Ellipsoid Method and the Amazing Oracles

‘Ellipsoid method is revisited. Besides that, three separation oracles are investigated for applications. They are robust optimization, semidefinite programming, and network optimization. Discuss the stability issue. Finally, the parallel cut is described.’

# Introduction

The ellipsoid method has a bad reputation. The method is commonly considered to be slow for large-scale convex problems compared to the interior-point method [@unknown]. And this is not fair. First of all, unlike the interior-point method, the ellipsoid method does not need to explicitly evaluate all constraint functions at each iteration. All it needs is a *separation oracle* that provides a *cutting-plane* (§ 2). This can make the method attractive for some problems with a large or even infinite number of constraints. Second, while the ellipsoid method itself cannot exploit the sparsity of the problem, the separation oracle can exploit certain types of structures of the problems…

In § 3.1, robust optimization…

In § 3.2, we show that for the network parametric problems, the cutting-plane can be obtained by finding a negative cycle of a directed graph. Efficient algorithms exist in which the locality of network and other properties can be exploited.

In § 3.3, problems involving matrix inequalities are discussed. Recall that positive definiteness of symmetry matrices can be checked efficiently using Cholesky or more precisely factorization. Let a symmetric matrix . Recall that if the factorization process of stops at row due to a non-positive diagonal entry is encountered, then is not positive-definite. Futhermore, by using the lazy evaluation technique, one can construct a cutting plane in rather than . Thus, it can be used for efficient oracle implementations.

The implementation issues of the ellipsoid method is discussed in § 4. The method is one kind of cutting-plane methods, where the search space is an ellipsoid usually represented by:

where is the center of the ellipsoid. The matrix is symmetric positive-definite. At each iteration, and are updated according to the oracle. While the updating of ellipsoids is simple, and the implementations have been found for decades, we show that by splitting into multiplying by , i.e.,

one can further reduce the updating cost by the flops (floating point operations) of .

In addition, the use of parallel cuts is discussed in Section 4.2. When a pair of parallel inequalities, one of which is violated, two constraints can be used simultaneously to update the ellipsoid. Some papers reported that this technique did not provide significant improvement. We show that, for cases where the upper and lower bounds of some constraints are tight, as in the case of some filter designs, the use of parallel cuts actually can improve runtime dramatically. Furthermore, we show that, if the method is implemented correctly, each update, whether using deep cuts or parallel cuts, requires at most one square-root operation.

In many practice engineering problems, some design variables may be restricted to discrete forms. Since all the cutting-plane method need is a separation oracle, it works for discrete problems as well…

# Cutting-plane Method Revisited

## Convex Feasibility Problem

Let be a convex set. Consider the feasibility problem:

1. Find a point in , or
2. Determine that is empty (i.e., no feasible solution)

When a *separation oracle* is *queried* at , it either

1. Asserts that , or
2. Returns a separating hyperplane between and :

The pair is called a *cutting-plane*, since it eliminates the half-space from our search. We have the following observations:

* If ( is on the boundary of half-space that is cut), cutting-plane is called *neutral-cut*.
* If ( lies in the interior of half-space that is cut), cutting-plane is called \*\*.
* If ( lies in the exterior of half-space that is cut), cutting-plane is called *shadow-cut*.

The is usually given by a set of inequalities or for , where is a convex function. A vector is called a *sub-gradient* of a convex function at if . Hence, the cut is given by .

Note that if is differentiable, we can simply take . Cutting-plane method comprises two key components: separation oracle and a search space initially big enough to cover . For example,

* Polyhedron = .
* Interval = (for one-dimensional problem).
* Ellipsoid = .

Generic Cutting-plane method:

* **Given** initial known to contain .
* **Repeat**
  1. Choose a point in .
  2. Query the separation oracle at .
  3. **If** , quit.
  4. **Else**, update to a smaller set that covers:
  5. **If** or it is small enough, quit.

Todo: What if the search space is large enough?

## From Feasibility to Optimization

Consider:

We treat the optimization problem as a feasibility problem with an additional constraint . Here, could be a convex function or a quasi-convex function. is the best-so-far value of . We can reformulate the problem as:

where is the -sublevel set of .

For each , is a nonincreasing function of , *i.e.*, whenever . Note that if and only if (monotonicity). One easy way to solve the optimization problem is to apply the binary search on .

Another possible way is, to update the best-so-far whenever a feasible solution is found such that . We assume that the oracle takes responsibility for that.

Generic Cutting-plane method (Optim)

* **Given** initial known to contain .
* **Repeat**
  1. Choose a point in
  2. Query the separation oracle at
  3. **If** , update such that .
  4. Update to a smaller set that covers:
  5. **If** or it is small enough, quit.

## Example: Profit Maximization

This example is taken from [@Aliabadi2013Robust]. Consider the following *short-run* profit maximization problem:

where is the market price per unit, is the scale of production, and are output elasticities, and are the i-th input quantity and output price, is the Cobb-Douglas production function, and is a constant that limits the quantity of . The above formulation is not in convex form. First, we rewrite the problem:

By the change of variables, we can obtain the following convex form of [@eq-profit-max-in-orginal-form]:

where and . Some readers may recognize that we can also write the problem in a geometric program by introducing one additional variable [@Aliabadi2013Robust].

# Amazing Oracles

* Robust convex optimization
  + oracle technique: affine arithmetic
* Parametric network potential problem
  + oracle technique: negative cycle detection
* Semidefinite programming
  + oracle technique: Cholesky factorization

## Robust Convex Optimization

Consider:

where represents a set of varying parameters. We can reformulate the problem as:

### Algorithm

The oracle only needs to determine:

* If for some and , then
* the cut =
* If for some , then
* the cut =
* Otherwise, is feasible, then
* Let .
* .
* The cut =

### Example: Robust Profit Maximization

Consider again the profit maximization problem in § 2.3. Now suppose that the parameters are subject to interval uncertainties:

The problem formulation of the robust counterpart considering the worst-case scenario is:

In [@Aliabadi2013Robust], the authors present a *piecewise linear approximation* approach. It involves a lot of programming work, but the results are inaccurate. However, this can easily be solved using the cutting-plane method. Note that in this simple example, the worst-case scenario occurs when:

* ,
* , ,
* if , , else
* if , , else

We can even reuse the original oracle to compose the robust counterpart.

class profit\_rb\_oracle:  
 def \_\_init\_\_(self, params, a, v, vparams):  
      p, A, k = params  
 e1, e2, e3, e4, e5 = vparams  
      params\_rb = p - e3, A, k - e4  
 self.a = a  
 self.e = [e1, e2]  
 self.P = profit\_oracle(params\_rb, a, v + e5)  
  
 def \_\_call\_\_(self, y, t):  
      a\_rb = self.a.copy()  
 for i in [0, 1]:  
 if y[i] <= 0:  
      a\_rb[i] += self.e[i]  
 else:  
      a\_rb[i] -= self.e[i]  
 self.P.a = a\_rb  
 return self.P(y, t)

Note that the argmax may be non-convex and therefore difficult to solve. For more complex problems, one way is to use affine arithmetic for help [@liu2007robust].

## Multi-parameter Network Problems

Given a network represented by a directed graph . Consider :

where is the weight function of edge .

Assume that the network is large but the number of parameters is small. Given and , the problem has a feasible solution if and only if contains no negative cycle. Let be a set of all cycles of . We can formulate the problem as:

where is a cycle of :

### Negative Cycle Detection Algorithm

The negative cycle detection is the most time-consuming part of the proposed method, so it is very important to choose the proper negative cycle detection algorithm. There are lots of methods to detect negative cycles in a weighted graph [@cherkassky1999negative], in which Tarjan’s algorithm [@Tarjan1981negcycle] is one of the fastest algorithms in practice [@alg:dasdan\_mcr; @cherkassky1999negative].

The separation oracle only needs to determine:

* If there exists a negative cycle under , then
* the cut =
* If , then the cut = .
* Otherwise, is feasible, then
  + .
  + The cut =

### Example: symmetric scalings under the min-max-ratio criterion

This example is taken from [@orlin1985computing]. Given a matrix . A *symmetric scaling* of is a matrix of the form where is a nonnegative diagonal matrix with the same dimension. According to the *min-max criterion*, the aim is to minimize the largest absolute value of ’s elements [@orlin1985computing] (Program 3).

Another possible criterion is to minimize the ratio of largest absolute value of the element to the smallest. One motivation for using this criterion is that high ratios cause difficulties in performing the simplex method. With this *min-max-ratio* criterion, the symmetric scaling problem can be formulated as [@orlin1985computing] (Program 8):

Let denotes . By taking the logarithm of the variables, we can transform the above programming into a two-parameter network optimization problem:

where . The authors of [@orlin1985computing] claim that they have developed an efficient algorithm for solving such multi-parameter problem, but we could not find any follow-up publication on this. Interestingly, by using the cutting-plane method, one can easily extend the single-parameter network algorithm to a multi-parameter one.

In this application, is:

We can find fast algorithms for finding negative cycle in [@dasdan1998faster; @dasdan2004experimental]. More applications to clock skew scheduling can be found in [@zhou2015multi].

## Problems Involving Matrix Inequalities

Consider the following problem:

where is a matrix-valued function, denotes is positive semidefinite. Recall that a matrix is positive semidefinite if and only if for all . We can transform the problem into:

Consider is concave for all w.r.t. , then the above problem is a convex programming. Reduce to *semidefinite programming* if is linear w.r.t. , i.e., .

### Cholesky Factorization Algorithm

An alternative form, eliminating the need to take square roots, is the symmetric indefinite factorization:

If is real, the following recursive relations apply for the entries of and :

Again, the pattern of access allows the entire computation to be performed in-place if desired.

The following is the algorithm written in Python:

def factor(self, getA):  
 T = self.T  
 for i in range(self.n): # from 0 to n-1  
 for j in range(i+1): # from 0 to i  
        d = getA(i, j) - np.dot(T[:j, i], T[j, :j])  
        T[i, j] = d  
 if i != j:  
        T[j, i] = d / T[j, j]  
 if d <= 0.: # strictly positive  
 self.p = i  
 return  
 self.p = self.n

The vector can be found. The following is the algorithm written in Python:

def witness(self):  
    p = self.p  
    n = p + 1  
 v = np.zeros(n)  
    v[p] = 1  
 for i in range(p, 0, -1): # backward substitution  
        v[i-1] = -np.dot(self.T[i-1, i:n], v[i:n])  
 return v, -self.T[p, p]

The oracle only needs to:

* Perform a *row-based* Cholesky factorization such that .
* Let denotes a submatrix .
* If Cholesky factorization fails at row ,
  + there exists a vector , such that
    - , and
    - .
  + The cut =

### Example: Matrix Norm Minimization

Let . Problem can be reformulated as

Binary search on can be used for this problem.

### Example: Estimation of Correlation Function

## Random Field [@Schabenberger05]

*Random field*, also known as *stochastic process*, can be regarded as an indexed family of random variables denoted as {}, where is a subset of -dimensional Euclidean space . To specify a stochastic process, the joint probability distribution function of any finite subset must be given in a consistent way, which is called *distribution* of the process. For ease of analysis, a random field is often assumed to be with *Gaussian* distribution and is called Gaussian random field.

A random field has several key properties useful in practical problems. The field is *stationary* under translations, or *homogeneous*, if the distribution is unchanged when the point set is translated. The field is *isotropic* if the distribution is invariant under any rotation of the whole points in the parameter space. We study the homogeneous isotropic field in this paper.

The *covariance* and *correlation* of a stochastic process are defined by

and

respectively for all , where denotes the expectation of . Thus a process is homogeneous if and depend only on the separation vector . Furthermore, it is isotropic if and depend upon only through its length , i.e.,

If we denote , the variance of , as , then the relationship between covariance and correlation is .

When the two components are considered, the measurement data can still be regarded as a Gaussian random field, but the correlation function will have a discontinuity at the origin. We call this phenomenon “nugget effect” [@Diggle07].

Let , where ’s are the unknown coefficients to be fitted ’s are a family of basis functions. The covariance matrix can be recast as:

where .

# Ellipsoid Method Revisited

Some History of Ellipsoid Method [@BGT81]. Introduced by Shor and Yudin and Nemirovskii in 1976. Used to show that linear programming (LP) is polynomial-time solvable (Kachiyan 1979), settled the long-standing problem of determining the theoretical complexity of LP. In practice, however, the simplex method runs much faster than the method, although its worst-case complexity is exponential.

## Basic Ellipsoid Method

An ellipsoid is specified as a set

where is the center of the ellipsoid and is a positive definite matrix.

Updating the ellipsoid (deep-cut)

Calculation of minimum volume ellipsoid covering:

Let , . We can make the following observations:

1. If (shallow cut), then no smaller ellipsoid can be found.
2. If , then intersection is empty.
3. Otherwise,
4. where

Even better, split into two variables . Let , , .

Reduce multiplications per iteration. Note that:

* The determinant of decreases monotonically.
* The range of is

## Central Cut

A Special case of when . Deserve a separate implement because it is much simpler. Let , ,

## Parallel Cuts

Oracle returns a pair of cuts instead of just one. The pair of cuts is given by and such that:

for all .

Only linear inequality constraint can produce such parallel cut:

Usually, provide faster convergence.

![Parallel cuts](data:application/pdf;base64,)

Parallel cuts

Updating the ellipsoid.

Let , .

* If , intersection is empty.
* If , no smaller ellipsoid can be found.
* If , it reduces to deep-cut with .

Otherwise,

where

### Example: FIR filter design

A typical structure of digital Finite Impulse Response (FIR) filter is shown in Fig. 2, where the coefficients must be determined to meet given specifications. Usually, they can be manually designed using windowing or frequency-sampling techniques [@oppenheim1989discrete].

However, the experience and knowledge of designers are highly demanded in this kind of design methods. Moreover, there is no guarantee about the design’s quality. Therefore, the optimization-based techniques (e.g. [@wu1999fir], more reference) have attracted tons of research effort. In this kind of methods, facilitated with growing computing resource and efficient optimization algorithms, the solution space can be effectively explored.

![A typical structure of an FIR filter [@mitra2006digital].](data:application/pdf;base64,)

A typical structure of an FIR filter [@mitra2006digital].

In optimization algorithms, what is particularly interesting is the convex optimization. If a problem is in a convex form, it can be efficiently and optimally solved. Convex optimization techniques are also implementable in designing FIR filters, including Parks-McClellan algorithm [@park1972chebyshev], METEOR [@steiglitz1992meteor] and peak-constrained least-squares (PCLS) [@selesnick1996constrained; @adams1998peak]. In the mentioned articles, with the help of exchange algorithms (e.g. Remez exchange algorithm), certain FIR filter design problems can be formed as linear or quadratic programs. They are two simple forms of convex optimization problems, which can be optimally solved with existing algorithms, such as the interior-point method [@boyd2009convex]. Tempted by the optimality, more efforts were devoted to form the problem convex. Particularly, in [@wu1999fir], via spectral factorization [@goodman1997spectral], the problem of designing an FIR filter with magnitude constraints on frequency-domain is formulated as a convex optimization problem. More examples are provided in [@davidson2010enriching].

Its time response is

where is the filter coefficients. Its frequency response is

where , is the order of the filter. The design of a filter with magnitude constraints is often formulated as a constraint optimization problem as the form

where is the vector of design variables, represents the characteristics of the desirable filter and is the performance metric to be optimized. For example, the magnitude constraints on frequency domain are expressed as

where and are the lower and upper (nonnegative) bounds at frequency respectively. Note that is periodic and . Therefore, we can only consider the magnitude constraint on [@wu1999fir].

Generally, the problem might be difficult to solve, since we can only obtain the global optimal solution with resource consuming methods, such as branch-and-bound [@davidson2010enriching]. However, the situation is totally different if the problem is convex, where and are convex functions. In such a case, the problem can be optimally solved with many efficient algorithms.

Attracted by the benefits, the authors of [@wu1999fir] transformed (?), originally non-convex, into a convex form via spectral factorization:

where and are the autocorrelation coefficients. Especially, can be determined by , with the following equation vice versa [@wu1999fir]:

where for or .

![Result](data:application/pdf;base64,)

Result

### Example: Maximum Likelihood estimation

Consider

Note that the 1st term is concave, the 2nd term is convex. However, if there are enough samples such that is a positive definite matrix, then the function is convex within . Therefore, the following problem is convex:

## Discrete Optimization

Many engineering problems can be formulated as a convex/geometric programming, e.g. digital circuit sizing. Yet in an ASIC design, often there is only a limited set of choices from the cell library. In other words, some design variables are discrete. We can formulate the discrete version as Mixed-Integer Convex programming (MICP) by mapping the design variables to integers.

What’s wrong with the existing methods? Mostly based on relaxation. Then use the relaxed solution as a lower bound and use the branch-and-bound method for the discrete optimal solution. Note that the branch-and-bound method does not utilize the convexity of the problem. What if I can only evaluate constraints on discrete data?

Usually a relaxed optimal solution (convex) is obtained first. Then the optimized discrete solution is obtained by searching exhaustively the neighborhood. However, sometimes the constraints are tight so that the relaxed continuous optimal solution is far from the discrete one. Enumeration of the discrete domain is difficult.

Consider:

where and are “convex”. Note that some design variables are discrete. The oracle looks for the nearby discrete solution of with the cutting-plane:

Note that the cut may be a shallow cut. Suggestion: use different cuts as possible for each iteration (e.g. round-robin the evaluation of constraints).

### Example: Multiplier-less FIR Filter Design

However, there are still many filter design problems that are non-convex, such as multiplier-less FIR filter design problems. Note that in Fig. 2, each coefficient associated with a multiplier unit makes the filter power hungry, especially in Application Specific Integrated Circuits (ASIC). Fortunately, it can be implemented multiplier-less if each coefficient is quantized and represented as a sum of Singed Power-of-Two (SPT). Such a coefficient can be uniquely represented by a Canonic Signed-Digit (CSD) code [@george1960csd] with the smallest number of non-zero digits. In such a case, it confines the multiplication to add and shift operations. An example is shown in Fig. **¿fig:multi-shift?**. A coefficient 0.40625 = 13/32 can be written as . Consequently, as shown in Fig. **¿fig:shift?**, the multiplier can be replaced with three shifters and two adders, which are with much lower cost. However, the coefficient quantization constraint, which is non-convex, makes the convex optimization algorithm cannot be directly applied. A similar scenario is considering the finite word-length effect [@lim1982finite].

Attracted by the benefits of this “multiplierlessness”, many efforts have been devoted to its design techniques. For its general problems, integer programming (e.g. [@kodek1980design; @lim1982finite; @lim1983fir; @lim1999signed]) can be implemented to achieve the optimal solution. However, it demands excessive computing resources. Other heuristic techniques, such as genetic algorithm [@xu1995design] and dynamic-programming-like method [@chen1999trellis], are also with low efficiency. If the quantization constraint is the only non-convex constraint in the design problem, a lower bound can be efficiently obtained by solving the relaxed problem [@davidson2010enriching]. Then to make the solution feasible, it can be rounded to the nearest CSD codes or treated as a starting point of a local search algorithm for a better solution [@kodek1981comparison]. However, both of the methods can not guarantee the feasibility of the final solution. Besides, the local search problem is still non-convex. Therefore, the adopted algorithm could also be inefficient, such as branch-and-bound in [@kodek1981comparison].

![Result](data:application/pdf;base64,)

Result

# Concluding Remarks

Should be known to student. Ellipsoid method is not competitor but companion of interior point methods.

TBD.

# References