Graph structure in polynomial systems: chordal networks

Pablo A. Parrilo

Laboratory for Information and Decision Systems
Electrical Engineering and Computer Science
Massachusetts Institute of Technology

Based on joint work
with Diego Cifuentes (MIT)

SIAM Annual Meeting - July 2016
Many application domains require the solution of large-scale systems of polynomial equations.

Among others: robotics, power systems, chemical engineering, cryptography, etc.
Background: structured polynomial systems

Many application domains require the solution of large-scale systems of polynomial equations.

Among others: robotics, power systems, chemical engineering, cryptography, etc.

Today: **MS111/MS126** - Structured Polynomial Equations and Applications
Polynomial systems and graphs

A polynomial system defined by $m$ equations in $n$ variables:

$$f_i(x_0, \ldots, x_{n-1}) = 0, \quad i = 1, \ldots, m$$
A polynomial system defined by \( m \) equations in \( n \) variables:

\[
f_i(x_0, \ldots, x_{n-1}) = 0, \quad i = 1, \ldots, m
\]

Construct a graph \( G \) ("primal graph") with \( n \) nodes:

- Nodes are variables \( \{x_0, \ldots, x_{n-1}\} \).
- For each equation, add a clique connecting the variables appearing in that equation.
A polynomial system defined by \( m \) equations in \( n \) variables:

\[
f_i(x_0, \ldots, x_{n-1}) = 0, \quad i = 1, \ldots, m
\]

Construct a graph \( G \) ("primal graph") with \( n \) nodes:

- Nodes are variables \( \{x_0, \ldots, x_{n-1}\} \).
- For each equation, add a clique connecting the variables appearing in that equation.

Example:

\[
I = \langle x_0^2x_1x_2 + 2x_1 + 1, \quad x_1^2 + x_2, \quad x_1 + x_2, \quad x_2x_3 \rangle
\]
Questions

“Abstracted” the polynomial system to a (hyper)graph.
“Abstracted” the polynomial system to a (hyper)graph.

- Can the graph structure help solve this system?
- For instance, to optimize, or to compute Groebner bases?
- Or, perhaps we can do something better?
- Preserve graph (sparsity) structure?
- Complexity aspects?
(Hyper)Graphical modelling

Pervasive idea in many areas, in particular: numerical linear algebra, graphical models, constraint satisfaction, database theory, . . .

Key notions: chordality and treewidth.

Many names: Arnborg, Beeri/Fagin/Maier/Yannakakis, Blair/Peyton, Bodlaender, Courcelle, Dechter, Freuder, Lauritzen/Spiegelhalter, Pearl, Robertson/Seymour, . . .

Remarkably (AFAIK) almost no work in computational algebraic geometry exploits this structure.

Reasonably well-known in discrete (0/1) optimization, what happens in the continuous side? (e.g., Waki et al., Lasserre, Bienstock, Vandenberghe, Lavaei, etc)
(Hyper)Graphical modelling

Pervasive idea in many areas, in particular: numerical linear algebra, graphical models, constraint satisfaction, database theory, . . .

Key notions: chordality and treewidth.

Many names: Arnborg, Beeri/Fagin/Maier/Yannakakis, Blair/Peyton, Bodlaender, Courcelle, Dechter, Freuder, Lauritzen/Spiegelhalter, Pearl, Robertson/Seymour, . . .

Remarkably (AFAIK) almost no work in computational algebraic geometry exploits this structure.
(Hyper)Graphical modelling

Pervasive idea in many areas, in particular: numerical linear algebra, graphical models, constraint satisfaction, database theory, ... 

Key notions: chordality and treewidth.

Many names: Arnborg, Beeri/Fagin/Maier/Yannakakis, Blair/Peyton, Bodlaender, Courcelle, Dechter, Freuder, Lauritzen/Spiegelhalter, Pearl, Robertson/Seymour, ...

Remarkably (AFAIK) almost no work in computational algebraic geometry exploits this structure.

Reasonably well-known in discrete (0/1) optimization, what happens in the continuous side? (e.g., Waki et al., Lasserre, Bienstock, Vandenberghe, Lavaei, etc)
Chordality

Let $G$ be a graph with vertices $x_0, \ldots, x_{n-1}$. A vertex ordering

$$x_0 > x_1 > \cdots > x_{n-1}$$

is a perfect elimination ordering if for all $l$, the set

$$X_l := \{x_l\} \cup \{x_m : x_m \text{ is adjacent to } x_l, x_l > x_m\}$$

is such that the restriction $G|_{X_l}$ is a clique.
Chordality

Let $G$ be a graph with vertices $x_0, \ldots, x_{n-1}$. A vertex ordering

$$x_0 > x_1 > \cdots > x_{n-1}$$

is a perfect elimination ordering if for all $l$, the set

$$X_l := \{x_l\} \cup \{x_m : x_m \text{ is adjacent to } x_l, \ x_l > x_m\}$$

is such that the restriction $G|_{X_l}$ is a clique.

A graph is chordal if it has a perfect elimination ordering.
Chordality

Let $G$ be a graph with vertices $x_0, \ldots, x_{n-1}$. A vertex ordering

$$x_0 > x_1 > \cdots > x_{n-1}$$

is a perfect elimination ordering if for all $l$, the set

$$X_l := \{x_l\} \cup \{x_m : x_m \text{ is adjacent to } x_l, \ x_l > x_m\}$$

is such that the restriction $G|_{X_l}$ is a clique.

A graph is chordal if it has a perfect elimination ordering.

(Equivalently, in numerical linear algebra: Cholesky factorization has no “fill-in”)

Cifuentes, Parrilo (MIT)
Chordality, treewidth, and a meta-theorem

A chordal completion of $G$ is a chordal graph with the same vertex set as $G$, and which contains all edges of $G$. 

Informally, treewidth quantitatively measures how "tree-like" a graph is. 

Meta-theorem: NP-complete problems are "easy" on graphs of small treewidth.
Chordality, treewidth, and a meta-theorem

A chordal completion of $G$ is a chordal graph with the same vertex set as $G$, and which contains all edges of $G$.

The treewidth of a graph is the clique number (minus one) of its smallest chordal completion.

Informally, treewidth quantitatively measures how “tree-like” a graph is.
Chordality, treewidth, and a meta-theorem

A chordal completion of $G$ is a chordal graph with the same vertex set as $G$, and which contains all edges of $G$.

The treewidth of a graph is the clique number (minus one) of its smallest chordal completion.

Informally, treewidth quantitatively measures how “tree-like” a graph is.

Meta-theorem:
NP-complete problems are “easy” on graphs of small treewidth.
Recall the subset sum problem, with data $A = \{a_1, \ldots, a_n\} \subset \mathbb{Z}$. Is there a subset of $A$ that adds up to 0?

Letting $s_i$ be the partial sums, we can write a polynomial system:

$$
0 = s_0 \\
0 = (s_i - s_{i-1})(s_i - s_{i-1} - a_i) \\
0 = s_n
$$

The graph associated with these equations is a path (treewidth=1)

$$
S_0 \longrightarrow S_1 \longrightarrow S_2 \longrightarrow \cdots \longrightarrow S_n
$$

But, subset sum is NP-complete… :(
For linear equations, “good” elimination preserves graph structure (perfect!)

Ex: Consider $I = \langle x_0 x_2 - 1, x_1 x_2 - 1 \rangle$, whose associated graph is the path $x_0 - x_2 - x_1$.

Every Groebner basis must contain the polynomial $x_0 - x_1$, breaking the sparsity structure.

Q: Are there alternative descriptions that “play nicely” with graphical structure?
For linear equations, “good” elimination preserves graph structure (perfect!)

For polynomials, however, Groebner bases can destroy chordality.

Ex: Consider

\[ l = \langle x_0 x_2 - 1, x_1 x_2 - 1 \rangle, \]

whose associated graph is the path \( x_0 \rightarrow x_2 \rightarrow x_1 \).
Bad news? (II)

For linear equations, “good” elimination preserves graph structure (perfect!)

For polynomials, however, Groebner bases can destroy chordality.

Ex: Consider

\[ l = \langle x_0 x_2 - 1, x_1 x_2 - 1 \rangle, \]

whose associated graph is the path \( x_0 \longrightarrow x_2 \longrightarrow x_1 \).

Every Groebner basis must contain the polynomial \( x_0 - x_1 \), breaking the sparsity structure.
For *linear* equations, “good” elimination preserves graph structure (perfect!)

For polynomials, however, Groebner bases can destroy chordality.

Ex: Consider

\[ I = \langle x_0 x_2 - 1, x_1 x_2 - 1 \rangle, \]

whose associated graph is the path \( x_0 \rightarrow x_2 \rightarrow x_1 \).

*Every* Groebner basis must contain the polynomial \( x_0 - x_1 \), breaking the sparsity structure.

**Q:** Are there alternative descriptions that “play nicely” with graphical structure?
How to resolve this (apparent) contradiction?

“Trees are good” \(\iff\) “Trees can be NP-hard"
How to resolve this (apparent) contradiction?

“Trees are good” $\iff$ “Trees can be NP-hard”

Underlying hero/culprit: dynamic programming (DP), and more refined cousins (nonserial DP, belief propagation, etc).
How to resolve this (apparent) contradiction?

“Trees are good” \[???\] “Trees can be NP-hard”

Underlying hero/culprit: dynamic programming (DP), and more refined cousins (nonserial DP, belief propagation, etc).

Key: “nice” graphical structure allows DP to work in principle. But, we also need to control the complexity of the objects DP is propagating. Without this, we’re doomed!

[Ubiquitous theme: “complicated” value functions in optimal control, “message complexity” in statistical inference, …]
How to get around this?

Need to impose conditions on the geometry!
How to get around this?

Need to impose conditions on the geometry!

In the algebraic setting, a natural condition: degree of projections onto clique subspaces.
How to get around this?

Need to impose conditions on the geometry!

In the algebraic setting, a natural condition: degree of projections onto clique subspaces.

Consider the full solution set (an algebraic variety).

Require the projections onto the subspaces spanned by the maximal cliques to have bounded degree.
How to get around this?

Need to impose conditions on the geometry!

In the algebraic setting, a natural condition: degree of projections onto clique subspaces.

Consider the full solution set (an algebraic variety).

Require the projections onto the subspaces spanned by the maximal cliques to have bounded degree.

- For discrete domains (e.g., 0/1 problems), always satisfied.
- Holds in other cases, e.g., low-rank matrices (determinantal varieties).
Two approaches

- **Chordal elimination and Groebner bases** (arXiv:1411:1745)
  - New *chordal elimination* algorithm, to exploit graphical structure
  - Conditions under which chordal elimination succeeds
  - For a certain class, complexity is *linear* in number of variables! (exponential in treewidth)
  - Implementation and experimental results

- **Chordal networks** (arXiv:1604.02618)
  - New representation/decomposition for polynomial systems
  - Efficient algorithms to compute them. Can use them for root counting, dimension, radical ideal membership, etc.
  - Links to BDDs (binary decision diagrams) and extensions
Example 1: Coloring a cycle

Let $C_n = (V, E)$ be the cycle graph and consider the ideal $I$ given by the equations

\[ x_i^3 - 1 = 0, \quad i \in V \]
\[ x_i^2 + x_i x_j + x_j^2 = 0, \quad ij \in E \]

These equations encode the proper 3-colorings of the graph. Note that coloring the cycle graph is very easy!
Example 1: Coloring a cycle

Let $C_n = (V, E)$ be the cycle graph and consider the ideal $I$ given by the equations

$$x_i^3 - 1 = 0, \quad i \in V$$
$$x_i^2 + x_i x_j + x_j^2 = 0, \quad ij \in E$$

These equations encode the proper 3-colorings of the graph. Note that coloring the cycle graph is very easy!

However, a Gröbner basis is not so simple: one of its 13 elements is

$$x_0 x_2 x_4 x_6 + x_0 x_2 x_4 x_7 + x_0 x_2 x_4 x_8 + x_0 x_2 x_5 x_6 + x_0 x_2 x_5 x_7 + x_0 x_2 x_5 x_8 + x_0 x_2 x_6 x_8 + x_0 x_2 x_7 x_8 + x_0 x_2 x_2^2 + x_0 x_3 x_4 x_6 + x_0 x_3 x_4 x_7 + x_0 x_3 x_5 x_6 + x_0 x_3 x_5 x_7 + x_0 x_3 x_5 x_8 + x_0 x_3 x_7 x_8 + x_0 x_3 x_2^2 + x_0 x_4 x_6 x_8 + x_0 x_4 x_7 x_8 + x_0 x_4 x_2^2 + x_0 x_5 x_6 x_8 + x_0 x_5 x_7 x_8 + x_0 x_6 x_8 + x_0 x_6 x_2^2 + x_0 x_7 x_8 + x_0 x_1 x_2 x_4 x_6 + x_1 x_2 x_4 x_7 + x_1 x_2 x_4 x_8 + x_1 x_2 x_5 x_6 + x_1 x_2 x_5 x_7 + x_1 x_2 x_5 x_8 + x_1 x_2 x_6 x_8 + x_1 x_2 x_7 x_8 + x_1 x_2 x_8^2 + x_1 x_3 x_4 x_6 + x_1 x_3 x_4 x_7 + x_1 x_3 x_4 x_8 + x_1 x_3 x_5 x_6 + x_1 x_3 x_5 x_7 + x_1 x_3 x_5 x_8 + x_1 x_3 x_6 x_8 + x_1 x_3 x_7 x_8 + x_1 x_3 x_2^2 + x_1 x_4 x_6 x_8 + x_1 x_4 x_7 x_8 + x_1 x_4 x_8^2 + x_1 x_5 x_6 x_8 + x_1 x_5 x_7 x_8 + x_1 x_5 x_2^2 + x_1 x_6 x_8 + x_1 x_7 x_8 + x_1 + x_2 x_4 x_6 x_8 + x_2 x_4 x_7 x_8 + x_2 x_4 x_8^2 + x_2 x_5 x_6 x_8 + x_2 x_5 x_7 x_8 + x_2 x_5 x_8^2 + x_2 x_6 x_8 + x_2 x_7 x_8 + x_2 + x_3 x_4 x_6 x_8 + x_3 x_4 x_7 x_8 + x_3 x_4 x_8^2 + x_3 x_5 x_6 x_8 + x_3 x_5 x_7 x_8 + x_3 x_5 x_8^2 + x_3 x_6 x_8 + x_3 x_7 x_8 + x_3 + x_4 x_6 x_8 + x_4 x_7 x_8 + x_4 + x_5 x_6 x_8 + x_5 x_7 x_8 + x_5 + x_6 + x_7 + x_8$$
Example 1: Coloring a cycle

There is a nicer representation, that respects its graphical structure. The solution set can be decomposed into triangular sets:

$$\mathcal{V}(I) = \bigcup_T \mathcal{V}(T)$$

where the union is over all maximal directed paths in the figure. The number of triangular sets is 21, which is the 8-th Fibonacci number.
Chordal networks

A new representation of structured polynomial systems!

- What do they look like?
  - “Enlarged” elimination tree, with polynomial sets as nodes.
  - Efficient encoding of components in paths/subtrees.

How can you compute them?

A nice algorithm to compute chordal networks.

Remarkably, many polynomial systems admit “small” chordal networks, even though the number of components may be exponentially large.

What are they good for?

Can be effectively used to solve feasibility, counting, dimension, elimination, radical membership, . . .

Linear time algorithms (exponential in treewidth)

Implementation and experimental results.
Chordal networks

A new representation of structured polynomial systems!

- What do they look like?
  - “Enlarged” elimination tree, with polynomial sets as nodes.
  - Efficient encoding of components in paths/subtrees.

- How can you compute them?
  - A nice algorithm to compute chordal networks.
  - Remarkably, many polynomial systems admit “small” chordal networks, even though the number of components may be exponentially large.

What are they good for?

Can be effectively used to solve feasibility, counting, dimension, elimination, radical membership, ...
A new representation of structured polynomial systems!

- What do they look like?
  - “Enlarged” elimination tree, with polynomial sets as nodes.
  - Efficient encoding of components in paths/subtrees.

- How can you compute them?
  - A nice algorithm to compute chordal networks.
  - Remarkably, many polynomial systems admit “small” chordal networks, even though the number of components may be exponentially large.

- What are they good for?
  - Can be effectively used to solve feasibility, counting, dimension, elimination, radical membership, . . .
  - Linear time algorithms (exponential in treewidth)
  - Implementation and experimental results.
The elimination tree of a graph $G$ is the following directed spanning tree:

For each $\ell$ there is an arc from $x_\ell$ towards the largest $x_p$ that is adjacent to $x_\ell$ and $p > \ell$.

Note that the elimination tree is rooted at $x_{n-1}$.
A \textit{G-chordal network} is a directed graph $\mathcal{N}$, whose nodes are polynomial sets in $\mathbb{K}[X]$, such that:

- each node $F$ is given a $\text{rank}(F) \in \{0, \ldots, n-1\}$, s.t. $F \subset \mathbb{K}[X_{\text{rank}(F)}]$.
- for any arc $(F_\ell, F_p)$ we have that $x_p$ is the parent of $x_\ell$ in the elimination tree of $G$, where $\ell = \text{rank}(F_\ell)$, $p = \text{rank}(F_p)$.

A chordal network is \textit{triangular} if each node consists of a single polynomial $f$, and either $f = 0$ or its largest variable is $x_{\text{rank}(f)}$. 
Chordal networks (Example)

\[ g(a, b, c) := a^2 + b^2 + c^2 + ab + bc + ca \]
Computing chordal networks (Example)

\[ I = \langle x_2 - x_3, x_1 - x_2, x_1^2 - x_1, x_0x_2 - x_2, x_0^3 - x_0 \rangle \]

The output of the algorithm will be

This represents the decomposition of \( I \) into the triangular sets

\[
(x_3, x_2, x_1 - x_2, x_0^3 - x_0), \\
(x_3, x_2 - 1, x_1 - x_2, x_0 - 1), \\
(x_3 - 1, x_2 - 1, x_1 - x_2, x_0 - 1).
\]
Computing chordal networks (Example)

\[ x_0^3 - x_0, x_0 x_2 - x_2, x_2^2 - x_2 \]

\[ x_1 - x_2, x_2^2 - x_2 \]

\[ x_2^2 - x_2, x_2 x_3^2 - x_3 \]

0
Computing chordal networks (Example)

\[ x_3^3 - x_0, x_0 x_2 - x_2, x_2^2 - x_2 \]
\[ x_1 - x_2, x_2^2 - x_2 \]
\[ x_2^2 - x_2, x_2 x_3^2 - x_3 \]
\[ 0 \]

\[ \rightarrow \ t r i a \]

\[ x_3^3 - x_0, x_2 \]
\[ x_0 - 1, x_2 - 1 \]
\[ x_1 - x_2, x_2^2 - x_2 \]
\[ x_2^2 - x_2, x_2 x_3^2 - x_3 \]
\[ 0 \]
Computing chordal networks (Example)

\[ x_0^3 - x_0, x_0x_2 - x_2, x_2^2 - x_2 \]
\[ x_1 - x_2, x_2^2 - x_2 \]
\[ x_2^2 - x_2, x_2x_3 - x_3 \]
\[ 0 \] \(\xrightarrow{\text{tria}}\)

\[ x_0^3 - x_0, x_0, x_0 - 1, x_2 - 1 \]
\[ x_1 - x_2, x_2^2 - x_2 \]
\[ x_2^2 - x_2, x_2x_3^2 - x_3 \]
\[ x_2^2 - x_2, x_2x_3 - x_3, x_2 \]
\[ x_2^2 - x_2, x_2x_3^2 - x_3, x_2 - 1 \]
\[ 0 \] \(\xrightarrow{\text{elim}}\)

\[ x_0^3 - x_0 \]
\[ x_1 - x_2, x_2^2 - x_2 \]
\[ x_2^2 - x_2, x_2x_3 - x_3, x_2 \]
\[ x_2^2 - x_2, x_2x_3^2 - x_3, x_2 - 1 \]
\[ 0 \] \(\xrightarrow{\text{merge}}\)
Computing chordal networks (Example)

\[
\begin{align*}
\text{tria} & \quad x_0^3 - x_0, x_0 x_2 - x_2, x_2^2 - x_2 \\
& \quad x_1 - x_2, x_2^2 - x_2 \\
& \quad x_2 - x_2, x_2 x_3^2 - x_3 \\
& \quad 0 \\
\end{align*}
\]

\[
\begin{align*}
\text{elim} & \quad x_0^3 - x_0, x_2 \\
& \quad x_0 - 1, x_2 - 1 \\
& \quad x_1 - x_2, x_2^2 - x_2 \\
& \quad x_2 - x_2, x_2 x_3^2 - x_3 \\
& \quad x_2^2 - x_2, x_2 x_3^2 - x_3, x_2 - 1 \\
& \quad 0 \\
\end{align*}
\]

\[
\begin{align*}
\text{elim} & \quad x_0^3 - x_0 \\
& \quad x_0 - 1 \\
& \quad x_1 - x_2 \\
& \quad x_2^2 - x_2, x_2 x_3^2 - x_3, x_2 \\
& \quad x_2^2 - x_2, x_2 x_3^2 - x_3, x_2 - 1 \\
& \quad 0 \\
\end{align*}
\]
Computing chordal networks (Example)

\[
\begin{align*}
&x_0^3 - x_0, x_0 x_2 - x_2, x_2^2 - x_2 \\
&x_1 - x_2, x_2^2 - x_2 \\
&x_2 - x_2, x_2 x_3^2 - x_3 \\
&0 \\
\end{align*}
\]

\(\xrightarrow{\text{tria}}\)

\[
\begin{align*}
&x_0^3 - x_0, x_2 \\
&x_0 - 1, x_2 - 1 \\
&x_1 - x_2, x_2^2 - x_2 \\
&x_2 - x_2, x_2 x_3^2 - x_3 \\
&0 \\
\end{align*}
\]

\(\xrightarrow{\text{elim}}\)

\[
\begin{align*}
&x_0^3 - x_0 \\
&x_0 - 1 \\
&x_1 - x_2, x_2^2 - x_2 \\
&x_2^2 - x_2, x_2 x_3^2 - x_3, x_2 \\
&x_2^2 - x_2, x_2 x_3^2 - x_3, x_2 - 1 \\
&0 \\
\end{align*}
\]

\(\xrightarrow{\text{tria}}\)

\[
\begin{align*}
&x_0^3 - x_0 \\
&x_0 - 1 \\
&x_1 - x_2 \\
&x_2, x_3 \\
&x_2 - 1, x_3 \\
&x_2 - 1, x_3 - 1 \\
&0 \\
\end{align*}
\]

\(\xrightarrow{\text{elim}}\)

\[
\begin{align*}
&x_0^3 - x_0 \\
&x_0 - 1 \\
&x_1 - x_2 \\
&x_2, x_3 \\
&x_2 - 1, x_3 \\
&x_2 - 1, x_3 - 1 \\
&0 \\
\end{align*}
\]
Computing chordal networks (Example)

\[
\begin{align*}
x_0^3 - x_0, x_0x_2 - x_2, x_2^2 - x_2 \\
x_1 - x_2, x_2^2 - x_2 \\
x_2^2 - x_2, x_2x_3^2 - x_3 \\
0
\end{align*}
\]

\[
\begin{align*}
x_0^3 - x_0, x_0 - x_2, x_1 - x_2, x_2^2 - x_2 \\
x_2^2 - x_2, x_2x_3^2 - x_3, x_2 - 1 \\
0
\end{align*}
\]

\[
\begin{align*}
x_0^3 - x_0 \\
x_1 - x_2 \\
x_2^2 - x_2, x_2x_3^2 - x_3, x_2 - 1 \\
0
\end{align*}
\]

\[
\begin{align*}
x_0^3 - x_0, x_0 - 1, x_2 - 1 \\
x_1 - x_2, x_2^2 - x_2 \\
x_2^2 - x_2, x_2x_3^2 - x_3, x_2 - 1 \\
0
\end{align*}
\]

\[
\begin{align*}
x_0^3 - x_0 \\
x_1 - x_2 \\
x_2^2 - x_2, x_2x_3^2 - x_3, x_2 - 1 \\
0
\end{align*}
\]

\[
\begin{align*}
x_0^3 - x_0 \\
x_1 - x_2 \\
x_2^2 - x_2, x_2x_3^2 - x_3, x_2 - 1 \\
0
\end{align*}
\]

\[
\begin{align*}
x_0^3 - x_0 \\
0
\end{align*}
\]
Computing chordal networks (Example)

\[
\begin{align*}
&x_0^3 - x_0, x_0x_2 - x_2, x_2^2 - x_2 \\
\triangleright &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0, x_0 - 1, x_2 - 1 \\
\triangleright &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0 \\
\elim &x_0 - 1 \\
\quad &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0, x_0 - 1 \\
\triangleright &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0 \\
\elim &x_0 - 1 \\
\quad &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0, x_0 - 1 \\
\triangleright &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0, x_0 - 1 \\
\elim &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0 \\
\elim &x_0 - 1 \\
\quad &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0, x_0 - 1 \\
\triangleright &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0 \\
\elim &x_0 - 1 \\
\quad &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0, x_0 - 1 \\
\triangleright &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]

\[
\begin{align*}
&x_0^3 - x_0 \\
\elim &x_0 - 1 \\
\quad &x_1 - x_2, x_2^2 - x_2 \\
\quad &x_2 - x_2, x_2x_3^2 - x_3, x_2^2 - x_2 \\
\end{align*}
\]
Given a triangular chordal network $\mathcal{N}$ of a polynomial system, the following problems can be solved in linear time:

- Compute the cardinality of $\mathcal{V}(I)$.
- Compute the dimension of $\mathcal{V}(I)$
- Describe the top dimensional component of $\mathcal{V}(I)$.

We also developed efficient algorithms to

- Solve the radical ideal membership problem ($h \in \sqrt{I}$?)
- Compute the equidimensional components of the variety.
Links to BDDs

Very interesting connections with *binary decision diagrams* (BDDs).

- A clever representation of Boolean functions/sets, usually much more compact than naive alternatives
- Enabler of very significant practical advances in (discrete) formal verification and model checking
- “One of the only really fundamental data structures that came out in the last twenty-five years” (D. Knuth)

For the special case of *monomial ideals*, chordal networks are equivalent to (reduced, ordered) BDDs. But in general, more powerful!
Implementation and examples

Implemented in Sage, using Singular and PolyBoRi (for $\mathbb{F}_2$).

- Graph colorings (counting $q$-colorings)
- Cryptography ("baby" AES, Cid et al.)
- Sensor Network localization
- Discretization of polynomial equations
- Reachability in vector addition systems
- Algebraic statistics
Given a set of vectors $B \subset \mathbb{Z}^n$, construct a graph with vertex set $\mathbb{N}^n$ in which $u, v \in \mathbb{N}^n$ are adjacent if $u - v \in \pm B$.

**Ex:** Determine whether $f_n \in I_n$, where

$$f_n := x_0 x_1^2 x_2^3 \cdots x_{n-1}^n - x_0^n x_1^{n-1} \cdots x_{n-1},$$

$$I_n := \{x_i x_{i+3} - x_{i+1} x_{i+2} : 0 \leq i < n\},$$

and where the indices are taken modulo $n$.

We compare our radical membership test with Singular (Gröbner bases) and Epsilon (triangular decompositions).

<table>
<thead>
<tr>
<th>$n$</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>55</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChordalNet</td>
<td>0.7</td>
<td>3.0</td>
<td>8.5</td>
<td>14.3</td>
<td>21.8</td>
<td>29.8</td>
<td>37.7</td>
<td>48.2</td>
<td>62.3</td>
<td>70.6</td>
<td>84.8</td>
</tr>
<tr>
<td>Singular</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td>17.9</td>
<td>1036.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Epsilon</td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
<td>2.0</td>
<td>54.4</td>
<td>160.1</td>
<td>5141.9</td>
<td>17510.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Consider the binomial ideal $I^{n_1,n_2}$ that models a 2D dimensional generalization of the birth-death Markov process. We fix $n_2 = 1$.

We can compute all irreducible components of the ideal faster than specialized packages (e.g., Macaulay2’s “Binomials”)

<table>
<thead>
<tr>
<th>$n$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>#components</td>
<td>3</td>
<td>11</td>
<td>40</td>
<td>139</td>
<td>466</td>
<td>1528</td>
<td>4953</td>
</tr>
<tr>
<td>time</td>
<td>0:00:00</td>
<td>0:00:01</td>
<td>0:00:04</td>
<td>0:00:13</td>
<td>0:02:01</td>
<td>0:07:35</td>
<td>12:22:19</td>
</tr>
</tbody>
</table>

For ChordalNet and Binomials
Consider the binomial ideal $I^{n_1,n_2}$ that models a 2D dimensional generalization of the birth-death Markov process. We fix $n_2 = 1$.

We can compute all irreducible components of the ideal faster than specialized packages (e.g., Macaulay2’s “Binomials”)

<table>
<thead>
<tr>
<th>$n$</th>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>#components</td>
<td>3</td>
<td>11</td>
<td>40</td>
<td>139</td>
<td>466</td>
<td>1528</td>
<td>4953</td>
<td></td>
</tr>
<tr>
<td>ChordalNet</td>
<td>0:00:00</td>
<td>0:00:01</td>
<td>0:00:04</td>
<td>0:00:13</td>
<td>0:02:01</td>
<td>0:37:35</td>
<td>12:22:19</td>
<td></td>
</tr>
<tr>
<td>Binomials</td>
<td>0:00:00</td>
<td>0:00:00</td>
<td>0:00:01</td>
<td>0:00:12</td>
<td>0:03:00</td>
<td>4:15:36</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Our methods are particularly efficient for computing the highest dimensional components.

<table>
<thead>
<tr>
<th>$n$</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>#comps</td>
<td>404</td>
<td>684</td>
<td>964</td>
<td>1244</td>
<td>1524</td>
<td>2442</td>
<td>5372</td>
<td>8702</td>
<td>12432</td>
</tr>
<tr>
<td>time</td>
<td>0:01:07</td>
<td>0:04:54</td>
<td>0:15:12</td>
<td>0:41:52</td>
<td>1:34:05</td>
<td>0:05:02</td>
<td>0:41:41</td>
<td>3:03:29</td>
<td>9:53:09</td>
</tr>
</tbody>
</table>
Summary

- (Hyper)graphical structure may simplify optimization/solving
- Under assumptions (treewidth + algebraic structure), tractable!
- New data structures: chordal networks
- Yields practical, competitive, implementable algorithms
- Ongoing and future work: other polynomial solving approaches (e.g., homotopies, full numerical algebraic geometry...)

If you want to know more:


Thanks for your attention! (and please come to MS111/MS126!)
Summary

- (Hyper)graphical structure *may* simplify optimization/solving
- Under assumptions (treewidth + algebraic structure), tractable!
- New data structures: chordal networks
- Yields practical, competitive, implementable algorithms
- Ongoing and future work: other polynomial solving approaches (e.g., homotopies, full numerical algebraic geometry...)

If you want to know more:


Thanks for your attention!
Summary

- (Hyper)graphical structure may simplify optimization/solving
- Under assumptions (treewidth + algebraic structure), tractable!
- New data structures: chordal networks
- Yields practical, competitive, implementable algorithms
- Ongoing and future work: other polynomial solving approaches (e.g., homotopies, full numerical algebraic geometry...)

If you want to know more:


Thanks for your attention! (and please come to MS111/MS126!)