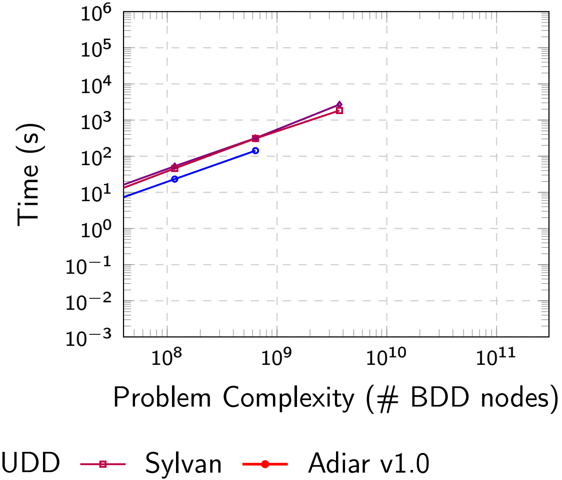
Predicting Memory Demands of BDD Operations using Maximum Graph Cuts

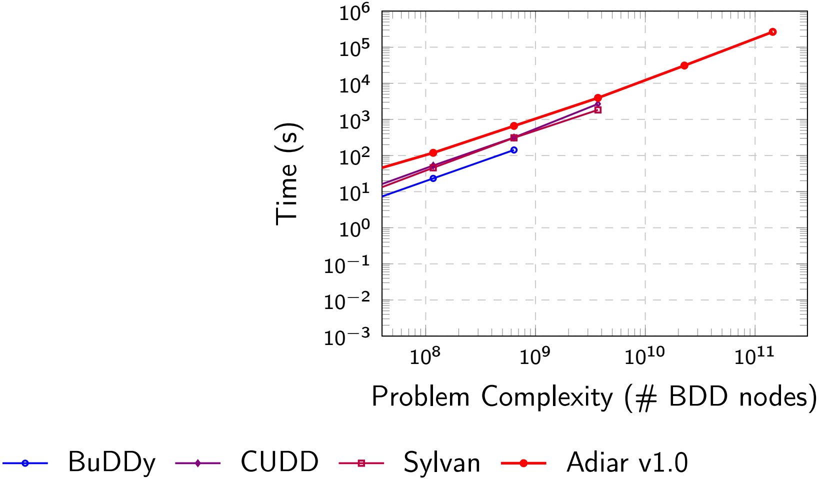
Steffan Christ Sølvsten and Jaco van de Pol

ATVA 2023

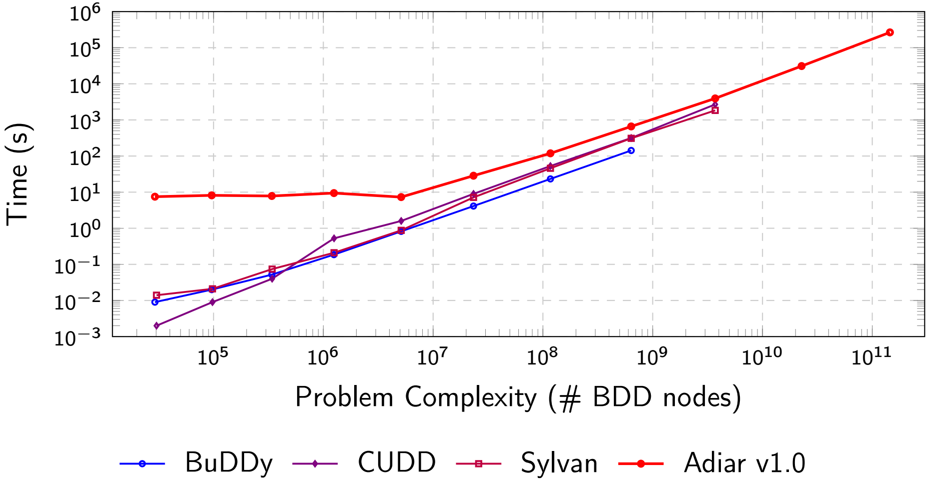
AARHUS
UNIVERSITY



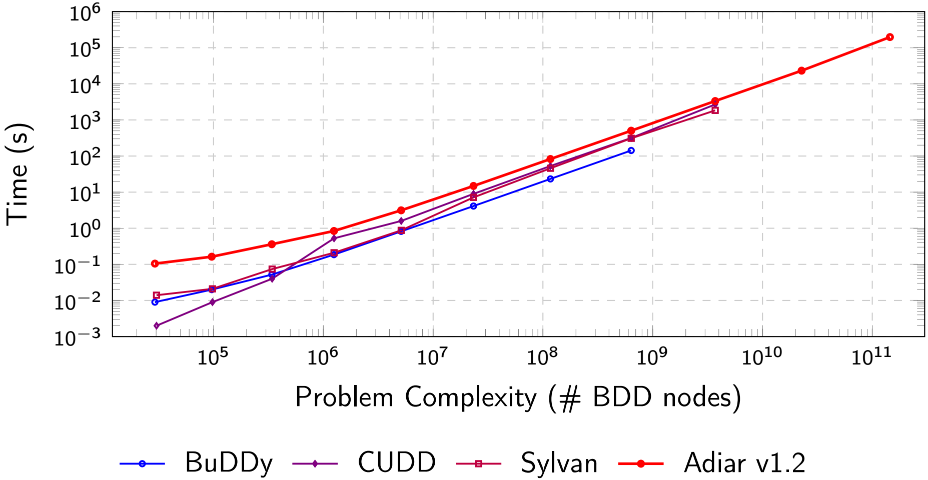
Running Time to solve *N-Queens* problems.



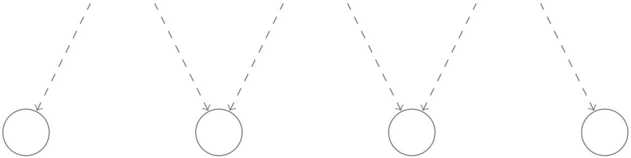
Running Time to solve *N-Queens* problems.

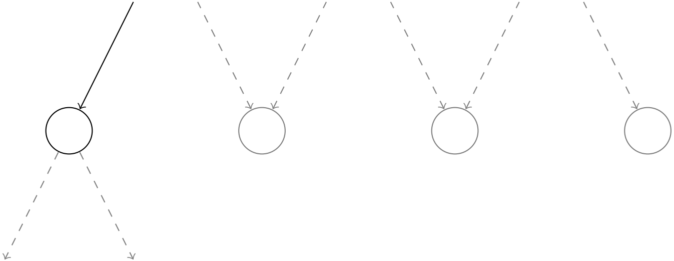


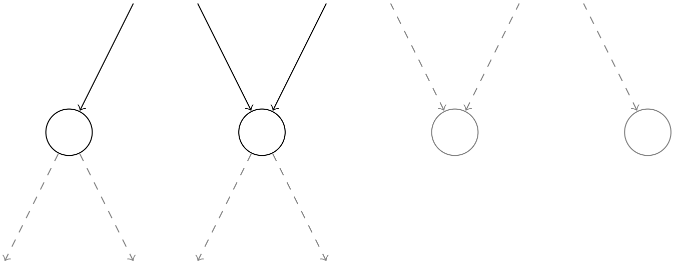
Running Time to solve *N-Queens* problems.

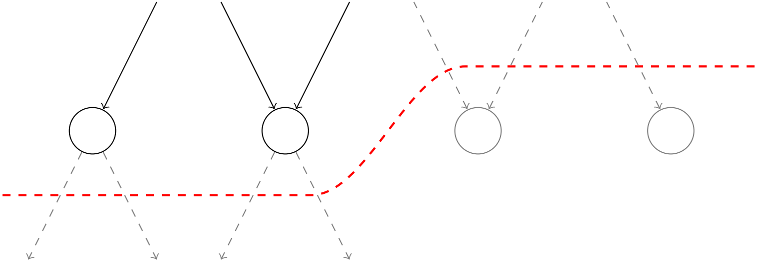


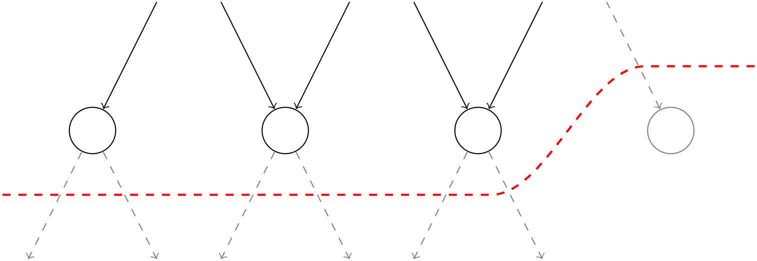
Running Time to solve *N-Queens* problems.

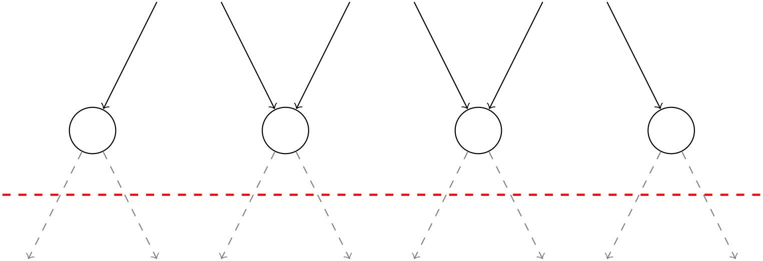




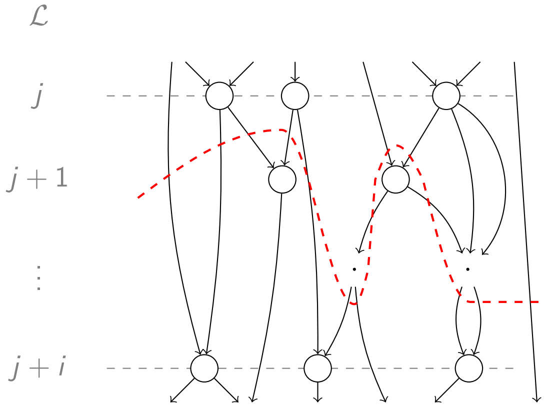




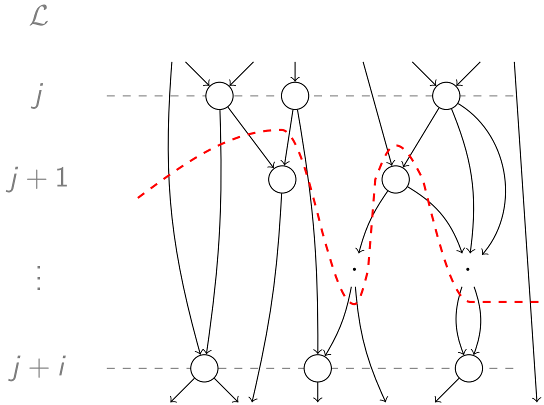




i-level cut



i-level cut

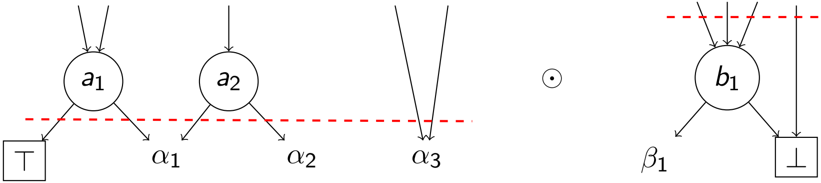


Lemma (Sølvsten, Van de Pol 2023)
The maximum i-level cut problem is in P for $i \in \{1, 2\}$.

Theorem (Lampis, Kaouri, Mitsou 2011)
The maximum i-level cut problem is NP-complete for $i \geq 4$.

Theorem (Sølvsten, Van de Pol 2023)
Given maximum 2-level cuts size C_f for f and C_g for g, the maximum 2-level cut for $f \odot g$ is less than or equal to $C_f \cdot C_g$.

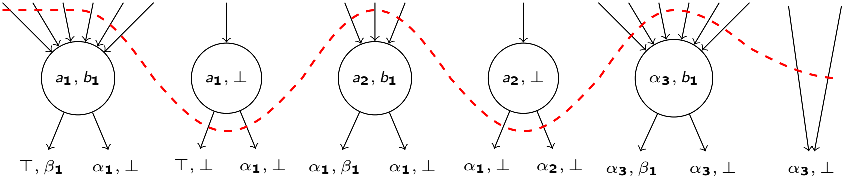
Proof.



□

Theorem (**Sølvsten, Van de Pol 2023**)
Given maximum 2-level cuts size C_f for f and C_g for g, the maximum 2-level cut for $f \odot g$ is less than or equal to $C_f \cdot C_g$.

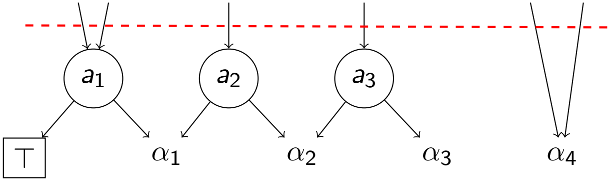
Proof.



□

Lemma (Sølvsten, Van de Pol 2023)
The maximum 2*-level cut for* f *is at most* $\frac{3}{2}$ *larger than its maximum* 1*-level cut.*

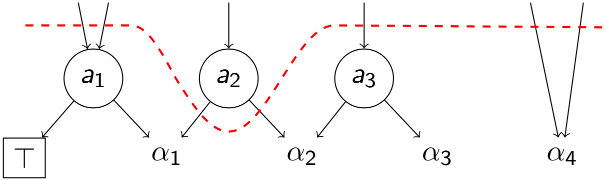
Proof.
The maximum 1-level cut bounds the number of available in-going and out-going edges.



□

Lemma (Sølvsten, Van de Pol 2023)
The maximum 2-level cut for f is at most $\frac{3}{2}$ larger than its maximum 1-level cut.

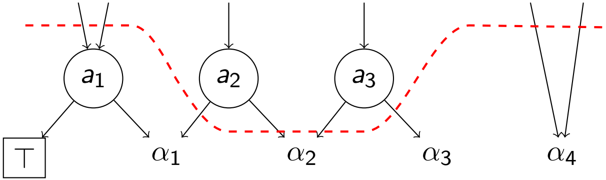
Proof.
The maximum 1-level cut bounds the number of available in-going and out-going edges.



□

Lemma (Sølvsten, Van de Pol 2023)
The maximum 2-level cut for f is at most $\frac{3}{2}$ larger than its maximum 1-level cut.

Proof.
The maximum 1-level cut bounds the number of available in-going and out-going edges.



□

Possible to process a

1.1 GiB BDD

with only

128 MiB Memory

Adiar v1.0 : 56.5 hours

Running time to verify the 15 smallest EPFL instances.

Adiar v1.0 : 56.5 hours

Adiar v1.2 : 4.0 hours (−93%)[1]

Running time to verify the 15 smallest EPFL instances.

[1] 52.1 of these hours were saved on just verifying the `sin` circuit alone.

Steffan Christ Sølvsten

✉ `soelvsten@cs.au.dk`

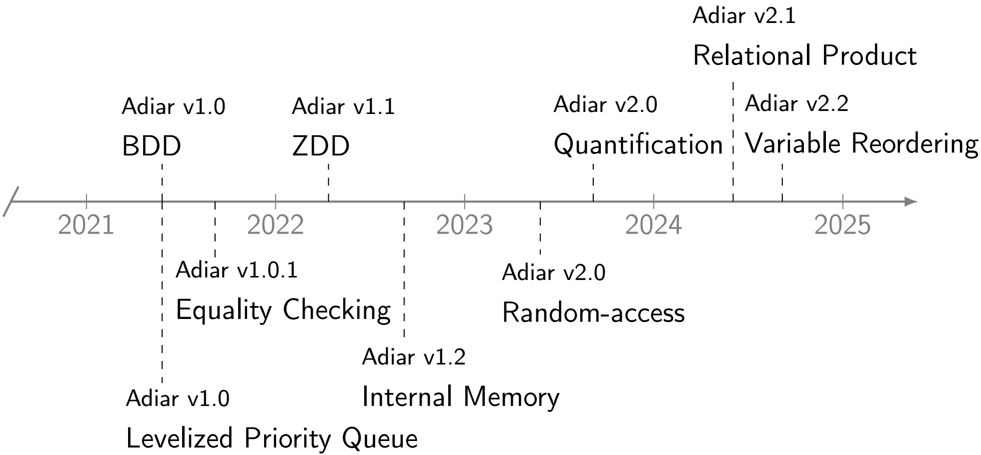
🌐 ssoelvsten.github.io

Adiar

</> github.com/ssoelvsten/adiar

📕 ssoelvsten.github.io/adiar

AARHUS
UNIVERSITY



Adiar v1.0
BDD

Adiar v1.0
Levelized Priority Queue

Adiar v1.0.1
Equality Checking

Adiar v1.1
ZDD

Adiar v1.2
Internal Memory

Adiar v2.0
Random-access

Adiar v2.0
Quantification

Adiar v2.1
Relational Product

Adiar v2.2
Variable Reordering

Steffan Christ Sølvsten

✉ `soelvsten@cs.au.dk`

🌐 ssoelvsten.github.io

Adiar

</> github.com/ssoelvsten/adiar

📖 ssoelvsten.github.io/adiar

AARHUS
UNIVERSITY