

Visualizing Dynamic Programming On Tree Decompositions

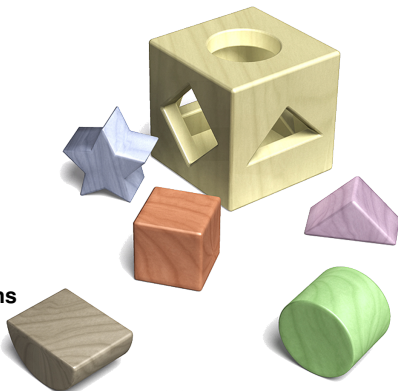
Martin Rübke

Fakultät Informatik

Technische Universität Dresden

Germany

- ▶ **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 soon gets very large
- ▶ Finding the cause of the problem is a time consuming challenge

Background

=¿ Kombinatorische Prob. (lösbare #P Klasse schwerer (teilweise deutlich komplexer =¿
Referenz Projected Model Counting Markus (hybride Erw dpdb))) - Algos

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**

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

Example: Vertex-Cover problem

gpusat2 - Improving Upon Previous Ideas

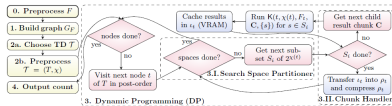


Figure 1: Architecture of our DP-based solver for parallel execution. Yellow colored boxes indicate tasks that are required as initial step for the DP-run or to finally read the model count from the computed results. The parts framed by a dashed box illustrate the DP-part. Boxes colored in red indicate computations that run on the CPU. Boxes colored in blue indicate computations that are executed on the GPU (with waiting CPU).

- only primal graph (simpler solving DP)
- customized tree decompositions
- adapted memory-management
- improved precision handling

Using databases for intermediate results

Generator SQL Qs = \dot{c} Datenbank Templating in Python

```
- #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

► SAT

► #SAT

► Vertex cover

```
- #εTab#:      SELECT 1 AS cnt
- #intrTab#:   SELECT 1 AS val UNION ALL ... UNION ALL 0
- #localProbFilter#: NOT ([u1] = [v1]) AND ... AND NOT ([un] = [vn])
- #aggrExp#:   SUM(cnt) AS cnt
- #extProj#:   τ1.cnt * ... * τℓ.cnt AS cnt
```

(b) Problem #o-Col

```
- #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]
```

(c) Problem MinVC

github: https://github.com/hmarkus/dp_on_dbs

Challenge1

Generisches Datenformat /Strings Visu
Arbeits Space groß
=¿ Anwendungen

Challenge2

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

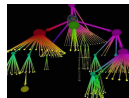
Challenge3



© Gephi.org - a tool for data analysts and scientists keen to explore and understand graphs.¹



LaBRI
université
BORDEAUX



Tulip - Better Visualization Through Research.²



³ Vis.js



Sigma.js



vasturiano/3d-force-graph⁴

=₂ Related Work Schluss / Wiss Arbeiten -₂ Nicht speziell Angeschaut / Format aus Solvern extrahiert - kann trotzdem sehr generisch sein (dpdb speziell)

¹<https://gephi.org/>

²<https://tulip.labri.fr/TulipDrupal/>

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

Visualization

Manually for gpusat

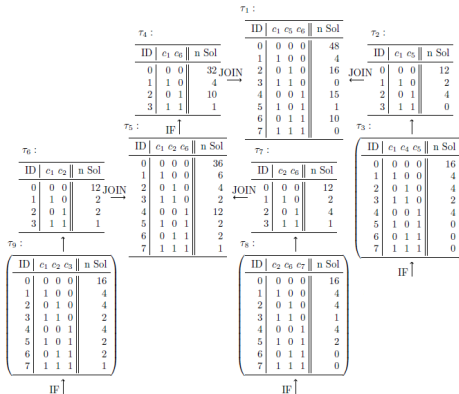


Figure: Handcrafted #SAT example-run from Markus Zisser⁵

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

⁵"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/>

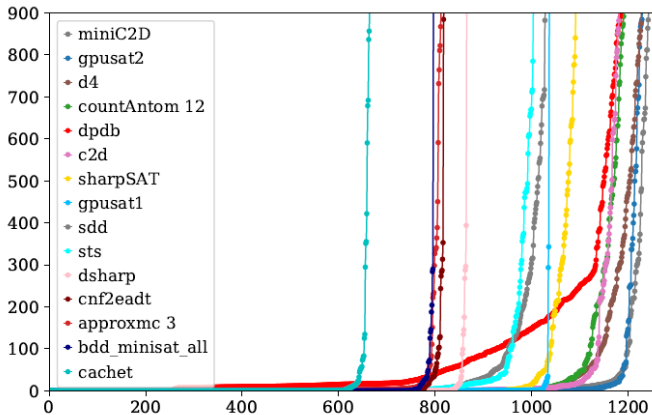
=¿ Automatische Methoden werden häufig schwerer als gedacht.

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

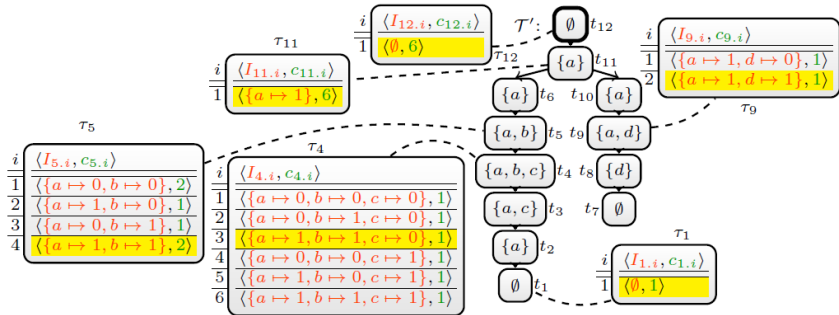


Figure: Handcrafted #SAT example-run from dpdb⁶

⁶"Exploiting Database Management Systems and Treewidth for Counting",
Fichte, Hecher, Thier, Woltran

Bibliography

