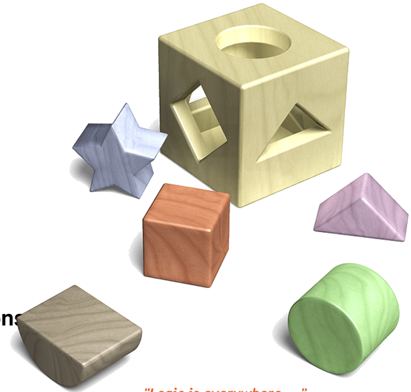
Visualizing Dynamic Programming On Tree Decompositions

Martin Röbke
Fakultät Informatik
Technische Universität Dresden
Germany

- **REWORK AFTER CHAPTERS**
- WHAT **was the motivation**
- WHAT **could be used otherwise?**
- WHO **benefits from visualization?**
- METHODOLOGY **challenges and solutions**
- WHAT **could be developed next?**



"Logic is everywhere ..."

Motivation

- **DP-on-TD-algorithms can solve Model Counting and various combinatorial problems**
- **Implementations of those are competing with modern solvers**
- **But: those are fairly hard to implement efficiently**
- **Practical debug output quickly becomes very large (GB)**
- **Finding the cause of the problem is a time consuming challenge**

The B.T. probably focused too much on the convenience features, and not on the urgent need for better debugging and visualization needs for those algorithms.

Background

The algorithms of interest solve problems of:

- **combinatorics (NP-problems)**
- **model-counting (#P-problems)**

Recent promising results for Projected Model Counting by Markus Hecher[1].

[1] Hecher M., Thier P., Woltran S. (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. SAT 2020. Lecture Notes in Computer Science, vol 12178. Springer, Cham. https://doi.org/10.1007/978-3-030-51825-7_25

Tree Decompositions

A tree decomposition is a tree obtained from an arbitrary graph s.t.

1. **Each vertex must occur in some bag**
2. **For each edge, there is a bag containing both endpoints**
3. **Connected: Subgraph "restricted" to any vertex must be connected**

Graphic

Graphs for Boolean Formulas

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

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

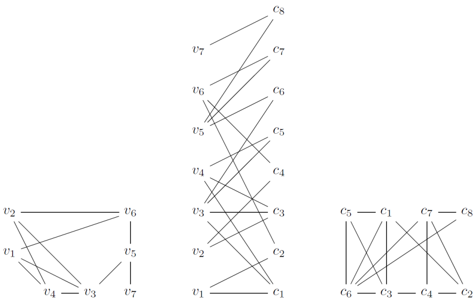
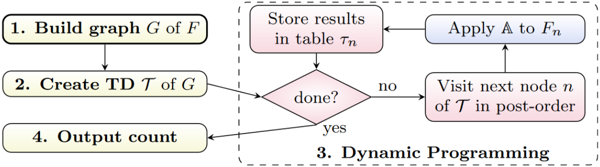
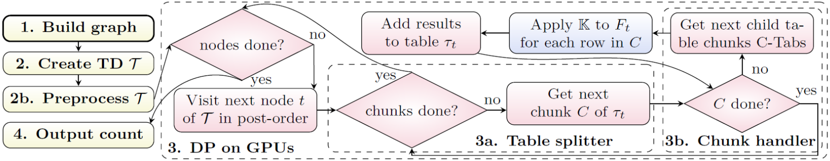


Figure: The primal (left), incidence (middle) and dual (right) graph

gpusat2 - Solving on GPU



- **Customized tree decompositions**
- **Adapted memory-management**
- **Improved precision handling**



[1] 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**

. . .

Diff. debug information and problem types

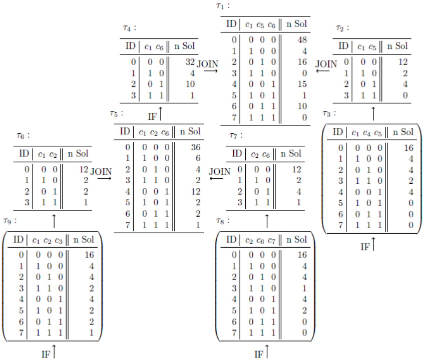
Display user-defined strings where applicable
Different ?????? Grafik bag, Grafik solution

Challenge2

- Wie robust ist die Datenverarbeitung in der Visu Restrictions in strings for ids - Was Gedanken bei der Visu waren

Visualization

Manually for one run



τ_1:

ID	c_1	c_5	c_6	n Sol
0	0	0	0	48
1	1	0	0	4
2	0	1	0	16
3	1	1	0	0
4	0	0	1	15
5	1	0	1	1
6	0	1	1	10
7	1	1	1	0

τ_4: (JOIN →)

ID	c_1	c_6	n Sol
0	0	0	32
1	1	0	4
2	0	1	10
3	1	1	1

τ_2: (JOIN ←)

ID	c_1	c_5	n Sol
0	0	0	12
1	1	0	2
2	0	1	4
3	1	1	0

IF

τ_5:

ID	c_1	c_2	c_6	n Sol
0	0	0	0	36
1	1	0	0	6
2	0	1	0	4
3	1	1	0	2
4	0	0	1	12
5	1	0	1	2
6	0	1	1	2
7	1	1	1	1

τ_6: (JOIN →)

ID	c_1	c_2	n Sol
0	0	0	12
1	1	0	2
2	0	1	2
3	1	1	1

τ_7: (JOIN ←)

ID	c_2	c_6	n Sol
0	0	0	12
1	1	0	2
2	0	1	4
3	1	1	1

τ_3:

ID	c_1	c_4	c_5	n Sol
0	0	0	0	16
1	1	0	0	4
2	0	1	0	4
3	1	1	0	2
4	0	0	1	4
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0

IF

τ_9:

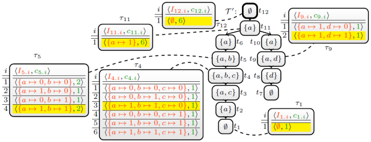
ID	c_1	c_2	c_3	n Sol
0	0	0	0	16
1	1	0	0	4
2	0	1	0	4
3	1	1	0	2
4	0	0	1	4
5	1	0	1	2
6	0	1	1	2
7	1	1	1	1

IF

τ_8:

ID	c_2	c_6	c_7	n Sol
0	0	0	0	16
1	1	0	0	4
2	0	1	0	4
3	1	1	0	1
4	0	0	1	4
5	1	0	1	2
6	0	1	1	0
7	1	1	1	0

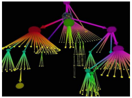
IF



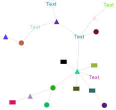
dpdb????[2]

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

© Gephi.org - a tool for data analysts and scientists keen to explore and understand graphs.[3]



Tulip - Better Visualization Through Research. [4]



[5] Vis.js Sigma.js vasturiano/3d-force-graph [6]

=¿ Related Work Schluss / Wiss Arbeiten -¿ Nicht speziell Angeschaut / Format aus Solvern extrahiert - kann trotzdem sehr generisch sein (dpdb speziell)

[3] https://gephi.org/

[4] https://tulip.labri.fr/TulipDrupal/

[5] https://neo4j.com/developer/tools-graph-visualization/

Visualization in Action

MinVC example size 90 (expected 82)

Visualization in Action

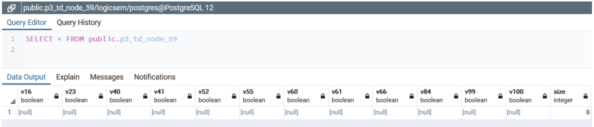
1. **Inspect visualization**

Visualization in Action

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

Visualization in Action

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

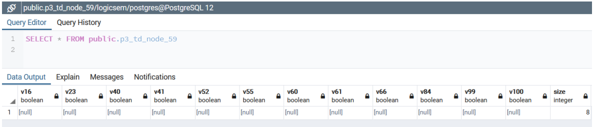



```sql
SELECT * FROM public.p3_td_node_59
```

	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

Visualization in Action

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



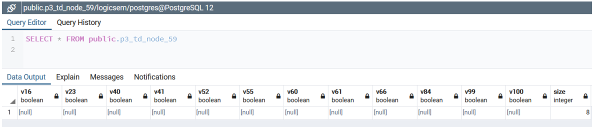

```sql
SELECT * FROM public.p3_td_node_59
```

	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**

Visualization in Action

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


public.p3_td_node_59/logicsem/postgres@PostgreSQL 12

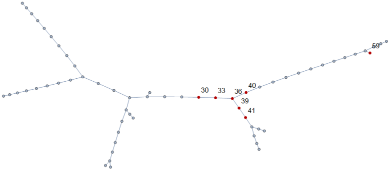
Query Editor | Query History

```
SELECT * FROM public.p3_td_node_59
```

Data Output | Explain | Messages | Notifications

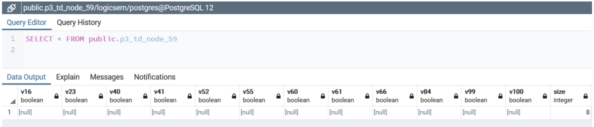
	v16 boolean	v23 boolean	v40 boolean	v41 boolean	v52 boolean	v55 boolean	v60 boolean	v61 boolean	v66 boolean	v84 boolean	v99 boolean	v100 boolean	size integer
1	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	[null]	8

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



Visualization in Action

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



public.p3_td_node_59/logicsem/postgres@PostgreSQL 12

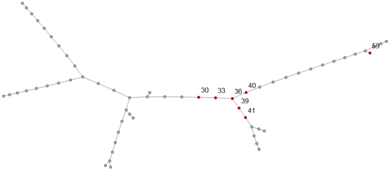
Query Editor | Query History

```
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**

Outlook

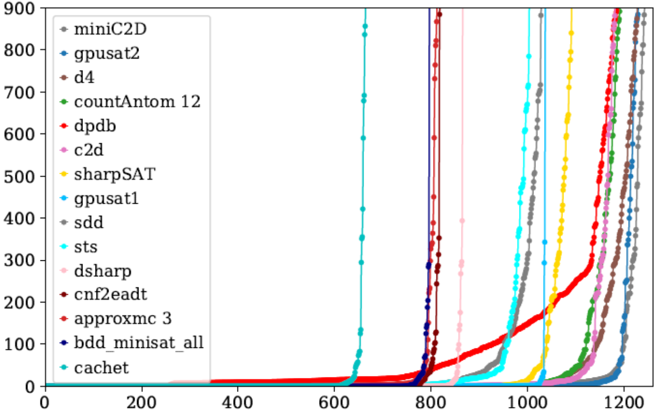
for relevant problems the static graph visualization will become to complicated.
https://data-science-blog.com/blog/2015/07/20/3d-visualisierung-von-graphen/
=¿ Automatische Methoden werden häufig schwerer als gedacht.
Für tiefere Debugging Tasks müsste evtl auch der Ansatz erneuert werden
=¿ Was wären weitere Fragestellungen was man ansehen möchte

Final slide

I would like to thank you for your attention.

Benchmark

Performance of all three programs on #SAT instances:



Visualization

Manually for dpdb

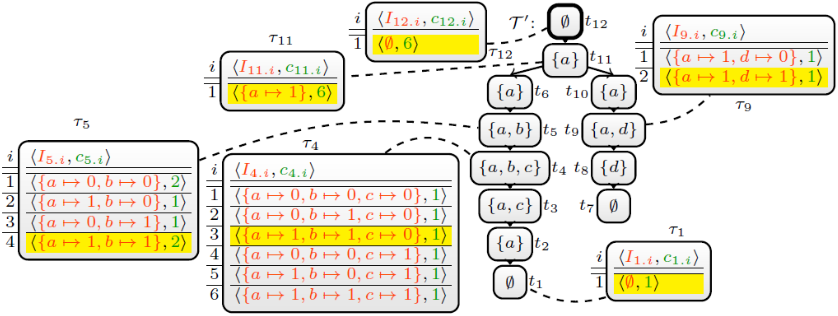


Figure: Handcrafted #SAT example-run from dpdb[7]

[7] "Exploiting Database Management Systems and Treewidth for Counting", Fichte, Hecher, Thier, Woltran

Bibliography