

Visualizing Dynamic Programming On Tree Decompositions

Martin Röbbke
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 Model Counting and various combinatorial problems and are provable efficient at it
- ▶ Implementations of those are competing with modern solvers
- ▶ **But:** those are fairly tedious to implement efficiently
- ▶ Often bugs in the implementation
- ▶ Practical program information quickly becomes very large (GB)

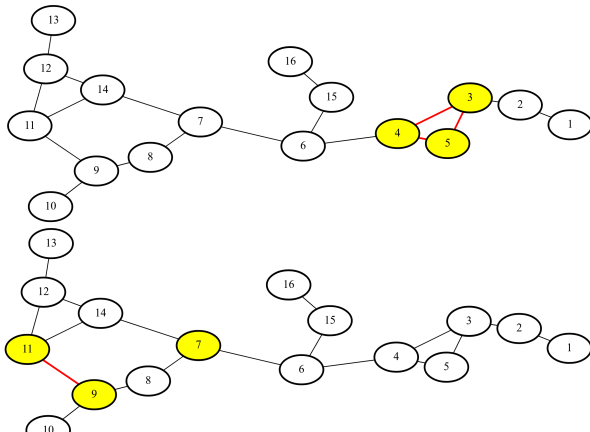
Contribution

Background

The algorithms of interest solve problems of:

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

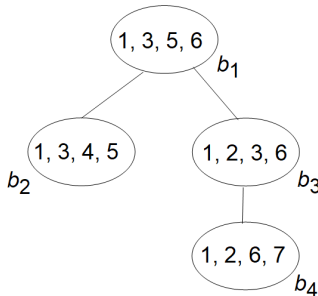
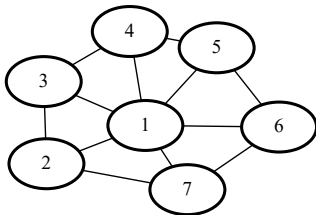
Modelcounting: Instead of **one** solution we want to count **all** solutions.



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 “restricted” to any 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:

$\{c_1 = \{v_1, v_3, \neg v_4\}, c_2 = \{\neg v_1, v_6\}, c_3 = \{\neg v_2, \neg v_3, \neg v_4\}, c_4 = \{\neg v_2, v_6\}, c_5 = \{\neg v_3, \neg v_4\}, c_6 = \{\neg v_3, v_5\}, c_7 = \{\neg v_5, \neg v_6\}, c_8 = \{v_5, v_7\}\}$

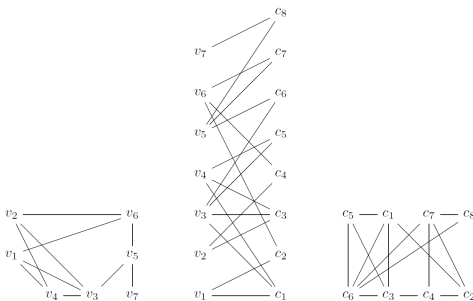
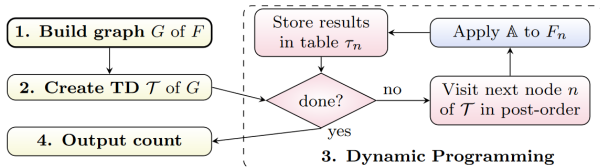


Figure: The primal (left), incidence (middle) and dual (right) graph

gpusat2 - Solving #SAT on GPU



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

⁰Images: Markus Zisser. *Solving the #SAT problem on the GPU with dynamic programming and OpenCL*. Technische Universität Wien, 2018.

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#: (l1,1 OR ... OR l1,k1) AND ... AND (ln,1 OR ... OR ln,kn)
- #aggrExp#:       SUM(cnt) AS cnt
- #extProj#:       τ1.cnt * ... * τℓ.cnt AS cnt

```

(a) Problem #SAT

```

- #εTab#:          SELECT 0 AS card
- #intrTab#:       SELECT 1 AS val UNION ALL 0
- #localProbFilter#: ([u1] OR [v1]) AND ... AND ([un] OR [vn])
- #aggrExp#:       MIN(card) AS card
- #extProj#:       τ1.card + ... + τℓ.card - (Σi=1ℓ |χ(ti) ∩ {a1}| - 1) *
                  τ1. [a1] - ... - (Σi=1ℓ |χ(ti) ∩ {ak}| - 1) * τ1. [ak]

```

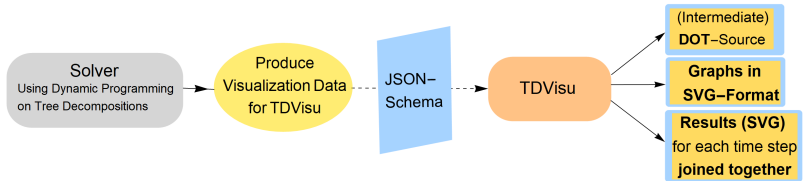
(b) Problem MinVC

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

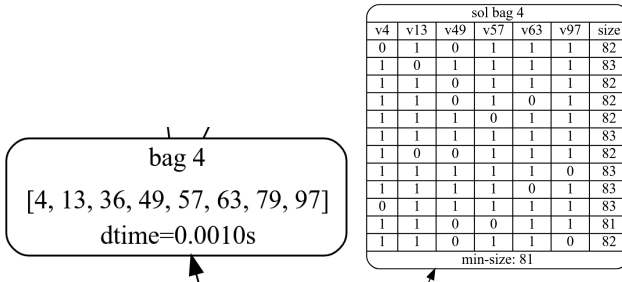
...

⁰“Exploiting Database Management Systems and Treewidth for Counting”,
Johannes Fichte et al. doi: 10.1007/978-3-030-39197-3_10.

Running tdvisu



TDVisu producing flexible and further processable formats



Challenge2

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

Visualization in Action

MinVC example size 90 (expected 82)

Visualization in Action

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

public.p3_td_node_59/pgsql/postgres@PostgreSQL 12

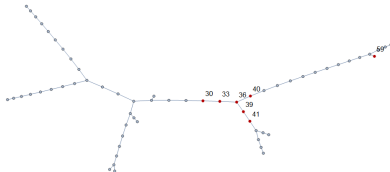
Query Editor Query History

```
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 |

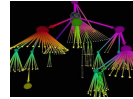
3. Cross reference with standalone tree-decomposition



4. Fix the root cause

Related Work

- ▶ Fichte, Johannes & Hecher, Markus & Morak, Michael & Woltran, Stefan. (2018). Exploiting Treewidth for Projected Model Counting and Its Limits. 10.1007/978-3-319-94144-8_11.
- ▶ Hecher, Markus. (2020). Treewidth-aware Reductions of Normal ASP to SAT - Is Normal ASP Harder than SAT after All?. 485-495. 10.24963/kr.2020/49.
- ▶ 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



Gephi.org¹ Tulip²



³ Vis.js



Sigma.js



vasturiano/3d-force-graph⁴

Briefly looked up different formats and graph software. 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.

¹<https://gephi.org/> - Tool for data analysts and scientists keen to explore and understand graphs.

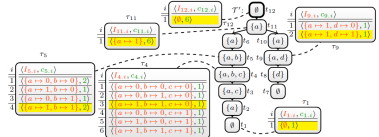
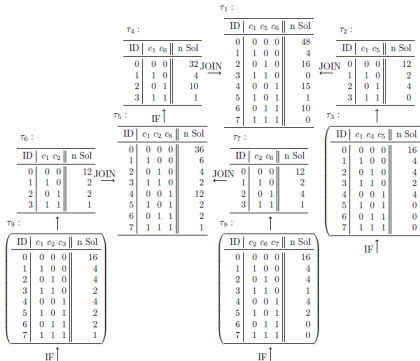
²tulip.labri.fr/TulipDrupal/ - Better Visualization Through Research.

³<https://neo4j.com/developer/tools-graph-visualization/>

⁴<https://github.com/vasturiano/3d-force-graph>

Visualization

Manually for one run



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

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

Outlook

Static \rightarrow Dynamic

Interesting Questions :

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

Summary

This thesis created tdvisu as a tool that

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

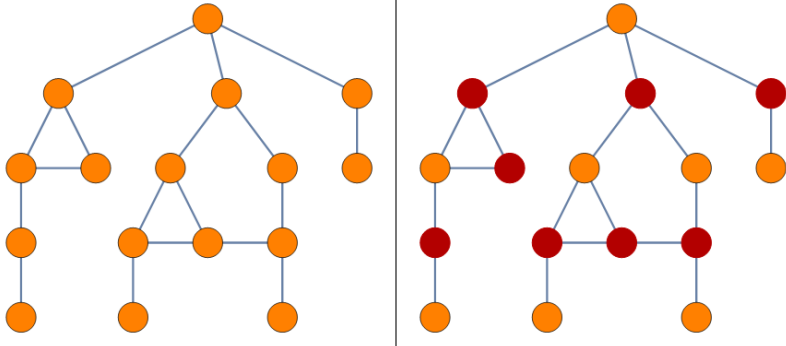
For further research it provides

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

Bibliography

See the citations in the thesis.

MinVC for example graph



Visualization

Manually for dpdb

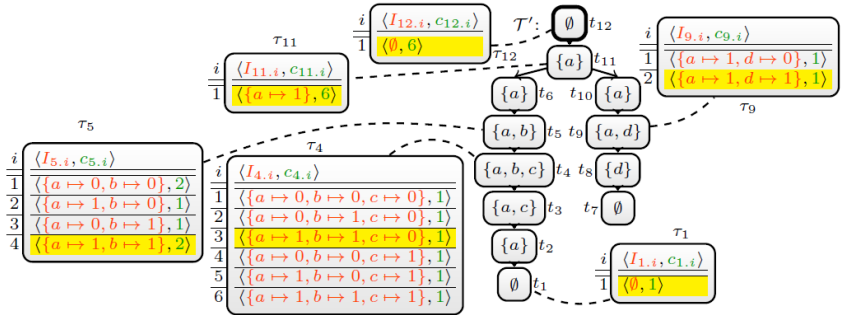
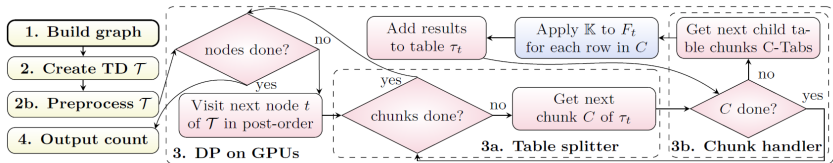


Figure: Handcrafted #SAT example-run from dpdb⁵

⁵"Exploiting Database Management Systems and Treewidth for Counting",
Fichte, Hecher, Thier, Woltran



Benchmark

Performance of all three programs on #SAT instances:

