Parameterized Complexity of Integer Linear Programming (ILP)

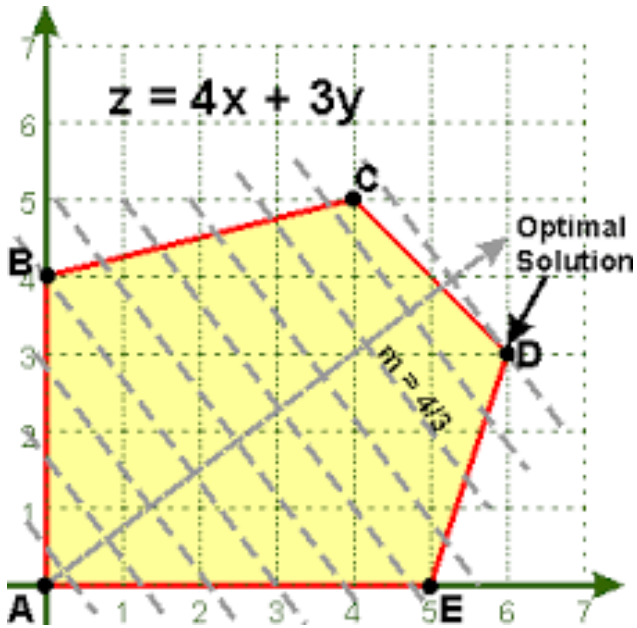
Sebastian Ordyniak

The University Of Sheffield.

Parameterized Graph Algorithms & Data Reduction
(Shonan Meeting 144, 2019)

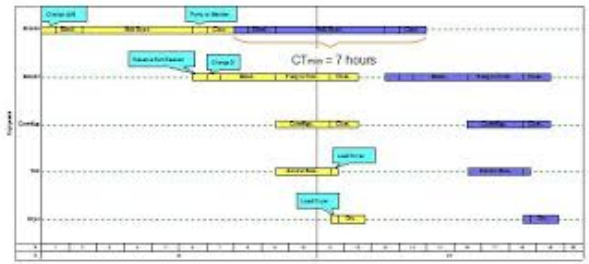
Integer Linear Programming (ILP)

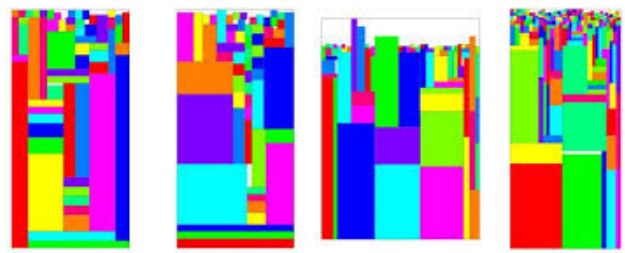
- archetypical problem for **NP**-complete optimization problems
- very general and successful paradigm for solving intractable optimization problems in practice

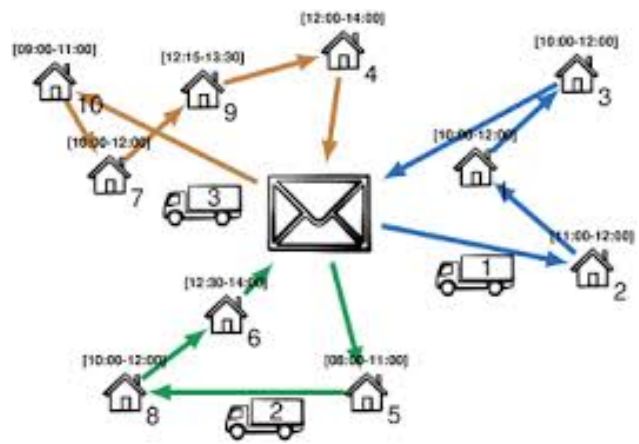


Applications

- process scheduling
- planning
- vehicle routing
- packing
- ...

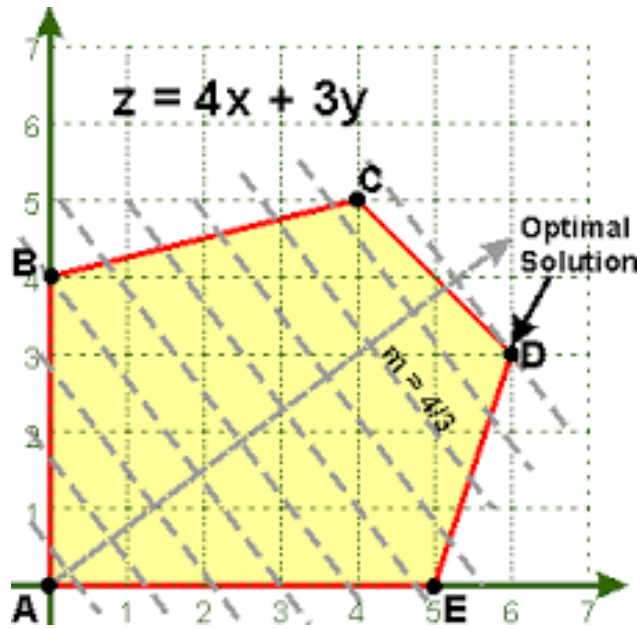






Problem Formulation: ILP

maximize	$\mathbf{c} \cdot \mathbf{x}$
subject to	$A\mathbf{x} \leq \mathbf{b}$
	$\mathbf{x} \in \mathbb{Z}^n$

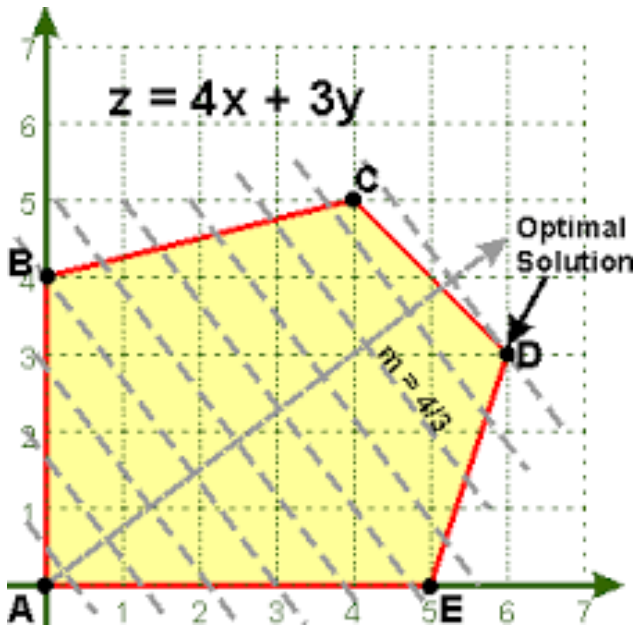


(where $A \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$, and $\mathbf{c} \in \mathbb{Z}^n$)

Problem Formulation: ILP

$$\begin{array}{ll} \text{maximize} & \mathbf{c} \cdot \mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \in \mathbb{Z}^n \end{array}$$

$$\begin{array}{ll} \text{maximize} & \mathbf{c} \cdot \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}; \quad \mathbf{x} \in \mathbb{Z}^n \end{array}$$



(where $A \in \mathbb{Z}^{m \times n}$, $\mathbf{b} \in \mathbb{Z}^m$, and $\mathbf{c} \in \mathbb{Z}^n$)

ILP: Example

$$\text{maximize } \sum_{1 \le i \le n} c_i x_i$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}$$

ILP: Example

$$\text{maximize } \sum_{1 \le i \le n} c_i x_i$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}$$

columns $\approx$ variables

ILP: Example

$$\text{maximize } \sum_{1 \le i \le n} c_i x_i$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}$$

rows $\approx$ constraints

ILP: Example

$$\text{maximize } \sum_{1 \le i \le n} c_i x_i$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}$$

$\ell_A \approx$ the maximum coefficient in A

ILP: Example

$$\text{maximize } \sum_{1 \le i \le n} c_i x_i$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}$$

maximization function; (maximum value $\approx$ maximum value of the maximization function for any feasible assignment)

ILP: Example

$$\text{maximize } \sum_{1 \leq i \leq n} x_i$$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{pmatrix}$$

without optimization function, we talk about ILP-feasibility

State-of-the-art

ILP and ILP-feasibility are NP-complete and until recently only very few tractable cases have been known:

- **totally unimodular** matrices (Papadimitriou, Steiglitz 1982),
- fixed number of variables (Lenstra 1983),

State-of-the-art: Block Matrices

Recently, various tractable classes based on **block matrices** have been introduced:

- n-fold, 2-stage stochastic, and 4-block N-fold ILP with fixed sized blocks and max coefficient (Hemmecke et al., 2010 and 2013;De Loera et al., 2013),
- tree-fold and multi-stage stochastic ILPs (Chen and Marx, 2018; Aschenbrenner and Hemmecke 2007)

State-of-the-art: Structural Restrictions

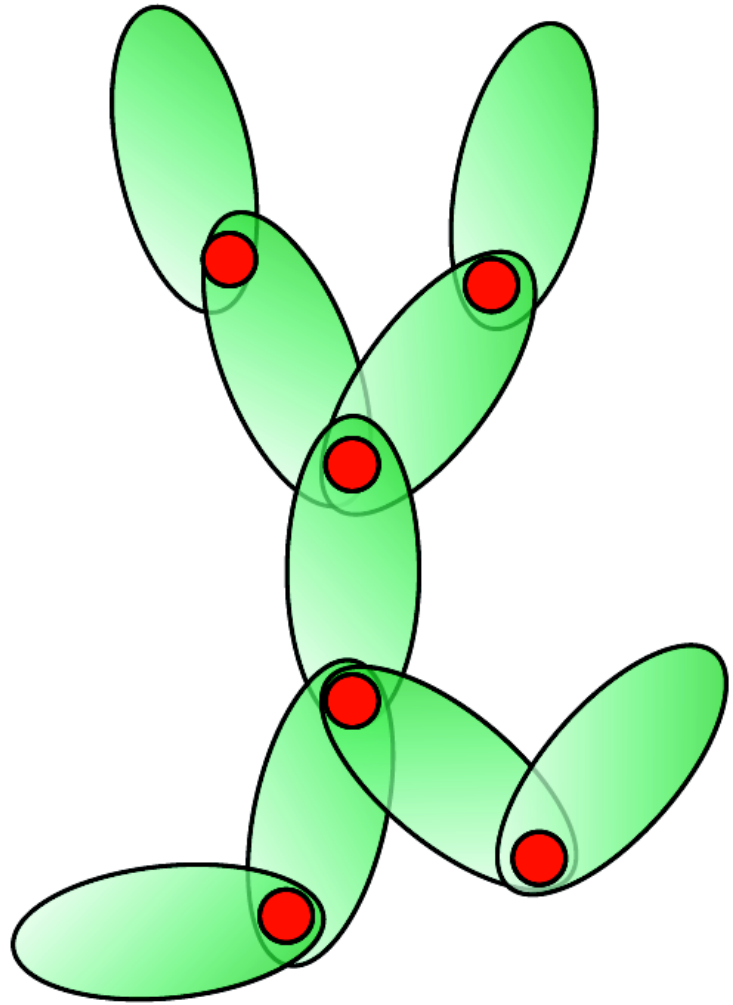
In parallel, various tractable classes based on restrictions on graphical representations of the constraint matrix have been introduced.

Namely, similar to SAT and CSP the following three graphical representations have been considered:

- **primal graph**,
- **dual graph**,
- **incidence graph**

Structural Parameters

- **fracture number**,
- **treedepth**,
- **treewidth**,
- clique-width,
- rank-width



Block Matrices vs. Structural Parameters

- interestingly all tractable fragments defined via block matrices can be defined in terms of structural restrictions . . .
- . . . while the reverse does not hold,
- the fragments obtained using structural restrictions are usually more natural/flexible and also allow the simple recognition and computation of the parameters,

Main Novel Tractable Classes (Using Blockmatrices)

Block Matrices vs. Fracture Number

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^N = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{B} & \cdots & \mathbf{B} \\ \mathbf{C} & \mathbf{D} & 0 & \cdots & 0 \\ \mathbf{C} & 0 & \mathbf{D} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C} & 0 & 0 & \cdots & \mathbf{D} \end{pmatrix}$$

Block Matrices vs. Fracture Number

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^N = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{B} & \cdots & \mathbf{B} \\ \mathbf{C} & \mathbf{D} & 0 & \cdots & 0 \\ \mathbf{C} & 0 & \mathbf{D} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C} & 0 & 0 & \cdots & \mathbf{D} \end{pmatrix}$$

4-block n-fold $\sim$ fracture number

Theorem (Hemmecke et al., 2010)

ILP is **XP** parameterized by ℓ_A and the max. number of rows/columns in $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$.

Block Matrices vs. Fracture Number

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^N = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{B} & \cdots & \mathbf{B} \\ \mathbf{C} & \mathbf{D} & 0 & \cdots & 0 \\ \mathbf{C} & 0 & \mathbf{D} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C} & 0 & 0 & \cdots & \mathbf{D} \end{pmatrix}$$

n-fold $\sim$ constraint fracture number

Theorem (De Loera et al., 2013)

ILP is **FPT** parameterized by ℓ_A and the max. number of rows/columns in $\mathbf{B}, \mathbf{D}$.

Block Matrices vs. Fracture Number

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^N = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{B} & \cdots & \mathbf{B} \\ \mathbf{C} & \mathbf{D} & 0 & \cdots & 0 \\ \mathbf{C} & 0 & \mathbf{D} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C} & 0 & 0 & \cdots & \mathbf{D} \end{pmatrix}$$

2-stage stochastic $\sim$ variable fracture number

Theorem (Hemmecke et al., 2013)

ILP is **FPT** parameterized by ℓ_A and the max. number of rows/columns in $\mathbf{C}, \mathbf{D}$.

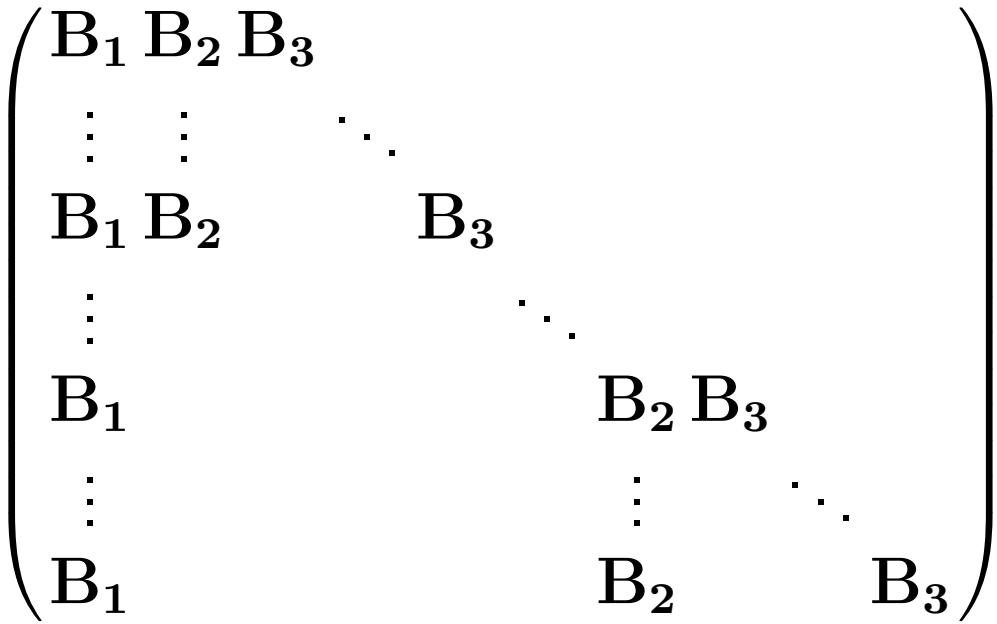
Block Matrices vs. Fracture Number

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^N = \begin{pmatrix} \mathbf{A} & \mathbf{B} & \mathbf{B} & \cdots & \mathbf{B} \\ \mathbf{C} & \mathbf{D} & 0 & \cdots & 0 \\ \mathbf{C} & 0 & \mathbf{D} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{C} & 0 & 0 & \cdots & \mathbf{D} \end{pmatrix}$$

Essentially, these are ILPs with few global variables and/or global constraints that interact uniformly with the rest.

- Several applications (e.g. for scheduling, social choice, closest string etc.).

Tree-fold and Multi-stage Stochastic ILPs



Multi-stage Stochastic $\sim$ **treedepth of primal graph**

Theorem (Koutechy, Levin, Onn (2018))

Multi-stage Stochastic ILP is **FPT** parameterized by ℓ_A, the number of rows of B_i, and the total number of columns of $B_1, \ldots, B_l$.

Tree-fold and Multi-stage Stochastic ILPs

$$\begin{pmatrix} \mathbf{B_1} & \cdots & \mathbf{B_1} & \cdots & \mathbf{B_1} & \cdots & \mathbf{B_1} \\ \mathbf{B_2} & \cdots & \mathbf{B_2} & & & & \\ \mathbf{B_3} & & & & & & \\ & \ddots & & & & & \\ & & \mathbf{B_3} & & & & \\ & & & \ddots & & & \\ & & & & \mathbf{B_2} & \cdots & \mathbf{B_2} \\ & & & & \mathbf{B_3} & & \\ & & & & & \ddots & \\ & & & & & & \mathbf{B_3} \end{pmatrix}$$

tree-fold $\sim$ **treedepth of dual graph**

Theorem (Koutechy, Levin, Onn (2018))

Tree-fold ILP is **FPT** parameterized by ℓ_A, the number of columns of B_i, and the total number of rows of $B_1, \ldots, B_l$.

Recent Applications

Scheduling:

- Scheduling Meets n-Fold Integer Programming (Knop, Koutechy 2018),
- Empowering the Configuration-IP – New PTAS Results for Scheduling with Setup Times (Jansen *et al.* 2019)

Social Choice and Combinarial Optimization (e.g. Closest String):

- Combinatorial n-Fold Integer Programming and Applications (Knop, Koutechy, Mnich 2017),
- Voting and Bribing in Single-Exponential Time (Knop, Koutechy, Mnich 2017)

Travelling Salesman Problem:

- Covering a Tree with rooted subtrees Parameterized Approximation Algorithms (Chen, Marx 2018)

Primal Graph: An illustrative Example

Primal Graph

$$A := \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & * \end{pmatrix}$$

The **primal graph** of an ILP instance $\mathcal{I}$, denoted by $\mathsf{P}(\mathcal{I})$, has:

- one vertex for every variable of $\mathcal{I}$,
- an edge between two variables x and y iff x and y occur together in a constraint of $\mathcal{I}$.

Primal Graph: State-of-the-Art

Using structural restrictions of the primal graph leads to two main fixed-parameter tractable cases for ILP:

- treewidth and domain,
- treedepth and ℓ_A (a.k.a. multi-stage stochastic ILP).

Primal Graph: State-of-the-Art

Using structural restrictions of the primal graph leads to two main fixed-parameter tractable cases for ILP:

- treewidth and domain,
- treedepth and ℓ_A (a.k.a. multi-stage stochastic ILP).

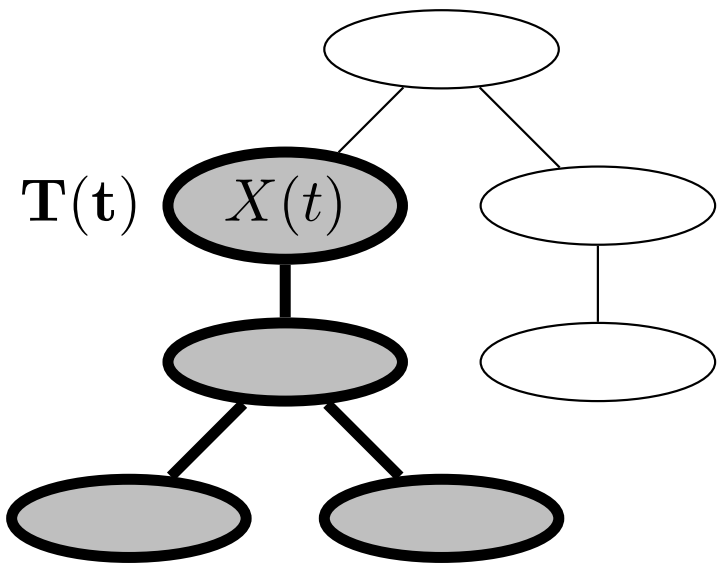
All other combinations can be shown to be **para-NP**-hard.

An Algorithm using Treewidth and Domain

Theorem (Jansen and Kratsch, 2015)

ILP is fixed-parameter tractable parameterized by treewidth and the maximum absolute domain value D of any variable ($\mathcal{O}((2D+1)^{\mathsf{tw}}|\mathcal{I}|)$.

- dynamic programming algorithm on a tree decomposition of the primal graph,
- For each bag of the tree decomposition stores which of the at most $(2D+1)^{\mathsf{tw}}$ many assignments of the variables in the bag can be extended to a feasible assignment for the subinstance represented by the current subtree.



An Algorithm using Treedepth

Theorem (Koutechy, Levin, Onn (2018))

ILP is fixed-parameter tractable parameterized by the treedepth of the primal graph and ℓ_A.

Multi-stage Stochastic ILPs

Given:

- A tree T of height ω with root r and
- ω integer matrices $B_1, \ldots, B_\omega$ with l rows and $n_1, \ldots, n_\omega$ columns, respectively

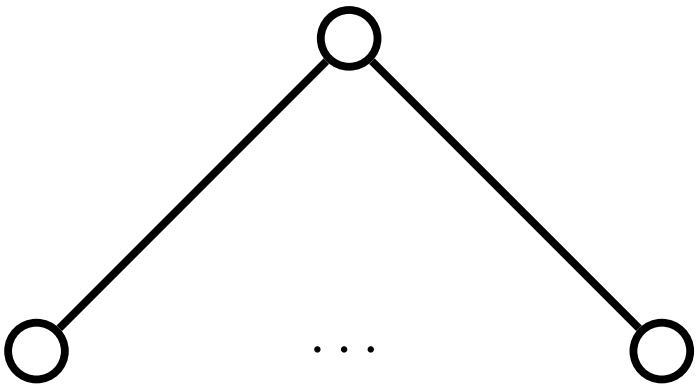
The constraint matrix A_r of a multi-stage stochastic ILP is defined inductively by setting:

- $A_l = B_\omega$ for every leaf l of T and
-

$$A_v = \begin{pmatrix} \mathbf{B_d} & \mathbf{A_{c_1}} & & \\ \mathbf{B_d} & & \mathbf{A_{c_2}} & \\ \vdots & & & \ddots \\ \mathbf{B_d} & & & & \mathbf{A_{c_n}} \end{pmatrix}$$

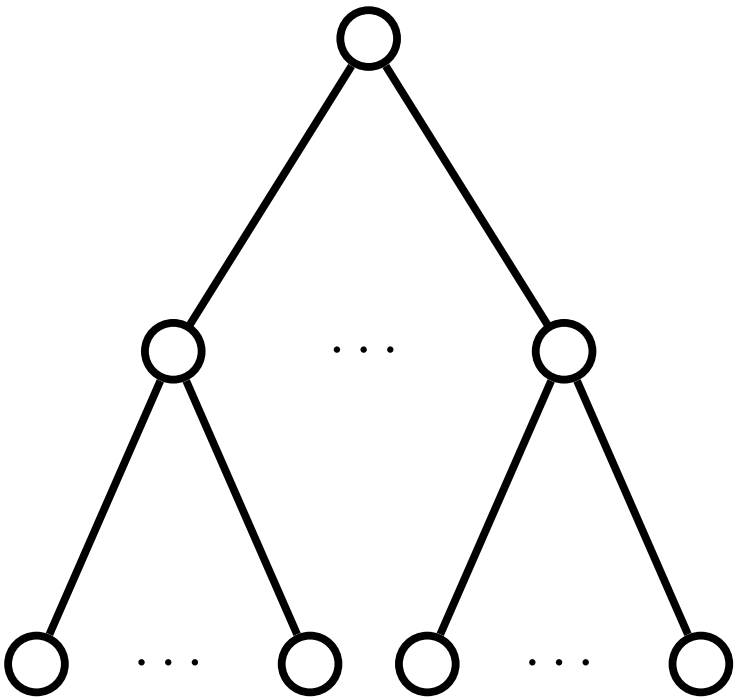
for an inner node v with depth d and children $c_1, \ldots, c_n$ in T.

Multi-stage Stochastic ILPs: Example



$$\begin{pmatrix} \mathbf{B_1} & \mathbf{B_2} & & \\ \mathbf{B_1} & & \mathbf{B_2} & \\ \vdots & & & \ddots \\ \mathbf{B_1} & & & \mathbf{B_2} \end{pmatrix}$$

Multi-stage Stochastic ILPs: Example



$$
\begin{array}{ccccccc}
\mathbf{B_1} & \mathbf{B_2} & \mathbf{B_3} & & & & \\
\vdots & \vdots & & \ddots & & & \\
\mathbf{B_1} & \mathbf{B_2} & & & \mathbf{B_3} & & \\
\vdots & & & & & \ddots & \\
\mathbf{B_1} & & & & & \mathbf{B_2} & \mathbf{B_3} \\
\vdots & & & & & \vdots & & \ddots \\
\mathbf{B_1} & & & & & \mathbf{B_2} & & \mathbf{B_3}
\end{array}
$$

Multi-stage Stochastic ILPs

Theorem (Koutecky, Levin, Onn (2018))

Multi-stage stochastic ILP is fixed-parameter tractable parameterized by ℓ_A, l, $n_1, \ldots, n_\omega$.

Remark

Fixed-parameter tractability has been known before, however, the algorithm improved upon the polynomial factor.

Treedepth

- a well-known structural parameter more restrictive than treewidth and pathwidth,
- many equivalent characterizations, e.g.:
  - **treedepth $\approx$ "length of a longest path"**,
  - treedepth $=$ cycle-rank,
  - treedepth is bounded if and only if there is a bounded width tree decomposition whose tree has bounded height

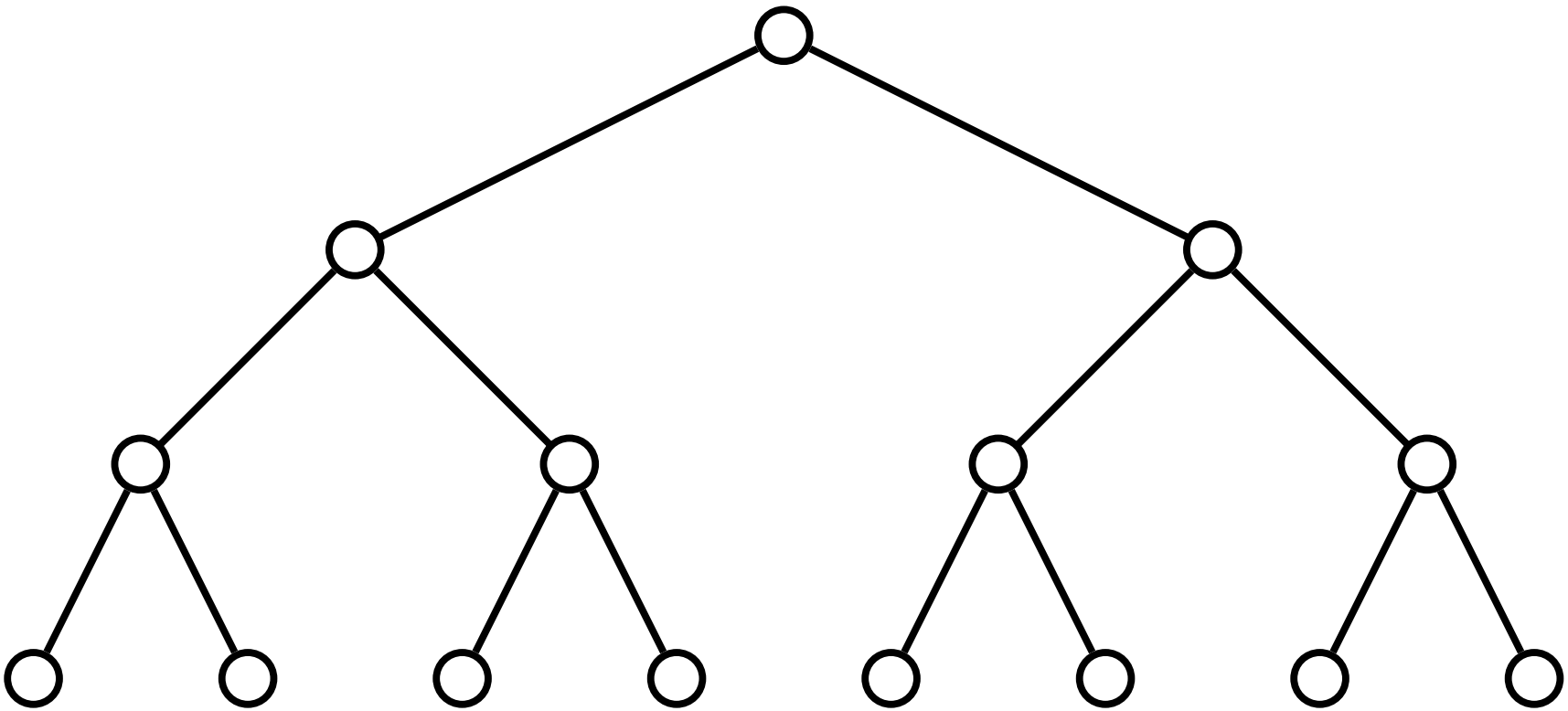
Treedepth decomposition

Definition

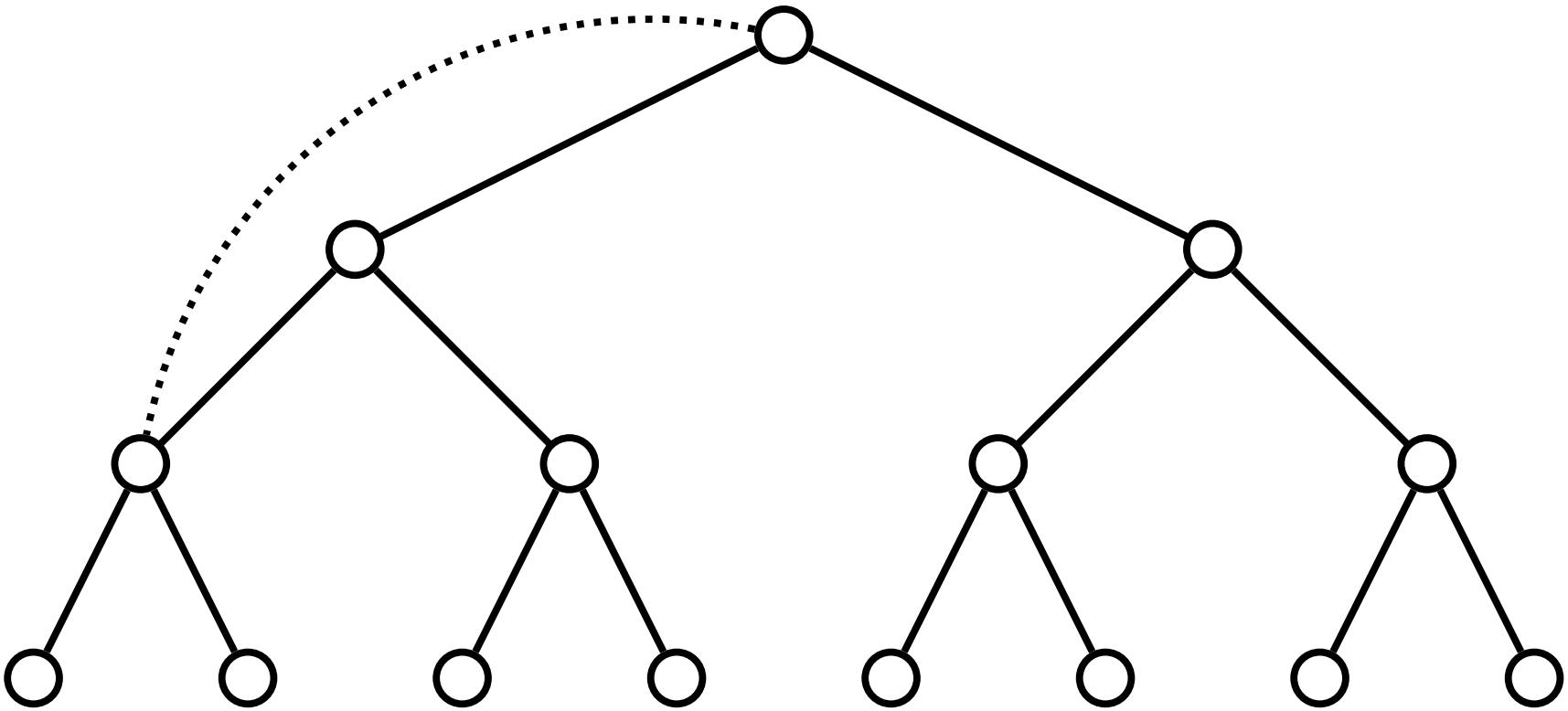
A graph G has treedepth at most k if and only if there is a rooted tree T on $V(G)$ of height at most k such that every edge in G is between ancestors and descendants of T.

The tree T is called a **treedepth decomposition** of G.

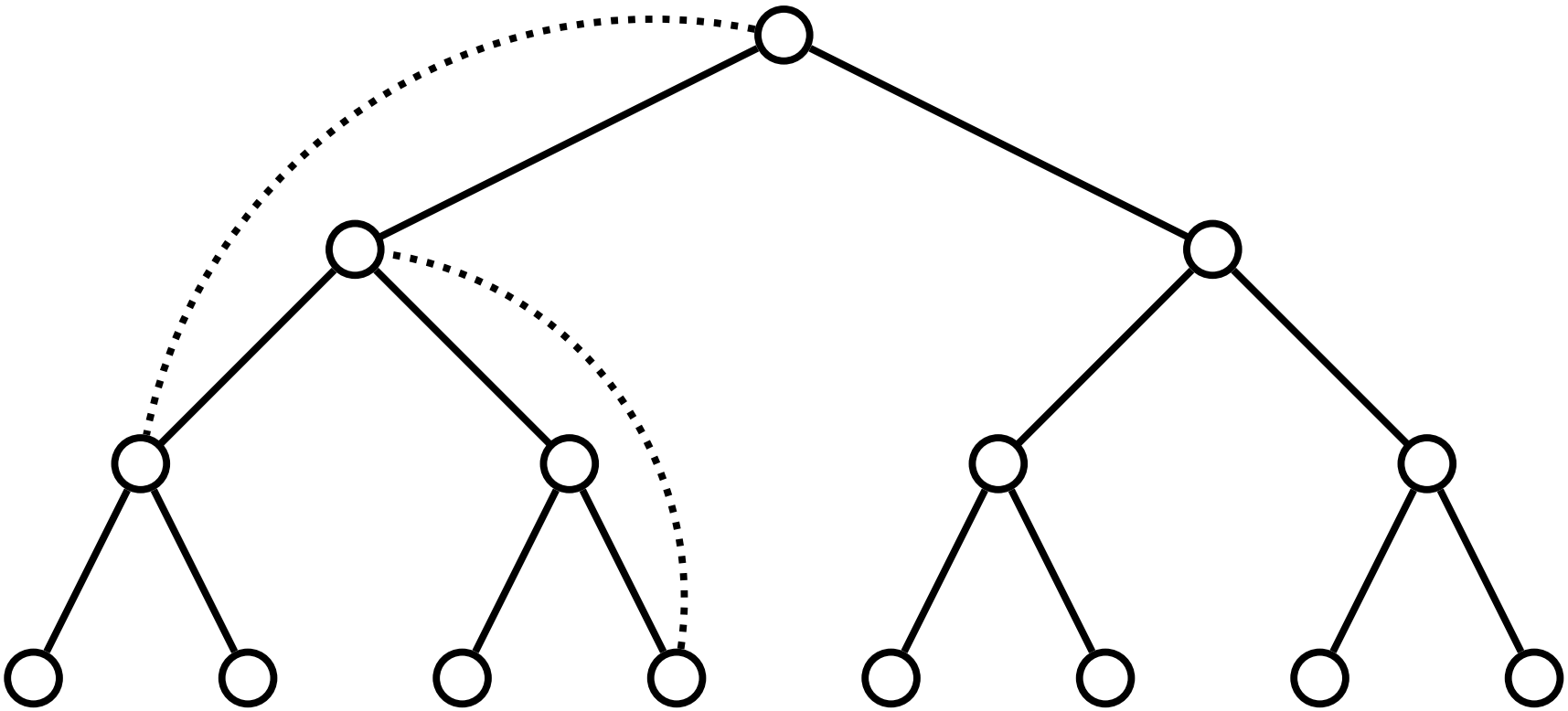
Treedepth decomposition



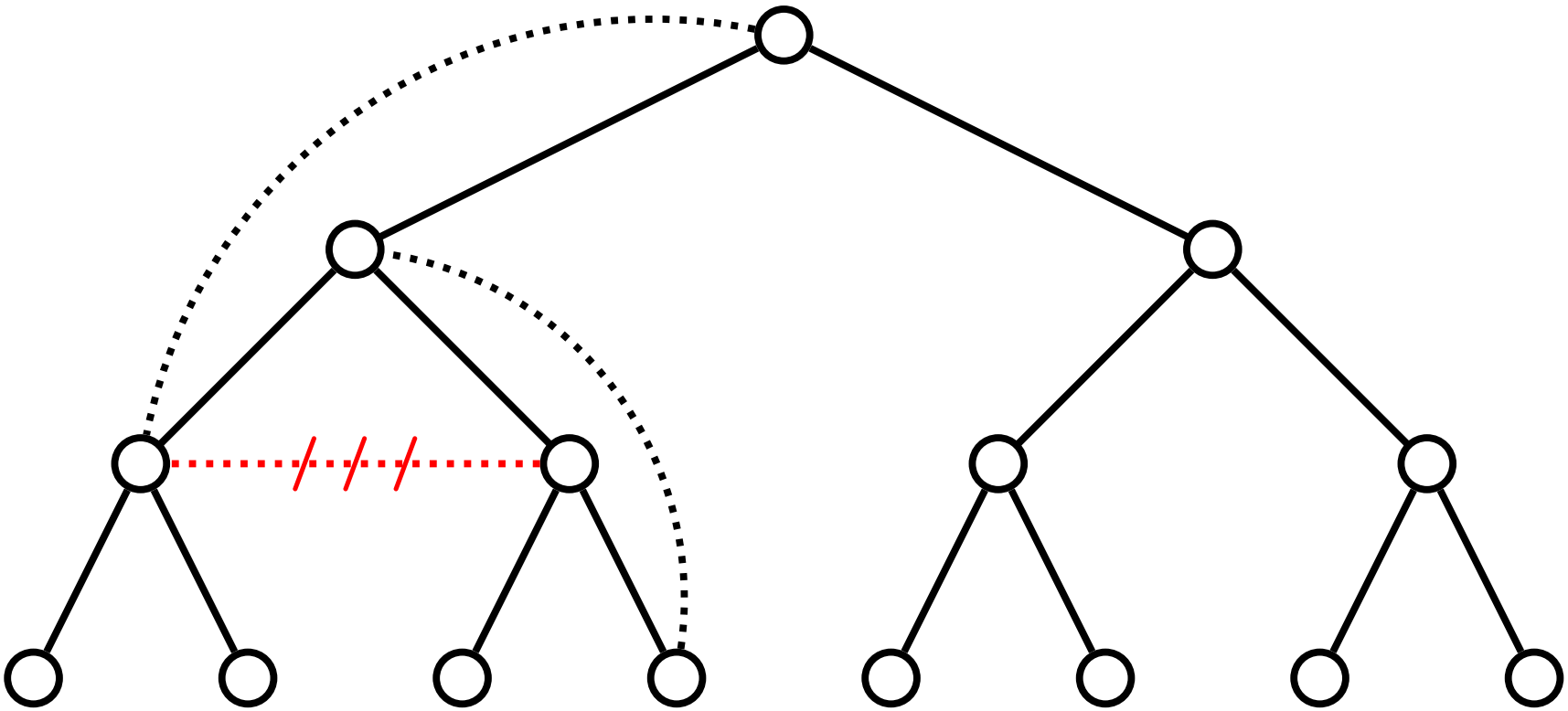
Treedepth decomposition



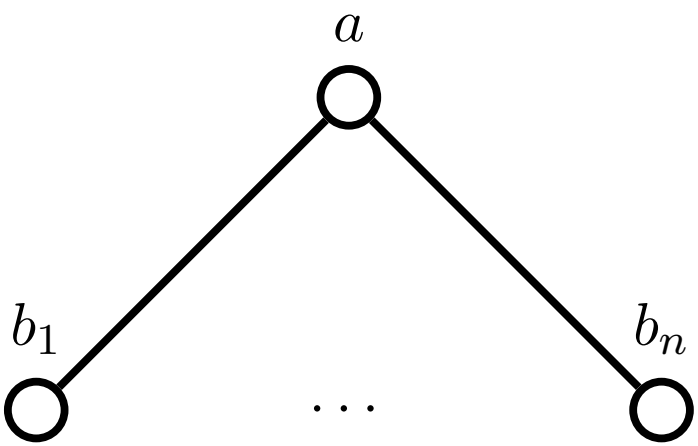
Treedepth decomposition



Treedepth decomposition

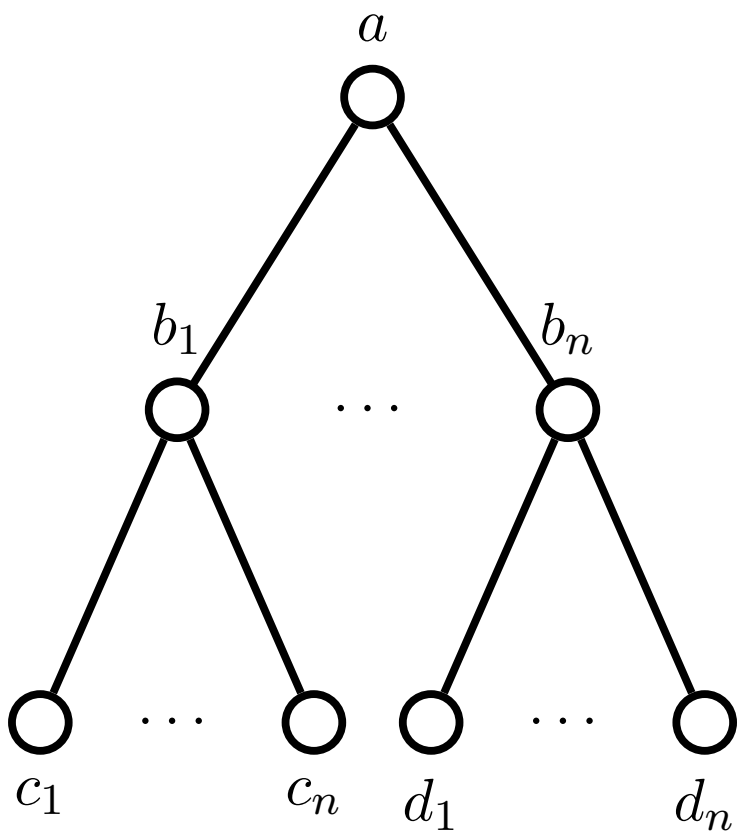


Treedepth and Matrices



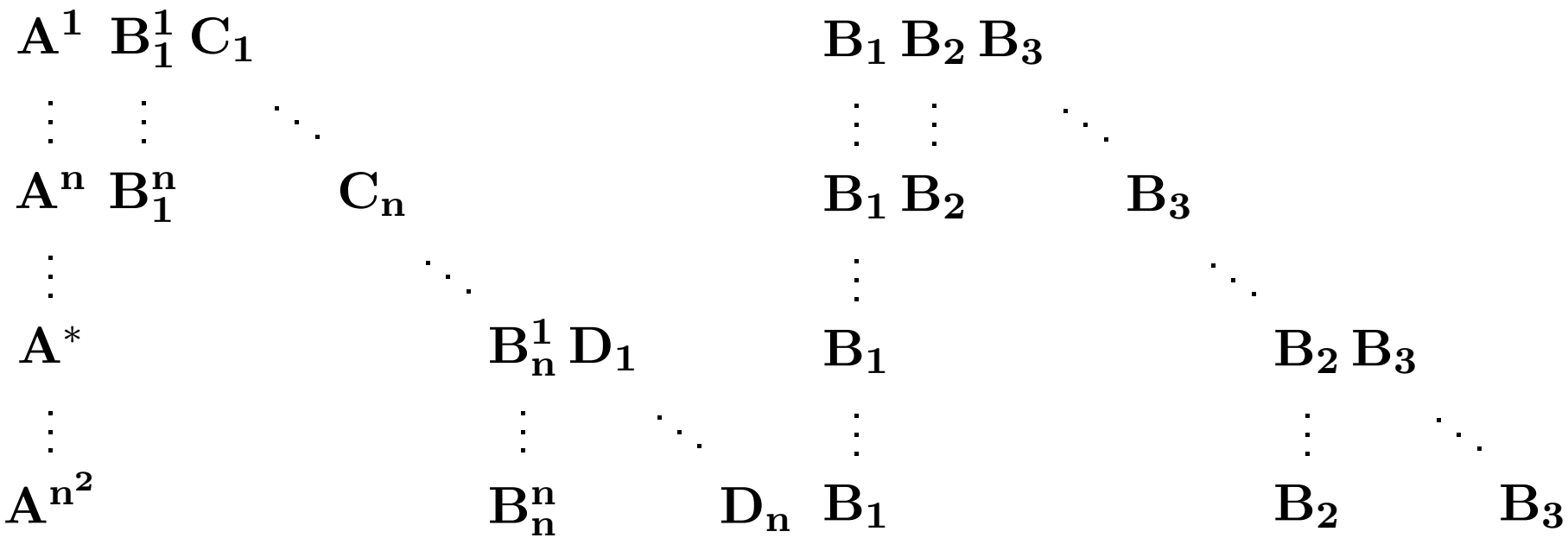
$$\begin{pmatrix} \mathbf{A^1} & \mathbf{B_1} & & \\ \mathbf{A^2} & & \mathbf{B_2} & \\ \vdots & & & \ddots \\ \mathbf{A^n} & & & \mathbf{B_n} \end{pmatrix}$$

Treedepth and Matrices

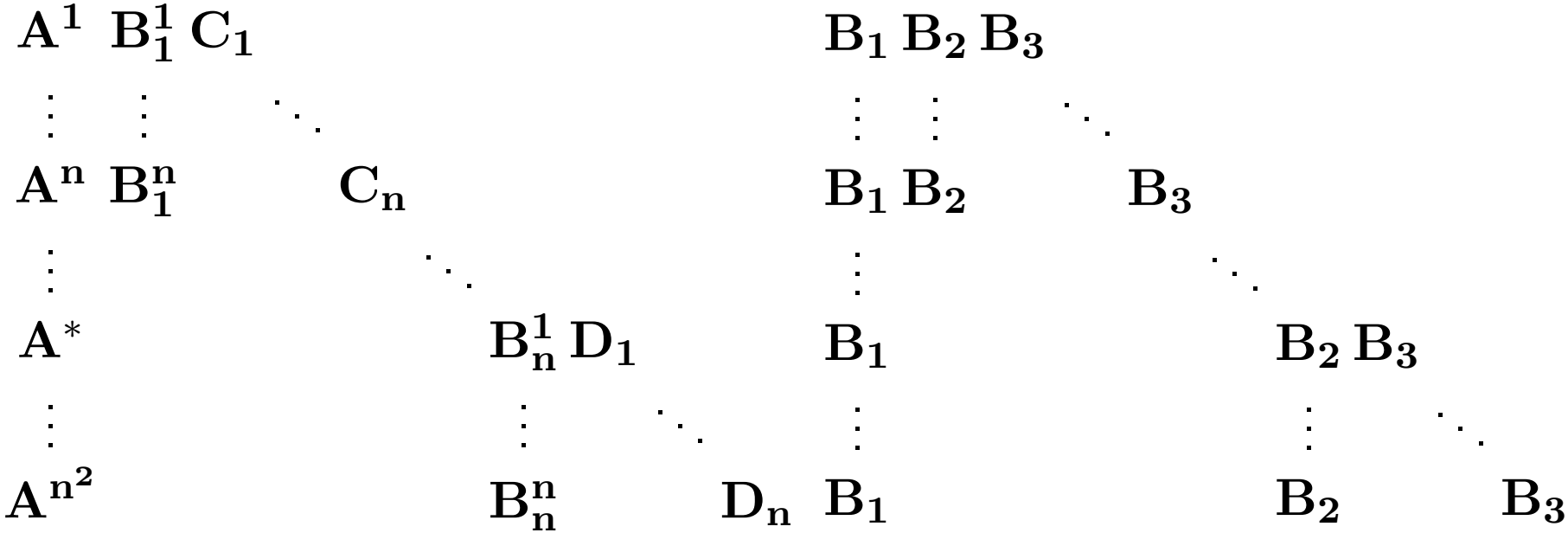


$$\begin{array}{ccccccc}
\mathbf{A^1} & \mathbf{B_1^1} & \mathbf{C_1} & & & & \\
\vdots & \vdots & & \ddots & & & \\
\mathbf{A^n} & \mathbf{B_1^n} & & & \mathbf{C_n} & & \\
\vdots & & & & & \ddots & \\
\mathbf{A^*} & & & & & \mathbf{B_n^1} & \mathbf{D_1} \\
\vdots & & & & & \vdots & & \ddots \\
\mathbf{A^{n^2}} & & & & & \mathbf{B_n^n} & & \mathbf{D_n}
\end{array}$$

From Treedepth to Multi-stage stochastic ILPs



From Treedepth to Multi-stage stochastic ILPs



Remark

Apart from using different matrices for variables at the same depth, this is almost the same as multi-stage stochastic ILPs.

From Treedepth to Multi-stage stochastic ILPs

Observation

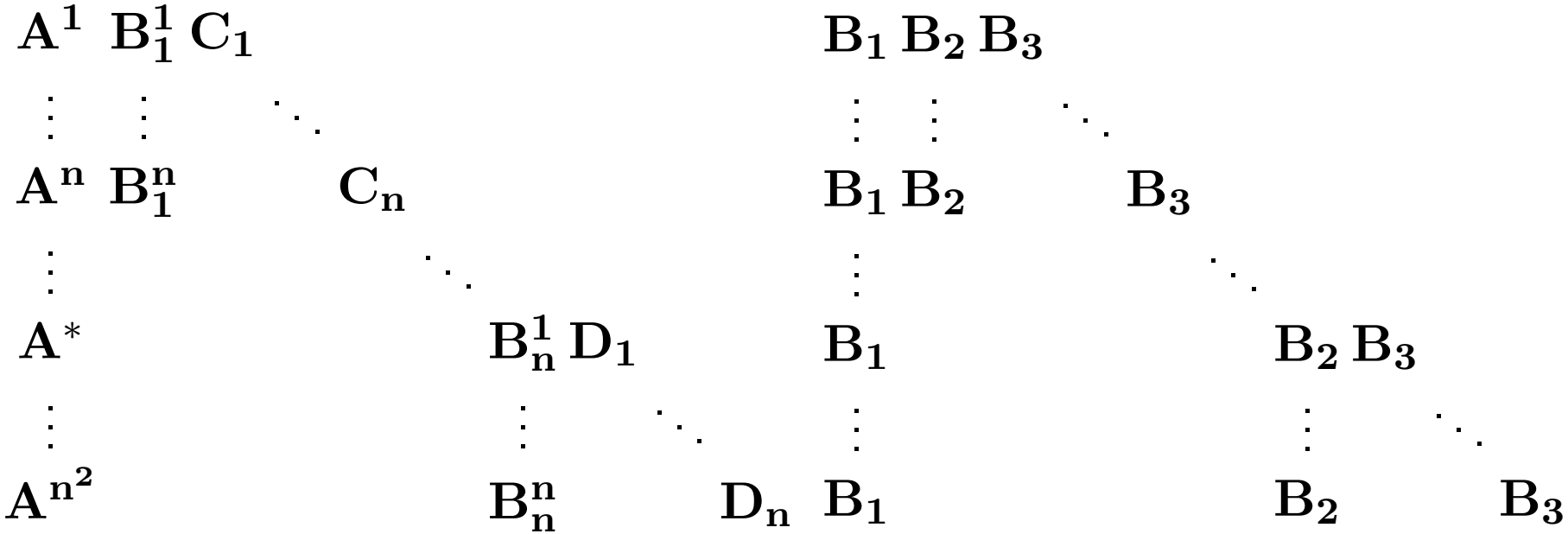
Since every root-to-leaf path in T contains at most ω variables, we obtain that:

$$(2\ell_A + 1)^\omega$$

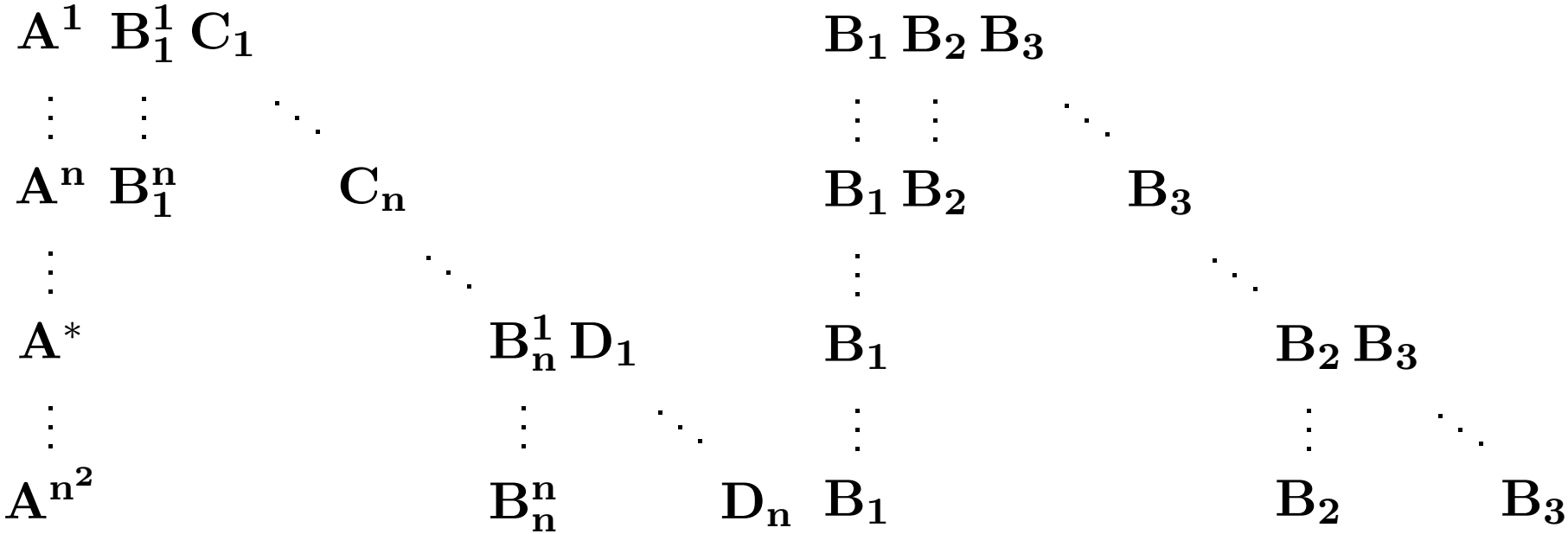
is the maximum number of constraints on the variables of such a path.

- Since we are only interested in an fpt-algorithm w.r.t. ω and ℓ_A, we can add all possible such constraints for every root to leaf path in the ILP,
- After doing so the block matrices associated to every root-to-leaf path are identical,
- (actually all block matrices become identical)

From Treedepth to Multi-stage stochastic ILPs



From Treedepth to Multi-stage stochastic ILPs



Hence, to model arbitrary ILPs with bounded treedepth, we only need to find a way to turn off constraints.

From Treedepth to Multi-stage stochastic ILPs

We can turn off constraints by adding a slack variable (occurring with coefficient 1) for every constraint:

- if the constraint needs to be turned on, we force the corresponding slack variable to be 0 by setting its lower bound and upper bound to 0,
- if the constraint needs to be turned off, we set the lower bound and upper bound of the corresponding slack variable to $-\infty$ and ∞, respectively

To achieve this, we only need to change the matrix B_ω to $(B_\omega|I)$, where I is the identity matrix with $(2\ell_A + 1)^\omega$ rows and columns.

From Treedepth to Multi-stage stochastic ILPs

$$\begin{pmatrix}
\mathbf{B_1} & \mathbf{B_2} & \mathbf{B_3}|\mathbf{I} & & & & & & \\
\vdots & \vdots & & \ddots & & & & & \\
\mathbf{B_1} & \mathbf{B_2} & & & \mathbf{B_3}|\mathbf{I} & & & & \\
\vdots & & & & & \ddots & & & \\
\mathbf{B_1} & & & & & & \mathbf{B_2} & \mathbf{B_3}|\mathbf{I} & \\
\vdots & & & & & & \vdots & & \ddots \\
\mathbf{B_1} & & & & & & \mathbf{B_2} & & \mathbf{B_3}|\mathbf{I}
\end{pmatrix}$$

From Treedepth to Multi-stage stochastic ILPs

Recall

Multi-stage stochastic ILP is fixed-parameter tractable parameterized by ℓ_A, l, $n_1, \ldots, n_\omega$.

Because:

- ℓ_A did not change,
- $l = (2\ell_A + 1)^\omega$, and
- $n_1 = \cdots = n_{\omega-1} = 1$ and $n_\omega = l + 1$

we obtain:

Theorem (Koutecky, Levin, Onn (2018))

ILP is fixed-parameter tractable parameterized by the treedepth of the primal graph (ω) and ℓ_A.

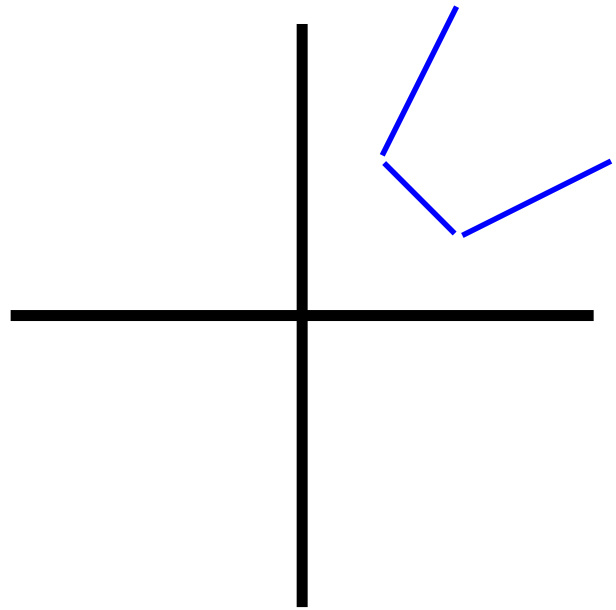
Solving Multi-stage Stochastic ILPs (or solving ILPs via Graver Best Oracles)

Graver Basis

Definition (partial order: $\sqsubseteq$)

$\mathbf{x} \sqsubseteq \mathbf{y}$ for $x, y \in \mathbb{Z}^n$ if:

- $\mathbf{x}_i \mathbf{y}_i \geq 0$ for every $i \in [n]$, i.e., $\mathbf{x}$ and $\mathbf{y}$ lie in the same orthant and
- $\mathbf{x}_i \leq \mathbf{y}_i$ for every $i \in [n]$.



Remarks

- "Natural order in each orthant of $\mathbb{Z}^n$",
- It is well-known that every subset of $\mathbb{Z}^n$ has finitely many $\sqsubseteq$-minimal elements.

Graver Basis

Definition (Graver basis)

The **Graver basis** of an $m \times n$-matrix A is the finite set $\mathcal{G}(A) \subset \mathbb{Z}^n$ of $\sqsubseteq$-minimal elements in $\{ \mathbf{x} \in \mathbb{Z}^n \mid A\mathbf{x} = 0, \mathbf{x} \neq 0 \}$.

Intuition

The Graver basis contains all "important" directions (separated by orthant) that preserve feasiblility of a solution w.r.t. constraint matrix.

Graver-best Step

Let $(A, \mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{c})$ be an ILP-instance having a feasible solution $\mathbf{x}$.

We say that $\mathbf{h} \in \mathbb{Z}^n$ is:

- a **feasible step** if $\mathbf{x} + \mathbf{h}$ is feasible,
- an **augmenting step** if it is a feasible step and $\mathbf{c}(\mathbf{x}+\mathbf{h}) \geq \mathbf{cx}$,
- a **Graver-best step** if it is an augmenting step and $\mathbf{c}(\mathbf{x} + \mathbf{h}) \geq \mathbf{c}(\mathbf{x} + \lambda\mathbf{g})$ for every $g \in \mathcal{G}(A)$ and every $\lambda \in \mathbb{Z}$.

Intuition

A Graver-best Step gives the best improvement that can be obtained by any step of the form $\lambda\mathbf{g}$, where $\mathbf{g} \in \mathcal{G}(A)$.

Graver-best Oracle

Definition

A **Graver-best Oracle** returns a Graver-best step $\mathbf{h}$ for a given ILP instance $(A, \mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{c})$ and feasible solution $\mathbf{x}$.

Theorem

ILP with a Graver-best oracle can be solved in strongly polynomial-time.

Graver-best Augmentation Procedure

Graver-best augmentation procedure

1. if there is no graver-best step for $\mathbf{x}$, return $\mathbf{x}$ as the optimum,
2. otherwise, set $\mathbf{x}$ to $\mathbf{x} + \mathbf{h}$ and go to 1.

Theorem (Graver-best augmentation procedure)

The Graver-best augmentation procedure finds an optimum solution for $\mathcal{I}$ in at most $(2n-2)\log F$ steps, where $F = \mathbf{cx} - \mathbf{cx}^*$ and x^* is any optimum solution.

Finding a Graver-best Step/Oracle

The general procedure to find a Graver-best step involves the following steps:

(S1) show a bound on the elements in $\mathcal{G}(A)$,

(S2) use the bound to compute a set of "interesting" step-lengths, i.e., step-lengths that can lead to a Graver-best step,

(S3) for every such step-length formulate an ILP whose optimum solution provides a Graver-best step w.r.t. to that step-length,

(S4) solve the ILP using the bound obtained in (S1) and additional structural insight

Example: Multi-stage Stochastic ILP (Step 1)

Lemma (Aschenbrenner and Hemmecke, 2007)

$$g_\infty(A) \leq f(\ell_A, l, n_1, \ldots, n_\omega)$$

for any Multi-stage stochastic constraint matrix A.

Remark

$$g_\infty(A) = \max_{\mathbf{g} \in \mathcal{G}(A)} \max_i |\mathbf{g}_i|$$

Example: Multi-stage Stochastic ILP (Step 2)

A step-length is interesting if it can be used to obtain a Graver-best step. Denote by $\Lambda(\mathbf{x})$ the set of interesting step-lengths for a feasible solution $\mathbf{x}$. Then:

Lemma

A set Λ of size at most $2(2M+1)n$ such that $\Lambda(\mathbf{x}) \subseteq \Lambda$ can be constructed in time $\mathcal{O}(Mn)$, where $M = g_\infty(A)$.

Proof

- any Graver-best step $\lambda\mathbf{g}$ must be tight in at least one coordinate i, i.e., either $(\lambda+1)\mathbf{g}_i + \mathbf{x}_i < \mathbf{l}_i$ or $(\lambda+1)\mathbf{g}_i + \mathbf{x}_i > \mathbf{u}_i$,
- for every $m \in \mathbb{Z}$ and every coordinate i there are at most two step-legths say $\lambda_L(m,i)$ and $\lambda_U(m,i)$ that are tight for the i-th coordinate,
- Hence the set Λ is given by the set $\{\, \lambda_L(m,i), \lambda_U(m,i) \mid -M \le m \le M \wedge 1 \le i \le n \,\}$.

Example: Multi-stage Stochastic ILP (Step 3)

Finding a Graver-best step for a step-length $\lambda \in \Lambda(\mathbf{x})$ is equivalent to solving:

$$
\begin{array}{ll}
\text{maximize} & \lambda \mathbf{c} \cdot \mathbf{g} \\
\text{subject to} & A\mathbf{g} = \mathbf{0} \\
& \mathbf{l} \leq \mathbf{x} + \lambda \mathbf{g} \leq \mathbf{u} \\
& -\mathbf{M} \leq \mathbf{g} \leq \mathbf{M} \\
& \mathbf{g} \in \mathbb{Z}^n
\end{array}
$$

Example: Multi-stage Stochastic ILP (Step 3)

Finding a Graver-best step for a step-length $\lambda \in \Lambda(\mathbf{x})$ is equivalent to solving:

$$\begin{array}{ll} \text{maximize} & \lambda \mathbf{c} \cdot \mathbf{g} \\ \text{subject to} & A\mathbf{g} = \mathbf{0} \\ & \mathbf{l} \leq \mathbf{x} + \lambda \mathbf{g} \leq \mathbf{u} \\ & -\mathbf{M} \leq \mathbf{g} \leq \mathbf{M} \\ & \mathbf{g} \in \mathbb{Z}^n \end{array}$$

Since the above ILP has bounded domain, it can be solved using the fpt-algorithm for treewidth and domain by using the following observation:

Observation

$$tw(A) \leq td(A) \leq \sum_{i=1}^{\omega} n_i$$

for any Multi-stage stochastic constraint matrix A.

Example: Multi-stage Stochastic ILP

Theorem (Koutecky, Levin, Onn (2018))

Multi-stage stochastic ILP is fixed-parameter tractable parameterized by ℓ_A, l, $n_1, \ldots, n_\omega$.

Summary: Primal Graph

Using structural restrictions of the primal graph leads to two main fixed-parameter tractable cases for ILP:

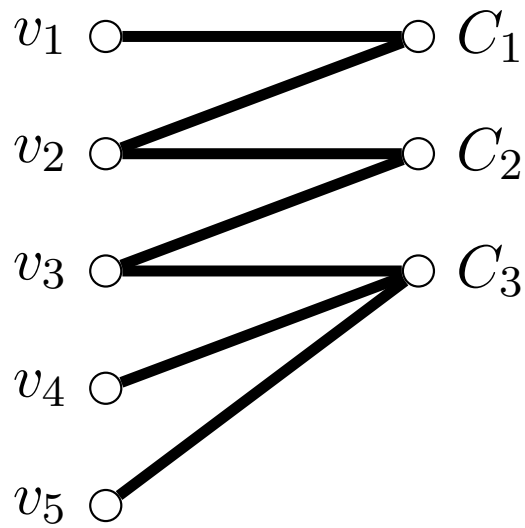
- treewidth and domain,
- treedepth and ℓ_A (a.k.a. multi-stage stochastic ILP).

All other combinations can be shown to be **para-NP**-hard.

Incidence Graph

Incidence Graph

$$A := \begin{pmatrix} * & * & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & * \end{pmatrix}$$



The **incidence graph** of an ILP instance $\mathcal{I}$, denoted by $\mathsf{I}(\mathcal{I})$, has:

- one vertex for every variable of $\mathcal{I}$,
- one vertex for every constraint of $\mathcal{I}$, and
- an edge between a variable x and a constraint c iff x occurs (has a non-zero coefficient) in c.

Incidence Graph vs. Primal Graph

- the incidence graph contains more information about the ILP instance,
- the treewidth of the incidence graph is always at most the treewidth of the primal graph plus one; but it can be arbitrary smaller,
- the main difference between incidence and primal treewidth is that incidence treewidth can be small even for instances with large arity

Incidence Graph: State-of-the-art

Using structural restrictions of the incidence graph leads to two main tractable cases for ILP:

- **FPT**: treewidth and Γ,
- **XP**: fracture number and ℓ_A (a.k.a. 4-block N-fold ILP).

Incidence Graph: State-of-the-art

Using structural restrictions of the incidence graph leads to two main tractable cases for ILP:

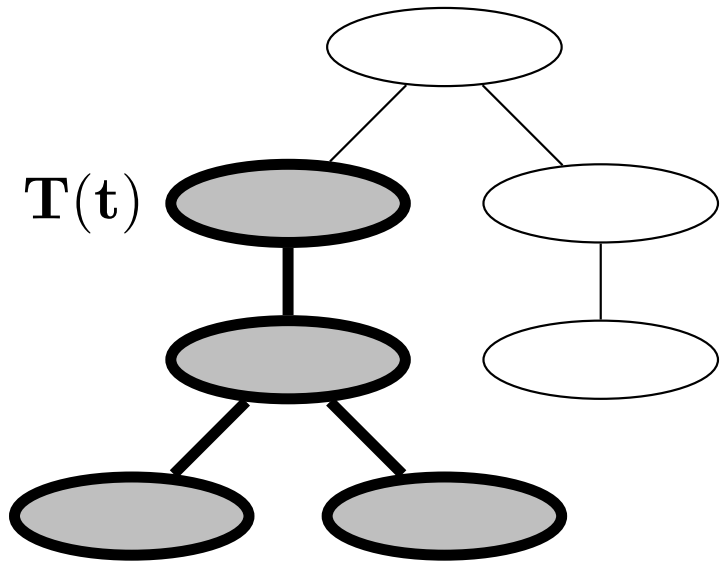
- **FPT**: treewidth and Γ,
- **XP**: fracture number and ℓ_A (a.k.a. 4-block N-fold ILP).

All other combinations can be shown to be **para-NP**-hard.

An Algorithm using incidence Treewidth (Main Idea)

Question?

Which information do we need to store for feasible assignments τ of $\mathcal{I}(t)$?



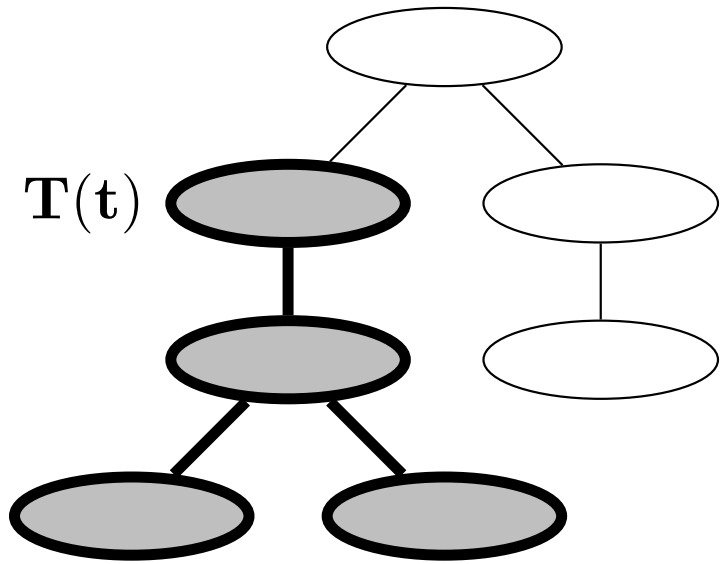
An Algorithm using incidence Treewidth (Main Idea)

Question?

Which information do we need to store for feasible assignments τ of $\mathcal{I}(t)$?

Answer:

- For the variables in $X(t)$ it is again sufficient to store their assignments since "future" constraints only share the variables in $X(t)$ with $V(t)$.



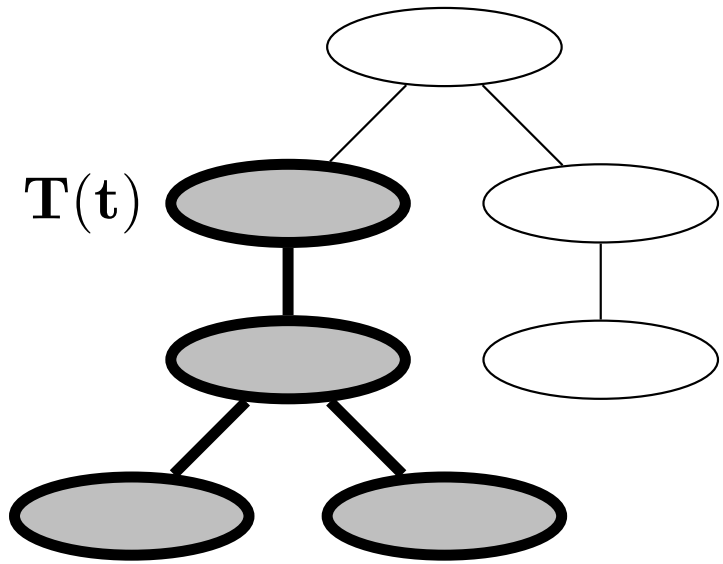
An Algorithm using incidence Treewidth (Main Idea)

Question?

Which information do we need to store for feasible assignments τ of $\mathcal{I}(t)$?

Answer:

- For the variables in $X(t)$ it is again sufficient to store their assignments since "future" constraints only share the variables in $X(t)$ with $V(t)$.
- However, since the constraints in $X(t)$ can be over variables both inside and outside of $V(t)$, we need to know all possible values they can evaluate to.



An Algorithm using Incidence Treewidth

Definition

For a constraint C and a (partial) assignment τ of (some of) the variables of C, let $C(\tau)$ denote the evaluation of C under τ.

Example

Let $C = 2x_1 + 3x_2 + 5x_3 = 8$ and let $\tau(x_1) = 5$ and $\tau(x_3) = 2$, then:

$$C(\tau) = 2\tau(x_1) + 5\tau(x_3) = 2 \cdot 5 + 5 \cdot 2 = 20$$

An Algorithm using Incidence Treewidth

Definition

For a constraint C and a (partial) assignment τ of (some of) the variables of C, let $C(\tau)$ denote the evaluation of C under τ.

Example

Let $C = 2x_1 + 3x_2 + 5x_3 = 8$ and let $\tau(x_1) = 5$ and $\tau(x_3) = 2$, then:

$$C(\tau) = 2\tau(x_1) + 5\tau(x_3) = 2 \cdot 5 + 5 \cdot 2 = 20$$

Definition

Let $\Gamma(\mathcal{I})$ be the maximum absolute value $C(\tau)$ over any constraint C of $\mathcal{I}$ and any feasible partial assignment τ of $\mathcal{I}$.

Discussion of the Algorithm

Theorem (Ganian, O., and Ramanujan, 2017)

An ILP instance $\mathcal{I}$ with n variables and m constraints can be solved in time:

$$\mathcal{O}(\Gamma(\mathcal{I})^{\mathsf{tw}(\mathcal{I})}(n+m))$$

Discussion of the Algorithm

Theorem (Ganian, O., and Ramanujan, 2017)

An ILP instance $\mathcal{I}$ with n variables and m constraints can be solved in time:

$$\mathcal{O}(\Gamma(\mathcal{I})^{\mathsf{tw}(\mathcal{I})}(n+m))$$

Remark

Because $\Gamma(\mathcal{I}) \leq \ell_A \times D \times n$, it follows that ILP can be solved in polynomial-time for bounded incidence treewidth provided that both ℓ and D are polynomially bounded in the input size.

Discussion of the Algorithm

Theorem (Ganian, O., and Ramanujan, 2017)

An ILP instance $\mathcal{I}$ with n variables and m constraints can be solved in time:

$$\mathcal{O}(\Gamma(\mathcal{I})^{\mathsf{tw}(\mathcal{I})}(n+m))$$

Remark

Because $\Gamma(\mathcal{I}) \leq \ell_A \times D \times n$, it follows that ILP can be solved in polynomial-time for bounded incidence treewidth provided that both ℓ and D are polynomially bounded in the input size.

Remark

Our previous hardness results for primal treewidth show that ILP becomes NP-complete again if only one of ℓ or D are allowed to grow exponentially in the input size.

Discussion of the Algorithm

Remark

If we want to find an fpt-algorithm parameterized by ℓ and some additional structural parameter of the incidence graph, we need to employ a more restrictive parameter than treewidth.

A natural candidate would be treedepth, however:

Theorem (Eiben et. al., IPCO 2019)

ILP-feasibility is NP-complete even if the incidence graph has treedepth at most 5 and the maximum absolute value of any coefficient is 1.

Nevertheless, we can show such a result for a slightly more restrictive parameter than treedepth, which we call the **fracture number**.

Summary: Incidence Graph

Using structural restrictions of the incidence graph leads to two main polynomial-time tractable cases for ILP:

- **FPT**: treewidth and Γ
- **XP**: fracture number and ℓ_A (a.k.a. N-fold 4-block ILP).

All other combinations can be shown to be **para-NP**-hard.

Dual Graph

Dual Graph (Transpose of Primal Graph)

$$A := \begin{pmatrix} * & 0 & 0 \\ * & * & 0 \\ 0 & * & * \\ 0 & 0 & * \\ 0 & 0 & * \end{pmatrix}$$

The **dual graph** of an ILP instance $\mathcal{I}$, denoted by $\mathsf{D}(\mathcal{I})$, has:

- one vertex for every constraint of $\mathcal{I}$,
- an edge between two constraints x and y iff x and y have a common variable.

Dual Graph vs. Primal Graph and Incidence Graph

- the dual graph is the transpose of the primal graph,
- the treewidth of the incidence graph is always at most the treewidth of the dual graph plus one; but it can be arbitrary smaller,
- the main difference between incidence and dual treewidth is that incidence treewidth can be small even for instances where variables occur in many constraints

Dual Graph: State-of-the-art

Using structural restrictions of the dual graph leads to two main fixed-parameter tractable cases for ILP:

- treewidth and Γ,
- treedepth and ℓ_A (a.k.a. tree-fold ILP).

Dual Graph: State-of-the-art

Using structural restrictions of the dual graph leads to two main fixed-parameter tractable cases for ILP:

- treewidth and Γ,
- treedepth and ℓ_A (a.k.a. tree-fold ILP).

All other combinations can be shown to be **para-NP**-hard.

Summary and Conclusions

- great potential for novel meta-theorems for NP-complete optimization problems,
- various recent applications of the results for problems in routing, scheduling, and social choice,
- the study of ILP w.r.t. to structural restrictions on the constraint matrix is still in its infancy.

Summary and Conclusions

- great potential for novel meta-theorems for NP-complete optimization problems,
- various recent applications of the results for problems in routing, scheduling, and social choice,
- the study of ILP w.r.t. to structural restrictions on the constraint matrix is still in its infancy.

Topics not covered here:

- Unary ILPs and stronger structural parameters such as **fracture number**,
- extension of tractable classes to Mixed ILP (e.g. **torso-width**), and
- algorithms using **signed clique-width**,

Open Problems and Future Work

- What is the practical relevance of the obtained algorithms?
- How can they be implemented to obtain the best results (in combination with known ILP solvers),
- Can the ideas be used to guide ILP-solvers?

Thank You!