Cutting Plane and Subgradient Methods

John E. Mitchell

Department of Mathematical Sciences
RPI, Troy, NY 12180 USA

October 12, 2009

Collaborators: Luc Basescu, Brian Borchers, Mohammad Oskoorouchi, Srini Ramaswamy, Kartik Sivaramakrishnan, Mike Todd
1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
The Traveling Salesman Problem

Lower bounds determined using cutting planes and branch-and-cut.

TSP Webpage
Bill Cook, Georgia Tech

Robert Bosch
Oberlin College
MIP Formulation

Want to solve the integer programming problem

$$\max_{y \in \mathbb{R}^m} \quad b^T y$$

subject to

$$A^T y \leq c$$

$$y_i \quad \text{integer for } i \in I$$
LP Relaxation

Relax integrality restriction:

Dual problem:

\[
\begin{align*}
\max_{y \in \mathbb{R}^m} & \quad b^T y \\
\text{s.t.} & \quad A^T y \leq c \\
& \quad a_0^T y \leq c_0
\end{align*}
\]

Primal problem:

\[
\begin{align*}
\min_{x \in \mathbb{R}^n, x_0} & \quad c^T x + c_0 x_0 \\
\text{s.t.} & \quad Ax + a_0 x_0 = b \\
& \quad x \geq 0
\end{align*}
\]

Add a cutting plane to the dual problem

Corresponds to column generation in the primal.
LP Relaxation

Relax integrality restriction:

Dual problem:
\[
\begin{align*}
\max_{y \in \mathbb{R}^m} & \quad b^T y \\
\text{s.t.} & \quad A^T y \leq c \\
& \quad a_0^T y \leq c_0
\end{align*}
\]

Primal problem:
\[
\begin{align*}
\min_{x \in \mathbb{R}^n, x_0} & \quad c^T x + c_0 x_0 \\
\text{s.t.} & \quad Ax + a_0 x_0 = b \\
& \quad x \geq 0
\end{align*}
\]

Add a cutting plane to the dual problem

Corresponds to column generation in the primal.
LP Relaxation

Relax integrality restriction:

Dual problem:

\[
\begin{align*}
\max_{y \in \mathbb{R}^m} & \quad b^T y \\
\text{s.t.} & \quad A^T y \leq c \\
& \quad a_0^T y \leq c_0
\end{align*}
\]

Primal problem:

\[
\begin{align*}
\min_{x \in \mathbb{R}^n, x_0} & \quad c^T x + c_0 x_0 \\
\text{s.t.} & \quad Ax + a_0 x_0 = b \\
& \quad x, x_0 \geq 0
\end{align*}
\]

Add a cutting plane to the dual problem.

Corresponds to column generation in the primal.
LP Relaxation

Relax integrality restriction:

Dual problem:
\[
\max_{y \in \mathbb{R}^m} \quad b^T y \\
\text{s.t.} \quad A^T y \leq c \\
\quad a_0^T y \leq c_0
\]

Primal problem:
\[
\min_{x \in \mathbb{R}^n, x_0} \quad c^T x + c_0 x_0 \\
\text{s.t.} \quad A x + a_0 x_0 = b \\
\quad x, x_0 \geq 0
\]

Add a cutting plane to the dual problem

Corresponds to column generation in the primal.
Cutting plane algorithm illustration

\[ \{ y \in \mathbb{R}^m : A^T y \leq c \} \]
Cutting plane algorithm illustration

\[ \{ y \in \mathbb{R}^m : A^T y \leq c \} \]
Cutting plane algorithm illustration

\[ \{ y \in \mathbb{R}^m : A^T y \leq c, \ a_i^T y \leq c_1 \} \]
Cutting plane algorithm illustration

\( \{ y \in \mathbb{R}^m : A^T y \leq c, \ a_1^T y \leq c_1 \} \)

Cutting plane

\( a_2^T y = c_2 \)

Soln to LP relaxation
Cutting plane algorithm illustration

\[ \{ y \in \mathbb{R}^m : A^T y \leq c, \ a_1^T y \leq c_1, \ a_2^T y \leq b_2 \} \]

Solution to LP relaxation
Cutting plane algorithm illustration

\[ \{ y \in \mathbb{R}^m : A^T y \leq c, \ a_1^T y \leq c_1, \ a_2^T y \leq b_2 \} \]

Solution to LP relaxation
Solution is integral, so optimal to IP
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
     - Interior point cutting plane methods
     - Warm starting
     - Theoretical results
     - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Valid constraints for MaxCut

Find the maximum cut in a graph.


Interactions known. Deduce spins.

\[ y_{uv} = \begin{cases} 
1 & u, v \text{ opposite spins} \\
0 & u, v \text{ same spin} 
\end{cases} \]

\[ y \leftrightarrow \text{incidence vector of cut.} \]

Any cut and any cycle intersect in an even number of edges.

Valid constraint:

\[ y_e + y_f + y_g - y_h \leq 2 \]
Valid constraints for MaxCut

Find the maximum cut in a graph.

Find ground state of Ising spin glass.

Vertices $\leftrightarrow$ spins: “Up” or “Down”.
Edges $\leftrightarrow$ interactions: $\pm 1$.

Interactions known. Deduce spins.

$$y_{uv} = \begin{cases} 1 & u, v \text{ opposite spins} \\ 0 & u, v \text{ same spin} \end{cases}$$

$y \leftrightarrow$ incidence vector of cut.

Any cut and any cycle intersect in an even number of edges.

Valid constraint:

$$y_e + y_f + y_g - y_h \leq 2$$
Integer programming formulation

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} \\
& \quad y \text{ binary}
\end{align*}
\]

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} \\
& \quad y \text{ satisfies additional linear constraints} \\
& \quad 0 \leq y \leq 1
\end{align*}
\]

Algorithm:
1. Initialize: just the box constraints \(0 \leq y \leq 1\).
2. Solve LP relaxation
3. Add violated constraints as required.
4. If not yet converged, return to Step 2.

Can typically solve 100x100 grids in about 5 minutes with \(b_e = \pm 1\).
Integer programming formulation

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} \\
& \quad y \text{ binary}
\end{align*}
\]

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} \\
& \quad y \text{ satisfies additional linear constraints} \\
& \quad 0 \leq y \leq 1
\end{align*}
\]

Algorithm:

1. Initialize: just the box constraints \(0 \leq y \leq 1\).
2. Solve LP relaxation
3. Add violated constraints as required.
4. If not yet converged, return to Step 2.

Can typically solve 100x100 grids in about 5 minutes with \(b_e = \pm 1\).
Integer programming formulation

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} \\
& \quad y \text{ binary}
\end{align*}
\]

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} & \quad \text{(IP)} \\
& \quad y \text{ satisfies additional linear constraints} \\
& \quad 0 \leq y \leq 1 & \quad \text{(LP)}
\end{align*}
\]

Algorithm:

1. Initialize: just the box constraints \(0 \leq y \leq 1\).
2. Solve LP relaxation
3. Add violated constraints as required.
4. If not yet converged, return to Step 2.

Can typically solve 100x100 grids in about 5 minutes with \(b_e = \pm 1\).
Integer programming formulation

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} \quad (IP) \\
& \quad y \text{ binary}
\end{align*}
\]

\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} b_e y_e \\
\text{subject to} & \quad y \text{ satisfies all cut-cycle inequalities} \quad (LP) \\
& \quad y \text{ satisfies additional linear constraints} \\
& \quad 0 \leq y \leq 1
\end{align*}
\]

Algorithm:

1. Initialize: just the box constraints \(0 \leq y \leq 1\).
2. Solve LP relaxation
3. Add violated constraints as required.
4. If not yet converged, return to Step 2.

Can typically solve 100x100 grids in about 5 minutes with \(b_e = \pm 1\).
Computational results

Compare interior point and simplex cutting plane algorithms. Times in seconds (rounded). $b_e = \pm 1$. One problem of each size.

<table>
<thead>
<tr>
<th>Grid</th>
<th>Interior</th>
<th>Simplex</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>50x50</td>
<td>4</td>
<td>18</td>
<td>4.87</td>
</tr>
<tr>
<td>60x60</td>
<td>31</td>
<td>271</td>
<td>8.90</td>
</tr>
<tr>
<td>70x70</td>
<td>54</td>
<td>786</td>
<td>14.75</td>
</tr>
<tr>
<td>80x80</td>
<td>51</td>
<td>639</td>
<td>12.72</td>
</tr>
<tr>
<td>90x90</td>
<td>223</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>100x100</td>
<td>187</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

**Interior point:**
one core of Mac Pro with 2x2.8 GHz Quad-Core Intel Xeon. Personal code.

**Simplex** (spin glass server):
Intel(R) Celeron(R) M CPU 440 @ 1.86GHz. CPLEX 9.1.

**Interior** is faster, with the ratio getting better as problem size increases.
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
     - Warm starting
     - Theoretical results
     - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Solve the LP relaxations using **interior point methods**: good for large scale LPs.

These methods work well when a lot of violated cuts can be added at once.

The LP relaxations need only be solved approximately, so the resulting cutting planes can be more centered.

**Combining interior point and simplex methods** is especially effective: use the interior point method initially; switch to simplex once sufficiently close to optimality.
Comparing the strength of simplex and interior point cutting planes

Simplex:

- Optimal vertex found by simplex
- Added cutting plane when using simplex

Interior point method:

- Interior point iterate
- Central trajectory
- Optimal face
- Added cutting plane when using interior point method
Comparing the strength of simplex and interior point cutting planes

**Simplex:**
- Optimal vertex found by simplex
- Added cutting plane when using simplex

**Interior point method:**
- Interior point iterate
- Optimal face
- Central trajectory
- Added cutting plane when using interior point method
Linear ordering problems

Ratio: Combined time / Simplex time

Different problem classes:
- 0% zeroes
- 10% zeroes
- 20% zeroes
- Type B
- Type C

Simplex time (secs)
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - **Warm starting**
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Warm starting an interior point method

Can’t be done
Warm starting an interior point method

Can be done to some degree!

Can reduce iteration counts by 50%, perhaps more.
Warm starting an interior point method

Can be done to some degree!

Can reduce iteration counts by 50%, perhaps more.
Warm starting an interior point method

Can be done to some degree!

Can reduce iteration counts by 50%, perhaps more.
Warm starting an interior point method

Can be done to some degree!

Can reduce iteration counts by 50%, perhaps more.
Warm starting an interior point method

Can be done to some degree!

Can reduce iteration counts by 50%, perhaps more.

Need to make some effort to recover centrality instead of immediately aiming for new optimal solution.

Useful to store potential restart points: earlier iterates, or earlier approximate analytic centers (Gondzio), or problem-specific points.
Warm starting an interior point method

Can be done to some degree!

Can reduce iteration counts by 50%, perhaps more.

Need to make some effort to recover centrality instead of immediately aiming for new optimal solution.

Useful to store potential restart points: earlier iterates, or earlier approximate analytic centers (Gondzio), or problem-specific points.

Want to get to final straighter portion of trajectory.
Using Dikin ellipsoid to restart

Current relaxation and iterate

\[ A^T y \leq c \]
Using Dikin ellipsoid to restart

Dikin ellipsoid:
inscribed ellipsoid centered at $\tilde{y}$
Using Dikin ellipsoid to restart

Add cutting plane

\[ a_0^T \bar{y} = a_0^T \bar{y} \]
Using Dikin ellipsoid to restart

Use Dikin ellipsoid as proxy for old constraints
Dikin Ellipsoid

Find restart direction $d$

$$a_0^T \hat{y} = a_0^T \bar{y}$$

Move as far off the added constraint as possible, while staying within the Dikin ellipsoid.
Another Dikin Ellipsoid

Find restart direction $d$
when adding multiple constraints

Minimize the potential function of the new slack variables, $- \sum \ln s_i$,
while staying within the Dikin ellipsoid. (Goffin and Vial)
Primal restart

\[
\begin{aligned}
\min_{x \in \mathbb{R}^n, x_0} & \quad c^T x + c_0 x_0 \\
\text{s.t.} & \quad Ax + a_0 x_0 = b \\
& \quad x, x_0 \geq 0
\end{aligned}
\]

Restart from a point \( x = \bar{x}, x_0 = 0 \).

Using a scaling matrix \( D \), get direction

\[
\Delta x := -D^2 A^T (AD^2 A^T)^{-1} a_0
\]

This can be derived from the Dikin ellipsoid.

Can be generalized to the addition of multiple columns.
**Primal restart**

$$\min_{x \in \mathbb{R}^n, x_0} \quad c^T x + c_0 x_0$$

s.t. \quad Ax + a_0 x_0 = b

\[ x, x_0 \geq 0 \]

Restart from a point \( x = \bar{x}, x_0 = 0 \).

Using a scaling matrix \( D \), get direction

\[
\Delta x := -D^2 A^T (AD^2 A^T)^{-1} a_0
\]

This can be derived from the Dikin ellipsoid.

Can be generalized to the addition of multiple columns.
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Theoretical results for the convex feasibility problem

Convex feasibility problem:

Given convex set $C$ and a polyhedral outer approximation to $Y$, either find a point in $Y$ or determine $Y$ is empty.
Theoretical results for the convex feasibility problem

Convex feasibility problem:
Given convex set $C$ and a polyhedral outer approximation to $Y$, either find a point in $Y$ or determine $Y$ is empty.
Theoretical results for the convex feasibility problem

Convex feasibility problem:

Given convex set $C$ and a polyhedral outer approximation to $Y$, either find a point in $Y$ or determine $Y$ is empty.
Theoretical results for the convex feasibility problem

Convex feasibility problem:

Given convex set $C$ and a polyhedral outer approximation to $Y$, either find a point in $Y$ or determine $Y$ is empty.
Convergence theorem (Goffin and Vial)

Use interior point cutting plane algorithm.

Use approximate analytic centers as iterates.

Add up to $p$ cuts at each call to oracle. Never drop cuts.

**Theorem**

*If $C$ contains a ball of radius $\varepsilon$ then a certain interior point cutting plane method converges after adding no more than $O\left(\frac{m^2p^2}{\varepsilon^2}\right)$ cutting planes.*

*Each recentering step requires at most $O(p \ln p)$ Newton steps.*
Convergence theorems (Atkinson and Vaidya)

Also allow dropping of constraints. Add at most one cut at a time.

**Theorem**

*If C contains a ball of radius \( \varepsilon \) then an interior point variant converges after adding no more than \( O(m \ln(\frac{1}{\varepsilon})^2) \) iterations, when extra conditions are used to determine constraints to add and drop.*

Volumetric center cutting plane methods:

Add or drop one constraint at a time.

**Theorem**

*If C contains a ball of radius \( \varepsilon \) then an interior point variant stops in \( O(m \ln(\frac{1}{\varepsilon})) \) calls to the oracle and \( O(m \ln(\frac{1}{\varepsilon})) \) approximate Newton steps.*
Convergence theorems (Atkinson and Vaidya)

Also allow dropping of constraints. Add at most one cut at a time.

**Theorem**

If $C$ contains a ball of radius $\varepsilon$ then an interior point variant converges after adding no more than $O(m \ln(\frac{1}{\varepsilon})^2)$ iterations, when extra conditions are used to determine constraints to add and drop.

Volumetric center cutting plane methods:

Add or drop one constraint at a time.

**Theorem**

If $C$ contains a ball of radius $\varepsilon$ then an interior point variant stops in $O(m \ln(\frac{1}{\varepsilon}))$ calls to the oracle and $O(m \ln(\frac{1}{\varepsilon}))$ approximate Newton steps.
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Links with Stabilization

Maximize concave function $f(y)$.

Subgradient inequality: $f(y) \leq f(\bar{y}) + \xi^T(y - \bar{y})$
Links with Stabilization

Maximize concave function $f(y)$.
Subgradient inequality: $f(y) \leq f(\bar{y}) + \xi^T(y - \bar{y})$
Links with Stabilization

Maximize concave function $f(y)$.
Subgradient inequality: $f(y) \leq f(\bar{y}) + \xi^T(y - \bar{y})$
Links with Stabilization

Maximize concave function $f(y)$.

Subgradient inequality: $f(y) \leq f(\bar{y}) + \xi^T(y - \bar{y})$
Links with Stabilization

Maximize concave function $f(y)$.
Subgradient inequality: $f(y) \leq f(\bar{y}) + \xi^T(y - \bar{y})$
Links with Stabilization

Maximize concave function $f(y)$.

Subgradient inequality:

$$f(y) \leq f(\bar{y}) + \xi^T(y - \bar{y})$$
Links with Stabilization

Maximize concave function $f(y)$.
Subgradient inequality: $f(y) \leq f(\bar{y}) + \xi^T(y - \bar{y})$
Stabilization / regularization

Iterates jump around if solve piecewise linear overestimator to optimality.

Try to stabilize the process.

One possibility: subtract proximal term $\frac{u}{2}||y - \bar{y}||^2$ from objective. Used in bundle methods.

Alternative: use interior point method and only solve the relaxations approximately.
Stabilization with interior point methods
Stabilization with interior point methods
Stabilization with interior point methods

Find approximate analytic center

lower bound
Stabilization with interior point methods

Find approximate analytic center
Stabilization with interior point methods
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions

Mitchell (RPI)
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
SDP relaxation of MaxCut

$y_e = 1$ if edge $e$ in cut, 0 otherwise.
Let $x_{uv} = 2y_{uv} - 1$ for each pair of vertices $u, v$.

$x_{uv}$ can be regarded as entries in matrix $X$:

Let $z$ be a vector: $z_u = +1$ on one side of cut, $z_u = -1$ on other side.

Then $X = zz^T$.

\[
\begin{align*}
\max_X & \quad C \cdot X & \quad \text{(Frobenius inner product)} \\
\text{subject to} & \quad X_{vv} = 1 \quad \forall v \in V \\
& \quad X \succeq 0 \quad \text{(ie, } X \text{ symmetric, psd)} \\
& \quad \text{rank}(X) = 1
\end{align*}
\]

Relaxation: drop rank requirement.
Then get within 0.878 of optimality if all $c_e \geq 0$ (Goemans and Williamson).
SDP relaxation of MaxCut

\( y_e = 1 \) if edge \( e \) in cut, 0 otherwise.
Let \( x_{uv} = 2y_{uv} - 1 \) for each pair of vertices \( u, v \).

\( x_{uv} \) can be regarded as entries in matrix \( X \):

Let \( z \) be a vector: \( z_u = +1 \) on one side of cut, \( z_u = -1 \) on other side.

Then \( X = zz^T \).

\[
\begin{align*}
\max_X & \quad C \bullet X \\
\text{subject to} & \quad X_{vv} = 1 \quad \forall v \in V \\
& \quad X \succeq 0 \quad \text{(ie, } X \text{ symmetric, psd)} \\
& \quad \text{rank}(X) = 1
\end{align*}
\]

Relaxation: drop rank requirement.
Then get within 0.878 of optimality if all \( c_e \geq 0 \) (Goemans and Williamson).
Applications of Semidefinite Programming

- **Combinatorial optimization**: Obtain stronger relaxations than using linear programming for MaxCut, Satisfiability, Independent Set, ...
- Control theory
- Structural optimization
- Electronic structure calculation and ground states
- Relaxations of other optimization problems: For example, quadratically constrained quadratic programs.
- Statistics: For example, principal component analysis.
- Data mining: Distance metric learning.
Applications of Semidefinite Programming

- **Combinatorial optimization**: Obtain stronger relaxations than using linear programming for MaxCut, Satisfiability, Independent Set, ...
- **Control theory**
  - Structural optimization
  - Electronic structure calculation and ground states
  - Relaxations of other optimization problems: For example, quadratically constrained quadratic programs.
- **Statistics**: For example, principal component analysis.
- **Data mining**: Distance metric learning.
Applications of Semidefinite Programming

- **Combinatorial optimization**: Obtain stronger relaxations than using linear programming for MaxCut, Satisfiability, Independent Set, ...
- Control theory
- Structural optimization
- Electronic structure calculation and ground states
- **Relaxations of other optimization problems**: For example, quadratically constrained quadratic programs.
- **Statistics**: For example, principal component analysis.
- **Data mining**: Distance metric learning.
Semidefinite Programming

A semidefinite program has a linear objective function and linear constraints.

The variables can be written in the form of a matrix.

This matrix of variables is constrained to be positive semidefinite.

SDPs can be solved in polynomial time using interior point methods.

SDPs generalize linear programming.

Can be generalized further to conic programming:

linear objective function, linear constraints, the variables belong to a convex cone.
### SDP primal-dual pair

<table>
<thead>
<tr>
<th>Primal</th>
<th>Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \min \ C \bullet X )</td>
<td>( \max \ b^T y )</td>
</tr>
<tr>
<td>s.t. ( \mathcal{A}(X) = b )</td>
<td>s.t. ( \mathcal{A}^*(y) + S = C )</td>
</tr>
<tr>
<td>( X \succeq 0 )</td>
<td>( S \succeq 0 )</td>
</tr>
</tbody>
</table>

**Notation:**

- \( C \bullet X = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij}X_{ij} = \text{trace}(CX) \).
- \( \mathcal{A}(X) \) is an \( m \)-vector. Components \( A_i \bullet X \).
- \( \mathcal{A}^*(y) \) is an \( n \times n \) matrix. \( \mathcal{A}^*(y) = \sum_{i=1}^{m} y_i A_i \).
Assumptions

1. The matrices $A_i$ are linearly independent.

2. Primal and dual problems both have strictly feasible solutions. Consequently: the primal and dual values agree.

3. Every primal feasible solution satisfies $\text{trace}(X) = 1$. Equivalent to assuming the primal problem has a bounded feasible region. SDP relaxations of combinatorial optimization problems can be rescaled to satisfy this assumption.

First two assumptions are standard in interior point literature for SDP. Third one is due to Helmberg and Rendl (2000).
Assumptions

1. The matrices $A_i$ are linearly independent.

2. Primal and dual problems both have strictly feasible solutions. Consequently: the primal and dual values agree.

3. Every primal feasible solution satisfies $\text{trace}(X) = 1$. Equivalent to assuming the primal problem has a bounded feasible region. SDP relaxations of combinatorial optimization problems can be rescaled to satisfy this assumption.

First two assumptions are standard in interior point literature for SDP. Third one is due to Helmberg and Rendl (2000).
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Why cutting planes and surfaces?

- Interior point algorithms solve SDPs in polynomial time.
- But, computational time becomes impractical for larger problems.

- Alternatively, use relaxations or approximations to get a good solution reasonably quickly.
- These relaxation approaches will also typically give an indication of a final duality gap.
Why cutting planes and surfaces?

- Interior point algorithms solve SDPs in \textit{polynomial time}.
- But, computational time becomes \textit{impractical for larger problems}.
- Alternatively, use \textit{relaxations or approximations} to get a good solution reasonably quickly.
- These relaxation approaches will also typically give an indication of a \textit{final duality gap}.
An example of an SDP feasible region

\[ S = \begin{bmatrix} y_1 & y_2 & 0 \\ y_2 & y_1 - 3 & 0 \\ 0 & 0 & y_1 - 4 \end{bmatrix} \]

Feasible region in \( y \)-space

(4,2)

(4,-2)
Tangent line approximation

\[ S = \begin{bmatrix} y_1 & y_2 & 0 \\ y_2 & y_1 - 3 & 0 \\ 0 & 0 & y_1 - 4 \end{bmatrix} \]

Linear approximation
A relaxation of SDP

Restricted primal, relaxed dual: For a given matrix $P$:

$$\begin{align*}
\min_V & \quad C \bullet PVP^T \\
\text{s.t.} & \quad A(PVP^T) = b \\
& \quad V \succeq 0
\end{align*}$$

$$\begin{align*}
\max_{y,S} & \quad b^T y \\
\text{s.t.} & \quad A^*(y) + S = C \\
& \quad P^TSP \succeq 0
\end{align*}$$

Restrict $X$ to be equal to $PVP^T$. Note $V \succeq 0 \Rightarrow X \succeq 0$.

Relax $S \succeq 0$ to $P^TSP \succeq 0$.

Possibly have additional restrictions on $V$ and corresponding weakening of the condition $P^TSP \succeq 0$.

The matrix $P$ is updated over the course of the algorithm.
A relaxation of SDP

Restricted primal, relaxed dual: For a given matrix $P$:

\[
\begin{align*}
\min_V & \quad C \bullet PVP^T \\
\text{s.t.} & \quad \mathcal{A}(PVP^T) = b \\
& \quad V \succeq 0
\end{align*}
\]

\[
\begin{align*}
\max_{y,S} & \quad b^Ty \\
\text{s.t.} & \quad \mathcal{A}^*(y) + S = C \\
& \quad P^TSP \succeq 0
\end{align*}
\]

Restrict $X$ to be equal to $PVP^T$. Note $V \succeq 0 \Rightarrow X \succeq 0$.

Relax $S \succeq 0$ to $P^TSP \succeq 0$.

Possibly have additional restrictions on $V$ and corresponding weakening of the condition $P^TSP \succeq 0$.

The matrix $P$ is updated over the course of the algorithm.
Algorithmic framework

1. Approximately solve the modified problem, get $X = PVP^T$, $S$.

2. If $S$ is not positive semidefinite:
   Add one or more columns to $P$ corresponding to eigenvectors of $S$ with negative eigenvalue.
   If desired, delete columns of $P$, or aggregate them into a linear term.
   Return to Step 1.

3. If duality gap too large, tighten the tolerance and return to Step 1.

4. STOP with an approximate solution to SDP.
Algorithmic framework

1. Approximately solve the modified problem, get $X = PVP^T$, $S$.

2. If $S$ is not positive semidefinite:
   
   Add one or more columns to $P$ corresponding to eigenvectors of $S$ with negative eigenvalue.

   If desired, delete columns of $P$, or aggregate them into a linear term.

   **Return to Step 1.**

3. If duality gap too large, tighten the tolerance and return to Step 1.

4. **STOP** with an approximate solution to SDP.
Algorithmic framework

1. Approximately solve the modified problem, get $X = PVP^T$, $S$.

2. If $S$ is not positive semidefinite:
   - Add one or more columns to $P$ corresponding to eigenvectors of $S$ with negative eigenvalue.
   - If desired, delete columns of $P$, or aggregate them into a linear term.
   - Return to Step 1.

3. If duality gap too large, tighten the tolerance and return to Step 1.

4. STOP with an approximate solution to SDP.
Why update $P$ with eigenvectors?

(See Helmberg (2000), for example).

Lagrangian relaxation of primal SDP:

$$\Theta(y) := \min_X b^T y + (C - \sum_{i=1}^{m} y_i A_i) \bullet X$$
subject to

$$I \bullet X = 1$$
$$X \succeq 0.$$  

Lagrangian dual problem:

$$\max_y \Theta(y) = b^T y + \lambda_{\min}(C - \sum_{i=1}^{m} y_i A_i).$$

($\lambda_{\min}$ denotes the minimum eigenvalue.)
Lagrangian dual problem

\[
\max_y \Theta(y) = b^T y + \lambda_{\min}(C - \sum_{i=1}^{m} y_i A_i).
\]

Under our assumptions, optimal value of Lagrangian dual problem is equal to optimal value of SDP. So no duality gap.

Lagrangian dual function is concave and nonsmooth. Overestimate it; improve overestimate with subgradient inequalities.

Subdifferential of \(\Theta(y)\) is derived from the eigenspace corresponding to the minimum eigenvalue.

If columns of \(P\) form an orthonormal basis for this eigenspace, the subdifferential inequality is \(P^T SP \succeq \theta I\), where \(\theta\) is estimate of \(\lambda_{\min}(C - \sum_{i=1}^{m} y_i A_i)\).
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
     - Theoretical results with conic cuts
     - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Variant 1: No additional restrictions on $V$

$$\begin{align*}
\min \quad & C \bullet P VP_T \\
\text{s.t.} \quad & A (P VP_T) = b \\
& V \succeq 0
\end{align*}$$

$$\begin{align*}
\max \quad & b^T y \\
\text{s.t.} \quad & A^*(y) + S = C \\
& P^T SP \succeq 0
\end{align*}$$

$$X = \begin{bmatrix}
P \\ V \\ P^T
\end{bmatrix}$$

Approximate the original SDP constraint by a single lower-dimensional SDP constraint.

If rank($P$) is equal to the dimension of the original $X$ then this formulation is exactly equivalent to the original SDP.

Typically, also include an extra linear term in the primal problem, and corresponding linear constraint in the dual, $W \bullet S \succeq 0$. 
Experience with lower-dimensional SDP constraint

Eg: Spectral bundle method (Helmberg, Rendl, Kiwiel)

Solve dual relaxation with additional proximal term:

\[
\begin{align*}
\max & \quad b^T y - \frac{u}{2} \| y - \bar{y} \|^2 \\
\text{subject to} & \quad A^T y + S = C \\
& \quad P^T S P \succeq 0 \\
& \quad W \cdot S \geq 0
\end{align*}
\]

\(\bar{y}\): current iterate.

\(u > 0, W \succeq 0\): dynamically updated.

Solved efficiently by primal-dual interior point method.

Converges under various assumptions.

Using \(W\) lets algorithm approach an optimal solution even when the number of columns in \(P\) is restricted.

Excellent computational results for SDP relaxations of combinatorial optimization problems.
Experience with lower-dimensional SDP constraint

Eg: Spectral bundle method (Helmberg, Rendl, Kiwiel)

Solve dual relaxation with additional \textbf{proximal term}:

\[
\begin{align*}
\max \quad & b^T y - \frac{u}{2} ||y - \bar{y}||^2 \\
\text{subject to} \quad & \mathcal{A}^T y + S = C \\
& P^T SP \succeq 0 \\
& W \cdot S \geq 0
\end{align*}
\]

$\bar{y}$: current iterate.

$u > 0$, $W \succeq 0$: dynamically updated.

Solved efficiently by primal-dual interior point method.

\textbf{Converges} under various assumptions.

Using $W$ lets algorithm approach an optimal solution even when the number of columns in $P$ is restricted.

Excellent computational results for SDP relaxations of combinatorial optimization problems.
Variant 2: Require $V$ to be diagonal

$$V = \begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}$$

\[
\begin{align*}
\min & \quad C \bullet PVP^T \\
\text{s.t.} & \quad A(PVP^T) = b \\
& \quad V \succeq 0, \text{ diagonal}
\end{align*}
\]

\[
\begin{align*}
\max & \quad b^T y \\
\text{s.t.} & \quad A^*(y) + S = C \\
& \quad \text{diagonal entries of } P^T SP \geq 0
\end{align*}
\]

Gives a linear programming relaxation of the dual SDP.

Each column $p$ of $P$ gives a constraint $p^T Sp \geq 0$, a linear constraint on the entries in $S$. 
Experience with LP relaxation

Eg: Sivaramakrishnan (2008)

Preprocess to get a block angular SDP:

\[
\begin{align*}
\min_{X_i} & \quad \sum_i C_i \cdot X_i \\
\text{s.t.} & \quad \sum_i A_i(X_i) = b \\
& \quad \sum_i d_i^T w_i \\
& \quad X_i \preceq 0 \quad \forall i
\end{align*}
\]

\[
\begin{align*}
\max_{y, w_i} & \quad b^T y + \sum_i d_i^T w_i \\
\text{s.t.} & \quad A_i^T y + B_i^T w_i \preceq C_i \quad \forall i
\end{align*}
\]

Block angular constraints and linking constraints
Experience with LP relaxation

Eg: Sivaramakrishnan (2008)

Preprocess to get a block angular SDP:

\[
\begin{align*}
\min_{X_i} & \quad \sum_i C_i \bullet X_i \\
\text{s.t.} & \quad \sum_i A_i(X_i) = b \\
\end{align*}
\]

\[
\begin{align*}
\max_{y, w_i} & \quad b^T y + \sum_i d_i^T w_i \\
\text{s.t.} & \quad A_i^T y + B_i^T w_i \preceq C_i \quad \forall i \\
B_i(X_i) & = d_i \quad \forall i \\
X_i & \succeq 0 \quad \forall i
\end{align*}
\]

Block angular constraints and linking constraints
Experience with LP relaxation

Example: Sivaramakrishnan (2008)

Preprocess to get a block angular SDP:

$$\min_{X_i} \sum_i C_i \cdot X_i$$

$$\max_{y,w_i} \ b^T y + \sum_i d_i^T w_i$$

$$\text{s.t. } \sum_i A_i(X_i) = b$$

$$\text{s.t. } A_i^T y + B_i^T w_i \preceq C_i \ \forall i$$

$$B_i(X_i) = d_i \ \forall i$$

$$X_i \succeq 0 \ \forall i$$

Block angular constraints and linking constraints
Block angular method (continued)

Lagrangian dual function $\Theta(y)$:

$$\Theta(y) := b^T y + \sum_{i=1}^{r} \min\{(C_i - A_i^T y) \cdot X_i : A_i(X_i) = b, X_i \succeq 0\}$$

No duality gap under assumptions.
Construct **piecewise linear approximations** by solving in **parallel** the disaggregated semidefinite programming subproblems

$$\Theta_i(y) := \min\{(C_i - A_i^T y) \cdot X_i : A_i(X_i) = b, X_i \succeq 0\}$$

for different choices of $y$ arising as solutions of a **master problem**.
Get subgradients $-A_i(X_i)$ of $\Theta_i(y)$.

**Master problem**: use stabilized column generation procedure.

Can solve SDPs with $n, m > 10000$ to duality gap of $10^{-2}$ or $10^{-3}$. 

Mitchell (RPI)
Variant 3: Require $V$ to be block-diagonal

$$V = \begin{bmatrix} \ldots \end{bmatrix}$$

$$\begin{align*}
\min & \quad C \bullet PVP^T \\
\text{s.t.} & \quad \mathcal{A}(PVP^T) = b \\
& \quad V \succeq 0, \text{ block diagonal}
\end{align*}$$

$$\begin{align*}
\max & \quad b^T y \\
\text{s.t.} & \quad \mathcal{A}^*(y) + S = C \\
\text{diagonal blocks of } P^T SP \succeq 0
\end{align*}$$

Approximate the original SDP constraint by several lower-dimensional SDP constraints.

For example, diagonal blocks of size two correspond to second order cone constraints. $M = \begin{bmatrix} a & b \\ b & c \end{bmatrix} \succeq 0 \iff a \geq 0, c \geq 0, ac \geq b^2$. (Oskoorouchi, Goffin, Mitchell)
Experience with SOCP constraints

<table>
<thead>
<tr>
<th></th>
<th>lc</th>
<th>socc</th>
<th>p</th>
<th>dim</th>
<th>gap</th>
<th>SOCCSM</th>
<th>SDPLR</th>
<th>SDPT3</th>
<th>SeDuMi</th>
</tr>
</thead>
<tbody>
<tr>
<td>300, 50, 100</td>
<td>14</td>
<td>46</td>
<td>3</td>
<td>109</td>
<td>9.2e-4</td>
<td>16</td>
<td>29</td>
<td>69</td>
<td>352</td>
</tr>
<tr>
<td>300, 100, 600</td>
<td>4</td>
<td>31</td>
<td>4</td>
<td>121</td>
<td>7.8e-4</td>
<td>56</td>
<td>78</td>
<td>225</td>
<td>843</td>
</tr>
<tr>
<td>300, 200, 1100</td>
<td>4</td>
<td>44</td>
<td>4</td>
<td>169</td>
<td>8.6e-4</td>
<td>191</td>
<td>146</td>
<td>503</td>
<td>1834</td>
</tr>
<tr>
<td>300, 300, 800</td>
<td>11</td>
<td>79</td>
<td>8</td>
<td>518</td>
<td>9.2e-4</td>
<td>739</td>
<td>--</td>
<td>656</td>
<td>--</td>
</tr>
<tr>
<td>300, 300, 1000</td>
<td>5</td>
<td>62</td>
<td>7</td>
<td>335</td>
<td>9.2e-4</td>
<td>561</td>
<td>--</td>
<td>747</td>
<td>--</td>
</tr>
<tr>
<td>500, 50, 200</td>
<td>4</td>
<td>33</td>
<td>4</td>
<td>128</td>
<td>7.6e-4</td>
<td>77</td>
<td>144</td>
<td>247</td>
<td>1245</td>
</tr>
<tr>
<td>500, 100, 1000</td>
<td>9</td>
<td>21</td>
<td>3</td>
<td>69</td>
<td>7.3e-4</td>
<td>67</td>
<td>358</td>
<td>1227</td>
<td>2504</td>
</tr>
<tr>
<td>500, 200, 500</td>
<td>11</td>
<td>81</td>
<td>5</td>
<td>384</td>
<td>9.1e-4</td>
<td>643</td>
<td>--</td>
<td>1555</td>
<td>--</td>
</tr>
<tr>
<td>500, 200, 2000</td>
<td>9</td>
<td>30</td>
<td>3</td>
<td>94</td>
<td>8.6e-4</td>
<td>398</td>
<td>--</td>
<td>1791</td>
<td>--</td>
</tr>
<tr>
<td>500, 300, 1000</td>
<td>1</td>
<td>44</td>
<td>7</td>
<td>221</td>
<td>4.7</td>
<td>646</td>
<td>--</td>
<td>2558</td>
<td>--</td>
</tr>
<tr>
<td>800, 10, 800</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>4.2e-4</td>
<td>7</td>
<td>159</td>
<td>553</td>
<td>2798</td>
</tr>
<tr>
<td>800, 50, 500</td>
<td>1</td>
<td>22</td>
<td>3</td>
<td>58</td>
<td>9.1e-4</td>
<td>77</td>
<td>433</td>
<td>1453</td>
<td>3449</td>
</tr>
<tr>
<td>800, 100, 800</td>
<td>2</td>
<td>32</td>
<td>4</td>
<td>114</td>
<td>9.4e-4</td>
<td>313</td>
<td>--</td>
<td>2811</td>
<td>--</td>
</tr>
<tr>
<td>800, 200, 1000</td>
<td>1</td>
<td>28</td>
<td>4</td>
<td>107</td>
<td>3.4e-3</td>
<td>467</td>
<td>--</td>
<td>5814</td>
<td>--</td>
</tr>
<tr>
<td>800, 200, 1500</td>
<td>1</td>
<td>24</td>
<td>5</td>
<td>98</td>
<td>4.5e-3</td>
<td>424</td>
<td>--</td>
<td>6332</td>
<td>--</td>
</tr>
</tbody>
</table>
The additional linear term

Most implementations include an additional linear term, $\alpha W$.

Use $X = PVP^T + \alpha W$, with $\alpha \geq 0$ a variable.

Used to aggregate columns of $P$:
Instead of dropping a column $p$, add multiple of $pp^T$ to $W$.

In addition to saving old information, this also makes it easier to restart.
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Theoretical results for the convex feasibility problem

Given a convex set \( Y \subseteq \mathbb{R}^m \) containing a ball of radius \( \varepsilon \), find a point \( y \in Y \) or determine that \( Y \) is empty.
Find approximate analytic center $\bar{y}$
Add a conic cut at \( \bar{y} \)
Update outer approximation

... and repeat
Conic relaxation

The feasibility problem is approximated by a conic program:

\[
\begin{align*}
\text{max} & \quad 0 \\
\text{subject to} & \quad A^* y + s = c \\
& \quad s \in K
\end{align*}
\]

where \( K \) is a full-dimensional self-scaled cone.

Note that \( K \) may be a product of smaller cones.

Possible cones include \( \mathbb{R}_+^n \), SDP cone, SOCP cone.

Potential functions:
- \( \mathbb{R}_+^n \) has \( f^*(s) = - \sum_{i=1}^n \ln s_i \)
- \( S \succeq 0 \) has \( f^*(S) = - \ln \det(S) \).

Analytic center minimizes the potential function.
Using the Dikin Ellipsoid

Find restart direction $d$ when adding conic constraint

Minimize the potential function of the new slack variables, while staying within the Dikin ellipsoid.
Convergence

- **Convergence to new analytic center:**
  Can solve an NLP in the new variables in order to find a *restart direction*. Number of steps to get convergence to new approximate analytic center is *linear in the “size” of the added constraint*.

- **Global convergence for feasibility problem:**
  Polynomial in the dimension $m$,
  the required *tolerance* $\epsilon$,
  and a *condition number* based on the added constraints.

Proof uses upper and lower bounds on the dual potential function.

(Oskoorouchi and Goffin for SDP and SOCP, Basescu and Mitchell for more general problems)
**Convergence**

- **Convergence to new analytic center:**
  Can solve an NLP in the new variables in order to find a restart direction. Number of steps to get convergence to new approximate analytic center is linear in the “size” of the added constraint.

- **Global convergence for feasibility problem:**
  Polynomial in the dimension $m$,
  the required tolerance $\epsilon$,
  and a condition number based on the added constraints.

Proof uses upper and lower bounds on the dual potential function.

(Oskoorouchi and Goffin for SDP and SOCP, Basescu and Mitchell for more general problems)
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - **Condition number**

3. Smoothing techniques and subgradient methods

4. Conclusions
The need for a condition number

- Dual potential function $f^*(c - A^*y)$ is equal to sum of dual potential functions over each of the added constraints.

- Upper bound obtained from assumption of an $\epsilon$-ball: If $c - A^*(y + \epsilon u) \succeq_K 0$ for any unit vector $u$ then $f^*_K(c - A^*y) \leq f^*_K(\epsilon A^*u) = f^*_K(A^*u) - \varphi f_K \ln \epsilon$.

- Want $f^*_K(A^*u)$ small in order to get a good upper bound.

- Define condition number $\mu_K$ so that $\ln \mu_K := \inf \{ f^*_K(A^*u) : A^*u \succeq_K 0, \|u\| = 1 \}$. 
The need for a condition number

- **Dual potential function** $f^*(c - A^*y)$ is equal to sum of dual potential functions over each of the added constraints.

- **Upper bound** obtained from assumption of an $\epsilon$-ball:
  If $c - A^*(y + \epsilon u) \succeq_K 0$ for any unit vector $u$ then
  $$f_K^*(c - A^*y) \leq f_K^*(\epsilon A^*u) = f_K^*(A^*u) - \vartheta f_K \ln \epsilon.$$

- Want $f_K^*(A^*u)$ small in order to get a good upper bound.

- Define **condition number** $\mu_K$ so that
  $$\ln \mu_K := \inf \{ f_K^*(A^*u) : A^*u \succeq_K 0, \|u\| = 1 \}.$$
The need for a condition number

- Dual potential function $f^*(c - A^*y)$ is equal to sum of dual potential functions over each of the added constraints.

- Upper bound obtained from assumption of an $\epsilon$-ball:
  If $c - A^*(y + \epsilon u) \succeq_K 0$ for any unit vector $u$ then
  $$f_K^*(c - A^*y) \leq f_K^*(\epsilon A^*u) = f_K^*(A^*u) - \vartheta f_K \ln \epsilon.$$

- Want $f_K^*(A^*u)$ small in order to get a good upper bound.

- Define condition number $\mu_K$ so that
  $$\ln \mu_K := \inf\{f_K^*(A^*u) : A^*u \succeq_K 0, \|u\| = 1\}. $$
The need for a condition number

- **Dual potential function** $f^*(c - A^*y)$ is equal to sum of dual potential functions over each of the added constraints.

- **Upper bound** obtained from assumption of an $\epsilon$-ball:
  If $c - A^*(y + \epsilon u) \succeq_K 0$ for any unit vector $u$ then
  $$f^*_K(c - A^*y) \leq f^*_K(\epsilon A^*u) = f^*_K(A^*u) - \vartheta f_K \ln \epsilon.$$  

- Want $f^*_K(A^*u)$ small in order to get a good upper bound.

- Define **condition number** $\mu_K$ so that
  $$\ln \mu_K := \inf\{ f^*_K(A^*u) : A^*u \succeq_K 0, \|u\| = 1 \}. $$
An SDP constraint with a bad condition number

Look at \( \{ d \in \mathbb{R}^4 : A^*(d) := \sum_{i=1}^{4} A_i d_i \succeq 0 \} \) for

\[
A_1 = \begin{bmatrix} 1 & 0 \\ 0 & \delta \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & \delta \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}
\]

If \( ||d|| = 1 \) then \( \det(A^*(d')) \) is at most \( O(\delta) \).

Thus, for any choice of direction, need to move quite far to get a reasonable potential function value.

Can remove need for the condition number by selective orthonormalization, which unfortunately weakens the constraints.
An SDP constraint with a bad condition number

- Look at \( \{ d \in \mathbb{R}^4 : A^*(d) := \sum_{i=1}^{4} A_i d_i \succeq 0 \} \) for
  - \( A_1 = \begin{bmatrix} 1 & 0 \\ 0 & \delta \end{bmatrix} \)
  - \( A_2 = \begin{bmatrix} 0 & 1 \\ 1 & \delta \end{bmatrix} \)
  - \( A_3 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \)
  - \( A_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \)

- If \( ||d|| = 1 \) then \( \det(A^*(d')) \) is at most \( O(\delta) \).

Thus, for any choice of direction, need to move quite far to get a reasonable potential function value.

Can remove need for the condition number by selective orthonormalization, which unfortunately weakens the constraints.
An SDP constraint with a bad condition number

Look at \( \{ d \in \mathbb{R}^4 : A^*(d) := \sum_{i=1}^4 A_i d_i \succeq 0 \} \) for

\[
A_1 = \begin{bmatrix} 1 & 0 \\ 0 & \delta \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & \delta \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}
\]

If \( \|d\| = 1 \) then \( \det(A^*(d')) \) is at most \( O(\delta) \).

Thus, for any choice of direction, need to move quite far to get a reasonable potential function value.

Can remove need for the condition number by selective orthonormalization, which unfortunately weakens the constraints.
An SDP constraint with a bad condition number

Look at \( \{ d \in \mathbb{R}^4 : A^*(d) := \sum_{i=1}^{4} A_i d_i \succeq 0 \} \) for

\[
A_1 = \begin{bmatrix} 1 & 0 \\ 0 & \delta \end{bmatrix} \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & \delta \end{bmatrix} \quad A_3 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad A_4 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}
\]

If \( ||d|| = 1 \) then \( \det(A^*(d)) \) is at most \( O(\delta) \).

Thus, for any choice of direction, need to move quite far to get a reasonable potential function value.

Can remove need for the condition number by selective orthonormalization, which unfortunately weakens the constraints.
Outline

1. Interior Point Cutting Plane and Column Generation Methods
   - Introduction
   - MaxCut
   - Interior point cutting plane methods
   - Warm starting
   - Theoretical results
   - Stabilization

2. Cutting Surfaces for Semidefinite Programming
   - Semidefinite Programming
   - Relaxations of dual SDP
   - Computational experience
   - Theoretical results with conic cuts
   - Condition number

3. Smoothing techniques and subgradient methods

4. Conclusions
Subgradient methods

Subgradient methods used to solve $\min_x \{ f(x) : x \in C \}$

$C$: bounded closed convex set
$f(x)$ is a continuous convex function on $C$.

Given feasible $\bar{x}$, find subgradient $\xi$ of $f(.)$ at $\bar{x}$.
Update $x \leftarrow \bar{x} - \tau \xi$ for steplength $\tau$.

Converges under certain conditions on $\tau$.

Work required at each iteration:
(i) find $f(\bar{x})$ and $\xi$ (eg, using an oracle)
(ii) update the iterate.

Advantage: low cost of (ii).
Disadvantage: the number of iterations may be large.
Nesterov’s smoothing technique

Complexity to get within $\varepsilon$ of optimality:
$O(1/\varepsilon^2)$ for best possible generic subgradient scheme.

If $f(x)$ smooth and $C$ satisfies certain conditions:
Complexity is $O(1/\sqrt{\varepsilon})$.

So, construct smooth approximation to $f(x)$. Complexity: $O(1/\varepsilon)$.

Complexity depends on Lipschitz constant of gradient of approximation.

Used on certain Lagrangian dual problems, variational inequalities, semidefinite programming, conic programming, Nash equilibria computation . . .
Smoothing techniques and subgradient methods

**Smoothing technique for SDPs**

Consider the SDP:

\[
\min_y f(y) := \lambda_{\text{max}}(C + A^T y)
\]

Smoothed function is

\[
f_\mu(y) = \mu \ln\left(\sum_{i=1}^{n} e^{\lambda_i(C+ A^T y)/\mu}\right)
\]

where \(\lambda_i(M)\) is \(i\)th eval of \(M\)

More accurate for smaller values of \(\mu\).

Calculating gradient requires calculating matrix exponential.


Computationally: works well for approximate solution to large SDPs.
Outline

1 Interior Point Cutting Plane and Column Generation Methods
   • Introduction
   • MaxCut
   • Interior point cutting plane methods
   • Warm starting
   • Theoretical results
   • Stabilization

2 Cutting Surfaces for Semidefinite Programming
   • Semidefinite Programming
   • Relaxations of dual SDP
   • Computational experience
   • Theoretical results with conic cuts
   • Condition number

3 Smoothing techniques and subgradient methods

4 Conclusions
Conclusions for Cutting Plane Methods

- Interior point methods provide an excellent choice for stabilizing a column generation approach.
- The strength of interior point methods for linear programming means that these column generation approaches scale well, with theoretical polynomial or fully polynomial convergence depending on the variant.
- Combining interior point and simplex column generation methods has proven especially effective, with the interior point methods used early on and the simplex method used later. The development of methods that automatically combine the two linear programming approaches would be useful.
- Theoretical development of a more intuitive polynomial interior point cutting plane algorithm is desirable.
Conclusions for Cutting Plane Methods

- Interior point methods provide an excellent choice for stabilizing a column generation approach.
- The strength of interior point methods for linear programming means that these column generation approaches scale well, with theoretical polynomial or fully polynomial convergence depending on the variant.
- Combining interior point and simplex column generation methods has proven especially effective, with the interior point methods used early on and the simplex method used later. The development of methods that automatically combine the two linear programming approaches would be useful.
- Theoretical development of a more intuitive polynomial interior point cutting plane algorithm is desirable.
Conclusions for Cutting Plane Methods

- Interior point methods provide an excellent choice for stabilizing a column generation approach.
- The strength of interior point methods for linear programming means that these column generation approaches scale well, with theoretical polynomial or fully polynomial convergence depending on the variant.
- Combining interior point and simplex column generation methods has proven especially effective, with the interior point methods used early on and the simplex method used later. The development of methods that automatically combine the two linear programming approaches would be useful.
- Theoretical development of a more intuitive polynomial interior point cutting plane algorithm is desirable.
Conclusions for Cutting Plane Methods

- Interior point methods provide an excellent choice for stabilizing a column generation approach.
- The strength of interior point methods for linear programming means that these column generation approaches scale well, with theoretical polynomial or fully polynomial convergence depending on the variant.
- Combining interior point and simplex column generation methods has proven especially effective, with the interior point methods used early on and the simplex method used later. The development of methods that automatically combine the two linear programming approaches would be useful.
- Theoretical development of a more intuitive polynomial interior point cutting plane algorithm is desirable.
Cutting Surface Conclusions

- Research in cutting surface and subgradient methods for quickly finding good solutions to semidefinite programming problems is an active area.
- The links between the cutting surface approaches and the smoothed subgradient methods can be developed further.
- Complexity results for cutting surface methods have been developed, but can probably be improved further. Eg, remove the condition number without weakening the constraints.
- These methods have been developed for conic problems where the cones are self-dual, and it would be interesting to extend the methods and results to more general classes of conic optimization problems.
Cutting Surface Conclusions

- Research in cutting surface and subgradient methods for quickly finding good solutions to semidefinite programming problems is an active area.

- The links between the cutting surface approaches and the smoothed subgradient methods can be developed further.

- Complexity results for cutting surface methods have been developed, but can probably be improved further. Eg, remove the condition number without weakening the constraints.

- These methods have been developed for conic problems where the cones are self-dual, and it would be interesting to extend the methods and results to more general classes of conic optimization problems.
Cutting Surface Conclusions

- Research in cutting surface and subgradient methods for quickly finding good solutions to semidefinite programming problems is an active area.
- The links between the cutting surface approaches and the smoothed subgradient methods can be developed further.
- Complexity results for cutting surface methods have been developed, but can probably be improved further. Eg, remove the condition number without weakening the constraints.
- These methods have been developed for conic problems where the cones are self-dual, and it would be interesting to extend the methods and results to more general classes of conic optimization problems.
Cutting Surface Conclusions

- Research in cutting surface and subgradient methods for quickly finding good solutions to semidefinite programming problems is an active area.

- The links between the cutting surface approaches and the smoothed subgradient methods can be developed further.

- Complexity results for cutting surface methods have been developed, but can probably be improved further. Eg, remove the condition number without weakening the constraints.

- These methods have been developed for conic problems where the cones are self-dual, and it would be interesting to extend the methods and results to more general classes of conic optimization problems.