Visualizing Dynamic Programming On Tree Decompositions

Martin Röbke
Fakultät Informatik
Technische Universität Dresden
Germany

- WHAT **is the motivation**
- WHO **benefits from visualization?**
- CHALLENGES **and solutions**
- WHAT **could be used otherwise?**
- OUTLOOK **and ideas**

Motivation

- DP-on-TD-algorithms can solve various combinatorial problems like model counting
- Efficient (if the treewidth is small)
- Competitive with other modern solvers

- **But tedious and hard to implement if done efficiently**

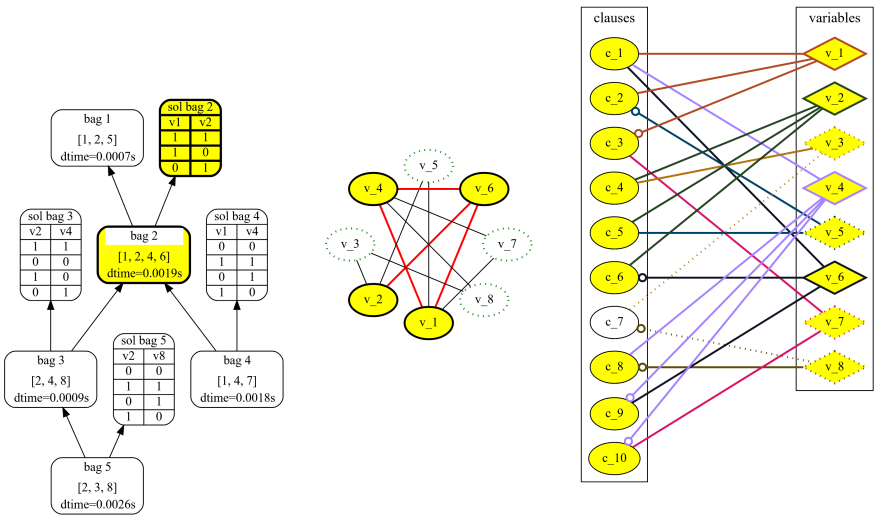
- Often bugs in the implementation
- DP for model counting is extremely space demanding (much more than SAT)

Ideas

- **Inspect intermediate data during solving process**
- **Represent the input, tree decomposition and created solutions**
- **Lightweight but customizable output format**

Ideas

- **Inspect intermediate data during solving process**
- **Represent the input, tree decomposition and created solutions**
- **Lightweight but customizable output format**



Contribution

This thesis created tdvisu as a tool that

- **integrates into existing implementations**
- **statically exports data from runs**
- **compiles simple DOT files and SVG graphics**

For further research it provides

- **starting point for more complex investigations of**
  - **bug spotting**
  - **and fixing by using visualizations**

Background

The algorithms of interest solve problems of:

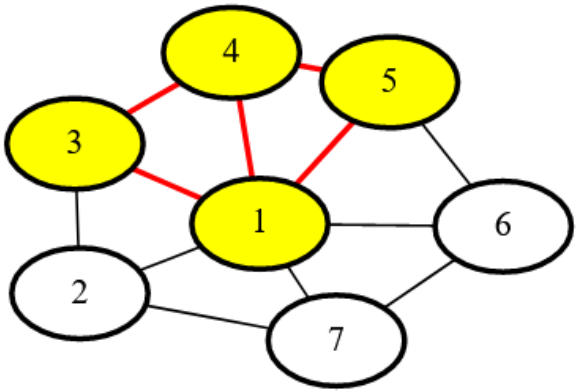
- **NP-complete**
- **#P-complete problems -** instead of **one** solution we want to **count all** solutions

Background

The algorithms of interest solve problems of:

- **NP-complete**
- **#P-complete problems -** instead of **one** solution we want to **count all** solutions

Example of two snapshots of getting a minimal vertex cover via DP:

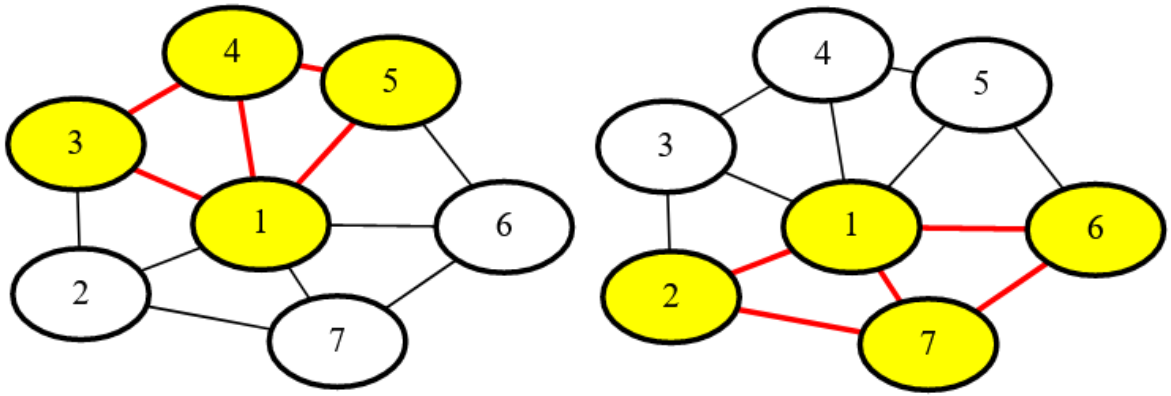


Background

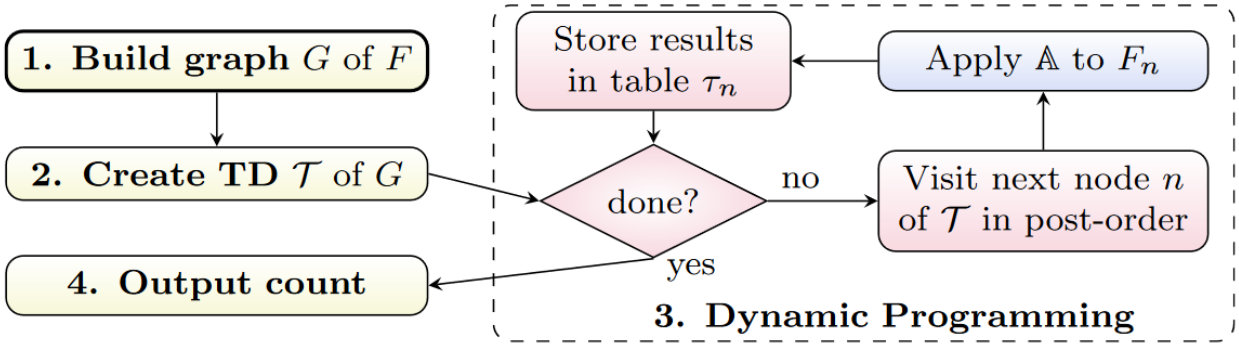
The algorithms of interest solve problems of:

- **NP-complete**
- **#P-complete problems -** instead of **one** solution we want to **count all** solutions

Example of two snapshots of getting a minimal vertex cover via DP:



gpusat2 - Solving #SAT on GPU



- **Customized tree decompositions**

Images: Markus Zisser. *Solving the #SAT problem on the GPU with dynamic programming and OpenCL*. Technische Universität Wien, 2018.

dpdb

Database templates in Python
Generating SQL queries

1. **Create graph representation**
2. **Decompose graph**
3. **Solve sub-problems**
4. **Combine rows**

- #εTab#: `SELECT 1 AS cnt`
- #intrTab#: `SELECT 1 AS val UNION ALL 0`
- #localProbFilter#: $(l_{1,1}$ `OR ... OR` $l_{1,k_1})$ `AND ... AND` $(l_{n,1}$ `OR ... OR` $l_{n,k_n})$
- #aggrExp#: `SUM(cnt) AS cnt`
- #extProj#: τ_1`.cnt * ... *` τ_ℓ`.cnt AS cnt`

(a) Problem #SAT

- #εTab#: `SELECT 0 AS card`
- #intrTab#: `SELECT 1 AS val UNION ALL 0`
- #localProbFilter#: $([\![u_1]\!]$ `OR` $[\![v_1]\!])$ `AND ... AND` $([\![u_n]\!]$ `OR` $[\![v_n]\!])$
- #aggrExp#: `MIN(card) AS card`
- #extProj#: τ_1`.card + ... +` τ_ℓ`.card -` $(\Sigma_{i=1}^{\ell}|\chi(t_i) \cap \{a_1\}|$ `- 1) *` $\tau_1.[\![a_1]\!]$ `- ... -` $(\Sigma_{i=1}^{\ell}|\chi(t_i) \cap \{a_k\}|$ `- 1) *` $\tau_1.[\![a_k]\!]$

(b) Problem MinVC

- **SAT and #SAT**
- **#o-Coloring**
- **Vertex cover**

. . .

"Exploiting Database Management Systems and Treewidth for Counting", Johannes Fichte et al. doi: 10.1007/978-3-030-39197-3_10.

Tree Decomposition

Gives the DP algorithm a partial ordering for sub-problems.

1. **Each vertex must occur in some bag**

Tree Decomposition

Gives the DP algorithm a partial ordering for sub-problems.

1. **Each vertex must occur in some bag**
2. **For each edge, there is a bag containing both endpoints**

Tree Decomposition

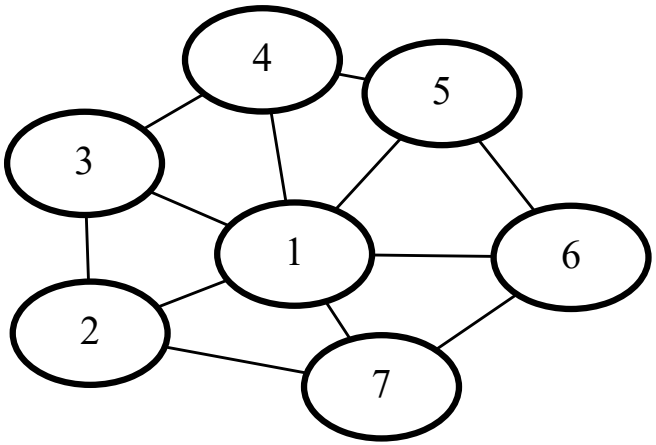
Gives the DP algorithm a partial ordering for sub-problems.

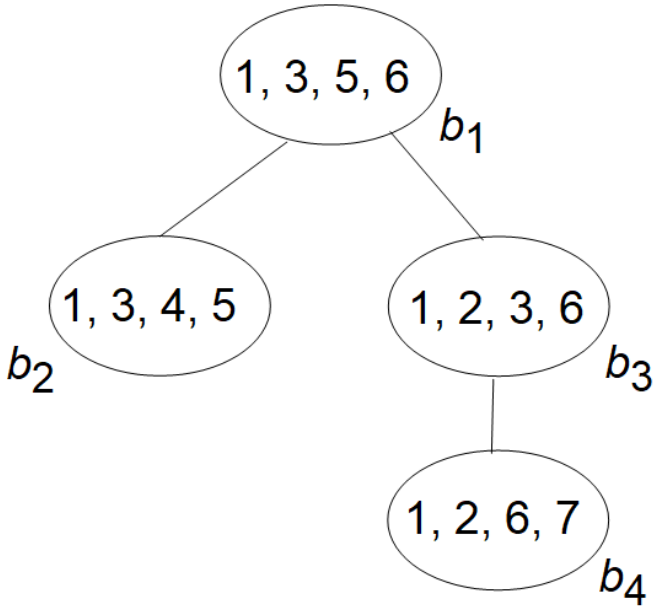
1. **Each vertex must occur in some bag**
2. **For each edge, there is a bag containing both endpoints**
3. **Subgraph of bags containing a given vertex must be connected**

Tree Decomposition

Gives the DP algorithm a partial ordering for sub-problems.

1. **Each vertex must occur in some bag**
2. **For each edge, there is a bag containing both endpoints**
3. **Subgraph of bags containing a given vertex must be connected**





Width of a TD is: size of largest bag -1

width $=$ **3**

Graphs for Boolean Formulas

- **Example set of CNF-clauses:**

$\{c1 = \{v1, v3, \neg v4\}, c2 = \{\neg v1, v6\}, c3 = \{\neg v2, \neg v3, \neg v4\}, c4 = \{\neg v2, v6\}, c5 = \{\neg v3, \neg v4\}, c6 = \{\neg v3, v5\}, c7 = \{\neg v5, \neg v6\}, c8 = \{v5, v7\}\}$

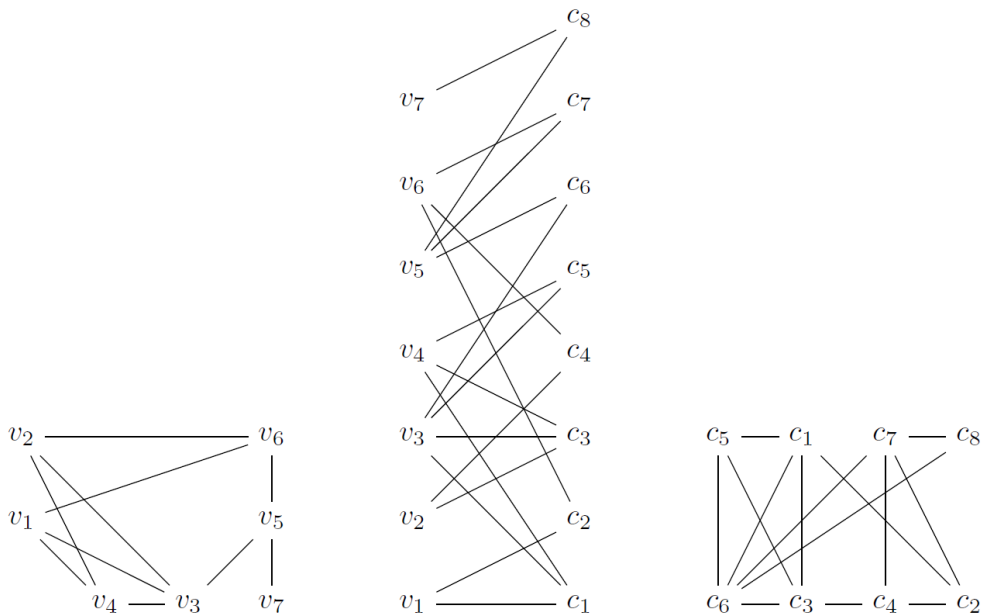


Figure: The primal (left), incidence (middle) and dual (right) graph

TDVisu

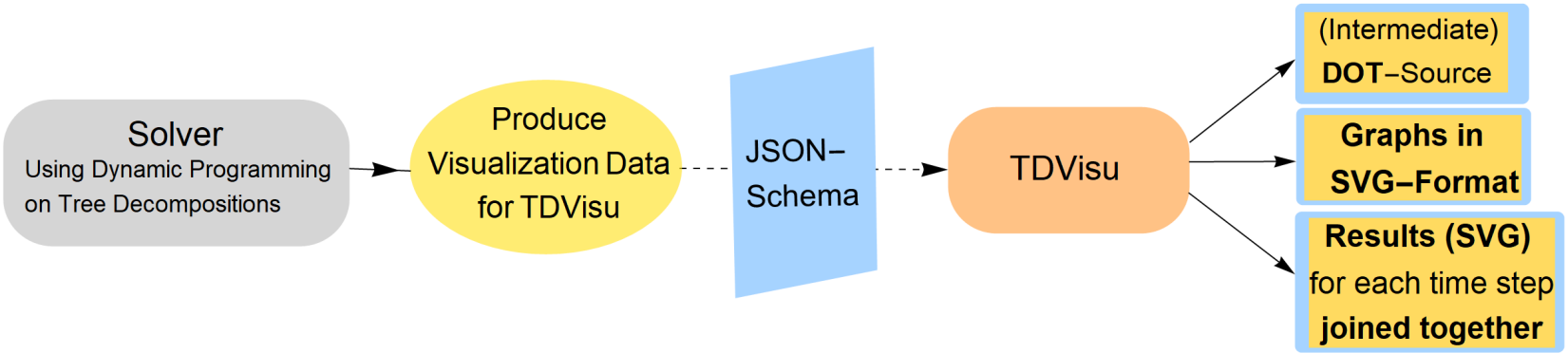


Figure: TDVisu producing flexible and further processable formats

TDVisu

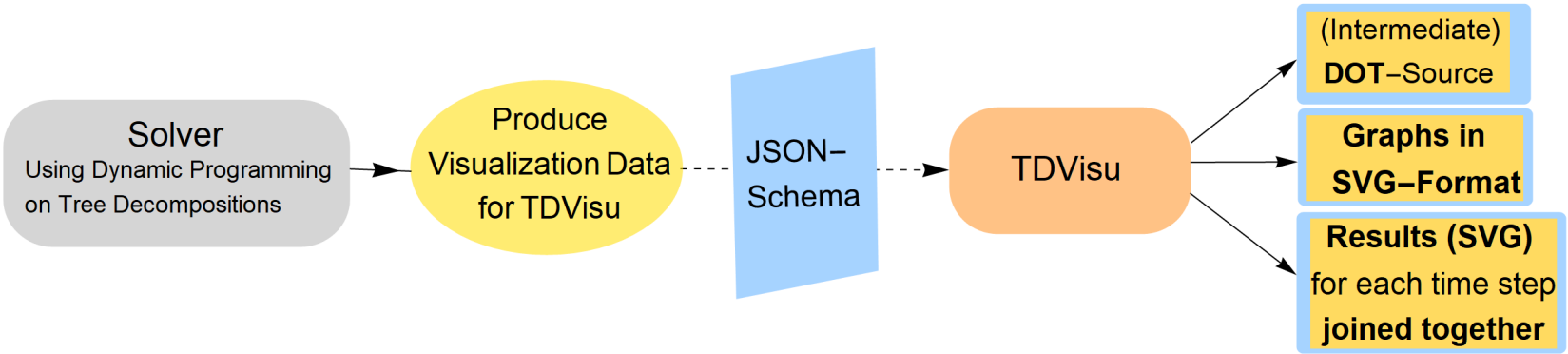


Figure: TDVisu producing flexible and further processable formats

TDVisu

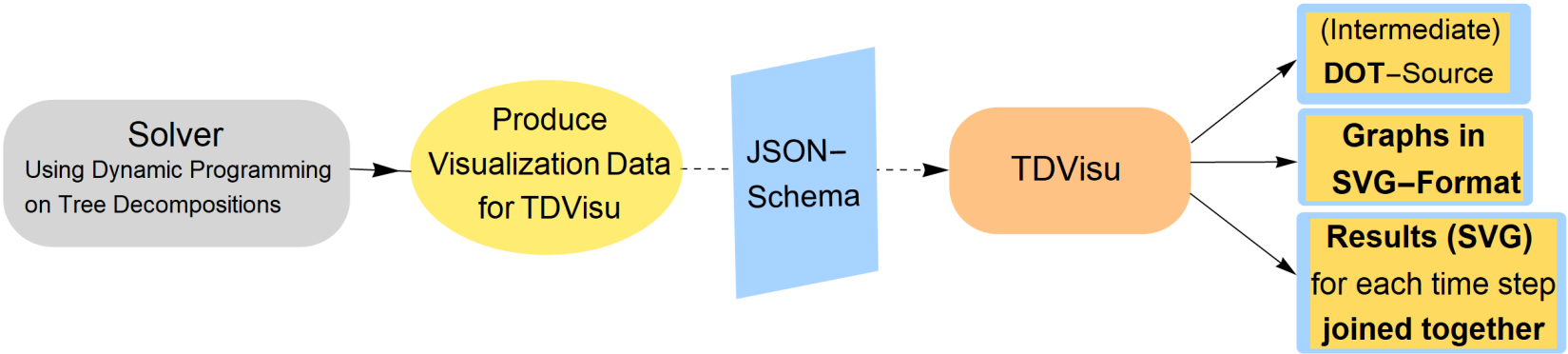


Figure: TDVisu producing flexible and further processable formats

JSON-Schema specifying:

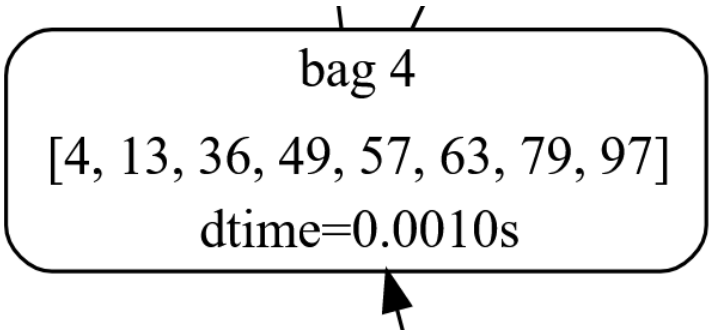
timeline, tree decomposition

incidence graph, general graph

orientation, maximum lines, maximum columns, emphasis, svgjoin info

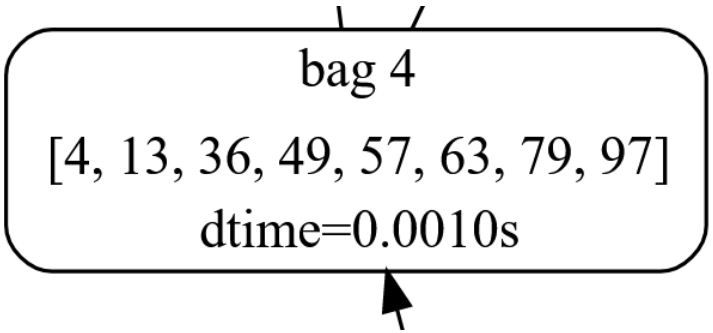
Graphics for TD

- **Most cells are interpreted as strings.**
- **Extendable header and footer.**



Graphics for TD

- **Most cells are interpreted as strings.**
- **Extendable header and footer.**



sol bag 4

v4	v13	v49	v57	v63	v97	size
0	1	0	1	1	1	82
1	0	1	1	1	1	83
1	1	0	1	1	1	82
1	1	0	1	0	1	82
1	1	1	0	1	1	82
1	1	1	1	1	1	83
1	0	0	1	1	1	82
1	1	1	1	1	0	83
1	1	1	1	0	1	83
0	1	1	1	1	1	83
1	1	0	0	1	1	81
1	1	0	1	1	0	82

min-size: 81

Figure: Bag node and solution node

Capabilities and Limitations

Integration of solvers via TDVisu.schema.json [1]

Capabilities:

- **Extracting basic (extendable) information (TD, solution nodes, order+time of processing...) from gpusat + dpdb**
- **Constructing and enriching the with solver information**
- **Adding multiple graphs for e.g. problems on Boolean formulas**
- **Providing a discrete timeline**
- **Parameters to control the layout and coloring of the data**

Limitations:

- **Can not further animate for example the origin of solutions**
- **Maneuvering in very large graphs is not very ergonomic with static content**
- **In the optimal case, a comprehensive test suite should be run prior to this.**

raw.githubusercontent.com/VaeterchenFrost/tdvisu/master/TDVisu.schema.json

Visualization in Action

MinVC example size 90 (expected 82)

Debugging with Visualization

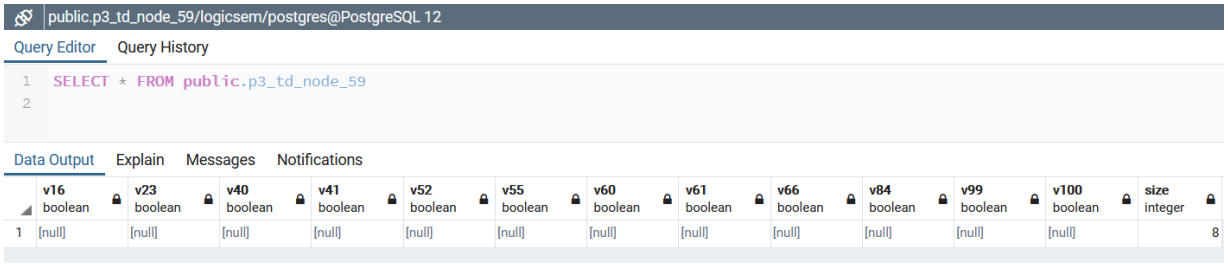
1. **Inspect visualization**

Debugging with Visualization

1. **Inspect visualization**
2. **Verify findings in solver (in this case dpdb)**

Debugging with Visualization

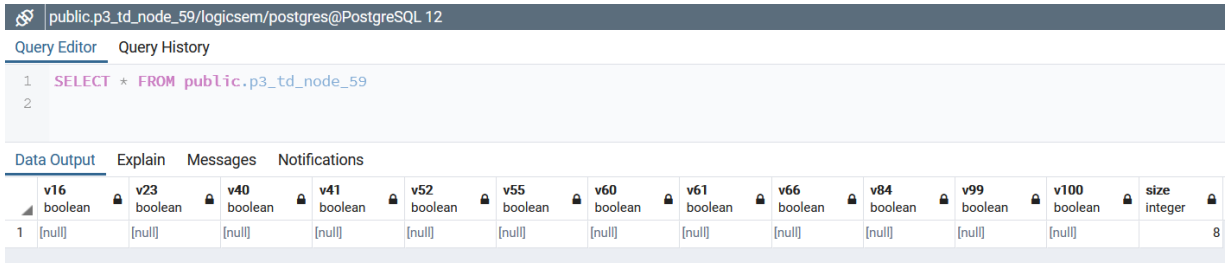
1. **Inspect visualization**
2. **Verify findings in solver (in this case dpdb)**



	v16 boolean	v23 boolean	v40 boolean	v41 boolean	v52 boolean	v55 boolean	v60 boolean	v61 boolean	v66 boolean	v84 boolean	v99 boolean	v100 boolean	size integer
1	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	8

Debugging with Visualization

1. **Inspect visualization**
2. **Verify findings in solver (in this case dpdb)**



	v16 boolean	v23 boolean	v40 boolean	v41 boolean	v52 boolean	v55 boolean	v60 boolean	v61 boolean	v66 boolean	v84 boolean	v99 boolean	v100 boolean	size integer
1	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	8

3. **Cross reference with standalone tree-decomposition**

Debugging with Visualization

1. **Inspect visualization**
2. **Verify findings in solver (in this case dpdb)**

public.p3_td_node_59/logicsem/postgres@PostgreSQL 12

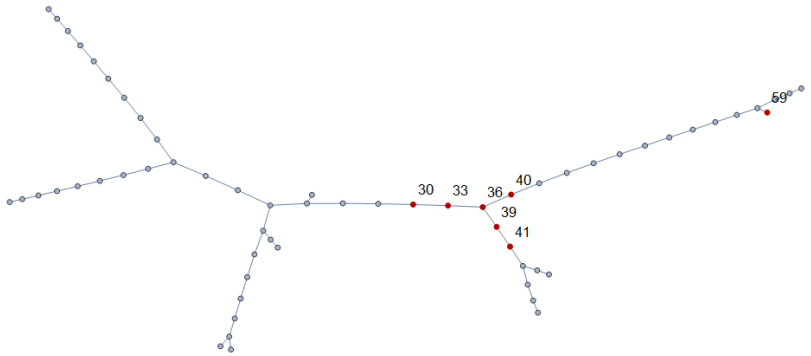
Query Editor | Query History

```
1  SELECT * FROM public.p3_td_node_59
2
```

Data Output | Explain | Messages | Notifications

	v16 boolean	v23 boolean	v40 boolean	v41 boolean	v52 boolean	v55 boolean	v60 boolean	v61 boolean	v66 boolean	v84 boolean	v99 boolean	v100 boolean	size integer
1	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	8

3. **Cross reference with standalone tree-decomposition**



Debugging with Visualization

1. **Inspect visualization**
2. **Verify findings in solver (in this case dpdb)**

public.p3_td_node_59/logicsem/postgres@PostgreSQL 12

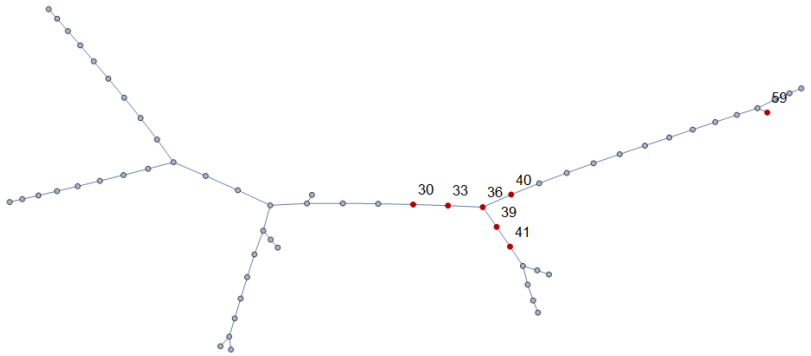
Query Editor | Query History

```
1  SELECT * FROM public.p3_td_node_59
2
```

Data Output | Explain | Messages | Notifications

	v16 boolean	v23 boolean	v40 boolean	v41 boolean	v52 boolean	v55 boolean	v60 boolean	v61 boolean	v66 boolean	v84 boolean	v99 boolean	v100 boolean	size integer
1	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	8

3. **Cross reference with standalone tree-decomposition**



4. **Fix the root cause**

Related Work on the Algorithms

- **J. Fichte, M. Hecher, M. Morak, S. Woltran (2018)**
  “Exploiting Treewidth for Projected Model Counting and Its Limits.”
- **M. Hecher (2020)**
  “Treewidth-aware Reductions of Normal ASP to SAT - Is Normal ASP Harder than SAT after All?”.
- **M. Hecher, P. Thier, S. Woltran (2020)**
  “Taming High Treewidth with Abstraction, Nested Dynamic Programming, and Database Technology.” In: Pulina L., Seidl M. (eds) Theory and Applications of Satisfiability Testing - SAT 2020.

Related Work on Visualizations

- M.-C. Harre, Jelschen, Winter. “ELVIZ: A querybased approach to model visualization”. (Jan. 2014)
- S. Diehl. “Software Visualization. Visualizing the Structure, Behaviour, and Evolution of Software.” Springer, 2007.
- J. Daida et al. “Visualizing Tree Structures in Genetic Programming”. (Mar. 2005)

Outlook

Static $\rightarrow$ Dynamic

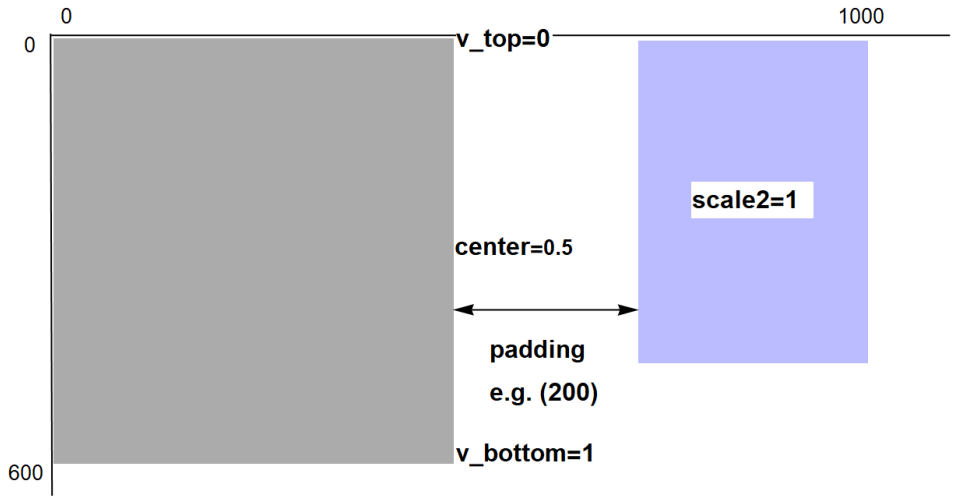
Interesting Questions :

- **Cross reference the creation of rows in parent nodes**
- **Enriching the visualization with more data for each node**
- **For more advanced debugging tasks you may also need to revise the approach**
- **Utilizing graph databases for debugging**

Final slide

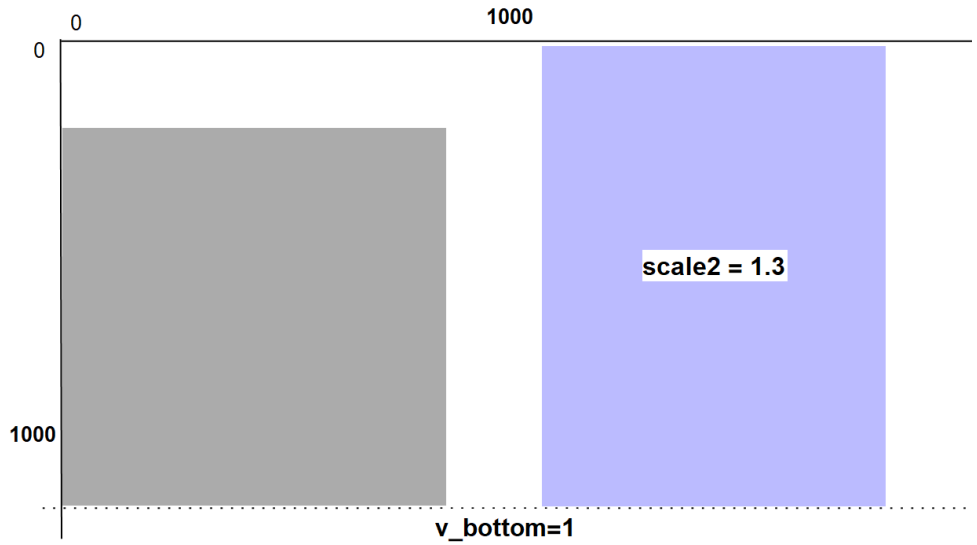
SVG-Join

- **Joining single graphs for each time step**

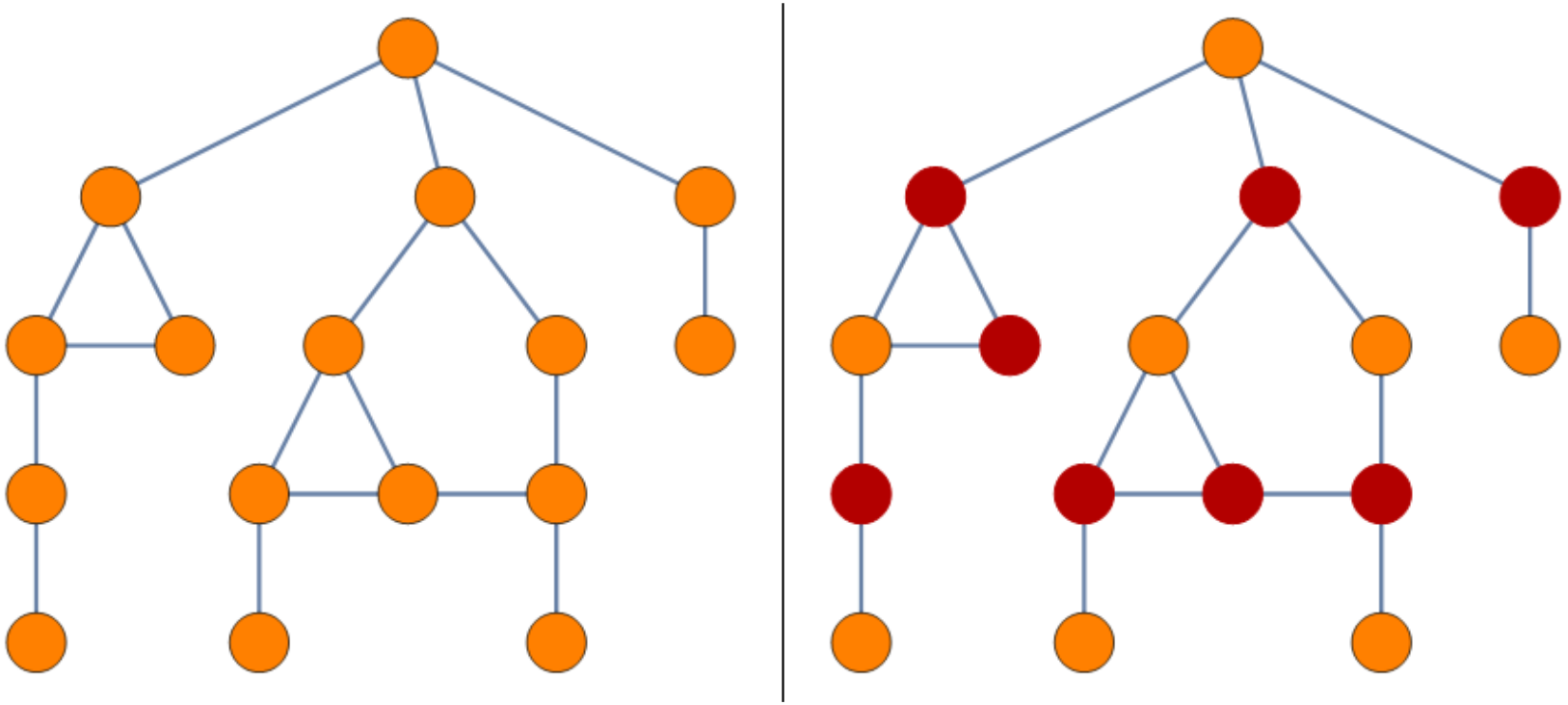


SVG-Join

- **Joining single graphs for each time step**

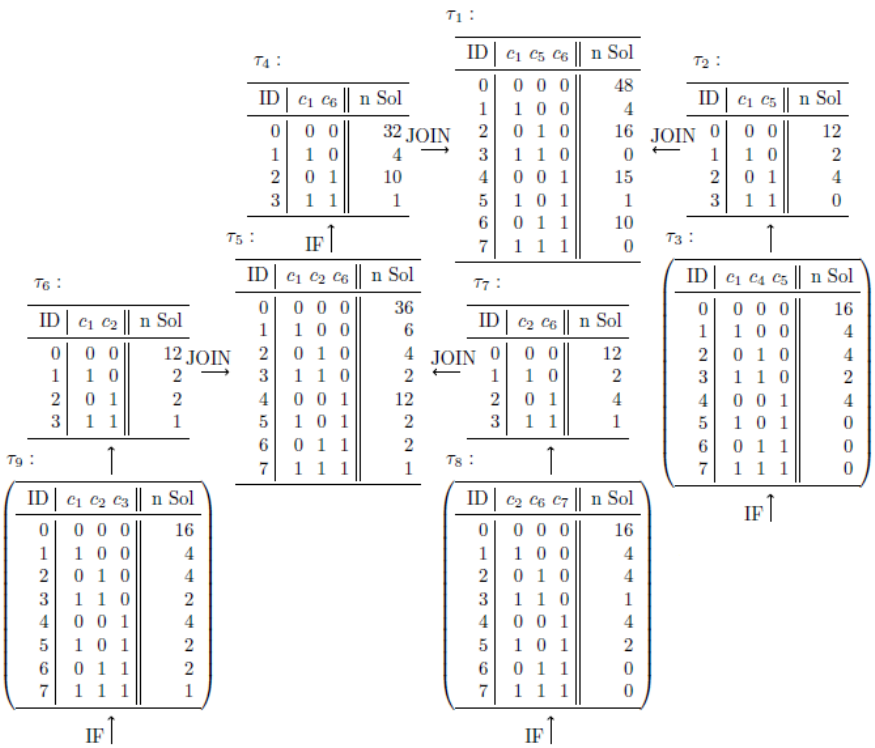


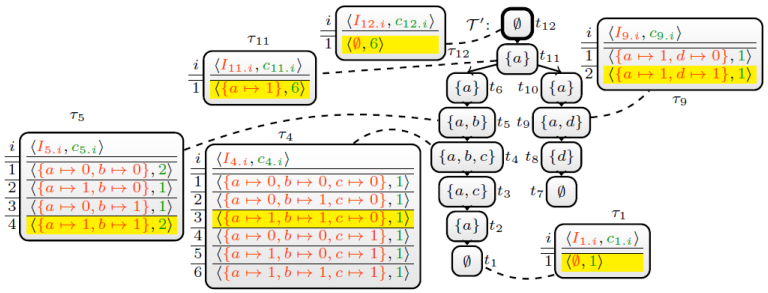
MinVC for example graph



Visualization

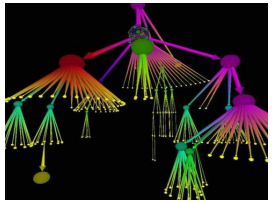
Manually for one run



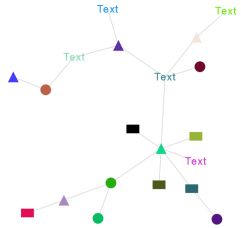


"Exploiting Database Management Systems and Treewidth for Counting", Johannes Fichte et al. doi: 10.1007/978-3-030-39197-3_10.

"Solving #SAT on the GPU with Dynamic Programming and OpenCL", Diploma Markus Zisser 2018 Technische Universität Wien, p.33



Gephi.org[1] Tulip [2]



[3] Vis.js Sigma.js vasturiano/3d-force-graph [4]

With the diverse / large node labels and special layout
the creation of a lightweight and customizable exchange format
took precedence over the integration into special layout software.

---

https://gephi.org/ - Tool for data analysts and scientists keen to explore and understand graphs.
tulip.labri.fr/TulipDrupal/ - Better Visualization Through Research.
https://neo4j.com/developer/tools-graph-visualization/
https://github.com/vasturiano/3d-force-graph

ELVIZ - Query based approach to software visualization

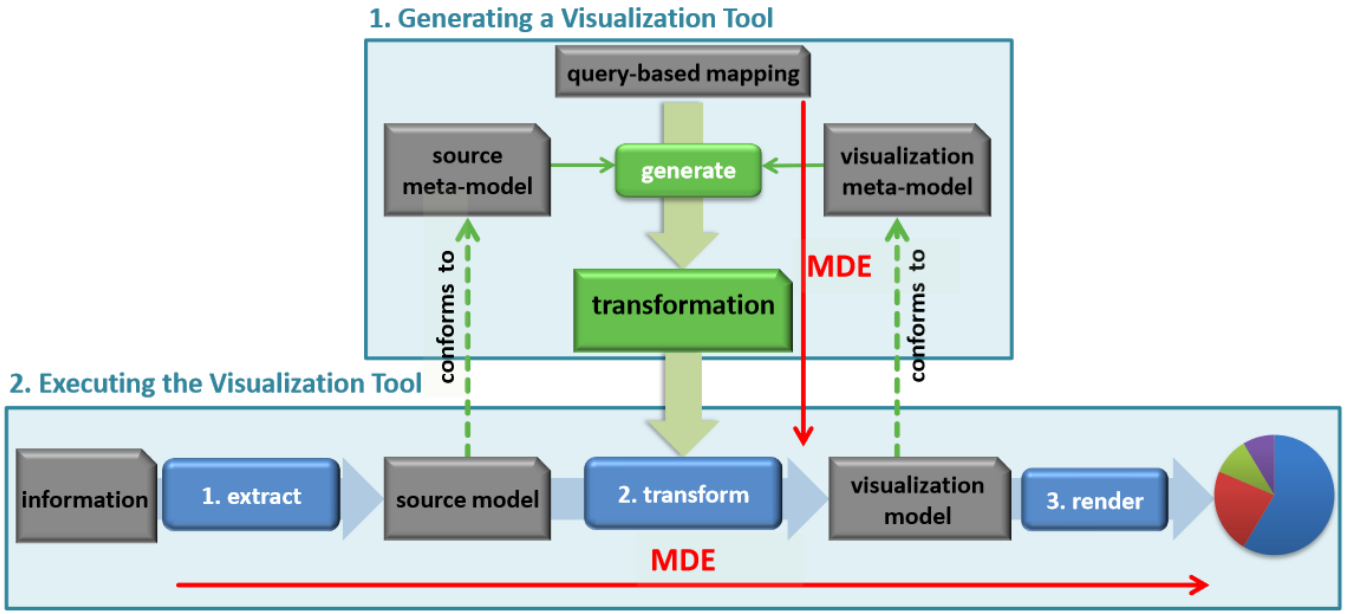


Figure: Overview of the ELVIZ-approach[5]

Fig 1 in: Marie-Christin Harre, J. Jelschen, A. Winter. “ELVIZ: A querybased approach to model visualization”. In: Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI) (Jan. 2014), pp. 105–120.

Bibliography

See the citations in the thesis.