Extension Complexity, MSO Logic, and Treewidth

Petr Kolman

Joint work with M. Koutecký, H. R. Tiwary

Department of Applied Mathematics
Faculty of Mathematics and Physics
Charles University

SWAT 2016
A polytope $Q \subseteq \mathbb{R}^{d+r}$ is an **extended formulation** of $P \subseteq \mathbb{R}^d$ if $P$ is a projection of $Q$ onto the first $d$ coordinates.

The **size** of $P$ is the number of its facet-defining inequalities.

The **extension complexity** of a polytope $P$, denoted by $\text{xc}(P)$, is the size of its smallest extended formulation.
Extended Formulations

What is the meaning?
- Complexity measure

Selected related results
- 1986-87 - Swart, attempts to prove $P=NP$ by giving polynomial size LP for TSP
- 1988 - Yannakakis, every symmetric EF for TSP has exponential size
- 2007 - Sellmann et al., EF for CSP of size $O(n^\tau)$ for graphs of treewidth $\tau$ using Sherali-Adams hierarchy
- 2012 - Fiorini et al., no polynomial-size EF for TSP
- 2014 - Rothvoß, no polynomial-size EF for matching polytope
- 2015 - K., Koutecký - EF for CSP of size $O(D^\tau n)$ for graphs of treewidth $\tau$ - includes vertex cover, independent set ...
- ...

Petr Kolman
Extension Complexity, MSO Logic, and Treewidth
**Input**
- A graph $G = (V, E)$ with $n$ vertices and treewidth $\tau$
- An MSOL formula $\varphi(\vec{X})$ with $m$ free set variables $X_1, \ldots, X_m$

**MSOL polytope**

$$P_{\varphi}(G) = \text{conv}\left(\{y \in \{0, 1\}^{nm} \mid y \text{ satisfies } \varphi\}\right).$$

Where $y^i_v = 1$ represents $v \in X_i$ and $y^i_v = 0$ represents $v \notin X_i$.

**Question**
- What is the extension complexity of $P_{\varphi}(G)$?
**Example**

### Formula and Graph

\[ 2\text{COL}(X_1, X_2) = (X_1 \cap X_2 = \emptyset) \land \forall x (x \in X_1 \lor x \in X_2) \land \]
\[ (\forall x \in X_1 \forall y \in X_1, x \neq y \rightarrow \neg E(x, y)) \land \]
\[ (\forall x \in X_2 \forall y \in X_2, x \neq y \rightarrow \neg E(x, y)) \]

\[ G = (\{u, v, w\}, \{\{u, v\}\}) \]

### Variables

\[ (y_u^1, y_u^2, y_v^1, y_v^2, y_w^1, y_w^2) \]

\[
\begin{array}{cccc}
(1, 0, 0, 1, 1, 0) & (0, 1, 1, 0, 0, 1) & (1, 0, 0, 1, 0, 1) & (0, 1, 1, 0, 1, 0) \\
\end{array}
\]

\[ P_\varphi(G) = \text{conv} (\{(1, 0, 0, 1, 1, 0), (0, 1, 1, 0, 0, 1), (1, 0, 0, 1, 0, 1), (0, 1, 1, 0, 1, 0)\}) \]
Main Result

**Theorem (K., Koutecký, Tiwary, 2016)**

For every graph $G$ on $n$ vertices with $\text{tw}(G) = \tau$ and for every MSOL formula $\varphi$,

$$\text{xc}(P_\varphi(G)) = f(|\varphi|, \tau) \cdot n$$

where $f$ is some computable function.

As a corollary, it yields the famous result about LinEMSOL problems:

**Theorem (Arnborg, Lagergren, and Seese, 1991)**

Every LinEMSOL problem is solvable in polynomial time for graphs of bounded treewidth.
Theorem (Courcelle, 1990)

*Every graph property definable in MSOL is decidable in linear time for graphs of bounded treewidth.*

Our theorem: Not a surprising result on the high level

*Merging common wisdom* from various CS areas

- Courcelle’s theorem = dynamic programming (parameterized complexity)
- dynamic programming = compact extended formulation (polyhedral combinatorics)

Our theorem: Far from obvious when it comes to details

- no black-box results for the above knowledge
The *cartesian product* of two polytopes $P_1$ and $P_2$

\[ P_1 \times P_2 = \text{conv}\{(x, y) \mid x \in \text{vert}(P_1), y \in \text{vert}(P_2)\} \]

The *glued product* of $P \in \mathbb{R}^{d_1+k}$ and $Q \in \mathbb{R}^{d_2+k}$, with respect to the last $k$ coordinates

\[ P \times_k Q = \text{conv}\{(x, y, z) \in \mathbb{R}^{d_1+d_2+k} \mid (x, z) \in \text{vert}(P), (y, z) \in \text{vert}(Q)\} \]

Lemma (Gluing lemma, Margot 1994, KKT 2016)

Let $P$ and $Q$ be 0/1-polytopes and let the $k$ glued coordinates in $P$ be labeled $z$, and the $k$ glued coordinates in $Q$ be labeled $w$. If $1^T z \leq 1$ is valid for $P$ and $1^T w \leq 1$ is valid for $Q$, then

\[ xc(P \times_k Q) \leq xc(P) + xc(Q). \]
Treewidth

Tree decomposition of $G = (V, E)$

A tree $T$, each node $a \in T$ has an assigned set of vertices $B(a) \subseteq V$, called a bag, some properties ...

Nice tree decomposition

- **Leaf node**: a leaf $a$ of $T$ with $B(a) = \emptyset$.
- **Introduce node**: an internal node $a$ of $T$ with one child $b$ for which $B(a) = B(b) \cup \{v\}$ for some $v \in B(a)$.
- **Forget node**: an internal node $a$ of $T$ with one child $b$ for which $B(a) = B(b) \setminus \{v\}$ for some $v \in B(b)$.
- **Join node**: an internal node $a$ with two children $b$ and $c$ with $B(a) = B(b) = B(c)$.

Subgraph of $G$ induced by a tree node

For a node $a \in V(T)$, we denote by $T_a$ the subtree of $T$ rooted in $a$, and by $G_a$ the subgraph of $G$ induced by all vertices in bags of $T_a$. 

---

Petr Kolman

Extension Complexity, MSO Logic, and Treewidth
### Idea

For a formula $\varphi$, a graph $G$ and a nice tree decomposition $T$ of $G$

- for every node $a$ of $T$ define a **small polytope** representing assignments of the bag vertices to the sets (i.e., free variables) that are extendible to a feasible assignment of all vertices in $G_a$
- process the tree $T$ in a **bottom up** fashion and **glue** the small polytopes by Gluing lemma
- as every polytope is small, by Gluing lemma the polytope in the root of $T$ is of size $O(n)$
- show that $P_\varphi(G)$ is a **projection** of the polytope in the root of $T$
A $[m]$-colored graph is a pair $(G, \vec{V})$ where $G = (V, E)$, $\vec{V} = (V_1, \ldots, V_m)$ and $V_i \subseteq V$. 

The diagram illustrates a graph with vertices colored in green, red, and blue, with the vertices in each color subset forming the $V_i$ sets.
A [m]-colored \( \tau \)-boundaried graph is a triple \((G, \vec{V}, \vec{p})\) where \((G, \vec{V})\) is an [m]-colored graph and \(\vec{p} = (p_1, \ldots, p_\tau)\) is a \(\tau\)-tuple of vertices of \(G\).
Compatible Graphs

$(G_1, \vec{U}, \vec{p})$ and $(G_2, \vec{W}, \vec{q})$ are compatible if subgraphs $G_1[\vec{p}]$ and $G_2[\vec{q}]$ are identical and colored the same way.
Join of compatible graphs \((G_1, \vec{U}, \vec{p})\) and \((G_2, \vec{W}, \vec{q})\) is \([m]\)-colored \(\tau\)-boundaried graph ...
Equivalence and Types of Graphs

**MSO\([k, \tau, m]\)**

All MSOL formulae over \([m]\)-colored \(\tau\)-boundaried graphs with \(qr \leq k\)

**Equivalence \(\equiv_{MSO}^k\)**

Two \([m]\)-colored \(\tau\)-boundaried graphs \(G_1^{[m]},\tau\) and \(G_2^{[m]},\tau\) are MSO\([k]\)-equivalent if they satisfy the same MSO\([k, \tau, m]\) formulae.

**Theorem (Libkin, 2004, implicitly in Courcelle, 1990)**

For any fixed \(\tau, k, m \in \mathbb{N}\), the equivalence relation \(\equiv_{MSO}^k\) has a finite number of equivalence classes.

We denote the equivalence classes by \(C = \{\alpha_1, \ldots, \alpha_w\}\), fixing an ordering such that \(\alpha_1\) is the class containing the empty graph.

**Type of a graph**

Given an \([m]\)-colored \(\tau\)-boundaried graph, its type (w.r.t. \(\equiv_{MSO}^k\)) is the class to which it belongs.
Equivalence and Types of Graphs

**MSO**[$k, \tau, m$]

All MSOL formulae over $[m]$-colored $\tau$-boundaried graphs with $qr \leq k$

**Equivalence** $\equiv^{MSO}_k$

Two $[m]$-colored $\tau$-boundaried graphs $G_1^{[m], \tau}$ and $G_2^{[m], \tau}$ are **MSO**[$k$]-equivalent if they satisfy the same **MSO**[$k, \tau, m$] formulae.

**Theorem** (Libkin, 2004, implicitly in Courcelle, 1990)

For any fixed $\tau, k, m \in \mathbb{N}$, the equivalence relation $\equiv^{MSO}_k$ has a finite number of equivalence classes.

We denote the equivalence classes by $C = \{\alpha_1 \ldots, \alpha_w\}$, fixing an ordering such that $\alpha_1$ is the class containing the empty graph.

**Type of a graph**

Given an $[m]$-colored $\tau$-boundaried graph, its type (w.r.t. $\equiv^{MSO}_k$) is the class to which it belongs.
Equivalence and Types of Graphs

**MSO**$[k, \tau, m]$

all MSOL formulae over $[m]$-colored $\tau$-boundaried graphs with $qr \leq k$

**Equivalence** $\equiv^{MSO}_k$

Two $[m]$-colored $\tau$-boundaried graphs $G_1^{[m], \tau}$ and $G_2^{[m], \tau}$ are **MSO**$[k]$-equivalent if they satisfy the same **MSO**$[k, \tau, m]$ formulae.

**Theorem (Libkin, 2004, implicitly in Courcelle, 1990)**

*For any fixed $\tau, k, m \in \mathbb{N}$, the equivalence relation $\equiv^{MSO}_k$ has a finite number of equivalence classes.*

We denote the equivalence classes by $C = \{\alpha_1 \ldots, \alpha_w\}$, fixing an ordering such that $\alpha_1$ is the class containing the empty graph.

**Type of a graph**

Given an $[m]$-colored $\tau$-boundaried graph, its type (w.r.t. $\equiv^{MSO}_k$) is the class to which it belongs.
Equivalence and Types of Graphs

**MSO\([k, \tau, m]\)**

all MSOL formulae over \([m]\)-colored \(\tau\)-boundaried graphs with \(qr \leq k\)

**Equivalence \(\equiv_k^{MSO}\)**

Two \([m]\)-colored \(\tau\)-boundaried graphs \(G_{1[m],\tau}\) and \(G_{2[m],\tau}\) are MSO\([k]\)-equivalent if they satisfy the same MSO\([k, \tau, m]\) formulae.

**Theorem (Libkin, 2004, implicitly in Courcelle, 1990)**

For any fixed \(\tau, k, m \in \mathbb{N}\), the equivalence relation \(\equiv_k^{MSO}\) has a finite number of equivalence classes.

We denote the equivalence classes by \(C = \{\alpha_1, \ldots, \alpha_w\}\), fixing an ordering such that \(\alpha_1\) is the class containing the empty graph.

**Type of a graph**

Given an \([m]\)-colored \(\tau\)-boundaried graph, its type (w.r.t. \(\equiv_k^{MSO}\)) is the class to which it belongs.
Lemma (Libkin, 2004)

If $G_a^{[m],\tau} \equiv_{MSO}^k G_a'^{[m],\tau}$ and $G_b^{[m],\tau} \equiv_{MSO}^k G_b'^{[m],\tau}$, then

$$(G_a^{[m],\tau} \oplus G_b^{[m],\tau}) \equiv_{MSO}^k (G_a'^{[m],\tau} \oplus G_b'^{[m],\tau}).$$

Meaning

The type of a join of two $[m]$-colored $\tau$-boundaried graphs is determined by only a small amount of information about the two graphs, namely their types.
### Feasible Types of Tree Decomposition Nodes

#### Feasible type of a node $b \in V(T)$
- every $\alpha \in C$ such that there exist $X_1, \ldots, X_m \subseteq V(G_b)$: $(G_b, \tilde{X}, B(b))$ is of type $\alpha$ where $\tilde{X} = (X_1, \ldots, X_m)$
- Notation: $\mathcal{F}(b)$ - the set of feasible types of the node $b$ where every type is represented by a binary vector $t_b \in \{0, 1\}^{|C|}$

#### Feasible triple of types for a join node $c$ with children $a, b$
- every triple $(\gamma_1, \gamma_2, \alpha)$ such that
  - $\alpha \in \mathcal{F}(c)$, $\gamma_1 \in \mathcal{F}(a)$ and $\gamma_2 \in \mathcal{F}(b)$, and
  - $\gamma_1$, $\gamma_2$ and $\alpha$ are mutually compatible,
  - and there exist $\tilde{X}^1$, $\tilde{X}^2$ realizing $\gamma_1$ and $\gamma_2$ on $a$ and $b$, such that $\tilde{X} = (X_1^1 \cup X_1^2, \ldots, X_m^1 \cup X_m^2)$ realizes $\alpha$ on $c$.
- Notation: $\mathcal{F}_t(c)$ - the set of feasible triples of types of the join node $c$.

#### Feasible pairs of types for a forget and introduce node $c$
- analogously ... $\mathcal{F}_p(c)$
Feasible Types of Tree Decomposition Nodes

Feasible type of a node \( b \in V(T) \)

- every \( \alpha \in C \) such that there exist \( X_1, \ldots, X_m \subseteq V(G_b) \):
  \((G_b, \tilde{X}, B(b))\) is of type \( \alpha \) where \( \tilde{X} = (X_1, \ldots, X_m) \)
- Notation: \( \mathcal{F}(b) \) - the set of feasible types of the node \( b \) where every type is represented by a binary vector \( t_b \in \{0, 1\}^{|C|} \)

Feasible triple of types for a join node \( c \) with children \( a, b \)

- every triple \((\gamma_1, \gamma_2, \alpha)\) such that
  - \( \alpha \in \mathcal{F}(c) \), \( \gamma_1 \in \mathcal{F}(a) \) and \( \gamma_2 \in \mathcal{F}(b) \), and
  - \( \gamma_1, \gamma_2 \) and \( \alpha \) are mutually compatible,
  - and there exist \( \tilde{X}^1, \tilde{X}^2 \) realizing \( \gamma_1 \) and \( \gamma_2 \) on \( a \) and \( b \), such that \( \tilde{X} = (X_1^1 \cup X_1^2, \ldots, X_m^1 \cup X_m^2) \) realizes \( \alpha \) on \( c \).
- Notation: \( \mathcal{F}_t(c) \) - the set of feasible triples of types of the join node \( c \).

Feasible pairs of types for a forget and introduce node \( c \)

analogously \( \mathcal{F}_p(c) \)
Feasible Types of Tree Decomposition Nodes

**Feasible type of a node** \( b \in V(T) \)
- every \( \alpha \in C \) such that there exist \( X_1, \ldots, X_m \subseteq V(G_b) \): \((G_b, \bar{X}, B(b))\) is of type \( \alpha \) where \( \bar{X} = (X_1, \ldots, X_m) \)
- Notation: \( \mathcal{F}(b) \) - the set of feasible types of the node \( b \) where every type is represented by a binary vector \( t_b \in \{0, 1\}^{|C|} \)

**Feasible triple of types for a join node** \( c \) with children \( a, b \)
- every triple \( (\gamma_1, \gamma_2, \alpha) \) such that
  - \( \alpha \in \mathcal{F}(c) \), \( \gamma_1 \in \mathcal{F}(a) \) and \( \gamma_2 \in \mathcal{F}(b) \), and
  - \( \gamma_1, \gamma_2 \) and \( \alpha \) are mutually compatible,
  - and there exist \( \bar{X}^1, \bar{X}^2 \) realizing \( \gamma_1 \) and \( \gamma_2 \) on \( a \) and \( b \), such that \( \bar{X} = (X_1^1 \cup X_2^1, \ldots, X_m^1 \cup X_m^2) \) realizes \( \alpha \) on \( c \).
- Notation: \( \mathcal{F}_t(c) \) - the set of feasible triples of types of the join node \( c \).

**Feasible pairs of types for a forget and introduce node** \( c \)
- analogously ... \( \mathcal{F}_p(c) \)
The Construction

The basic polytopes

- $b$ is a leaf:
  $$P_b = \{100 \ldots 0\}$$
- $b$ is an introduce or forget node:
  $$P_b = \text{conv} (\mathcal{F}_p(b))$$
- $b$ is a join node:
  $$P_b = \text{conv} (\mathcal{F}_t(b))$$

Lemma

*Extension complexity of the polytopes $P_b$’s is independent on $n$.*

Proof: The sizes of the sets $\mathcal{F}(a)$, $\mathcal{F}_p(a)$, $\mathcal{F}_t(a)$ are independent on $n$.

Gluing them into larger polytopes

- $b$ is a leaf:
  $$Q_b = P_b$$
- $b$ is an introduce or forget node:
  $$Q_b = Q_a \times |C| P_b$$
  where $a$ is the child of $b$ and the gluing is done along the coordinates $t_a$ in $Q_a$ and $d_b$ in $P_b$.
- $b$ is a join node:
  $$Q_b = Q_a \times |C| P_b \times |C| Q_c$$
  where ....
### The Construction

#### The basic polytopes

<table>
<thead>
<tr>
<th>$b$ is a <strong>leaf</strong>:</th>
<th>$P_b = {100 \ldots 0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$ is an <strong>introduce</strong> or <strong>forget</strong> node:</td>
<td>$P_b = \text{conv}(\mathcal{F}_p(b))$</td>
</tr>
<tr>
<td>$b$ is a <strong>join</strong> node:</td>
<td>$P_b = \text{conv}(\mathcal{F}_t(b))$</td>
</tr>
</tbody>
</table>

#### Lemma

*Extension complexity of the polytopes $P_b$'s is independent on $n$.*

**Proof:** The sizes of the sets $\mathcal{F}(a)$, $\mathcal{F}_p(a)$, $\mathcal{F}_t(a)$ are independent on $n$.

#### Gluing them into larger polytopes

<table>
<thead>
<tr>
<th>$b$ is a <strong>leaf</strong>:</th>
<th>$Q_b = P_b$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$ is an <strong>introduce</strong> or <strong>forget</strong> node. where $a$ is the child of $b$ and the gluing is done along the coordinates $t_a$ in $Q_a$ and $d_b$ in $P_b$.</td>
<td>$Q_b = Q_a \times</td>
</tr>
<tr>
<td>$b$ is a <strong>join</strong> node.</td>
<td>$Q_b = Q_a \times</td>
</tr>
</tbody>
</table>
**Lemma**

For every node \( b \in V(T) \) and every vertex \( y \) of the polytope \( Q_b \) there exist \( X_1, \ldots, X_m \subseteq V(G_b) \) such that \( (G_b, (X_1, \ldots, X_m), \sigma(B(b))) \) is of the type specified by the vector \( y \).

Proof. By induction and previous Lemma.

\[
Q_r
\]

\[
y = (0, 0, \ldots, 0, 1, 0, \ldots, )
\]

Applying Lemma to the root of the decomposition tree and a few more steps completes the proof of the main theorem.
Worth noting

- the extension complexity linear in the size of $G$
- optimization easy (LinEMSOL)
- the constructed polytope almost universal: apart from the last step (skipped), the construction depends on the quantifier rank of the formula only, not on the formula itself
Thank you!