On space efficiency of algorithms working on structural decompositions of graphs

Michał Pilipczuk   Marcin Wrochna

University of Warsaw

October 2015

Dynamic programming on treewidth

- 3-Coloring, Vertex Cover, IndSet, DomSet, 3-SAT, . . .
  can be solved in $c^{\mathtt{tw}}\mathrm{poly}(n)$ time.

Dynamic programming on treewidth

- 3-Coloring, Vertex Cover, IndSet, DomSet, 3-SAT, . . .
  can be solved in $c^{\mathtt{tw}}\mathrm{poly}(n)$ time and $c^{\mathtt{tw}}\mathrm{poly}(n)$ space.
- **Can we do this in better** (poly?) **space?**

Dynamic programming on treewidth

- 3-Coloring, Vertex Cover, IndSet, DomSet, 3-SAT, …
  can be solved in $c^{\mathtt{tw}}\mathrm{poly}(n)$ time and $c^{\mathtt{tw}}\mathrm{poly}(n)$ space.
- **Can we do this in better** (poly?) **space?**
- We can do $c^{\mathbf{n}}$ time and poly space…

Dynamic programming on treewidth

- 3-Coloring, Vertex Cover, IndSet, DomSet, 3-SAT, ...

  can be solved in $c^{\mathtt{tw}}\mathrm{poly}(n)$ time and $c^{\mathtt{tw}}\mathrm{poly}(n)$ space.
- **Can we do this in better** (poly?) **space?**
- We can do $c^{\mathbf{n}}$ time and poly space...
- Non-trivial space-efficient algos exist for other problems...

Time and space needed for algo on treewidth

3-COLORING:

- $2^{\mathcal{O}(\mathtt{tw})}\mathrm{poly}$ time and $2^{\mathcal{O}(\mathtt{tw})}\mathrm{poly}$ space by dynamic prog. ✓
- $2^{\mathcal{O}(\mathtt{tw}\cdot\log n)}$ time and $\mathtt{tw}\cdot\log n$ space by Div&Conq
- $2^{o(tw\cdot\log n)}$ time and $2^{o(tw)}\mathrm{poly}$ space?

Time and space needed for algo on treewidth

3-COLORING:

- $2^{\mathcal{O}(\texttt{tw})}\mathrm{poly}$ time and $2^{\mathcal{O}(\texttt{tw})}\mathrm{poly}$ space by dynamic prog. ✓
- $2^{\mathcal{O}(\texttt{tw}\cdot\log n)}$ time and $\texttt{tw}\cdot\log n$ space by Div&Conq ✓
- $2^{o(tw\cdot\log n)}$ time and $2^{o(tw)}\mathrm{poly}$ space?

Time and space needed for algo on treewidth

3-COLORING:

- $2^{\mathcal{O}(\mathtt{tw})}\mathrm{poly}$ time and $2^{\mathcal{O}(\mathtt{tw})}\mathrm{poly}$ space by dynamic prog. ✓
- $2^{\mathcal{O}(\mathtt{tw}\cdot\log n)}$ time and $\mathtt{tw}\cdot\log n$ space by Div&Conq ✓
- $2^{o(tw\cdot\log n)}$ time and $2^{o(tw)}\mathrm{poly}$ space?

Time and space needed for algo on treewidth

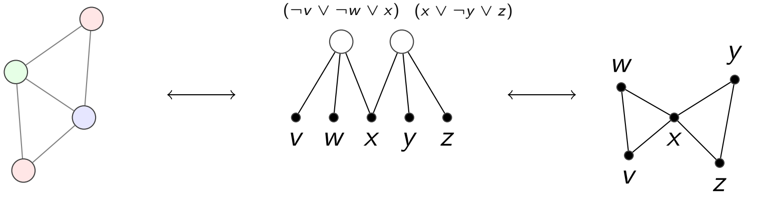
3-COLORING:

- $2^{\mathcal{O}(\mathtt{tw})}\mathrm{poly}$ time and $2^{\mathcal{O}(\mathtt{tw})}\mathrm{poly}$ space by dynamic prog. ✓
- $2^{\mathcal{O}(\mathtt{tw}\cdot\log n)}$ time and $\mathtt{tw}\cdot\log n$ space by Div&Conq ✓
- $2^{o(tw\cdot\log n)}$ time and $2^{o(tw)}\mathrm{poly}$ space?

Equivalence of problems

Standard reductions preserve treewidth linearly,
so it's the same question for SAT, etc.

3-Coloring	k-SAT for `tw` of incidence graph	k-SAT or SAT for `tw` of primal graph



Completeness

- Fix a function like $s(n) = \log^5 n$.
- pw-3Col[s] – the problem 3-COLORING with a path decomposition of width at most $s(n)$ on input.

Thm

pw-3Col[s] is complete for $\mathbf{N}[\underbrace{\text{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}]$

– problems solvable by **N**ondeterministic Turing machines with $\text{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

- for example: pw-3Col[log] is complete for **NL** (nondet logspace).

Completeness

- Fix a function like $s(n) = \log^5 n$.
- pw-3Col[s] – the problem 3-COLORING with a path decomposition of width at most $s(n)$ on input.

Thm

pw-3Col[s] is complete for $\mathbf{N}[\underset{time}{\mathrm{poly}}, \underset{space}{s}]$

– problems solvable by **N**ondeterministic Turing machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

- for example: pw-3Col[log] is complete for **NL** (nondet logspace).

Completeness

- Fix a function like $s(n) = \log^5 n$.
- pw-3Col[s] – the problem 3-Coloring with
  a path decomposition of width at most $s(n)$ on input.

Thm

pw-3Col[s] is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{time}, \underbrace{s}_{space}]$

– problems solvable by **N**ondeterministic Turing machines
with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

- for example: pw-3Col[log] is complete for **NL**
  (nondet logspace).

Completeness

- Fix a function like $s(n) = \log^5 n$.
- pw-3Col[s] – the problem 3-COLORING with a path decomposition of width at most $s(n)$ on input.

Thm

pw-3Col[s] is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{time}, \underbrace{s}_{space}]$

– problems solvable by **N**ondeterministic Turing machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

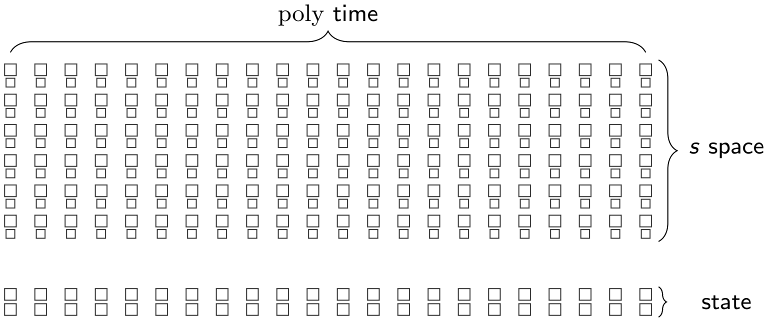
- for example: pw-3Col[log] is complete for **NL** (nondet logspace).

Completeness – Cook's theorem

Thm [Monien, Sudborough '85]

pw-SAT[s] is complete for $\mathbf{N[}\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}\mathbf{]}$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

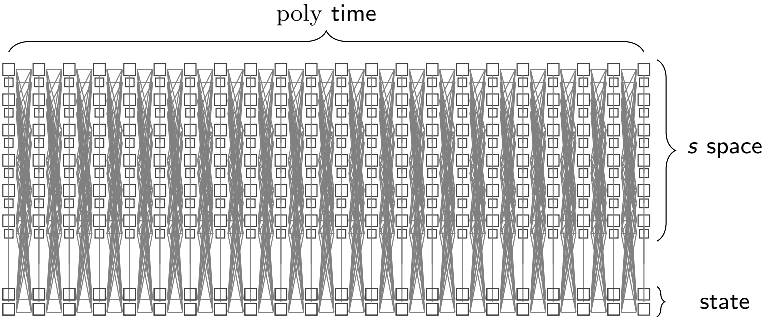


Completeness – Cook's theorem

Thm [Monien, Sudborough '85]

pw-SAT[s] is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}]$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

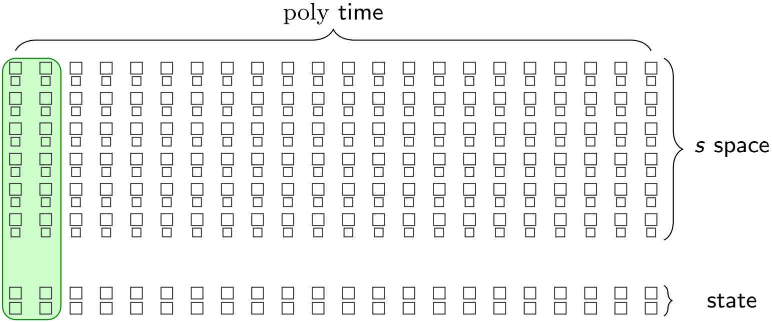


Completeness – Cook's theorem

Thm [Monien, Sudborough '85]

pw-SAT[s] is complete for **N[**$\underset{time}{\mathrm{poly}}$, $\underset{space}{s}$**]**

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

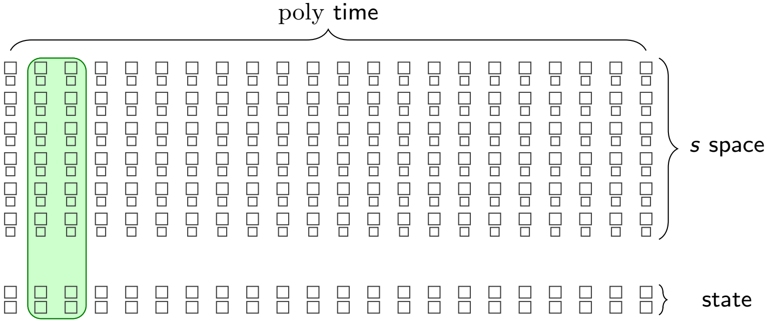


Completeness – Cook's theorem

Thm [Monien, Sudborough '85]

pw-SAT[s] is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{time}, \underbrace{s}_{space}]$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

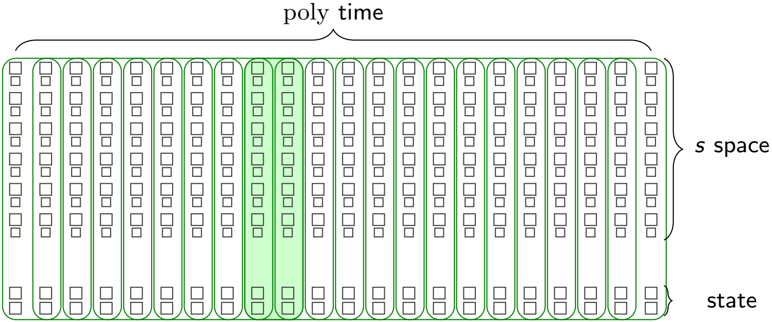


Completeness – Cook's theorem

Thm [Monien, Sudborough '85]

$\mathtt{pw}$-SAT$[s]$ is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}]$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.



Completeness – implications

Thm

pw-SAT[s] is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{time}, \underbrace{s}_{space}]$

- pw-3Col solvable in $2^{\mathcal{O}(s)}\mathrm{poly}$ time and $\mathrm{poly}(s, \lg n)$ space iff pw-3Col[log] solvable in poly time and $\mathrm{poly}\log$ space iff $\mathbf{NL} \subseteq \mathbf{D}[\underbrace{\mathrm{poly}}_{time}, \underbrace{\mathrm{poly}\log}_{space}]$ iff

  iff Directed Reachability can be solved in these bounds.
- In a restricted model we know $2^{o(\log^2 n)}$ time $n^{o(1)}$ space unconditional lower bounds.

Completeness – implications

Thm

pw-SAT[s] is complete for $\mathbf{N}[\underbrace{\text{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}]$

- pw-3Col solvable in $2^{\mathcal{O}(s)}\text{poly}$ time and $\text{poly}(s, \lg n)$ space iff
  pw-3Col[log] solvable in poly time and poly log space iff
  $\mathbf{NL} \subseteq \mathbf{D}[\underbrace{\text{poly}}_{\textit{time}}, \underbrace{\text{poly}\log}_{\textit{space}}]$ iff
  iff Directed Reachability can be solved in these bounds.
- In a restricted model we know $2^{o(\log^2 n)}$ time $n^{o(1)}$ space unconditional lower bounds.

Completeness – implications

Thm

pw-SAT[s] is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}]$

- pw-3Col solvable in $2^{\mathcal{O}(s)}\mathrm{poly}$ time and $\mathrm{poly}(s, \lg n)$ space iff
  pw-3Col[log] solvable in poly time and $\mathrm{poly}\log$ space iff
  $\mathbf{NL} \subseteq \mathbf{D}[\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{\mathrm{poly}\log}_{\textit{space}}]$ iff
  iff Directed Reachability can be solved in these bounds.
- In a restricted model we know $2^{o(\log^2 n)}$ time $n^{o(1)}$ space unconditional lower bounds.

How about treewidth?

Thm [Monien, Sudborough '85]

$\mathtt{pw}$-SAT[s] is complete for $\mathbf{N[}\underbrace{\mathrm{poly}}_{time}, \underbrace{s}_{space}\mathbf{]}$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

Thm [Allender, Chen, Lou, Papakonstantinou, Tang '14]

$\mathtt{tw}$-SAT[s] is complete for $\mathbf{NAuxPDA[}\underbrace{\mathrm{poly}}_{time}, \underbrace{s}_{space}\mathbf{]}$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space **and an unbounded push-down store**.

How about treewidth?

Thm [Monien, Sudborough '85]

$\texttt{pw}$-SAT[s] is complete for $\mathbf{N}[\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}]$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space.

Thm [Allender, Chen, Lou, Papakonstantinou, Tang '14]

$\texttt{tw}$-SAT[s] is complete for $\mathbf{NAuxPDA}[\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{s}_{\textit{space}}]$

– nondet machines with $\mathrm{poly}(n)$ time and $\mathcal{O}(s(n))$ space **and an unbounded push-down store**.

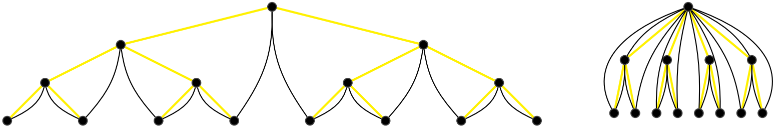
Our results

- tree-depth. . .
- Conjecture:
  "LONGEST COMMON SUBSEQ on input: alphabet, k strings cannot be solved in $n^{f(k)}$ time and $f(k)\mathrm{poly}$ space."

  Related to the *space-efficient* question for `pw`,
  for determinization.
  $\rightarrow$ completeness by [Elberfeld, Stockhusen, Tantau '14]

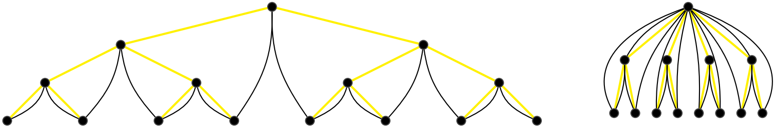
Tree-depth

Tree-depth



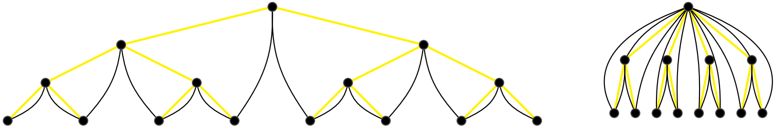
- Definition:
  - Intuitively: # of forget nodes on any root-leaf path of a nice tree decomposition.
  - min depth of a tree whose ancestor-closure contains G.
  - cops and robbers game, but a cop, once placed, cannot move.
  - Many others...
- Motivation:
  - Important in sparsity theory.
  - Some trichotomies for CSP complexity involve $\mathtt{tw}, \mathtt{pw}, \mathtt{td}$.
  - Seems to capture Divide and Conquer well.
  - $2^{\mathcal{O}(\mathtt{td})}$ time and poly space algorithms.

Tree-depth



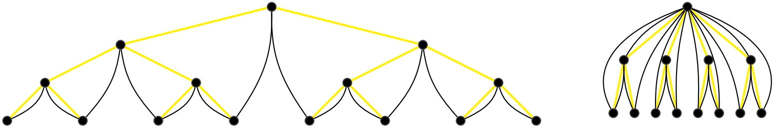
- Definition:
  - Intuitively: # of forget nodes on any root-leaf path of a nice tree decomposition.
  - min depth of a tree whose ancestor-closure contains G.
  - cops and robbers game, but a cop, once placed, cannot move.
  - Many others...
- Motivation:
  - Important in sparsity theory.
  - Some trichotomies for CSP complexity involve $\mathtt{tw}, \mathtt{pw}, \mathtt{td}$.
  - Seems to capture Divide and Conquer well.
  - $2^{\mathcal{O}(\mathtt{td})}$ time and poly space algorithms.

Tree-depth



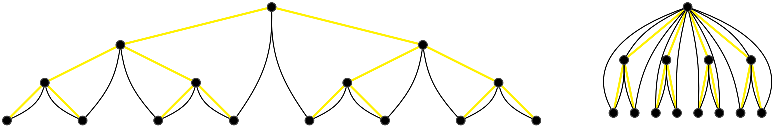
- Definition:
  - Intuitively: # of forget nodes on any root-leaf path of a nice tree decomposition.
  - min depth of a tree whose ancestor-closure contains G.
  - cops and robbers game, but a cop, once placed, cannot move.
  - Many others...
- Motivation:
  - Important in sparsity theory.
  - Some trichotomies for CSP complexity involve $\mathtt{tw}, \mathtt{pw}, \mathtt{td}$.
  - Seems to capture Divide and Conquer well.
  - $2^{\mathcal{O}(\mathtt{td})}$ time and poly space algorithms.

Tree-depth



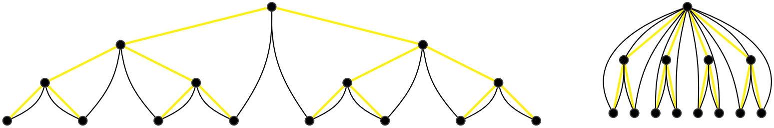
- Definition:
  - Intuitively: # of forget nodes on any root-leaf path of a nice tree decomposition.
  - min depth of a tree whose ancestor-closure contains G.
  - cops and robbers game, but a cop, once placed, cannot move.
  - Many others...
- Motivation:
  - Important in sparsity theory.
  - Some trichotomies for CSP complexity involve $\mathtt{tw}, \mathtt{pw}, \mathtt{td}$.
  - Seems to capture Divide and Conquer well.
  - $2^{\mathcal{O}(\mathtt{td})}$ time and poly space algorithms.

Tree-depth



- Definition:
  - Intuitively: # of forget nodes on any root-leaf path of a nice tree decomposition.
  - min depth of a tree whose ancestor-closure contains G.
  - cops and robbers game, but a cop, once placed, cannot move.
  - Many others...
- Motivation:
  - Important in sparsity theory.
  - Some trichotomies for CSP complexity involve $\mathtt{tw}, \mathtt{pw}, \mathtt{td}$.
  - Seems to capture Divide and Conquer well.
  - $2^{\mathcal{O}(\mathtt{td})}$ time and poly space algorithms.

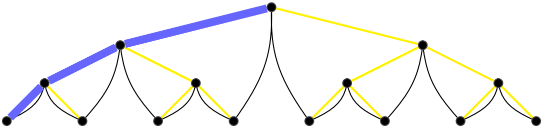
Tree-depth



- Definition:
  - Intuitively: # of forget nodes on any root-leaf path of a nice tree decomposition.
  - min depth of a tree whose ancestor-closure contains G.
  - cops and robbers game, but a cop, once placed, cannot move.
  - Many others...
- Motivation:
  - Important in sparsity theory.
  - Some trichotomies for CSP complexity involve $\mathtt{tw}, \mathtt{pw}, \mathtt{td}$.
  - Seems to capture Divide and Conquer well.
  - $2^{\mathcal{O}(\mathtt{td})}$ time and poly space algorithms.

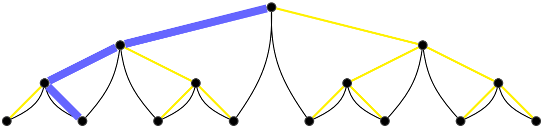
Tree-depth – dependencies

$$\mathtt{tw} \cdot \lg n \geq \mathtt{td} \geq \mathtt{pw} \geq \mathtt{tw}$$



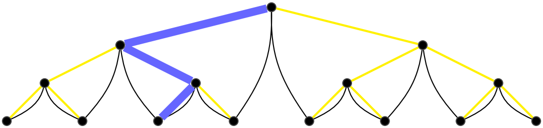
Tree-depth – dependencies

$$\texttt{tw} \cdot \lg n \geq \texttt{td} \geq \texttt{pw} \geq \texttt{tw}$$



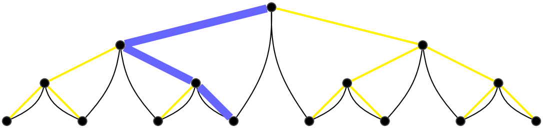
Tree-depth – dependencies

$$\mathtt{tw} \cdot \lg n \geq \mathtt{td} \geq \mathtt{pw} \geq \mathtt{tw}$$



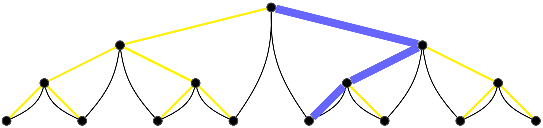
Tree-depth – dependencies

$$\texttt{tw} \cdot \lg n \geq \texttt{td} \geq \texttt{pw} \geq \texttt{tw}$$



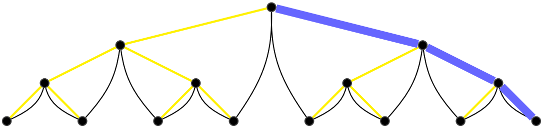
Tree-depth – dependencies

$$\mathtt{tw} \cdot \lg n \geq \mathtt{td} \geq \mathtt{pw} \geq \mathtt{tw}$$



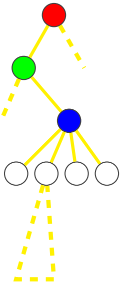
Tree-depth – dependencies

$$\mathtt{tw} \cdot \lg n \geq \mathtt{td} \geq \mathtt{pw} \geq \mathtt{tw}$$



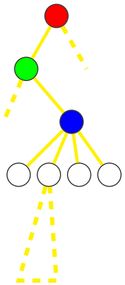
3COL on tree-depth

3COL can be solved in $\mathcal{O}(\mathtt{td} + \log)$ space (so $c^{\mathtt{td}}\mathrm{poly}$ time).



3COL on tree-depth

3COL can be solved in $\mathcal{O}(\mathtt{td} + \log)$ space (so $c^{\mathtt{td}}\mathrm{poly}$ time).



DOMSET can be solved in $3^{\mathtt{td}}\mathrm{poly}$ time and poly space.
→ Möbius transform [Fürer, Yu '14], [Lokshtanov, Nederlof '10]

Completeness for td

pw-SAT[s] is complete for **N[** $\underset{time}{\mathrm{poly}}, \underset{space}{s}$ **]**

tw-SAT[s] is complete for **NAuxPDA[** $\underset{time}{\mathrm{poly}}, \underset{space}{s}$ **]**

Let $s(n)$ be a nice polylogarithmic function $\geq \log^2 n$.

Thm

td-3Col[s] is complete for **NAuxSA[** $\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}$ **]**

– nondet machines with poly time, log space and a fully readable **stack of height** s.

Hierarchies

$$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}] = [\text{td-3}\textsc{Coloring}[s]]^L$$

$$\cap\!|$$

$$\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s}] = [\text{pw-3}\textsc{Coloring}[s]]^L$$

$$\cap\!|$$

$$\mathbf{NAuxPDA}[\underset{time}{\text{poly}}, \underset{space}{s}] = [\text{tw-3}\textsc{Coloring}[s]]^L$$

$$\cap\!|$$

$$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s \cdot \log}] = [\text{td-3}\textsc{Coloring}[s \cdot \log]]^L$$

Hierarchies – determinization

$$\textbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}] = [\text{td-3}\textsc{Coloring}[s]]^L \subseteq \textbf{D}[\underset{space}{s}]$$

$$\rotatebox{90}{$\subseteq$}$$

$$\textbf{N}[\underset{time}{\mathrm{poly}}, \underset{space}{s}] = [\text{pw-3}\textsc{Coloring}[s]]^L$$

$$\rotatebox{90}{$\subseteq$}$$

$$\textbf{NAuxPDA}[\underset{time}{\mathrm{poly}}, \underset{space}{s}] = [\text{tw-3}\textsc{Coloring}[s]]^L \subseteq \textbf{D}[\underset{time}{2^{\mathcal{O}(s)}}]$$

$$\rotatebox{90}{$\subseteq$}$$

$$\textbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s \cdot \log}] = [\text{td-3}\textsc{Coloring}[s \cdot \log]]^L \subseteq \textbf{D}[\underset{space}{s \cdot \log}]$$

Corollaries

- $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}] \subseteq \mathbf{D}[\underset{space}{s}]$
- $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}] = \mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\frac{s}{\log}}, \underset{height}{s}]$
- $\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}] = \mathbf{A}[\underset{time}{s}, \underset{treesize}{\text{poly}}]$

Corollaries

- $\mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}] \subseteq \mathbf{D}[\underset{space}{s}]$
- $\mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}] = \mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\frac{s}{\log}}, \underset{height}{s}]$
- $\mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}] = \mathbf{A}[\underset{time}{s}, \underset{treesize}{\mathrm{poly}}]$

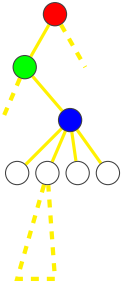
Corollaries

- $\mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}] \subseteq \mathbf{D}[\underset{space}{s}]$
- $\mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}] = \mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\frac{s}{\log}}, \underset{height}{s}]$
- $\mathbf{NAuxSA}[\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}] = \mathbf{A}[\underset{time}{s}, \underset{treesize}{\mathrm{poly}}]$

Containement

Solving td-3Col[s] in the **NAuxSA[**$\underbrace{\mathrm{poly}}_{\textit{time}}$, $\underbrace{\log}_{\textit{space}}$, $\underbrace{s}_{\textit{height}}$ **]** model:

- Enter the root
- When entering v:
  - guess its color, push it onto the stack,
  - compare with all ancestors,
  - recurse into each subtree.
- When leaving v: pop its color from the stack.

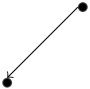


Proof of hardness

Push-pop tree

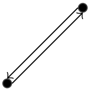
→ [Akatov, Gottlob '10]

push pop push push push pop push pop pop pop



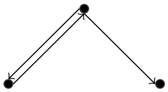
Push-pop tree

push pop push push push pop push pop pop pop



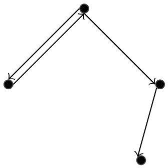
Push-pop tree

push pop push push push pop push pop pop pop



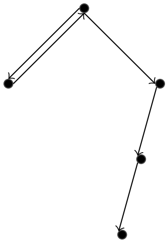
Push-pop tree

push pop push push push pop push pop pop pop



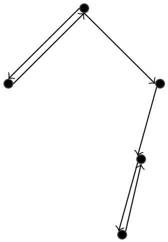
Push-pop tree

push pop push push push pop push pop pop pop



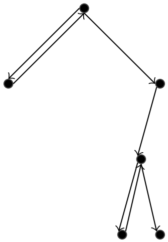
Push-pop tree

push pop push push push pop push pop pop pop



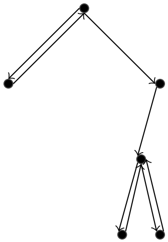
Push-pop tree

push pop push push push pop push pop pop pop



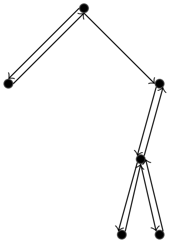
Push-pop tree

push pop push push push pop push pop pop pop



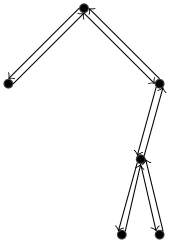
Push-pop tree

push pop push push push pop push pop pop pop



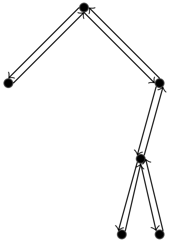
Push-pop tree

push pop push push push pop push pop pop pop



Push-pop tree

push pop push push push pop push pop pop pop



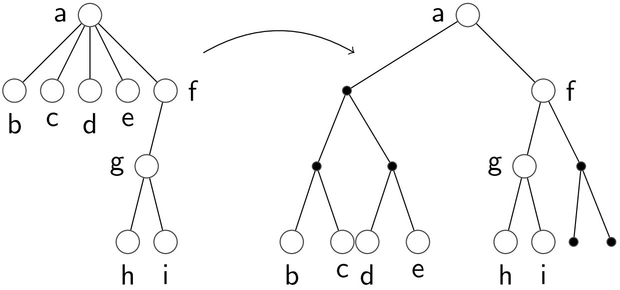
The arrows in order give the *Euler walk* of the tree.

Regularizing the tree

Lemma [Akatov, Gottlob '10], [Elberfeld, Jakoby, Tantau '10]

Every tree T of depth $\leq \lg|T|$ has an *embedding* into a full binary tree of depth $4\lg|T|$.

embedding = injection preserving ancestor relation and Euler walk.



Regularizing the machine

Consider a **NAuxSA[**$\underbrace{\mathrm{poly}}_{time}, \underbrace{\log}_{space}, \underbrace{s}_{height}$**]** machine. W.l.o.g. assume:

- It is a **NAuxSA[**$\underbrace{\mathrm{poly}}_{time}, \underbrace{s/\log}_{space}, \underbrace{s}_{height}$**]** machine,
- It pushes and pop blocks of $\lceil s/\lg \rceil$ symbols at a time, (by keeping the topmost symbols on a buffer on the tape)
- The push-pop tree (atomic 'block' pushes) has depth $\leq \lg n$.
- The push-pop tree is the full binary tree of depth $4 \lg n$.

Regularizing the machine

Consider a **NAuxSA[** $\underset{\textit{time}}{\text{poly}}, \underset{\textit{space}}{\log}, \underset{\textit{height}}{s}$ **]** machine. W.l.o.g. assume:

- It is a **NAuxSA[** $\underset{\textit{time}}{\text{poly}}, \underset{\textit{space}}{s/\log}, \underset{\textit{height}}{s}$ **]** machine,
- It pushes and pop blocks of $\lceil s/\lg \rceil$ symbols at a time, (by keeping the topmost symbols on a buffer on the tape)
- The push-pop tree (atomic 'block' pushes) has depth $\leq \lg n$.
- The push-pop tree is the full binary tree of depth $4 \lg n$.

Regularizing the machine

Consider a **NAuxSA**[$\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{\log}_{\textit{space}}, \underbrace{s}_{\textit{height}}$**]** machine. W.l.o.g. assume:

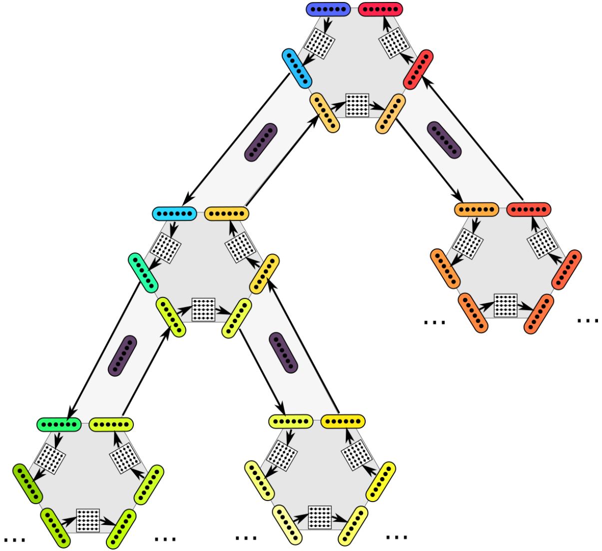
- It is a **NAuxSA**[$\underbrace{\mathrm{poly}}_{\textit{time}}, \underbrace{s/\log}_{\textit{space}}, \underbrace{s}_{\textit{height}}$**]** machine,
- It pushes and pop blocks of $\lceil s/\lg \rceil$ symbols at a time,
  (by keeping the topmost symbols on a buffer on the tape)
- The push-pop tree (atomic 'block' pushes) has depth $\leq \lg n$.
- The push-pop tree is the full binary tree of depth $4 \lg n$.

Regularizing the machine

Consider a $\textbf{NAuxSA[}\underbrace{\mathrm{poly}}_{time}, \underbrace{\log}_{space}, \underbrace{s}_{height}\textbf{]}$ machine. W.l.o.g. assume:

- It is a $\textbf{NAuxSA[}\underbrace{\mathrm{poly}}_{time}, \underbrace{s/\log}_{space}, \underbrace{s}_{height}\textbf{]}$ machine,
- It pushes and pop blocks of $\lceil s/\lg \rceil$ symbols at a time, (by keeping the topmost symbols on a buffer on the tape)
- The push-pop tree (atomic 'block' pushes) has depth $\leq \lg n$.
- The push-pop tree is the full binary tree of depth $4 \lg n$.

Wrapping computation around the push-pop tree



Conclusions

$$\mathbf{NAuxSA[}\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s}\mathbf{]} = [\text{td-3}\textsc{Coloring}[s]]^L \subseteq \mathbf{D[}\underset{space}{s}\mathbf{]}$$

$$\rotatebox{90}{$\subseteq$}$$

$$\mathbf{N[}\underset{time}{\mathrm{poly}}, \underset{space}{s}\mathbf{]} = [\text{pw-3}\textsc{Coloring}[s]]^L$$

$$\rotatebox{90}{$\subseteq$}$$

$$\mathbf{NAuxPDA[}\underset{time}{\mathrm{poly}}, \underset{space}{s}\mathbf{]} = [\text{tw-3}\textsc{Coloring}[s]]^L \subseteq \mathbf{D[}\underset{time}{2^{\mathcal{O}(s)}}\mathbf{]}$$

$$\rotatebox{90}{$\subseteq$}$$

$$\mathbf{NAuxSA[}\underset{time}{\mathrm{poly}}, \underset{space}{\log}, \underset{height}{s \cdot \log}\mathbf{]} = [\text{td-3}\textsc{Coloring}[s \cdot \log]]^L \subseteq \mathbf{D[}\underset{space}{s \cdot \log}\mathbf{]}$$

Conclusions

$$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s}] = [\text{td-3}\textsc{Coloring}[s]]^L \subseteq \mathbf{D}[\underset{space}{s}]$$

$$\cap\!|$$

$$\mathbf{N}[\underset{time}{\text{poly}}, \underset{space}{s}] = [\text{pw-3}\textsc{Coloring}[s]]^L$$

$$\cap\!|$$

$$\mathbf{NAuxPDA}[\underset{time}{\text{poly}}, \underset{space}{s}] = [\text{tw-3}\textsc{Coloring}[s]]^L \subseteq \mathbf{D}[\underset{time}{2^{\mathcal{O}(s)}}]$$

$$\cap\!|$$

$$\mathbf{NAuxSA}[\underset{time}{\text{poly}}, \underset{space}{\log}, \underset{height}{s \cdot \log}] = [\text{td-3}\textsc{Coloring}[s \cdot \log]]^L \subseteq \mathbf{D}[\underset{space}{s \cdot \log}]$$

Thank you!