The Interplay between Sparsity and Big Data in Systems Theory

M. Sznaier
Robust Systems Lab
ECE, Northeastern University
Motivation 1: SysId

Goal: Find a low order, stable model
Motivation 2: distributed sensing & control

Goal: impose a sparse structure
Motivation 3: decision making

How do we make (provably) correct decisions in a “data deluged” environments? (a hidden hybrid SysId problem)
Hard or Easy?

- Claim 1: These problems are (NP!) hard
Hard or Easy?

- Claim 1: These problems are (NP!) hard
- Claim 2: These problems can be solved in polynomial time
Hard or Easy?

- Claim 1: These problems are (NP!) hard
- Claim 2: These problems can be solved in polynomial time

Both can’t be right, can they?
Hard or Easy?

- Claim 1: These problems are **generically** NP-hard

- Claim 2: Many of these problems can be solved in polynomial time
Hard or Easy?

- **Q:** What makes a problem easy?
- **A:** Convexity?
Hard or Easy?

- Q: What makes a problem easy?
- A: Convexity? **Not Necessarily!**

Optimization over co-Positive matrices is NP-hard
Hard or Easy?

Q: What makes a problem easy?

A: Convexity + Self-Concordance?
Q: What makes a problem easy?

A: Convexity + Self-Concordance? Not Necessarily!

<table>
<thead>
<tr>
<th>Horizon</th>
<th>ADMM (secs)</th>
<th>SDP solver (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>280</td>
<td>1071.8</td>
<td>4177.0</td>
</tr>
<tr>
<td>350</td>
<td>1828.0</td>
<td>12686.9</td>
</tr>
<tr>
<td>420</td>
<td>2657.7</td>
<td>out of memory</td>
</tr>
</tbody>
</table>

In (convex) SysId Big Data may be as low as $10^2$
Hard or Easy?

- Q: What makes a problem hard?
- A: Lack of Convexity?
Hard or Easy?

- Q: What makes a problem hard?
- A: Lack of Convexity? Not Necessarily!

\[
\min \sum c_i x_i x_{i+1} \text{ subject to } x_i = \pm 1
\]

Non-convex but solving for 100000 variables takes 50 secs on a Mac
Hard or Easy?

Q: What makes a problem hard/easy?

A: Structure
   - Self Similarity
   - Sparsity

Both observed in many practical problems
   - Often they induces “good” convexity
   - Exploited in Machine Learning for “static” problems
Hard or Easy?

- **Challenge**
  - Separate easy/hard problems
  - Understand where does the complexity come from
  - Use this understanding to design “easy” problems

Main point of this talk: These issues are related to the sparsity structure of the problem
Intuition: look at QCQP

\[ p^* = \min_x x'Q_0x \text{ s.t. } x'Q_ix \leq 0 \ i = 1, \ldots n \]

\[ p^* = \min_x \text{Trace}(Q_0xx') \text{ s.t. } \text{Trace}(Q_ixx') \leq 0 \ i = 1, \ldots n \]

\[ p_{SDP} = \min_x \text{Trace}(Q_0X) \text{ s.t. } \text{Trace}(Q_iX) \leq 0, \ X \succeq 0 \]

Clearly \( p_{SDP} \leq p^* \) and \( p_{SDP} = p^* \) if \( \text{rank}(X)=1 \)
Intuition: look at QCQP

\[ p^* = \min_{x} x'Q_0x \text{ s.t. } x'Q_ix \leq 0 \ i = 1,..n \]

\[ p^* = \min_{x} \text{Trace}(Q_0xx') \text{ s.t. } \text{Trace}(Q_ixx') \leq 0 \ i = 1,..n \]

\[ p_{SDP} = \min_{x} \text{Trace}(Q_0X) \text{ s.t. } \text{Trace}(Q_iX) \leq 0, \ X \succeq 0 \]

Clearly \( p_{SDP} \leq p^* \) \text{ and } \( p_{SDP} = p^* \) \text{ if rank}(X)=1

Q: Can we get this for (almost) free?
Exploiting sparsity in QCQP

- Complexity related to the topology of a graph:
  - Each vertex corresponds to a variable
  - There is an edge (i,j) if there are terms involving $x_i x_j$
Exploiting sparsity in QCQP

- If the graph is a tree, then the SDP relaxation is exact

J. Lavaei, 2014
Exploiting sparsity in QCQP

- If the graph is a tree, then the SOCP relaxation is exact

- Example: \( \min \sum c_i x_i x_{i+1} \) subject to \( x_i = \pm 1 \)

Solving for 100,000 variables takes 50 secs on a Mac

Structure and Sparsity Matter
Many problems have a sparse structure (running intersection)

\[
\min_x p_1(x) + p_2(x) + \ldots + p_m(x) \quad s.t.
\]

\[
f_1(x^\alpha) \leq 0
\]

\[
f_2(x^\alpha) \leq 0
\]

\[
\vdots
\]

\[
f_m(x^\alpha) \leq 0
\]

where each \( p_i(.) \), \( f_i(.) \) depends only on a subset of variables such that

\( P_1 \)

\( f_1 \)

\( x_1, x_2, \ldots, x_k, x_{d}, x_{d+1}, \ldots, x_{d+k}, \ldots, x_{n-d+1}, x_n \)
Sparse polynomial optimization

- Many problems have a sparse structure (running intersection)

\[
\min_{x} p_1(x) + p_2(x) + \ldots + p_m(x) \quad s.t.
\]

\[
f_1(x^\alpha) \leq 0 \\
f_2(x^\alpha) \leq 0 \\
\vdots \\
f_m(x^\alpha) \leq 0
\]

where each \( p_i(.) \), \( f_i(.) \) depends only on a subset of variables such that

\[
P_1 \\
f_1
\]

\[
x_1, x_2, \ldots, x_k, x_{d+1}, \ldots, x_{d+k}, \ldots, x_{n-d+1}, x_n
\]
Sparse polynomial optimization

- Many problems have a sparse structure (running intersection)

\[
\min_x p_1(x) + p_2(x) + \ldots + p_m(x) \quad \text{s.t.} \\
\begin{align*}
  f_1(x^{\alpha}) &\leq 0 \\
  f_2(x^{\alpha}) &\leq 0 \\
  \vdots \\
  f_m(x^{\alpha}) &\leq 0
\end{align*}
\]

where each \( p_i(.) \), \( f_i(.) \) depends only on a subset of variables such that

\[
P_1 \quad P_2 \\
f_1 \quad f_2 \\
\begin{array}{ccccccc}
  x_1, x_2, \ldots, x_k, x_d, x_{d+1}, \ldots, x_{d+k}, \ldots, x_{n-d+1}, x_n
\end{array}
\]
Sparse polynomial optimization

- Running intersection is related to cliques in the \textit{(chordal completion of the) csp graph}
Sparse polynomial optimization

- Running intersection is related to cliques in the (chordal completion of the) csp graph

Size of the running intersection is given by the tree width
Sparse polynomial optimization

- Running intersection is related to cliques in the (chordal completion of the) CSP graph

Complexity dominated by the size of the clique, not the size of the problem
Connecting Information, Sparsity & Dynamics
Where should we pay attention?:

Features (edges, regions, etc.) are important.
Where should we pay attention?:

*Dynamics* are important too!
Sparse signal recovery:

- **Strong prior:**
  - Signal has a sparse representation
    \[ f = \sum c_i \psi_i \]
    only a few \( c_i \neq 0 \)

- **Signal Recovery:**
  - “sparsify” the coefficients
    \[
    \min \| [c_1, \ldots, c_n] \|_o \\
    \text{subject to: } f(x_i) = y_i
    \]
**Sparse signal recovery:**

- **Strong prior:**
  - Signal has a sparse representation
  \[ f = \sum c_i \psi_i \]
  only a few \( c_i \neq 0 \)

- **Signal Recovery:**
  - “sparsify” the coefficients
  \[ \min_{[c_1, \ldots, c_n]} \| c \|_o \]
  subject to: \( f(x_i) = y_i \)

**Sparse information extraction**

- **Strong prior:**
  - Actionable information is generated by low complexity dynamical systems.

- **Information extraction:**
  - “sparsify” the dynamics
  \[ \min_{y} \{ \text{rank}[M(y)] + \lambda \| E(y) \|_o \} \]
  - Where \( M(\cdot), E(\cdot) \) are affine in \( y \)
Example: Solving “Temporal Puzzles”
Example: Solving “Temporal Puzzles”

\[
\min_{c} \|c\|_1 \text{ subject to:} \\
v = Dc \\
\|Py - v\| \leq \epsilon \\
P \in \mathcal{P}
\]

\(D\) is a suitably chosen dynamic dictionary.
Example: Solving “Temporal Puzzles”

Dynamic sparsification
Information Extraction as an ID problem
- Model data streams as outputs of switched systems
- “Interesting” events ⇔ Model invariant(s) changes
- An identification/model (in)validation problem.
SARX Id problem:

- **Given:**
  - Bounds on noise ($||\eta||_\infty \leq \varepsilon$), sub-system order ($n_0$)
  - Input/output data ($u, y$)
  - Number of sub-models

- **Find:**
  - A piecewise affine model such that:

\[
y_t = \sum_{i=1}^{n_a} a_i(\sigma_t)y_{t-i} + \sum_{i=1}^{n_c} c_i(\sigma_t)u_{t-i} + f(\sigma_t) + \eta_t
\]

\[
0 = b(\sigma_t)^T r_t + \eta_t
\]
- Given $N$ points in $\mathbb{R}^n$, fit them to hyperplanes

- "Chicken and egg" problem
  - Do not known the point "labels"
  - Do not know the hyperplanes. 

$NP$ Hard!
Reformulation:

\[ y_t + \eta_t - \sum_{i=1}^{n_a} A_i(\sigma_1)y_{t-i} - \sum_{i=1}^{n_c} C_i(\sigma_1)u_{t-i} = 0 \]

or

\[ y_t + \eta_t - \sum_{i=1}^{n_a} A_i(\sigma_2)y_{t-i} - \sum_{i=1}^{n_c} C_i(\sigma_2)u_{t-i} = 0 \]

A hidden QCQP problem
QCQP reformulation:

\[ s_{1,t}(y_t + \eta_t - \sum_{i=1}^{n_a} A_i(\sigma_1) y_{t-i} - \sum_{i=1}^{n_c} C_i(\sigma_1) u_{t-i}) = 0 \]

and

\[ s_{2,t}(y_t + \eta_t - \sum_{i=1}^{n_a} A_i(\sigma_2) y_{t-i} - \sum_{i=1}^{n_c} C_i(\sigma_2) u_{t-i}) = 0 \]

Subject to: \[ s_{i,t} = s_{i,t}^2, \text{ and } \sum_i s_{i,t} = 1 \]

\[ s \in \{0, 1\} \]
QCQP reformulation:

\[
\begin{align*}
|s_{i,j}r_i^T x_j| & \leq \epsilon s_{i,j}, \forall i=1 \forall j=1 \\
\sum_{i=1}^{N_s} \sum_{j=1}^{N_p} s_{i,j}^2 & = s_{i,j}, \forall i=1 \forall j=1 \\
\sum_{i=1}^{N_s} s_{i,j} & = 1, \forall j=1 \\
r_i^T r_i & = 1, \forall i=1 \\
r_1(1) & \geq r_2(1) \geq \cdots \geq r_{N_s}(1) \geq 0
\end{align*}
\]
QCQP reformulation:

\[
\begin{align*}
|s_{i,j} r_i^T x_j| &\leq \epsilon s_{i,j}, \forall_{i=1}^{N_s} \forall_{j=1}^{N_p} \\
2 s_{i,j}^2 &= s_{i,j}, \forall_{i=1}^{N_s} \forall_{j=1}^{N_p} \\
\sum_{i=1}^{N_s} s_{i,j} &= 1, \forall_{j=1}^{N_p} \\
r_i^T r_i &= 1, \forall_{i=1}^{N_s} \\
r_1(1) &\geq r_2(1) \geq \cdots \geq r_{N_s}(1) \geq 0
\end{align*}
\]

\(x_j\) is an inlier in \(S_i\) if \(s_{i,j} = 1\)
**QCQP reformulation:**

\[
\begin{align*}
|s_{i,j}r_i^T x_j| & \leq \epsilon s_{i,j}, \forall i=1 \forall j=1 \\

s_{i,j}^2 & = s_{i,j}, \forall i=1 \forall j=1 \\

\sum_{i=1}^{N_s} s_{i,j} & = 1, \forall j=1 \\

r_i^T r_i & = 1, \forall i=1 \\

r_1(1) & \geq r_2(1) \geq \cdots \geq r_{N_s}(1) \geq 0
\end{align*}
\]

\(x_j\) is an inlier in \(S_i\) if \(s_{i,j} = 1\)

\(s_{i,j} \in \{0, 1\}\)
QCQP reformulation:

\[
|s_{i,j} r_i^T x_j| \leq \varepsilon s_{i,j}, \forall i=1 \ldots N_s, \forall j=1 \ldots N_p
\]

\[
s_{i,j}^2 = s_{i,j}, \forall i=1 \ldots N_s, \forall j=1 \ldots N_p
\]

\[
\sum_{i=1}^{N_s} s_{i,j} = 1, \forall j=1 \ldots N_p
\]

\[
r_i^T r_i = 1, \forall i=1 \ldots N_s
\]

\[
x_j \text{ is an inlier in } S_i \text{ if } s_{i,j} = 1
\]

\[
s_{i,j} \in \{0, 1\}
\]

each sample is assigned to one subspace

\[
r_1(1) \geq r_2(1) \geq \cdots \geq r_{N_s}(1) \geq 0
\]
QCQP reformulation:

\[ |s_{i,j} r_i^T x_j| \leq \epsilon s_{i,j}, \forall i=1 \forall j=1 \]

\[ s_{i,j}^2 = s_{i,j}, \forall i=1 \forall j=1 \]

\[ \sum_{i=1}^{N_s} s_{i,j} = 1, \forall j=1 \]

\[ r_i^T r_i = 1, \forall i=1 \]

\[ r_1(1) \geq r_2(1) \geq \cdots \geq r_{N_s}(1) \geq 0 \]

\( x_j \) is an inlier in \( S_i \) if \( s_{i,j} = 1 \)

\( s_{i,j} \in \{0, 1\} \)

Each sample is assigned to one subspace

Solvable using SoS / Moments techniques
Hidden Sparse Structure:

Model parameters

\[ s_1, t(y_t + \eta_t) - \sum_{i=1}^{n_a} A_i(\sigma_1)y_{t-i} - \sum_{i=1}^{n_c} C_i(\sigma_1)u_{t-i} = 0 \]
Hidden Sparse Structure:

Complexity determined by the order of the model.

Linear in the number of data points
Exploiting the Sparse Structure:

Original problem: Scales as $O((N_pN_s)^6)$

\[
P_0: \begin{cases}
    r_i^T r_i = 1, \forall i = 1, \ldots, N_s \\
    r_1(1) \geq r_2(1) \geq \cdots \geq r_{N_s}(1) \geq 0 \\
    \sum_{i=1}^{N_p} s_{i,j} r_i^T x_j \leq \epsilon s_{i,j}, \forall i = 1, \ldots, N_s \\
\end{cases}
\]

\[\forall j = 1: P_j: \begin{cases}
    s_{i,j}^2 \leq s_{i,j}, \forall i = 1, \ldots, N_s \\
    \sum_{i=1}^{N_s} s_{i,j} = 1
\end{cases}\]

Reduced problem: Scales as $O(N_p(N_s)^6)$

\[
\begin{align*}
\text{Tr}(\bar{Q}_{k,0} M_0) &\leq 0, \forall k = 1, \ldots, K_0 \\
M_0 &\succeq 0, M_0(1,1) = 1 \\
\text{rank}(M_0) &= 1 \\
\forall j = 1: \begin{cases}
    \text{Tr}(\bar{Q}_{k,j} M_j) &\leq 0, \forall k = 1, \ldots, K_j \\
    M_j &\succeq 0, M_j(1,1) = 1 \\
    M_j(1:nN_s + 1, 1:nN_s + 1) &= M_0
\end{cases}
\end{align*}
\]

Linear in the number of data points
Exploiting the Sparse Structure:

Original problem:
Scales as $O((N_pN_s)^6)$

\[
P_0 : \begin{cases} 
  r_i^T r_i = 1, \forall N_s \\
  r_1(1) \geq r_2(1) \geq \cdots \geq r_N(1) \geq 0 \\
  \forall j=1 : P_j : \begin{cases} 
    s_{i,j}^T r_i x_j \leq \epsilon s_{i,j}, \forall N_s \\
    s_{i,j}^2 = s_{i,j}, \forall N_s \\
    \sum_{i=1}^{N_s} s_{i,j} = 1 
  \end{cases} 
\end{cases}
\]

Reduced problem:
Scales as $O(N_p(N_s)^6)$

\[
\begin{cases} 
  \text{Tr}(\bar{Q}_{k,0}M_0) \leq 0, \forall K_0 \\
  M_0 \succeq 0, M_0(1,1) = 1 \\
  \text{rank}(M_0) = 1 \\
  \forall j=1 : \begin{cases} 
    \text{Tr}(Q_{k,j}M_j) \leq 0, \forall K_j \\
    M_j \succeq 0, M_j(1,1) = 1 \\
    M_j(1 : nN_s + 1, 1 : nN_s + 1) = M_0
  \end{cases} 
\end{cases}
\]

Caveat: still need to deal with a rank constraint
Example: Human Activity Analysis

![Image of a laboratory setting]

![Graph showing moment clustering with labels WALK, BEND, WALK]
(In)Validating SARX Models
Model (In)validation of SARX Systems

- **Given:**
  - A nominal switched model of the form:
    
    \[
    \tilde{y}_t = y_t + \eta_t \\
    y_t = \sum_{k=1}^{n_a} A_k(\sigma_t) y_{t-k} + \sum_{k=1}^{n_c} C_k(\sigma_t) u_{t-k} + f(\sigma_t) 
    \]
  - A bound on the noise \( ||\eta||_{\infty} \leq \varepsilon \)
  - Experimental Input/Output Data \( \{u_t, \tilde{y}_t\}_{t=t_0}^T \)

- **Determine:**
  - whether there exist noise and switching sequences consistent with a priori information and experimental data
Model (In)validation of SARX Systems

- **Given:**
  - A nominal switched model of the form:
    \[
    \begin{align*}
    y_t &= \sum_{k=1}^{n_a} A_k(\sigma_t)y_{t-k} + \sum_{k=1}^{n_c} C_k(\sigma_t)u_{t-k} + f(\sigma_t) \\
    \tilde{y}_t &= y_t + \eta_t
    \end{align*}
    \]
  - A bound on the noise (\(\|\eta\|_\infty \leq \epsilon\))
  - Experimental Input/Output Data \(\{u_t, \tilde{y}_t\}_{t=t_0}^T\)

- **Determine:**
  - whether there exist noise and switching sequences consistent with a priori information and experimental data

Reduces to SDP via Putinar’s Positivstellensatz
Given:
- A nominal switched model of the form:
  \[ y_t = \sum_{k=1}^{n_a} A_k(\sigma_t)y_{t-k} + \sum_{k=1}^{n_c} C_k(\sigma_t)u_{t-k} + f(\sigma_t) \]
  \[ \tilde{y}_t = y_t + \eta_t \]
- A bound on the noise (\(|\eta|_\infty \leq \varepsilon\))
- Experimental Input/Output Data \(\{u_t, \tilde{y}_t\}_{t=t_0}^T\)

Determine:
- whether there exist noise and switching sequences consistent with a priori information and experimental data

Reduces to SDP via Putinar’s Positivstellensatz

Guaranteed convergence for the \(n=T\) relaxation
(In)validation Certificates:

- The model is invalid if and only if

\[
\begin{align*}
    d^* &= \left\{ \begin{array}{l}
        \min_{s, \eta} \sum_{t=1}^{T} \sum_{i=1}^{n_s} e_{i,t}^2 \\
        \text{subject to:} \\
        s_{i,t} (g_{i,t} + h_{i,t} \eta_{t-n_a:t}) = e_{i,t} \\
        \sum_i s_{i,t} = 1 \\
        s_{i,t}^2 = 1 \\
        \|\eta\|_\infty \leq \epsilon
    \end{array} \right\} > 0
\end{align*}
\]
Model (In)validation of SARX Systems

Hidden sparse structure similar to the Id case

Complexity dominated by the order of the model

Noise from \( t \) to \( t-n \)
Example: Activity Monitoring

- **A priori switched model:** walking and waiting, 4% noise
- **Test sequences of hybrid behavior:**

  - **WALK, WAIT**
    - Not Invalidated
  - **RUN**
    - Invalidated
  - **WALK, JUMP**
    - Invalidated
Adding topological constraints:

- The model is invalid if and only if

\[
d^* = \left\{ \begin{array}{l}
\min_{\mathbf{s}, \mathbf{\eta}} \sum_{t=1}^{T} \sum_{i=1}^{n_s} e_{i,t}^2 \\
\text{subject to:}
\quad \mathbf{s}_{i,t} (g_{i,t} + h_{i,t} \mathbf{\eta}_{t-n_\alpha:t}) = e_{i,t} \\
\quad \sum_i s_{i,t} = 1 \\
\quad s_{i,t}^2 = 1 \\
\quad \|\mathbf{\eta}\|_\infty \leq \epsilon
\end{array} \right\} > 0
\]

plus additional linear constraints:

\[s_{i,t} + s_{j,t+1} \leq 1, \forall i \in I, \forall j \in J\]

These destroy sparsity patterns!
Example: Activity Monitoring

A Priori information

run

walk

Not Invalidated  (d=-3e-8)

Invalidated  (d=0.175)
Identifying Sparse Dynamical Networks
Formalization as a graph id problem:

Each time series becomes a **node** in a graph

Each edge is a **dynamical system**

\[
x_i(t) = \sum_{j=1,j \neq i}^{P} \sum_{n=k}^{N} (a_{ji}(n)x_j(t - n)) + u_i(t) + \eta_i(t)
\]
A Sparsification Problem:

- Find block sparse solutions to:
  \[ x = [X, I][a^t u]^t + \eta \]

- Efficient solutions using atomic norm minimization
- Atoms are the time series at other nodes
- Projection free Frank-Wolfe algorithm
Algorithm

\[
\begin{align*}
\min_{z} & \quad f(z) \\
\text{s.t.} & \quad \|z\|_A \leq \tau
\end{align*}
\]

\[
\begin{align*}
\min_{z} & \quad \|z - x_j\|_2 \\
\text{s.t.} & \quad \|z\|_{sA} \leq \tau
\end{align*}
\]

Frank–Wolfe Algorithm

1: Initialize:
   \[z^{(0)} \leftarrow \tau a_0\] for arbitrary \(a_0 \in \mathcal{A}\)

2: for \(k = 0, 1, 2, \cdots\) do

3: \[a \leftarrow \arg\min_{a \in \mathcal{A}} \langle \partial f(z^{(k)}), a \rangle\]

4: \[\alpha_k \leftarrow \arg \min_{\alpha \in [0,1]} f(z^{(k)} + \alpha [\tau a - z^{(k)}])\]

5: \[z^{(k+1)} \leftarrow z^{(k)} + \alpha_k [\tau a - z^{(k)}]\]

6: end for

Converges as \(O(1/n)\)
Algorithm

$$\min_z f(z) \quad \min_z ||z - x_j||_2$$

s.t. $||z||_A \leq \tau$  

s.t. $||z||_{sA} \leq \tau$

Frank-Wolfe Algorithm

1: Initialize:
$$z^{(0)} \leftarrow \tau a_0 \quad \text{for arbitrary } a_0 \in A$$

2: for $k = 0, 1, 2, \cdots$ do

3:
$$a \leftarrow \arg \min_{a \in A} \langle \partial f(z^{(k)}), a \rangle$$

4:
$$\alpha_k \leftarrow \arg \min_{\alpha \in [0,1]} f(z^{(k)} + \alpha [\tau a - z^{(k)}])$$

5:
$$z^{(k+1)} \leftarrow z^{(k)} + \alpha_k [\tau a - z^{(k)}]$$

6: end for

$L \leftarrow \arg \max_l \{||[\partial f(z^{(k)})]^{T} A_l||_1\}$
$c \leftarrow -\text{sign}([\partial f(z^{(k)})]^{T} A_L)$
$a \leftarrow A_L c$
Algorithm

\[
\begin{align*}
\min_z & \quad f(z) \\
\text{s.t.} & \quad \|z\|_A \leq \tau
\end{align*}
\]

Frank–Wolfe Algorithm

1: Initialize:
\[z^{(0)} \leftarrow \tau a_0 \quad \text{for arbitrary } a_0 \in A\]

2: for \( k = 0, 1, 2, \ldots \) do

3: \[a \leftarrow \arg\min_{a \in A} \langle \nabla f(z^{(k)}), a \rangle\]

4: \[\alpha_k \leftarrow \arg\min_{\alpha \in [0,1]} f(z^{(k)} + \alpha [\tau a - z^{(k)}])\]

5: \[z^{(k+1)} \leftarrow z^{(k)} + \alpha_k [\tau a - z^{(k)}]\]

6: end for

\[
\begin{align*}
L & \leftarrow \arg\max_i \{\|\nabla f(z^{(k)})^T A_i\|_1\} \\
c & \leftarrow -\text{sign}([\nabla f(z^{(k)})]^T A_L) \\
a & \leftarrow A_L c \\
\alpha_k & \leftarrow \max\{\min\left\{\frac{[\tau a - z^{(k)}]^T [x_j - z^{(k)}]}{\|\tau a - z^{(k)}\|_2^2}, 1\right\}, 0\}
\end{align*}
\]
Frank–Wolfe Algorithm

1: Initialize:
\[ z^{(0)} \leftarrow \tau a_0 \text{ for arbitrary } a_0 \in \mathcal{A} \]

2: for \( k = 0, 1, 2, \cdots \) do

3:
\[ a \leftarrow \arg \min_{a \in \mathcal{A}} \partial f(z^{(k)}), a \]

4:
\[ \alpha_k \leftarrow \arg \min_{\alpha \in [0,1]} f(z^{(k)} + \alpha [\tau a - z^{(k)}]) \]

5:
\[ z^{(k+1)} \leftarrow z^{(k)} + \alpha_k [\tau a - z^{(k)}] \]

6: end for

 Closed form solutions to each step

\[ L \leftarrow \arg \max \{ \| \partial f(z^{(k)})^T A_l \|_1 \} \]
\[ c \leftarrow -\text{sign}(\partial f(z^{(k)})^T A_L) \]
\[ a \leftarrow A_L c \]

\[ \alpha_k \leftarrow \max\{\min\{\frac{[\tau a - z^{(k)}]^T [x_j - z^{(k)}]}{\|\tau a - z^{(k)}\|_2^2}, 1\}, 0\} \]
Example

Interactions between human agents
More examples:
Tracking by detection

Reduces to an assignment problem with "dynamics-induced" weights

\[ \frac{\text{rank}(H_i) + \text{rank}(H_j)}{\text{rank}(\begin{bmatrix} H_i & H_j \end{bmatrix})} - 1 \]
Crowd photography sequencing
More examples where sparsity & self similarity help

- Semi-supervised SysId
- Wiener systems identification
- Identification with outliers
- Identification of PWA systems
- (In)validation of PWA systems
- Sparse network Id.
- Optimal sensor placement
- Controller design subject to sparsity constraints

All of these are known to be NP-hard, yet often solvable in polynomial time using sparsity based convex relaxations
What is Big Data?
Computational complexity is related to data interconnectivity, not data size!!
Computational complexity is related to data interconnectivity, not data size!!

Related to max clique size of an underlying graph
Big Data & Sparsity:

*Sparsity can provide a way around the curse of dimensionality*

- Challenge: how to build in and exploit the “right” sparsity
  - Graphs with small tree width (network design)
  - Low order models
- Submodularity also helps
  - what other properties can we exploit?
- An interesting connection between several communities:
  - Control, semi-algebraic optimization, machine learning,....
Big Data & Sparsity:

- Challenge: how to build in and exploit the “right” sparsity
  - Graphs with small tree width (network design)
  - Low order models
- Submodularity also helps
  - what other properties can we exploit?
- An interesting connection between several communities:
  - Control, semi-algebraic optimization, machine learning,....

Sparsity can provide a way around the curse of dimensionality.
SoS for “real sized” problems:

- Many promising advances towards making SoS/Moments practical:
  - ADMM, Frank-Wolfe, Factorizations

- Empirical experience: force M to be rank 1
  In many practical problems (e.g. subspace clustering) forcing a small matrix (much smaller than the running intersection) to have rank 1 guarantees rank(M)=1

- New developments covered elsewhere in this workshop
  - Ahmadi & Hall: DSoS and SDSoS
  - Lasserre: Krivine+Putinar P-satz (LP+fixed size SDC)

- Getting there, but more work needed. Keep tuned for more
Acknowledgements:

- Many thanks to:
  - Audience
  - Students: Y. Cheng, Y. Wang, X. Zhang
  - Colleagues: O. Camps, C. Lagoa, N. Ozay
  - Workshop organizers
  - Funding agencies (AFOSR, DHS, NSF)

More information as http://robustsystems.coe.neu.edu