A Combinatorial, Primal-Dual Approach to Semidefinite Programs

Satyen Kale
Microsoft Research

Joint work with
Sanjeev Arora
Princeton University
The Ubiquity of Semidefinite Programming

Max Cut [GW’95]
Balanced Partitioning [ARV’04]
Graph Coloring [KMS’98, ACC’06]

\[
\begin{align*}
\frac{dx(t)}{dt} &= Ax(t) \\
\text{Control Theory} \\
\text{SDP} \\
(a \land \neg b) &\land (\neg a \land \neg c) &\land (a \lor b) &\land (\neg c \lor c) &\land b
\end{align*}
\]

Constraint Satisfaction [ACMM’05]
Algorithms for SDP

Ellipsoid method
- $O(n^8)$ iterations

Interior point methods
- $O(\sqrt{m(n^3 + m)})$ time

Lagrangian Relaxation
- Reduction to eigenvectors
- $\text{poly}(1/\varepsilon)$ dependence on $\varepsilon$, limits applicability (e.g. Sparsest Cut)

Combinatorial, Primal-Dual algorithms
Primal-Dual algorithms for LP:

**Multicommodity Flows**

Objective: Maximize total flow, while respecting edge capacities
Primal-Dual algorithms for LP: Multicommodity Flows

- Edge weights $w_e = 1$
- Repeat until some $e$ reaches capacity:
  - Find shortest path $p$
  - Route $\varepsilon^2 / \log(m)$ flow on $p$
  - Update $w_e$ for all $e \in p$ as
    \[ w_e \leftarrow w_e \cdot (1 + \varepsilon) \]
- Output flow.

Thm [GK’98]: Stops in $\tilde{O}(m)$ rounds with $(1 - 2\varepsilon) \cdot \text{OPT}$ flow. Total running time is $\tilde{O}(m^2)$. 
Primal-Dual algorithms for LP: Multicommodity Flows

1. Primal-Dual algorithm

2. Combinatorial

3. Multiplicative Weights Update Rule

- Edge weights $w_e = 1$
- Repeat until some $e$ reaches capacity:
  - Find shortest path $p$
  - Route $\varepsilon^2 / \log(m)$ flow on $p$
  - Update $w_e$ for all $e \in p$ as $w_e \leftarrow w_e \cdot (1 + \varepsilon)$
- Output flow.

Thm [GK'98]: Stops in $\tilde{O}(m)$ rounds with $(1 - 2\varepsilon) \cdot \text{OPT}$ flow. Total running time is $\tilde{O}(m^2)$. 
Starting point for our work:

Analogous primal-dual algorithms for SDP?

Difficulties:

- Positive semidefiniteness hard to maintain
- Rounding algorithms exploit geometric structure (e.g. negative-type metrics)
- Matrix operations inefficient to implement

Our work: a generic scheme that yields fast, combinatorial, primal-dual algorithms for various optimization problems using SDP
### Our Results: Primal-Dual algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>Previous best: ( O(\sqrt{\log n}) ) apx</th>
<th>Our alg.: ( O(\sqrt{\log n}) ) apx</th>
<th>Our alg.: ( O(\log n) ) apx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undirected Sparsest Cut</td>
<td>( \tilde{O}(n^2) ) [AHK’04]</td>
<td>( \tilde{O}(n^2) )</td>
<td>( \tilde{O}(m + n^{1.5}) )</td>
</tr>
<tr>
<td>Undirected Balanced Sep.</td>
<td>( \tilde{O}(n^2) ) [AHK’04]</td>
<td>( \tilde{O}(n^2) )</td>
<td>( \tilde{O}(m + n^{1.5}) )</td>
</tr>
<tr>
<td>Directed Sparsest Cut</td>
<td>( \tilde{O}(n^{4.5}) )</td>
<td>( \tilde{O}(m^{1.5} + n^{2+\epsilon}) )</td>
<td>( \tilde{O}(m^{1.5}) )</td>
</tr>
<tr>
<td>Directed Balanced Sep.</td>
<td>( \tilde{O}(n^{4.5}) )</td>
<td>( \tilde{O}(m^{1.5} + n^{2+\epsilon}) )</td>
<td>( \tilde{O}(m^{1.5}) )</td>
</tr>
<tr>
<td>Min UnCut</td>
<td>( \tilde{O}(n^{4.5}) )</td>
<td>( \tilde{O}(n^3) )</td>
<td>---</td>
</tr>
<tr>
<td>Min 2CNF Deletion</td>
<td>( \tilde{O}(n^{4.5}) )</td>
<td>( \tilde{O}(nm^{1.5} + n^3) )</td>
<td>---</td>
</tr>
</tbody>
</table>
## Our Results: Primal-Dual algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>Previous best:</th>
<th>Our alg.: $O(\sqrt{\log n})$ apx</th>
<th>Our alg.: $O(\log n)$ apx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undirected Sparsest Cut</td>
<td>$\tilde{O}(n^2)$ [AHK'04]</td>
<td>$\tilde{O}(n^2)$</td>
<td>$\tilde{O}(m + n^{1.5})$</td>
</tr>
<tr>
<td>Undirected Balanced Sep.</td>
<td>$\tilde{O}(n^2)$</td>
<td>$\tilde{O}(m + n^{1.5})$</td>
<td></td>
</tr>
<tr>
<td>Directed Sparsest Cut</td>
<td>$\tilde{O}(n^{4.5})$</td>
<td>$\tilde{O}(m^{1.5})$</td>
<td>$\tilde{O}(m^{1.5})$</td>
</tr>
<tr>
<td>Directed Balanced Sep.</td>
<td>$\tilde{O}(n^{4.5})$</td>
<td>$\tilde{O}(m^{1.5} + n^{2+\varepsilon})$</td>
<td>$\tilde{O}(m^{1.5})$</td>
</tr>
<tr>
<td>Min Uncut</td>
<td>$\tilde{O}(n^{4.5})$</td>
<td>$\tilde{O}(n^3)$</td>
<td>---</td>
</tr>
<tr>
<td>Min 2CNF Deletion</td>
<td>$\tilde{O}(n^{4.5})$</td>
<td>$\tilde{O}(nm^{1.5} + n^3)$</td>
<td>---</td>
</tr>
</tbody>
</table>
Our Results: Near-linear time algorithm for Max Cut

- $\tilde{O}(n + m)$ time algorithm to approximate Max Cut SDP to $(1 + o(1))$ factor

- Previous best: $\tilde{O}(n^2)$ time algorithm by Klein, Lu '95
Prototype MW for LPs: Winnow [L’88]

\[ a_1 \cdot x \geq -\delta \]
\[ a_2 \cdot x \geq -\delta \]
\[ \vdots \]
\[ a_m \cdot x \geq -\delta \]
\[ x \geq 0 \]
\[ \sum_i x_i = 1 \]

Thm: Finds \( x \) in \( O(\rho^2 \log(n)/\delta^2) \) iterations.
\[ \rho = \max_{ij} |a_{ij}| \]

Oracle
Find \( i \) s.t. \( a_i \cdot x < -\delta \)

Convex comb. of constraints allowed:
\[ \sum_i y_i a_i \cdot x < -\delta \], where \( y_i \geq 0, \sum_i y_i = 1 \).
Hence Primal-Dual.

MW algorithm: boosting, hard-core sets, zero-sum games, flows, portfolio, ...

MW for LP:
Init: \( x = (1/n, \ldots, 1/n) \)
Update:
\[ x_j = x_j \cdot \exp(-\varepsilon a_{ij})/\Phi \]
\[ \Phi = \text{norm. factor} \]
Prototype Matrix MW for SDPs

\[ A_1 \cdot X \geq -\delta \]
\[ A_2 \cdot X \geq -\delta \]
\[ \vdots \]
\[ A_m \cdot X \geq -\delta \]
\[ X \succeq 0 \]
\[ \text{Tr}(X) = 1 \]

Thm: Finds \( X \) in \( O(\rho^2 \log(n)/\delta^2) \) iterations.

\[ \rho = \max_i \|A_i\| \]

Matrix MW for SDP:
Init: \( X = I/n \)
Update: \( X = \exp(-\varepsilon \sum_{s=1}^t A_i)/\Phi \)
\[ \Phi = \text{norm. factor} \]

Oracle Find \( i_t \) s.t. \( A_{i_t} \cdot X < -\delta \)

Convex comb. of constraints allowed:
\[ \sum_i y_i A_i \cdot X < -\delta, \text{ where } y_i \geq 0, \sum_i y_i = 1. \]
Hence Primal-Dual.

Analysis uses \( \Phi \) as potential fn.
The Matrix Exponential

Matrix MW for SDP:
Init: $X = I/n$
Update: $X = \exp\left(-\varepsilon \sum_{s=1}^{t} A_{i_s}\right) / \Phi$
$\Phi = \text{norm. factor}$

- **Matrix exponential:**
  \[ \exp(A) = I + A + A^2/2! + A^3/3! + \ldots \]

- **Always PSD:** $\exp(A) \succeq 0$

- **Golden-Thompson inequality:**
  \[ \text{Tr}(\exp(A+B)) \leq \text{Tr}(\exp(A)\exp(B)) \]

- **Computation:**
  - $O(n^3)$ time
  - Usually, can use J-L lemma
  - Only need $\exp(A)v$ products: $\tilde{O}(m)$ time
Approximation via SDP Relaxations

Input

0-1 Quadratic Program

SDP Solver

Primal-Dual SDP

Rounding Algorithm

SDP Opt
Primal-Dual SDP Framework

Input

Reduction

0-1 Quadratic Program

Relaxation

SDP: Vector vars

Rounding Algorithm

Primal-Dual SDP

Matrix MW

v₁

v₂

vₙ

y₁, …, yₘ

X

v₁

v₂

vₙ
Approximating Balanced Separator

- Cut \((S, S')\) is \(c\)-balanced if \(|S|, |S'| \geq cn\)
- **Min** \(c\)-Balanced Separator: \(c\)-balanced cut of min capacity
- Numerous applications:
  - Divide-and-conquer algorithms
  - Markov chains
  - Geometric embeddings
  - Clustering
  - Layout problems
  - ...

\(G = (V, E)\)
SDP for Balanced Separator

\[ \frac{1}{4} \|v_i - v_j\|^2 = 0 \text{ if } i, j \text{ on same side,} \]
\[ = 1 \text{ otherwise} \]

**SDP:**
\[
\min \sum_{i,j \in E} \frac{1}{4} \|v_i - v_j\|^2
\]
\[
\forall i: \|v_i\|^2 = 1
\]
\[
\forall i,j,k: \|v_i - v_j\|^2 + \|v_j - v_k\|^2 \geq \|v_i - v_k\|^2
\]
\[
\sum_{i,j} \frac{1}{4} \|v_i - v_j\|^2 \geq c(1-c)n^2
\]
Implementing Oracle

Primal:
\[
\min \sum_{i,j \in E} \frac{1}{4} \|v_i - v_j\|^2 \\
\forall i: \|v_i\|^2 = 1 \\
\forall ijk: \|v_i - v_j\|^2 + \|v_j - v_k\|^2 \geq \|v_i - v_k\|^2 \\
\checkmark \sum_{i,j} \frac{1}{4} \|v_i - v_j\|^2 \geq c(1-c)n^2
\]

Thm: Given \(v_i, \alpha\):
1. Max-flow \(\Rightarrow\) desired flow or cut of value \(O(\log(n) \cdot \alpha)\).
2. Multicommodity flow \(\Rightarrow\) desired flow or cut of value \(O(\sqrt{\log(n)} \cdot \alpha)\).
Conclusions and Future Work

- **Matrix MW algorithm:** more applications in
  - Solving SDPs: e.g. Min Linear Arrangement
  - Quantum algorithms: density matrix is a central concept
  - Learning: e.g. online variance minimization [WK COLT’06], online PCA [WK NIPS’06]
  - Other applications?

- **Linear time algorithm for Sparsest Cut?**
  - Our algorithm runs in $\tilde{O}(m + n^{1.5})$ time for an $O(\log n)$ approximation
Thank you!