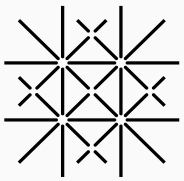
A Comparison of Abstraction Heuristics for Rubik's Cube

Clemens Büchner, Patrick Ferber, Jendrik Seipp, Malte Helmert

June 16, 2022

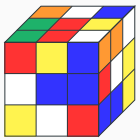


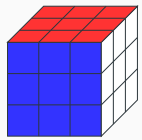
University of Basel

UNIVERSITÄT DES SAARLANDES

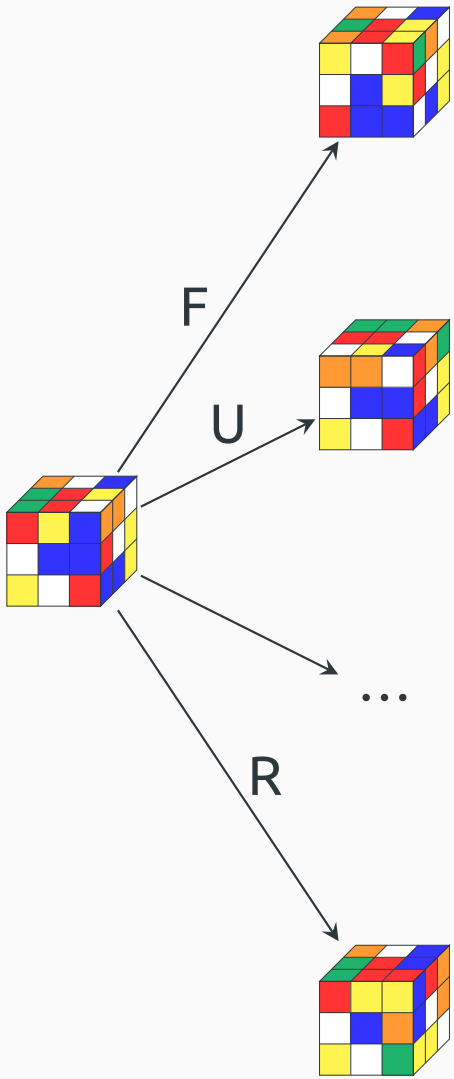
LINKÖPINGS UNIVERSITET

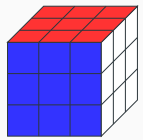
Introduction



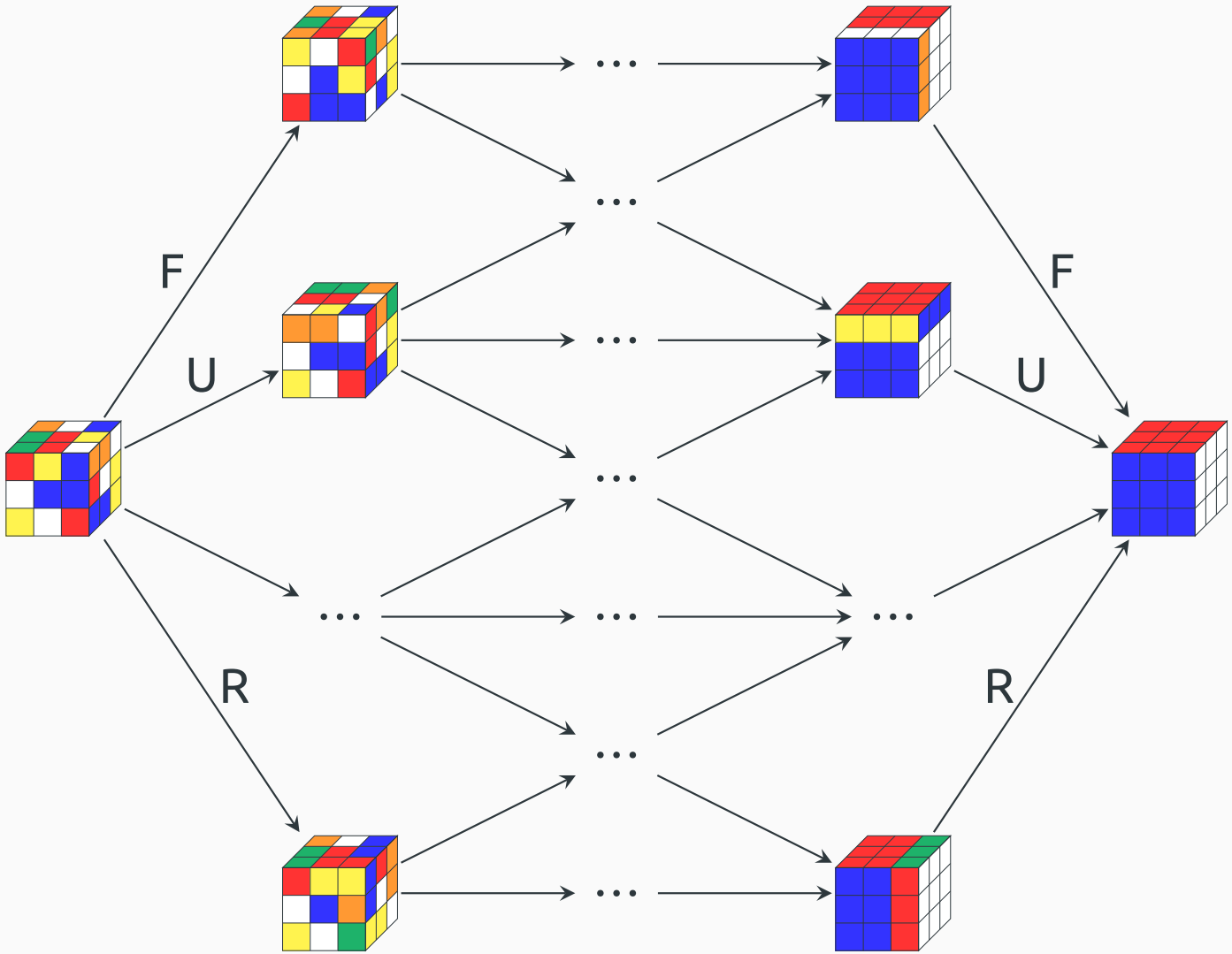


Introduction

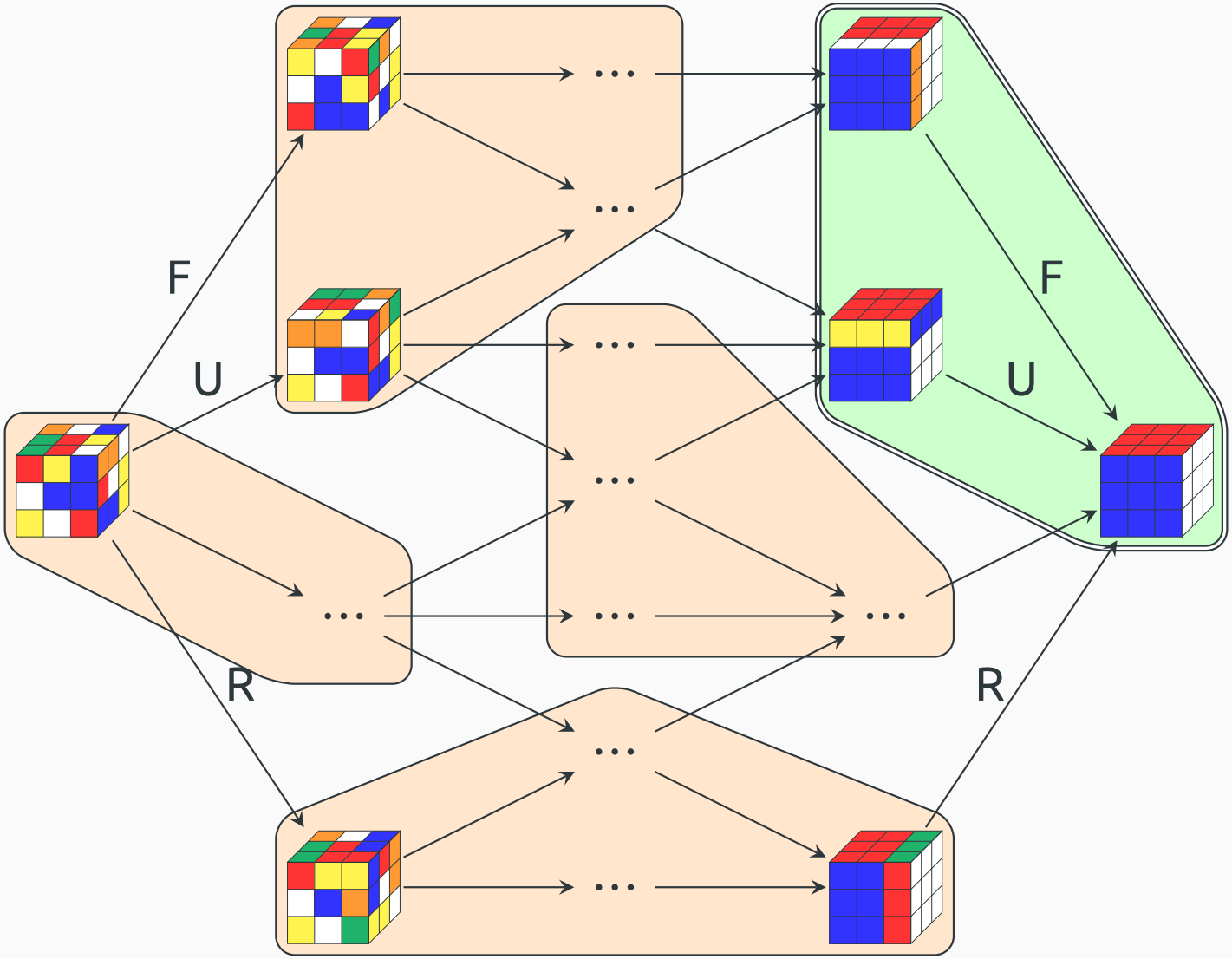




Introduction



Introduction



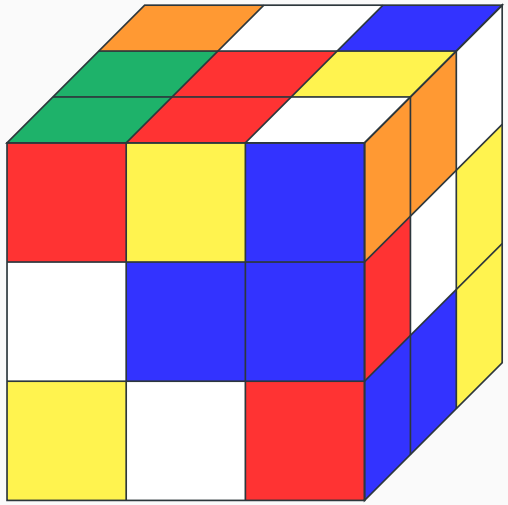
Contributions

- model of Rubik's Cube for general problem solvers
- generate Cartesian abstractions with conditional effects
- evaluate modern abstraction heuristics on Rubik's Cube

Rubik's Cube Model

20 variables, one per corner and edge

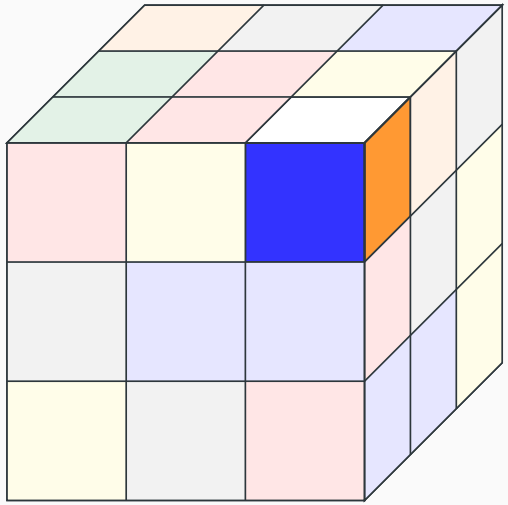
- domains $=$ positions $\times$ orientations



Rubik's Cube Model

20 variables, one per corner and edge

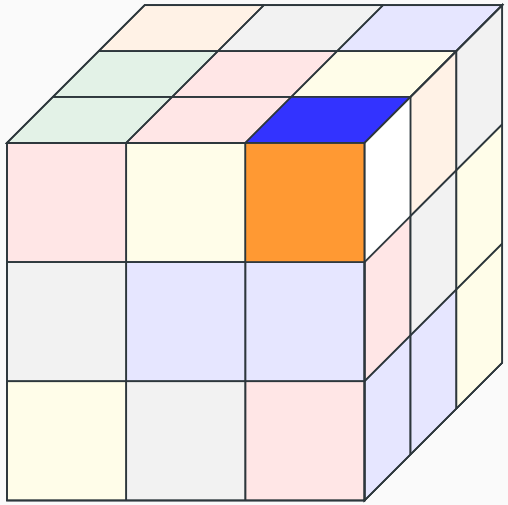
- domains $=$ positions $\times$ orientations



Rubik's Cube Model

20 variables, one per corner and edge

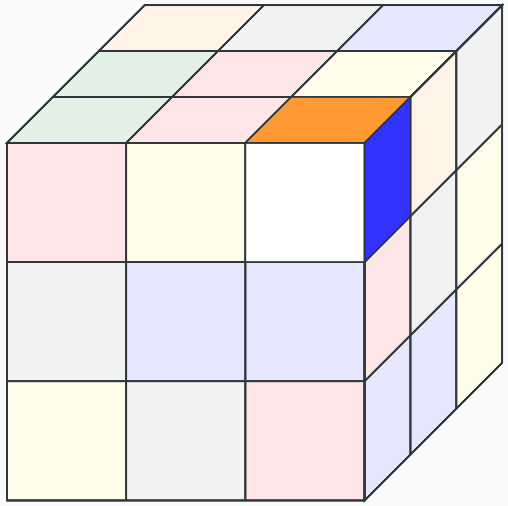
- domains $=$ positions $\times$ orientations



Rubik's Cube Model

20 variables, one per corner and edge

- domains $=$ positions $\times$ orientations



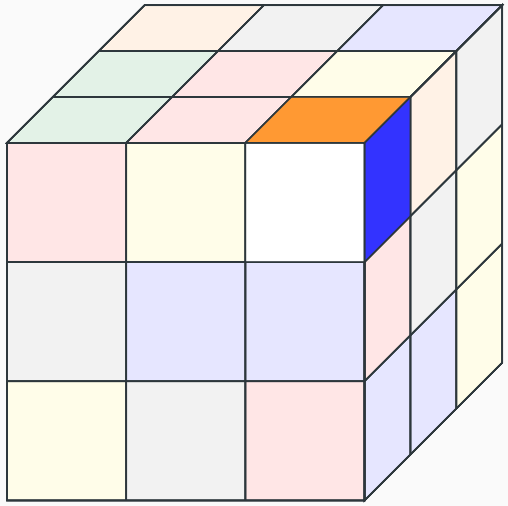
Rubik's Cube Model

20 variables, one per corner and edge

- domains = positions × orientations

18 operators, three per face

- no preconditions
- conditional effect for each cubie at each position in each orientation



Rubik's Cube Model

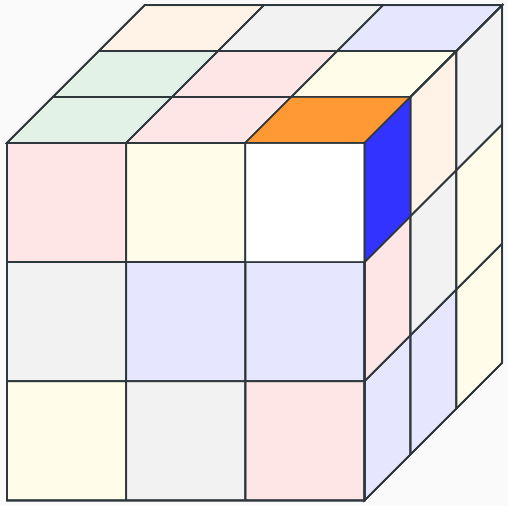
20 variables, one per corner and edge

- domains = positions × orientations

18 operators, three per face

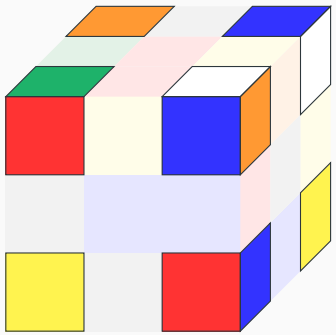
- no preconditions
- conditional effect for each cubie at each position in each orientation

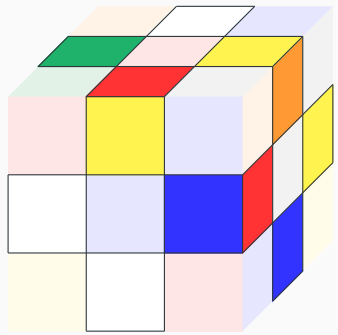
$\approx 4.3 \cdot 10^{19}$ reachable states



Solving Rubik's Cube Optimally

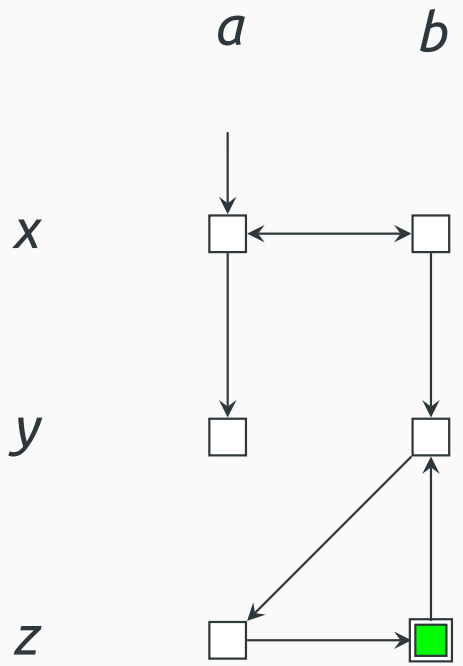
state of the art: pattern database heuristics (Korf 1997)





- split variables to obtain two simpler problems
- compute all goal distances in simplified problems
- use maximum as admissible heuristic in IDA* search

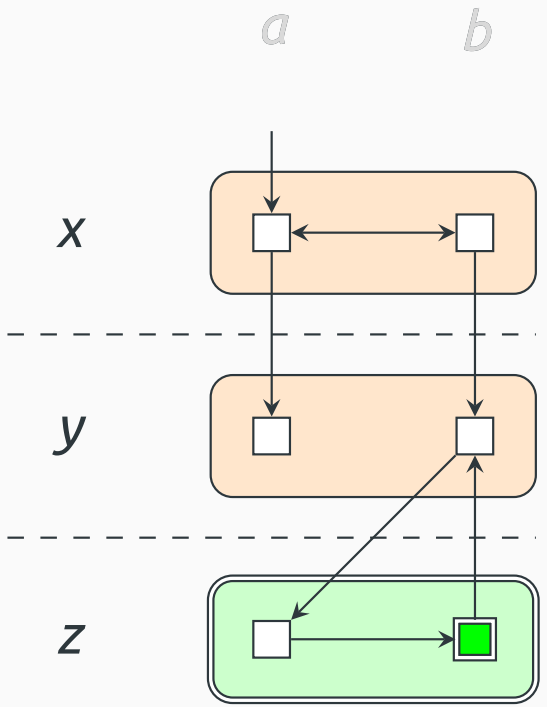
Overview of Abstraction Heuristics



Overview of Abstraction Heuristics

projections $\rightarrow$ PDBs

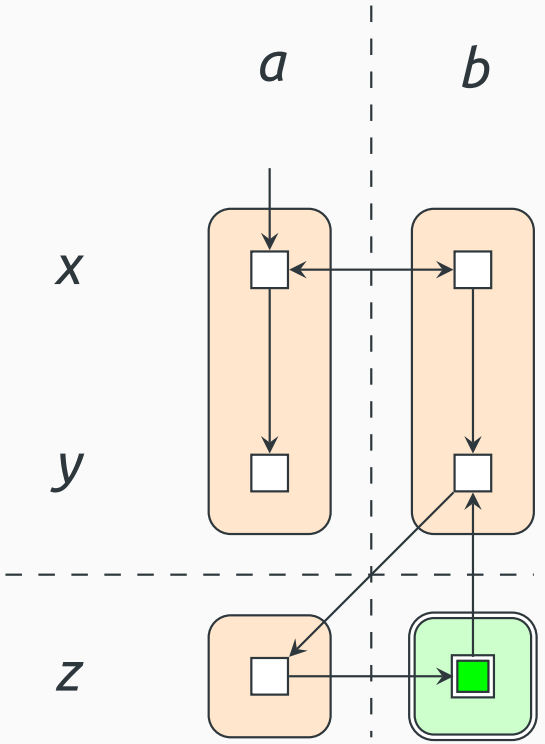
(Culberson and Schaeffer 1998)



Overview of Abstraction Heuristics

projections $\rightarrow$ PDBs
(Culberson and Schaeffer 1998)

domain abstractions
(Hernádvölgyi and Holte 2000)

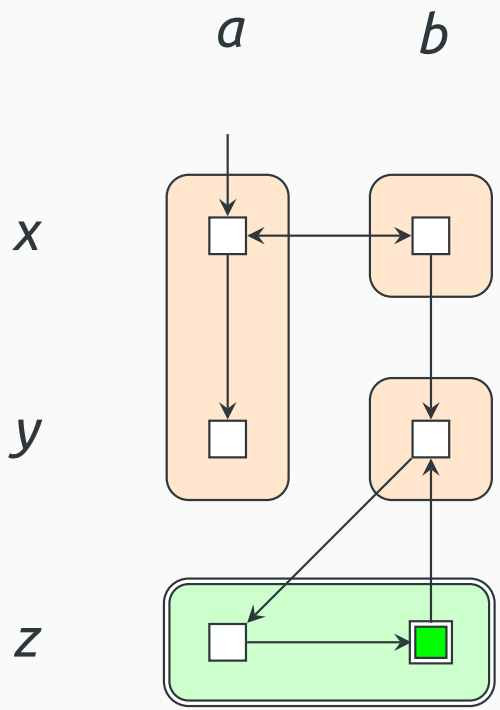


Overview of Abstraction Heuristics

projections $\rightarrow$ PDBs
(Culberson and Schaeffer 1998)

domain abstractions
(Hernádvölgyi and Holte 2000)

Cartesian abstractions
(Seipp and Helmert 2018)



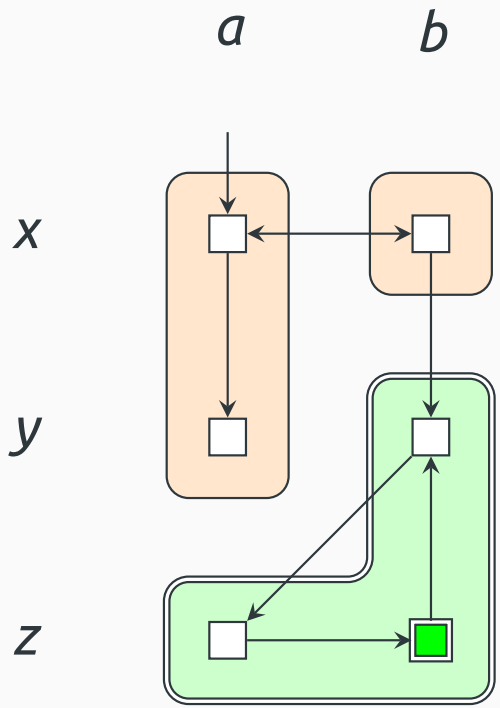
Overview of Abstraction Heuristics

projections $\rightarrow$ PDBs
(Culberson and Schaeffer 1998)

domain abstractions
(Hernádvölgyi and Holte 2000)

Cartesian abstractions
(Seipp and Helmert 2018)

merge-and-shrink abstractions
(Sievers and Helmert 2021)



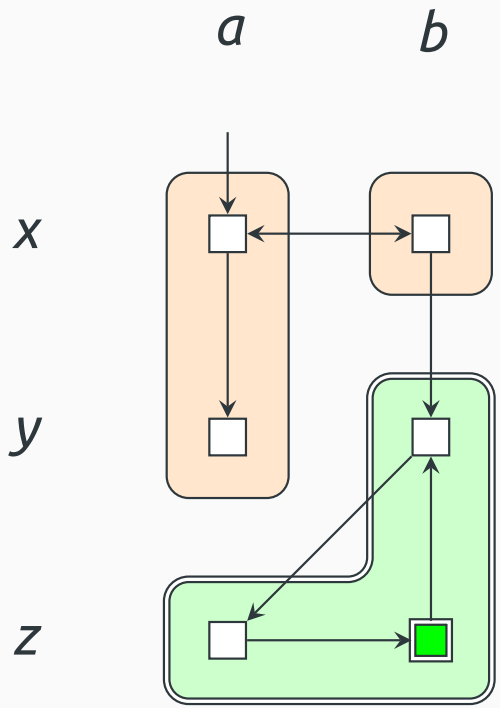
Overview of Abstraction Heuristics

projections $\rightarrow$ PDBs
(Culberson and Schaeffer 1998)

~~domain abstractions~~
~~(Hernádvölgyi and Holte 2000)~~

Cartesian abstractions
(Seipp and Helmert 2018)

merge-and-shrink abstractions
(Sievers and Helmert 2021)

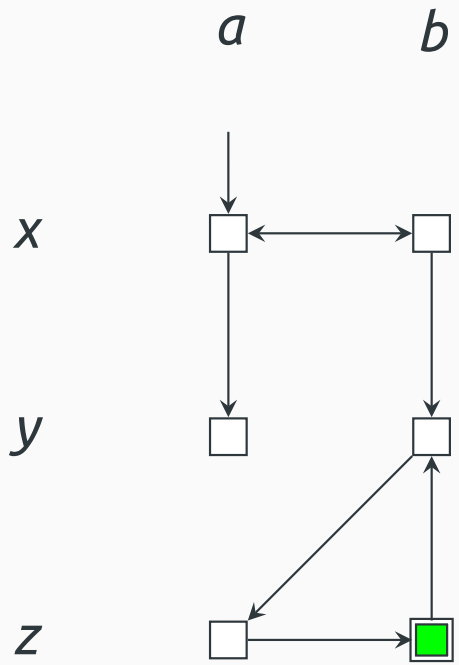


Cartesian CEGAR

Counterexample-Guided Cartesian Abstraction Refinement

Start with coarsest abstraction and iterate:

- find abstract plan
- execute in original
- if fails: fix flaw and repeat
- else: return solution

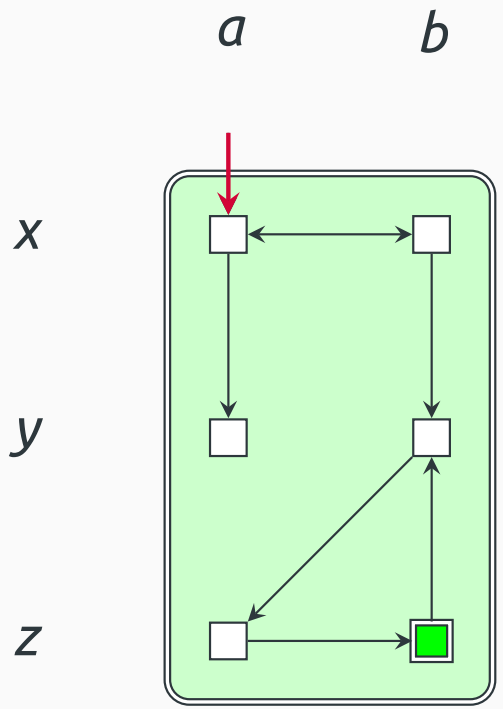


Cartesian CEGAR

Counterexample-Guided Cartesian Abstraction Refinement

Start with coarsest abstraction and iterate:

- find abstract plan
- execute in original
- if fails: fix flaw and repeat
- else: return solution

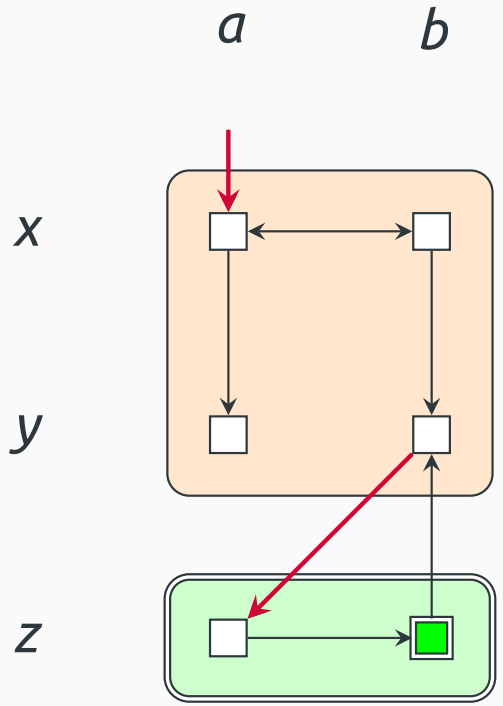


Cartesian CEGAR

Counterexample-Guided Cartesian Abstraction Refinement

Start with coarsest abstraction and iterate:

- find abstract plan
- execute in original
- if fails: fix flaw and repeat
- else: return solution

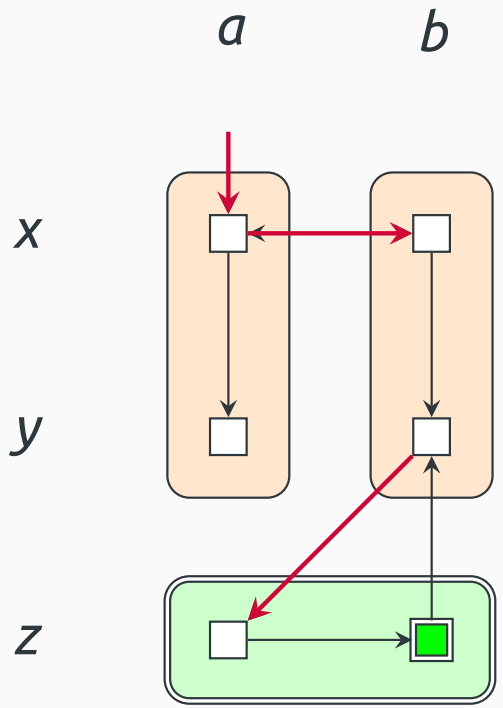


Cartesian CEGAR

Counterexample-Guided Cartesian Abstraction Refinement

Start with coarsest abstraction and iterate:

- find abstract plan
- execute in original
- if fails: fix flaw and repeat
- else: return solution

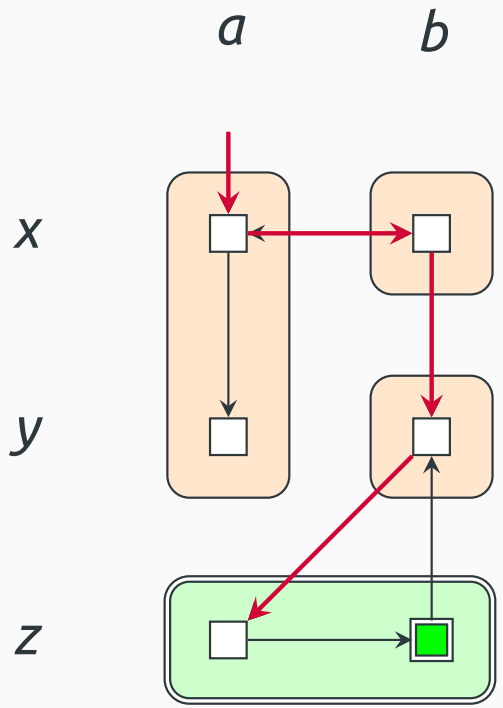


Cartesian CEGAR

Counterexample-Guided Cartesian Abstraction Refinement

Start with coarsest abstraction and iterate:

- find abstract plan
- execute in original
- if fails: fix flaw and repeat
- else: return solution



Cartesian CEGAR and Factored Effect Tasks

Problem

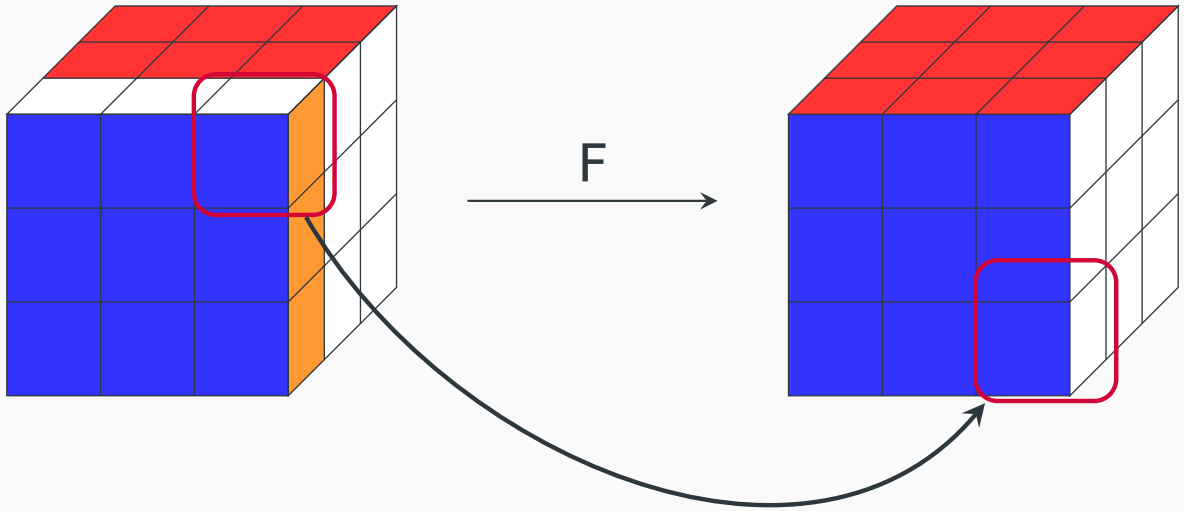
Regression through operators with general conditional effects is not Cartesian.

Cartesian CEGAR and Factored Effect Tasks

Problem

Regression through operators with general conditional effects is not Cartesian.

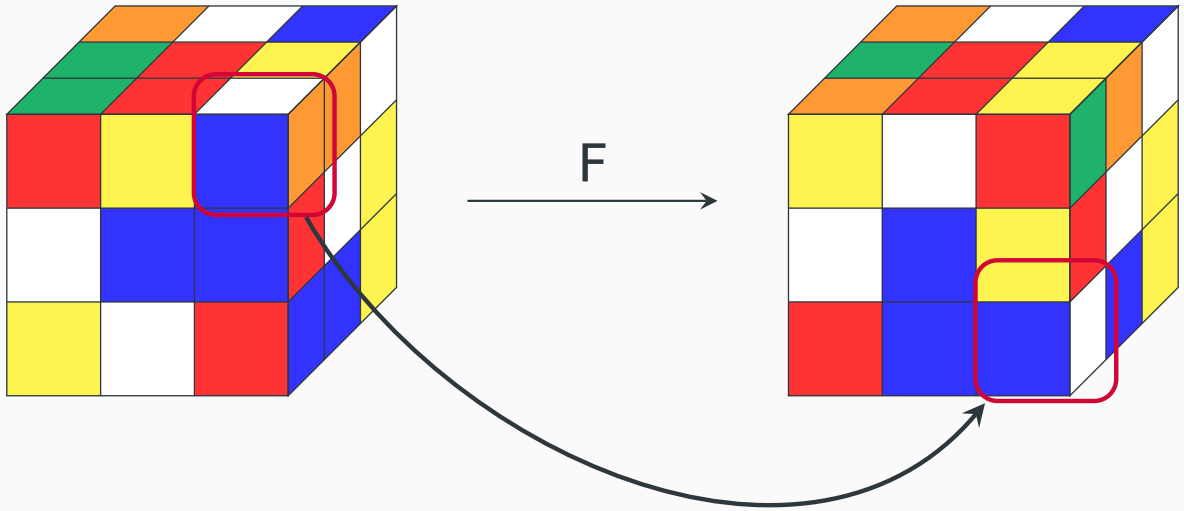
Special case Rubik's Cube: new position and orientation of cubie depends only on old position and orientation of itself.



Problem

Regression through operators with general conditional effects is not Cartesian.

Special case Rubik's Cube: new position and orientation of cubie depends only on old position and orientation of itself.



Cartesian CEGAR and Factored Effect Tasks

Problem

Regression through operators with general conditional effects is not Cartesian.

Special case Rubik's Cube: new position and orientation of cubie depends only on old position and orientation of itself.

Definition (Factored Effect Tasks)

A factored effect operator specifies in each effect condition exactly the variable changed by the effect.
A factored effect task has exclusively factored effect operators.

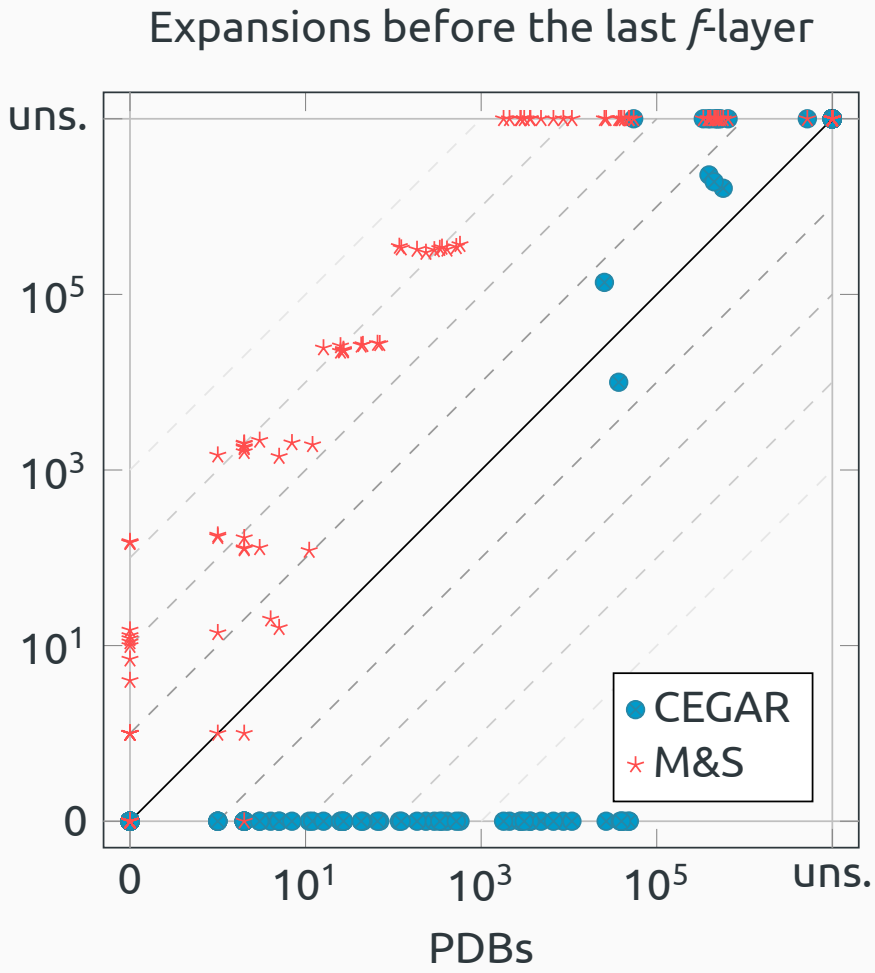
Theorem

Regression through factored effect operators is Cartesian.

Experiments

200 Rubik's Cube problems of increasing difficulty

	coverage
PDBs	123
CEGAR	113
M&S	90



Summary

- new Rubik's Cube benchmarks for planning in SAS$^+$
- Cartesian CEGAR adapted for factored effect tasks
- CEGAR yields perfect heuristic up to certain difficulty
- yet, fewer tasks solved by more general abstractions

Summary

- new Rubik's Cube benchmarks for planning in SAS$^+$
- Cartesian CEGAR adapted for factored effect tasks
- CEGAR yields perfect heuristic up to certain difficulty
- yet, fewer tasks solved by more general abstractions

Future work:

- consider more permutation domains for analysis
- extend with domain abstractions