Abstraction Heuristics for Factored Tasks

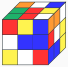
Clemens Büchner[1] Patrick Ferber[1] Jendrik Seipp[2] Malte Helmert[1]

June 4, 2024

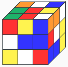
1 University of Basel

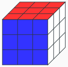
2 Linköpings Universitet

Introduction

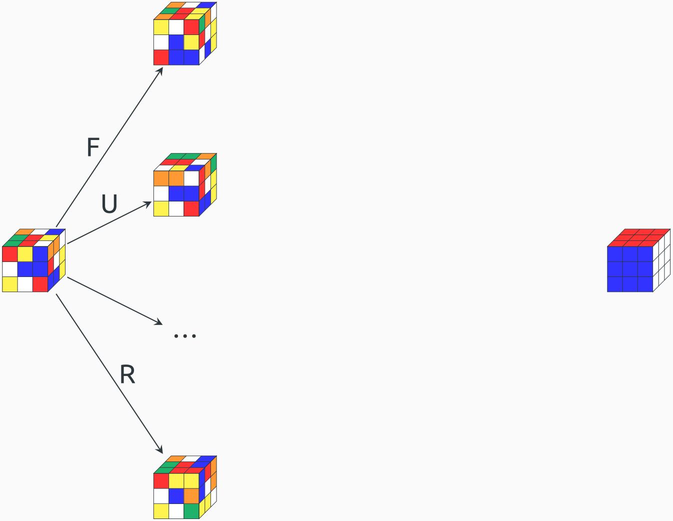


Introduction

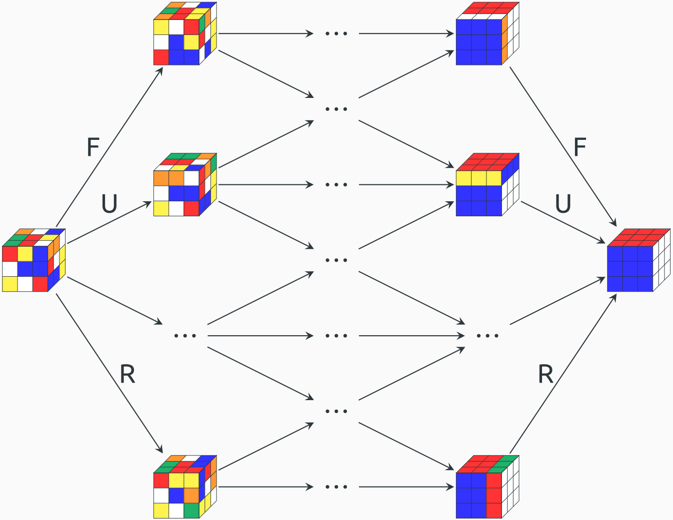




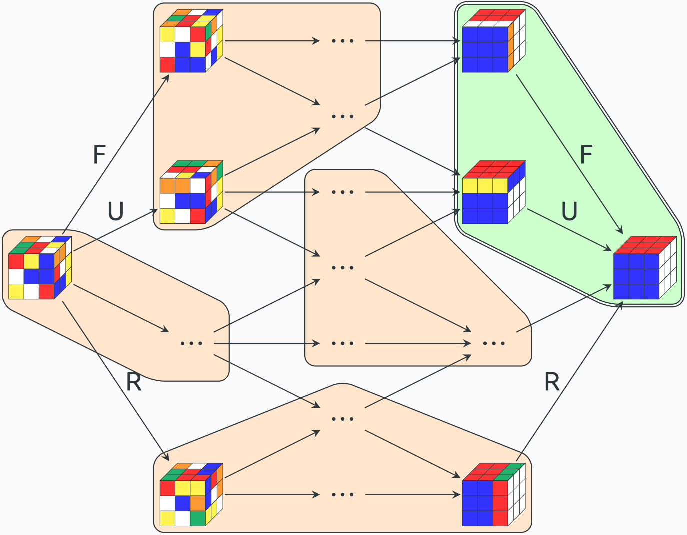
Introduction



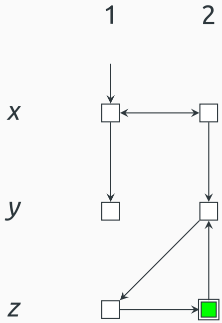
Introduction



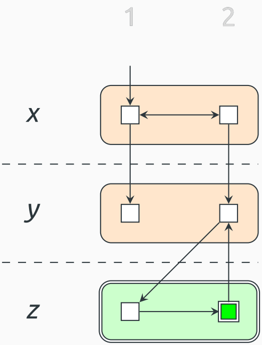
Introduction



Overview of Abstraction Heuristics



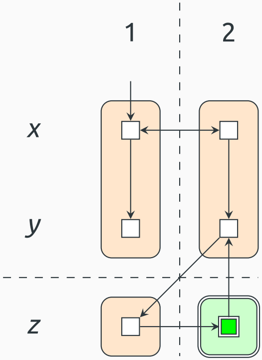
projections/pattern databases
(Culberson and Schaeffer 1998)



Overview of Abstraction Heuristics

projections/pattern databases
(Culberson and Schaeffer 1998)

domain abstractions
(Hernádvölgyi and Holte 2000)

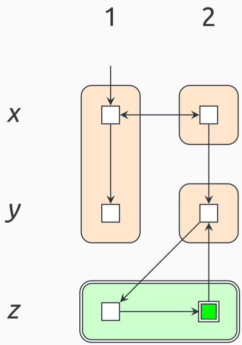


Overview of Abstraction Heuristics

projections/pattern databases
(Culberson and Schaeffer 1998)

domain abstractions
(Hernádvölgyi and Holte 2000)

Cartesian abstractions
(Seipp and Helmert 2013)



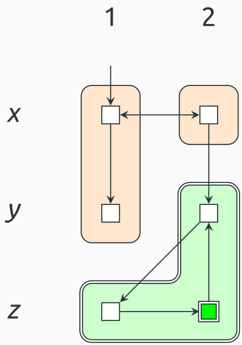
Overview of Abstraction Heuristics

projections/pattern databases
(Culberson and Schaeffer 1998)

domain abstractions
(Hernádvölgyi and Holte 2000)

Cartesian abstractions
(Seipp and Helmert 2013)

merge-and-shrink abstractions
(Dräger, Finkbeiner, and Podelski 2006)



Limitations of Abstraction Heuristics

- efficient domain-independent algorithms for SAS$^+$
- no compact models in SAS$^+$ for some problem domains
- some compact models rely on conditional effects

Issue with Compact Problem Representations

For tasks with general conditional effects, deciding whether a transition exists between two abstract states is NP-hard.

Limitations of Abstraction Heuristics

- efficient domain-independent algorithms for SAS$^+$
- no compact models in SAS$^+$ for some problem domains
- some compact models rely on conditional effects

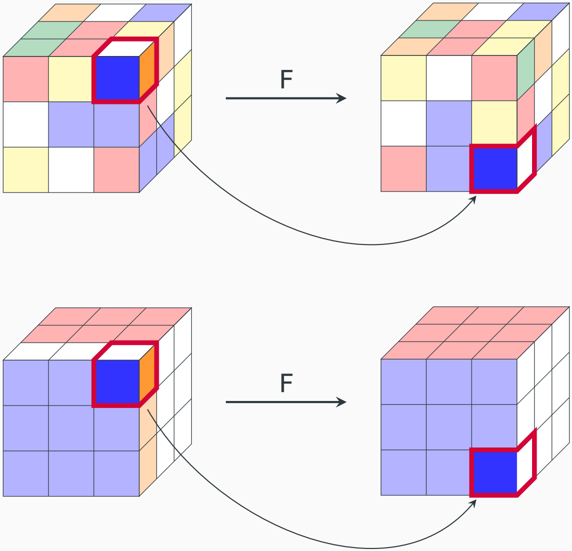
Issue with Compact Problem Representations

For tasks with general conditional effects, deciding whether a transition exists between two abstract states is NP-hard.

Good News!

Abstractions can be computed efficiently for factored tasks.

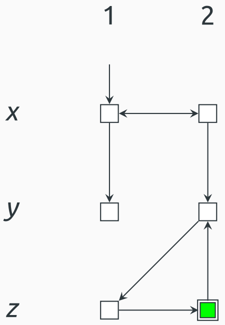
Factored Conditional Effects



Definition (factored task)

A factored task is a 4-tuple $\Pi = \langle \mathcal{V}, \mathcal{O}, I, G \rangle$ with

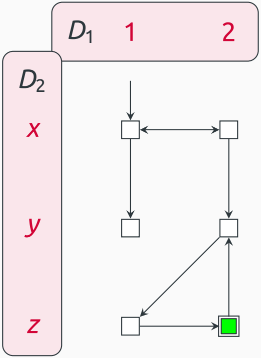
- variable space $\mathcal{V}$
- factored operators $\mathcal{O}$ consisting of
  - factored state relations with
  - associated costs
- factored state sets I and G



Factored Tasks

variable space

$\mathcal{V} = \langle V_1, \dots, V_n \rangle$
with domains $D_1, \dots, D_n$



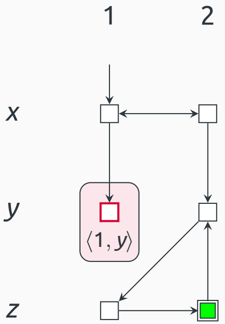
Factored Tasks

variable space

$\mathcal{V} = \langle V_1, \ldots, V_n \rangle$
with domains $D_1, \ldots, D_n$

state

$s = \langle d_1, \ldots, d_n \rangle$
with $d_i \in D_i$



Factored Tasks

variable space

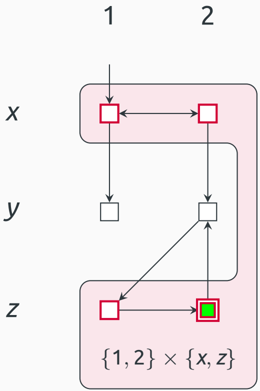
$\mathcal{V} = \langle V_1, \dots, V_n \rangle$
with domains $D_1, \dots, D_n$

state

$s = \langle d_1, \dots, d_n \rangle$
with $d_i \in D_i$

factored state set

$S = S_1 \times \cdots \times S_n$
with $S_i \subseteq D_i$



Factored Tasks

variable space

$\mathcal{V} = \langle V_1, \dots, V_n \rangle$
with domains $D_1, \dots, D_n$

state

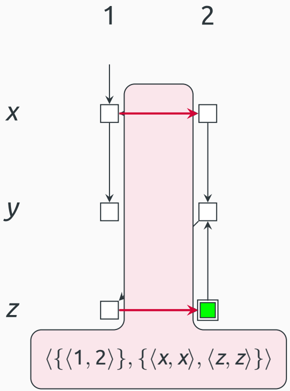
$s = \langle d_1, \dots, d_n \rangle$
with $d_i \in D_i$

factored state set

$S = S_1 \times \cdots \times S_n$
with $S_i \subseteq D_i$

factored state relation

$R = \langle R_1, \dots, R_n \rangle$
with $R_i \subseteq D_i \times D_i$



Properties of Factored Tasks

- alternative view as set of automata
- factored tasks generalize SAS$^+$
- additionally they support limited forms of
  - multiple initial states
  - disjunctive preconditions
  - conditional effects
  - angelic nondeterminism

Properties of Factored Tasks

- alternative view as set of automata
- factored tasks generalize SAS$^+$
- additionally they support limited forms of
  - multiple initial states
  - disjunctive preconditions
  - conditional effects
  - angelic nondeterminism

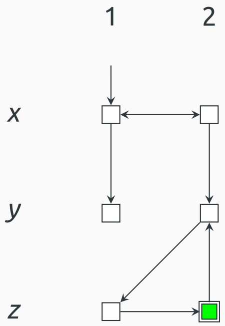
- as general as possible given independent variables
- progression and regression are symmetric
- Cartesian sets are everywhere
  - factored state sets I and G
  - operator preconditions
  - operator postconditions

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

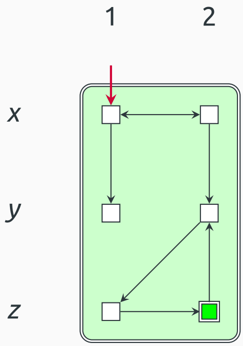


Cartesian CEGAR

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

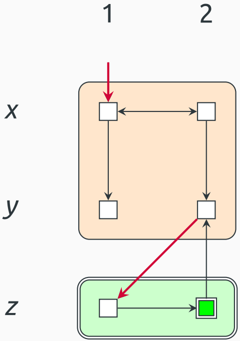


Cartesian CEGAR

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

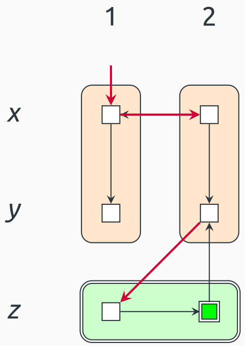


Cartesian CEGAR

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

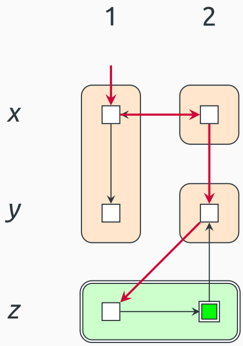


Cartesian CEGAR

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



Components of Cartesian CEGAR

- compact representation of abstract states
- check whether abstract state contains concrete state
- progression for executing plans
- regression for splitting abstract states given flaw

Components of Cartesian CEGAR

- compact representation of abstract states
- check whether abstract state contains concrete state
- progression for executing plans
- regression for splitting abstract states given flaw

Good News!

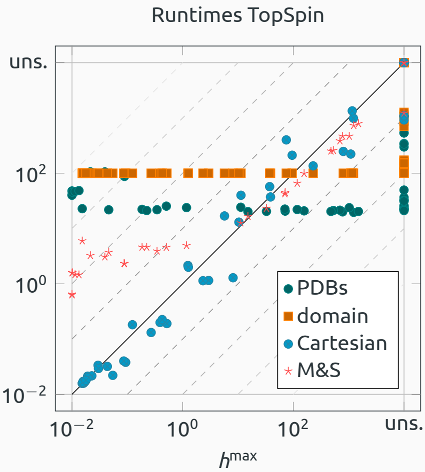
Factored tasks support all of the above efficiently.

- progression and regression yield factored state sets
- not true for tasks with general conditional effects

Experiments

new benchmark set
with 431 tasks

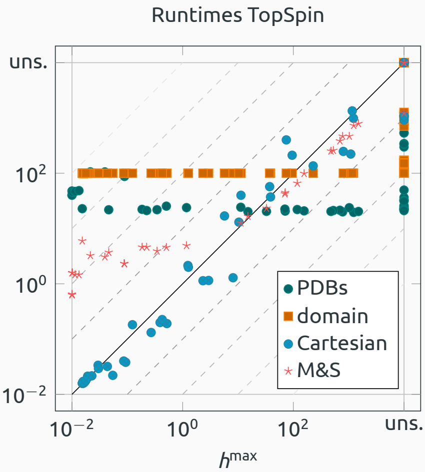
	coverage
PDBs	250
domain abs.	218
Cartesian abs.	189
M&S	175



Experiments

new benchmark set
with 431 tasks

	coverage
PDBs	250
SymBA*	220
domain abs.	218
Cartesian abs.	189
M&S	175
h^{max}	164
LM-Cut	134



Summary

- factored tasks generalize SAS^+
- Cartesian sets are everywhere in factored tasks
- common abstractions work efficiently for factored tasks

Future work:

- efficient abstractions beyond factored tasks
- heuristics for factored tasks beyond abstractions