $g \neq \vec{0}$ then if $f(y) < f(x)$ it is the case that $g^\top y \geq g^\top x$.

Note that the vector g returned by a separation oracle is not necessarily a subgradient. If $g \neq 0$ the condition that $f(y) \leq f(x)$ implies $g^\top y \geq g^\top x$ does not mean imply that $f(x) \geq f(y) + g^\top(y - x)$. The homework gives an instance of this.

Also note in our definition of separation oracle we did not impose that f is convex. It is natural to ask then whether or not a function admitting a separation oracle implies that it is convex. In the homework we show that a closed set is convex if and only if it is the intersection of half-spaces and consequently this definition does imply something about the convexity of level sets of f. However, while if f is convex then its sub-level sets are convex, the converse of this statement does not hold (as we show in the homework). The set of functions where sub-level sets are convex is known as quasiconvex functions (though we will not discuss them much in this class).

Furthermore, note that computing a separation oracle for various functions can be quite easy. For example, for a differentiable convex function we know that the gradient gives such a separation oracle. Furthermore, it can be shown that functions that are a maximum of various differentiable convex functions computing gradients and separation oracles can be easy (again, as we show in the homework).

Here we show that having a value oracle and a separation oracle for a function combined with a solution to the feasibility problem allows us to perform unconstrained function minimization.

Lemma 5 (Unconstrained Minimization)**.** *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ obtains its minimum value at some $x_* \in B_\infty(R)$. Further suppose that there is a box of radius r that is contained $B_\infty(R)$ such that every point in the box is ϵ-optimal. Then given a value oracle and a separation oracle for f and a $(\mathcal{T}_q, \mathcal{T}_t)$-solution to the (r, R, n)-feasibility problem we can compute an ϵ-optimal point for f with $O(T_q)$-queries to the oracles and $O(\mathcal{T}_t)$-time.*

Proof. There is a natural approach to solving this problem and it is the one we take. We simply run the algorithm for the feasibility problem with the separation oracle for f as the oracle for the feasibility problem, query the value at every point the algorithm for the feasibility problem queries and then output the point of minimum value.

Now if we ever query a point x and find that $g_x = \vec{0}$ then we have that $f(x) = f_*$ and this routine outputs a 0-optimal point in the desired running time. If this does not happen then we have that for some $x_1, ..., x_k \in B_\infty(R)$ it is the case that

$$S = B_\infty(R) \cap_{i \in [k]} \text{half}(g_{x_i}, g_{x_i}^\top x_i)$$

does not contain a box of radius r. However, since there is a box of radius r such that every point in that box is ϵ-optimal we have that there is some $y \notin S$ with $y \in B_\infty(R)$ such that $f(y) \leq f_* + \epsilon$. This implies that $y \notin \text{half}(g_{x_i}, g_{x_i}^\top x_i)$ for some $i \in [k]$. However, by definition of a function separation oracle this implies that $f(x_i) \leq f(y)$ and therefore x_i is ϵ-optimal. The result follows from the fact that our algorithm returns the point of minimum value. □

3.2 Constrained Minimization

Here we give another instance of reducing an optimization problem to the feasibility problem. We consider the problem of constrained minimization where we wish to solve $\min_{x \in S} f(x)$. We assume that we have a separation oracle for both f and S (as well as a value oracle for f).

Definition 6 (Separation Oracle (for a Set))**.** For a set $S \subseteq \mathbb{R}^n$ a *separation oracle* is an oracle that when queried at a point $x \in \mathbb{R}^n$ outputs a vector $g \in \mathbb{R}^n$, such that if $x \in S$ then $g = \vec{0}$ and if $x \notin S$ then $g \neq \vec{0}$ and $g^\top y \geq g^\top x$ for all $y \in S$.

Given a separation oracle for S and a separation oracle for f how should we solve the constrained minimization problem $\min_{x \in S} f(x)$? There is a standard trick to reduce this to unconstrained function minimization directly.

Lemma 7. *For $f : \mathbb{R}^n \to \mathbb{R}$ and $S \subseteq \mathbb{R}^n$ consider the function $\widetilde{f} : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ defined so that for all $x \in \mathbb{R}^n$*

$$\widetilde{f}(x) = \begin{cases} f(x) & \text{if } x \in S \\ \infty & \text{if } x \notin S \end{cases}.$$

We can implement a separation oracle for $\widetilde{f}$ using one call to at most one separation oracle call for S and one separation oracle call for f.

Proof. Our separation oracle for $\tilde{f}$ is simple. Given a query $x \in \mathbb{R}^n$ we first query the separation oracle for S to get $g \in \mathbb{R}^n$. If $g \neq 0$ we return g. Otherwise, we query the separation oracle for f and return it. Note that if $g \neq 0$ then it is the case that $x \notin S$ and $\tilde{f}(x) = \infty$. Consequently, $\tilde{f}(y) < \tilde{f}(x)$ if and only if $y \in S$ in which case $g^\top y \geq g^\top x$ as desired. In the other case $x \in S$ and since our modification from f to $\tilde{f}$ only increases the value of the function at points, our output is as desired by definition of a separation oracle. □

From this lemma we see that we can perform constrained minimization simply by performing unconstrained minimization on $\tilde{f}$.

Lemma 8 (Constrained Minimization)**.** *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ and S are such that $\min_{x \in S} f(x)$ is achieved for some $x_* \in S \cap B_\infty(R)$. Further suppose there is a a box of radius r that is contained $B_\infty(R) \cap S$ such that every point in the box has value at most $f(x_*) + \epsilon$. Then given a value oracle and a separation oracle for f and a separation oracle for S and a $(\mathcal{T}_q, \mathcal{T}_t)$-solution to the (r, R, n)-feasibility problem we can compute $y \in S$ with $f(y) \leq f(x_*) + \epsilon$ with $O(T_q)$-queries to the oracles and $O(\mathcal{T}_t)$-time.*

Proof. This follows directly from applying Lemma 5 to $\tilde{f}$ and implementing its separation oracle as defined in Lemma 7. Note that this works since the box containing the points of value at most $f(x_*) + \epsilon$ is ϵ-optimal for $\tilde{f}$. □

Note that again in this lemma we did not need to directly assume that f or S were convex. Instead the class of functions and sets for which this procedure works is something more broad and induced only by the assumption that the oracle exists.

Also, note that we can prove that the transformation from f to $\tilde{f}$ preserves convexity (but we did not need it directly for our analysis).

Lemma 9. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function defined on a convex set S. Now let $\widetilde{f} : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be the function that*

$$\widetilde{f}(x) = \begin{cases} x & \text{if } x \in S \\ \infty & \text{if } x \notin S \end{cases}$$

then $\widetilde{f}(x)$ is convex.

Proof. Let $x, y \in \mathbb{R}^n$ and $t \in [0, 1]$ we wish to show that

$$f(t \cdot y + (1 - t) \cdot x) \leq t \cdot f(y) + (1 - t) \cdot f(x).$$

Now, when $t \in \{0, 1\}$ the above holds trivially with equality. Furthermore, when $x \notin S$ or $y \notin S$ then the right hand side is ∞ and the lemma is trivially true. However, when $x \in S$ and $y \in S$ then by convexity $t \cdot x + (1 - t) \cdot y \in S$ and the result follows from the convexity of f. □

3.3 Bounded Convex Functions

While the last few examples showed can apply the feasibility problem to solve a broad range of optimization problems unfortunately they each rely on bounds on the size of the set of ϵ-optimal points. A natural lingering question is how to bound the sizes of these sets. Here we show how they can be bounded simply by using that f itself might be bounded. To prove this provide the following general lemma about ϵ-optimal sets.

Lemma 10. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and $\epsilon > 0$ be arbitrary. Now, if the set of ϵ-optimal points, i.e. $\text{level}_{\leq}(f, f_* + \epsilon)$, contains a ball of radius R in some norm $\|\cdot\|$, then for all $\alpha \in (0, 1)$ the set of $\alpha\epsilon$-optimal points contains a ball of radius αR in that norm.*

Proof. By assumption there is some x_0 such that $B_{\|\cdot\|}(R, x_0) \subseteq \text{level}_{\leq}(f, f_* + \epsilon)$. Now, consider $S = B_{\|\cdot\|}(\alpha R, \alpha x_0 + (1-\alpha)x_*)$, i.e. the ball of radius R around $\alpha \cdot x_0 + (1-\alpha)x_*$. Now we have that $x \in S$ if and only if

$$\|x - \alpha \cdot x_0 - (1-\alpha)x_*\| \leq \alpha R\,.$$

Further, this happens if and only if $x = \alpha \cdot y + (1-\alpha) \cdot x_*$ for some $y \in B_{\|\cdot\|}(R, x_0)$. Since $B_{\|\cdot\|}(R, x_0) \subseteq \text{level}_{\leq}(f, f_* + \epsilon)$ we have $f(y) \leq f_* + \epsilon$ and therefore by convexity

$$f(x) \leq \alpha \cdot f(y) + (1-\alpha) \cdot f(x_*) \leq f_* + \alpha\epsilon\,.$$

This implies $S \subseteq \text{level}_f(f_* + \alpha \cdot \epsilon)$ yielding the result. □

This lemma allows us to bound the size of ϵ-optimal sets in various contexts. We conclude with one simple illustrative example.

Lemma 11. *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable convex function that achieves its minimum value in $B_\infty(R)$. Further suppose that $|f(x)| \leq M$ for all $x \in \mathbb{R}^n$. Then given a gradient oracle and value oracle for f and a $(\mathcal{T}_q, \mathcal{T}_t)$-solution to the $(R, \frac{\epsilon R}{2M}, n)$-feasibility problem we can compute an epsilon optimal point with with $O(T_q)$-queries to the oracles and $O(\mathcal{T}_t)$-time.*

Proof. By assumption we have that $B_\infty(R)$ contains a box of radius R such that every point in $B_\infty(R)$ is $2M$-optimal. Consequently the set of ϵ-optimal points contains a box of radius $\frac{\epsilon R}{2M}$ by Lemma 10. The result then follows from Lemma 5 and the fact that the gradient of a convex function yields a separation oracle for that function. □

4 Solving the Feasibility Problem

Here we discuss algorithms for solving the feasibility problem. We call such algorithms, *cutting plane methods* (not to be confused with algorithms for solving integer linear programs). These algorithms all follow the following same broad algorithmic template to solve the feasibility problem.

- Start with some *initial set* $S_0 \subseteq \mathbb{R}^n$ that contains the box of radius R, i.e. $B_\infty(R) \subseteq S_0$
- Repeat the following for $k = 0, 1, 2, ...$
  - Query the oracle at $x_k \in S_k$ a point considered to be some type of *center* of S_k to get g_{x_k}
  - If $g_{x_k} = \vec{0}$ terminate the algorithm
  - Otherwise, update the set to compute $S_{k+1} \subseteq \mathbb{R}^n$ such that it contains the intersection of S_k and the half-space induced by g_{x_k}, i.e. $S_{k+1} \supseteq S_k \cap \text{half}(g_{x_k}, g_{x_k}^\top x_k)$.

- Argue that some notion of *size* of S_k is decreasing over time.

This is a fairly simple and broad framework. We start with a set that contains the box, query the center of the set, update the set to something (hopefully smaller) that contains the intersection of the old set and the half-space induced by the queried point, and repeat. How the algorithms vary is in term of what sets they use, what center they use, how the update to the set is done, and how progress in terms of decreasing the size of the S_k is measured. In the remainder of this section we consider various results for solving this problem. For some of these results we will just outline how they work, I am happy to provide additional references to the literature as requested.

4.1 Center of Gravity Method

Perhaps the most natural set to consider would simply be the accumulation of all information we have, i.e. the minimal set that conforms to the algorithmic template we have specified. We could pick $S_0 = B_\infty(R)$ and then let $S_{k+1} = S_k \cap \text{half}(g_{x_k}, g_{x_k}^\top x_k)$. In other words, we let S_k be the intersection of the box with all half-spaces we have seen so far.

Now what center should we compute? For intuition, let's consider what happens in the case when $n = 1$. In this case this algorithm will simply have each S_k be an interval, i.e. $S_k = [a_k, b_k]$ for some $a_k, b_k \in \mathbb{R}^n$. The natural center to get would be the center of the interval, i.e. $x_k = a_k + \frac{b_k - a_k}{2}$. Why is this a good choice? We know that any half-space through this point, when intersected with the interval will result in an interval of half the length. So in every iteration we could halve the length of the interval with this algorithm. This algorithm is known as "binary search" and it gives a $(O(\log(\frac{R}{r})), O((\log(\frac{R}{r}))$ time algorithm for solving the $(r, R, 1)$ feasibility problem as S_0 has length $2R$ and we can terminate with the interval has length $< 2r$ (as then it could not contain the box of radius r) and each iteration can clearly be performed in nearly linear time.

Thus, in some sense, this shows that the feasibility problem is asking for a high-dimensional analog of binary search. A natural analog of the 1-dimension binary search algorithm is to pick x_k as the *center of gravity* of S_k, this is simply the arithmetic mean, or average, over all points in S_k. Using this as the center gives the center of gravity method.

To analyze how well this method performs, we need to reason about what happens when we take a convex set and intersect it with a half-space through its center of gravity. Fortunately, to analyze this there is a beautiful result in convex geometry.

Theorem 12 (Grunbaum's Theorem)**.** *For any convex set $S \subseteq \mathbb{R}^n$ if $x \in S$ is the center of gravity of S and $g \in \mathbb{R}^n$ with $g \neq 0$ is arbitrary then* $\text{vol}(S \cap \text{half}(g, g^\top x)) \leq (1 - \frac{1}{e})\text{vol}(S)$. *In other words, the volume of S intersected with a half-space through its center of gravity has at most $1 - \frac{1}{\epsilon}$ of the volume of S.*

This immediately gives us the analog of binary search we were looking for. Where's in 1-dimension we could halve the interval length of S_k with a single query, here we can decrease the volume of S_k by a constant $(1 - \frac{1}{e}) \approx 0.632$. So we cannot make quite as much progress in high dimensions, but we can make a good deal of progress. This gives the following guarantee for solving the feasibility problem.

Theorem 13 (Center of Gravity Method Queries)**.** *We can solve the (r, R, n)-feasibility problem with $O(n \log(R/r))$ queries to the oracle.*

Proof. We run the center of gravity method where S_k is the intersection of all half-spaces we have computed as well as $B_\infty(R)$. Now $\text{vol}(B_\infty(R)) = (2R)^n$ and $\text{vol}(B_\infty(r)) = (2r)^n$ now after k iterations of computing the center, x_k, as the center of gravity we have that $\text{vol}(S_k) \leq (1 - \frac{1}{e})^k (2R)^n$ consequently. When $k \log(1 - \frac{1}{e}) + n \log(2R) < n \log(2r)$ we have that S_k cannot contain a ball of radius r and thus this happens when $k > -[\log(1 - \frac{1}{e})]^{-1} n \cdot \log(\frac{R}{r})$, i.e. $k = \Omega(n \cdot \log(\frac{R}{r}))$. □

This is (up to constants) the optimal number of queries that could be used to solve the feasibility problem. Note that every call to the oracle cut just cut a single dimension, so it clearly takes $\Omega(n)$ queries to see all the dimension. Further, to performa binary search in each dimension a total of $\Omega(n \cdot \log(\frac{R}{r}))$ queries would be needed; more careful reasoning can then be used to show this is optimal.

However, the running time for this method is quite slow. Unfortunately, actually computing the center of gravity is a difficult problem outside the scope of this class. However, there are sampling techniques to approximate it and this can be used to get a running time around $O(n^5)$ for an approximate center of gravity method that has $O(n \log(R/r))$ query complexity.

4.2 Ellipsoid Method

Another popular and well know method for solving the feasibility problem is the *ellipsoid method*. One way to possibly derive this method is to think about what would be needed to decrease the iteration costs in the center of gravity method. One of the reasons the center of gravity method is so expensive to implement is that the set being maintained is fairly complex. As the number of half-spaces computed increases, S_k increasingly looks like an arbitrary convex set.

To obtain a simpler method we could simply work with a simpler set of S_k. One nice family of convex sets is the set of ellipsoids, that is ball of bounded radius with respect to some rescaled Euclidean norm. More formally, for $\mathbf{A}_k \in \mathbb{R}^{n \times n}$ with $\mathbf{A}_k = \mathbf{A}_k^\top$ and $z^\top \mathbf{A}_k z > 0$ we have that all the eigenvalues of $\mathbf{A}$ are positive and it is not too hard to see that the set $\{y \mid \|y - x_k\|_{\mathbf{A}_k}^2 \leq 1\}$ is an ellipse centered around x_k with axis given by the eigenvectors and eigenvalues of $\mathbf{A}_k$, where we let $\|z\|_{\mathbf{A}_k} \stackrel{\text{def}}{=} \sqrt{z^\top \mathbf{A}_k z}$ for all z (see the chapter on norms).. Now a natural method would be to use such ellipses as the S_k. We could start with the trivial ball the contains $B_\infty(R)$ as the initial S_0 and then in each iteration use the center of the ellipse as the center x_k and compute the new S_{k+1} to be the smallest ellipse that contains the old ellipse and the new half-space. It turns out that a good new ellipse can always be computed in $O(n^2)$ time such that volume of the ellipse decreases by $(1 - \frac{1}{n})$. This method is known as the ellipsoid method and gives a $(O(n^2 \log(n\frac{R}{r}), O(n^4 \log(n\frac{R}{r}))$ solution to the $O(r, R, n)$ feasibility problem.

The ellipsoid method has essentially the cheapest iteration complexity known for methods that solve this problem with a $\log(\frac{R}{r})$ dependence in the query complexity (as opposed to $O(\text{poly}(\frac{R}{r}))$. However, it s query complexity is worse than that of the center of gravity method by a factor of n.

4.3 John Ellipse Method

Another idea to improve the iteration cost of the center of gravity method (while nearly preserving the optimal query complexity) would simply be to use a simpler center. One interesting idea is to compute the center of the maximum volume ellipse contained inside S_k. This ellipse is known as the *John Ellipse* and it is has the interesting property that if the ellipse is dilated by a factor of n then it contains S_k. It can also be shown that if this center is used the in each iteration the volume of the John Ellipse decreases by a constant. Since this tracks the volume of S_k up to a factor of n^n this method has a nearly optimal query complexity of $O(n \log(nR/r))$. However, since there is a convex program for computing the John ellipse, the iterations of this method are cheaper and each iteration can be implemented in time roughly $\tilde{O}(n^{\omega+1/2})$.

4.4 Volumetric Center

To decrease the iteration costs even further several insights were needed. There was a beautiful result of Vaidya in 1989 that did the following. First rather than maintaining all half-spaces computed and the box, the set he maintained was the intersection of some of the half-spaces that induce $B_\infty(R)$ and the half-spaces computed. Then, the center he computed came directly from an optimization problem. He defined a barrier

function p_k on S_k such that as $x \in S_k$ goes towards the boundary of S_k the value of $p_k(x)$ goes to infinity and let $x_k = \text{argmin}_{x \in S_k} p_k(x)$. This barrier is known as the volumetric barrier and the center is known as the volumetric center.

Now, one of the virtues of defining the center this way is that as half-spaces are added or removed, the center can be recomputed quickly by a few iterations of Newton's method, i.e. minimizing the second order Taylor approximation, on p_k. Furthermore, it can be shown that the ellipse induced by the second order Taylor approximation to p_k at x_k gives an approximate John ellipse, that is an ellipse contained inside S_k such that dilating it by a factor of n contains S_k.

To analyze his method he then showed that by measuring how much each half-space contributed to the approximate John ellipse and dropping unimportant half-spaces, he could insure that only $O(n)$ half-spaces ever needed to be maintained, the approximate John ellipse approximated S_k as desired, that every time a new half-space was added the volume of this ellipse decreased by a multiplicative constant and that every time a half-space was removed the volume of the ellipse may increase by a multiplicative constant (but it is smaller than the increment). Consequently, the volume of this ellipse decreases on average by a constant in every iteration and this gave a $O(n \log(nR/r), n^{\omega+1} \log(nR/r))$ solution to the (r, R, n) feasibility problem, where $\omega < 2.373$ is the matrix multiplication constant.

In practice, this idea of using a barrier is sometimes used, but that barrier is often different and the best theoretical guarantees for it are weaker.

4.5 State of the Art

Recently, in joint work with Yin Tat Lee and Sam Wong we showed how to further decrease the iteration costs and provided a $O(n \log(nR/r), n^3 \log^{O(1)}(nR/r))$ solution to the (r, R, n) feasibility problem. Our algorithm worked by further approximating Vaidya's method by only computing both the center and the importances of constraints approximately. Ultimately, our algorithm can be viewed as approximating Vaidya's method, which in turn approximated the John Ellipse method, which in turn approximated the center of gravity method while as much as possible trying to leverage simple properties of ellipses which we can leverage to make iterations cheap.