Visualizing Dynamic Programming On Tree Decompositions
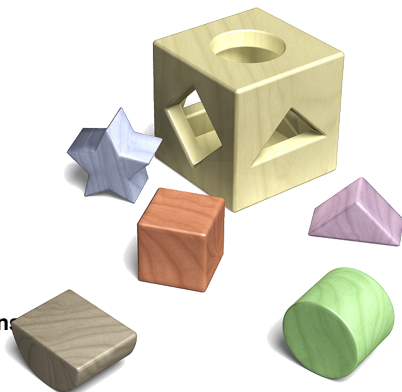
Martin Röbke
Fakultät Informatik
Technische Universität Dresden
Germany

► **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**

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 <u>bag</u>**
2. **For each edge, there is a bag containing both endpoints**
3. **<u>Connected</u>: Subgraph "restricted" to any vertex must be connected**

Graphic
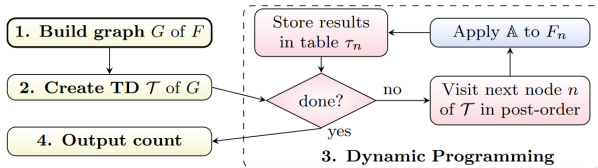
Graphs for Boolean Formulas

▶ **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 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**

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

      • • •

- #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#: $(\llbracket u_1 \rrbracket$ OR $\llbracket v_1 \rrbracket)$ AND ... AND $(\llbracket u_n \rrbracket$ OR $\llbracket v_n \rrbracket)$
- #aggrExp#:   MIN(card) AS card
- #extProj#: τ_1.card + ... + τ_ℓ.card - $(\Sigma_{i=1}^{\ell}|\chi(t_i) \cap \{a_1\}| - 1)$ *
                     $\tau_1.\llbracket a_1 \rrbracket - ... - (\Sigma_{i=1}^{\ell}|\chi(t_i) \cap \{a_k\}| - 1)$ * $\tau_1.\llbracket a_k \rrbracket$

(b) Problem MinVC

Generisches Datenformat /Strings Visu
=¿ Anwendungen

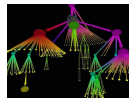=¿ Wie robust ist die Datenverarbeitung in der Visu =¿ Was Gedanken bei der Visu waren

Challenge3

© Gephi.org - a tool for data analysts and scientists keen to explore and understand graphs.[2]

Tulip - Better Visualization Through Research. [3]

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

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

[2]https://gephi.org/
[3]https://tulip.labri.fr/TulipDrupal/
[4]https://neo4i.com/developer/tools-graph-visualization/

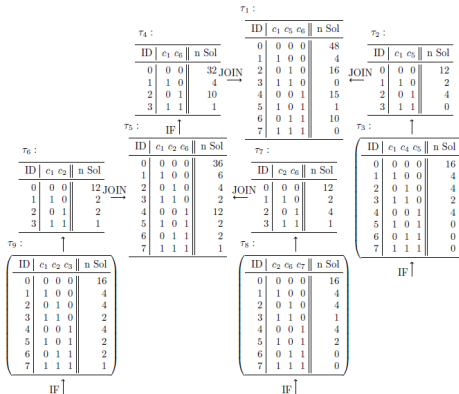# Visualization
Manually for gpusat

Figure: Handcrafted #SAT example-run from Markus Zisser[6]

=¿ Dynamic programming Grafiken / Verlaufsschema Kurz erklären -¿ Beweise dazu sind
aufwändig und recht speziell /

[6]"Solving #SAT on the GPU with Dynamic Programming and OpenCL".

Outlook

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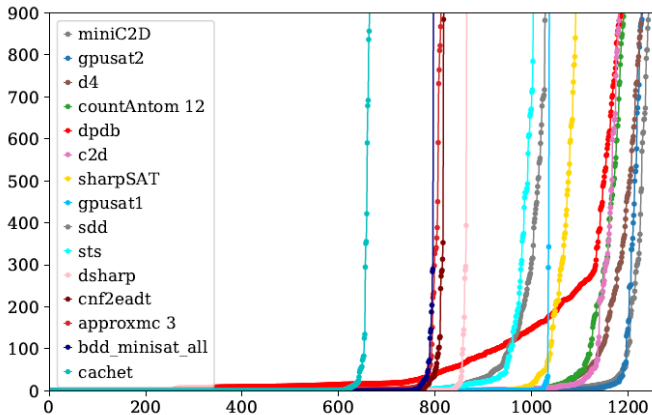https://data-science-blog.com/blog/2015/07/20/3d-visualisierung-von-graphen/
=¿ Automatische Methoden werden häufig schwerer als gedacht.
Für tiefere Debugging Tasks müsste evtl auch der Ansatz erneuert werden
=¿ Was wären weitere Fragestellungen was man ansehen möchte

Benchmark

Performance of all three programs on #SAT instances:
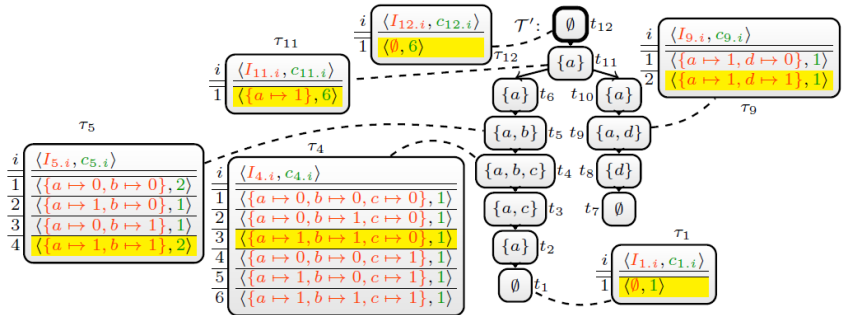
Visualization

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