Pattern Selection using CEGAR

Alexander Rovner

University of Basel

July 31, 2018

Planning Tasks

Definition: Planning Task

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

- $\mathcal{V}$: finite set of variables. Each variable $v \in \mathcal{V}$ has a finite domain $\mathcal{D}_v$
- $\mathcal{I}$: initial variable assignment
- $\mathcal{G}$: goal assignment
- $\mathcal{A}$: set of actions. Each $a \in \mathcal{A}$ consists of:
  - $pre(a)$: preconditions
  - $eff(a)$: effects
  - $cost(a)$: cost of performing a

Planning Tasks

Definition: Planning Task

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

- $\mathcal{V}$: finite set of variables. Each variable $v \in \mathcal{V}$ has a finite domain $\mathcal{D}_v$
- $\mathcal{I}$: initial variable assignment
- $\mathcal{G}$: goal assignment
- $\mathcal{A}$: set of actions. Each $a \in \mathcal{A}$ consists of:
  - $pre(a)$: preconditions
  - $eff(a)$: effects
  - $cost(a)$: cost of performing a

Goal: find a cost-optimal plan

Heuristics

- Task Π induces a state space $\mathcal{S}(\Pi)$ with $\prod_{v \in \mathcal{V}} |\mathcal{D}_v|$ states.
- Need to find a minimal cost path from initial state to a goal $\Rightarrow$ A* with an admissible heuristic

here: Pattern Database (PDB) heuristics

Pattern Databases

- discard some variables of the concrete task Π
- ...to obtain an abstract task Π^P
- Pattern $P \subseteq \mathcal{V}$ specifies which variables are kept in Π^P
- compute perfect heuristic h^* for all states of Π^P
- use h^* of Π^P as an admissible heuristic for Π

Choosing a Good Pattern

Which subset of $\mathcal{V}$ should be our pattern?

- small patterns lead to uninformative PDBs
- PDBs of large patterns are informative but computationally expensive

$\Rightarrow$ use combination of multiple PDB heuristics

Combining PDB Heuristics

Given collection of patterns C and corresponding PDB heuristics we can:

- take maximum (always admissible!)
- take sum (only admissible if patterns are additive)

Additivity

Additivity

Two patterns are additive if no action...

- changes variables from both patterns (eff-eff correlation)
- has a precondition on variables from one pattern and effects on variables of the other pattern (pre-eff correlation)

If two patterns P_1, P_2 are additive: $h^{P_1 \cup P_2}(s) = h^{P_1}(s) + h^{P_2}(s)$

Canonical Heuristic

Idea

Add PDB heuristics where possible and take max otherwise.

Example

$C = \{P_1, P_2, P_3\}$ where P_1 and P_2 are additive. Canonical heuristic is $h^C(s) = \max\{h^{P_1}(s) + h^{P_2}(s), h^{P_3}(s)\}$

Pattern Selection

Given a planning task Π, what pattern collection C should we use?
$\Rightarrow$ two ideas:

- taking sum is better than taking max
- generate a collection in which all patterns are pairwise additive

$\Rightarrow CEGAR^{fadd}$

- generate a collection of disjoint patterns
- no additivity enforcement
- more patterns to choose from

$\Rightarrow CEGAR^{nadd}$

CEGAR Framework

CEGAR Algorithm

1. generate initial pattern collection
2. find flaws in the collection
3. refine collection s.t. detected flaws do not occur again
4. repeat steps 2-3 until all flaws repaired or size limit reached
5. return final collection

Flaws

Flaw Detection

Given pattern P, task Π^P and its optimal plan τ^P we try to execute actions of τ^P in the concrete task Π.

What can go wrong?

- Some action a from the plan τ^P is not applicable because some precondition $pre(a)$ is not satisfied.
  $\Rightarrow$ precondition violation flaw
- Plan could be executed but did not lead to a goal state.
  $\Rightarrow$ goal violation flaw

Otherwise: solved during refinement

Causes of Flaws

Precondition Violation Flaws

Precondition violations can happen if some action of τ^P has a precondition on some $v \notin P$.

Causes of Flaws

Precondition Violation Flaws

Precondition violations can happen if some action of τ^P has a precondition on some $v \notin P$.

Goal Violation Flaws

Goal violations occur when some goal variable is not included in any pattern $P \in C$.

Causes of Flaws

Precondition Violation Flaws

Precondition violations can happen if some action of τ^P has a precondition on some $v \notin P$.

Goal Violation Flaws

Goal violations occur when some goal variable is not included in any pattern $P \in C$.

$\Rightarrow$ both flaw types occur because patterns are lacking certain important variables!

Causes of Flaws

Precondition Violation Flaws

Precondition violations can happen if some action of τ^P has a precondition on some $v \notin P$.

Goal Violation Flaws

Goal violations occur when some goal variable is not included in any pattern $P \in C$.

$\Rightarrow$ both flaw types occur because patterns are lacking certain important variables!

$\Rightarrow$ refinement $\equiv$ introduction of new variables

Abstraction Refinement: $CEGAR^{nadd}$

Reminder: $CEGAR^{nadd}$

We want to generate a collection C of disjoint patterns.

Refinement in $CEGAR^{nadd}$

Given flaw f with variable v_f

- If f is a goal violation: add pattern $\{v_f\}$ to collection
- If f is a precondition violation raised by P_f:
  a) if v_f not part of any pattern yet: add v_f to P_f
  b) if v_f already part of some P: merge P_f and P

Abstraction Refinement: $CEGAR^{fadd}$

Reminder: $CEGAR^{fadd}$

We want to generate a collection C of pairwise additive patterns.

Refinement in $CEGAR^{fadd}$

Given flaw f with variable v_f

1. create pattern $\{v_f\}$
2. select all patterns $P \in C$ that are not additive with $\{v_f\}$
3. merge $\{v_f\}$ with all selected patterns

Merging

Both algorithms merge patterns to preserve additivity/disjointedness

Merging

Both algorithms merge patterns to preserve additivity/disjointedness

Merging is bad!

- merging produces large patterns
- large patterns lead to large state spaces
  $\Rightarrow$ PDB construction becomes very expensive

Merge Avoidance

Merging cannot be avoided entirely, but how can we minimize it?

Ideas:

- better flaw selection strategy (LCF selection)
- completely ignore highly correlated variables (blacklisting)
- use different definition of additivity (partial additivity)

LCF Flaw Selection

Given a set of flaws, which flaw should be repaired next?

LCF Flaw Selection

Given a set of flaws, which flaw should be repaired next?
$\Rightarrow$ until now: pick a random flaw from the list

LCF Flaw Selection

Given a set of flaws, which flaw should be repaired next?
$\Rightarrow$ until now: pick a random flaw from the list

Pattern	Flaw Variable
P_1	v_1
P_2	v_1
P_3	v_1
P_4	v_2

LCF Flaw Selection

Given a set of flaws, which flaw should be repaired next?
⇒ until now: pick a random flaw from the list

Pattern	Flaw Variable
P_1	v_1
P_2	v_1
P_3	v_1
P_4	v_2

with random selection: 75% probability to pick a flaw with v_1
⇒ $CEGAR^{fadd}$: guaranteed merge of P_1, P_2 and P_3

LCF Flaw Selection

Given a set of flaws, which flaw should be repaired next?
$\Rightarrow$ until now: pick a random flaw from the list

Pattern	Flaw Variable
P_1	v_1
P_2	v_1
P_3	v_1
P_4	v_2

with random selection: 75% probability to pick a flaw with v_1
$\Rightarrow$ $CEGAR^{fadd}$: guaranteed merge of P_1, P_2 and P_3
would rather pick a flaw with the least common variable
$\Rightarrow$ Least-Common-First (LCF) Flaw Selection

Blacklisting

Blacklisting: Idea

Variables with many correlations are more likely to cause merges.
We put these variables on a blacklist B and ignore them.

Blacklisting

Blacklisting: Idea

Variables with many correlations are more likely to cause merges.
We put these variables on a blacklist B and ignore them.
$\Rightarrow$ precondition violation flaws cannot be raised for variables $v \in B$

Blacklisting

Blacklisting: Idea

Variables with many correlations are more likely to cause merges.
We put these variables on a blacklist B and ignore them.
$\Rightarrow$ precondition violation flaws cannot be raised for variables $v \in B$
$\Rightarrow$ $v \in B$ are never added to the collection

Partial Additivity

Idea

Relax definition of additivity, so that merging does not occur as frequently.

Partial Additivity

Reminder: Additivity

Two patterns are additive when no action...

- changes variables from both patterns
- has a precondition on variables from one pattern and effects on variables of the other pattern

If two patterns P_1, P_2 are additive: $h^{P_1 \cup P_2}(s) = h^{P_1}(s) + h^{P_2}(s)$

Partial Additivity

Partial Additivity

Two patterns are partially additive when no action...

- changes variables from both patterns
- ~~has a precondition on variables from one pattern and effects on variables of the other pattern~~

If P_1, P_2 are partially additive: $h^{P_1 \cup P_2}(s) \geq h^{P_1}(s) + h^{P_2}(s)$

$CEGAR^{padd}$ algorithm

$CEGAR^{padd}$ functions analogously to $CEGAR^{fadd}$ but aims to construct pairwise partially additive patterns.

$CEGAR^{padd}$ algorithm

$CEGAR^{padd}$ functions analogously to $CEGAR^{fadd}$ but aims to construct pairwise partially additive patterns.

Refinement in $CEGAR^{padd}$

Given flaw f with variable v_f

1. create pattern $\{v_f\}$
2. select all $P \in C$ that are not partially additive with $\{v_f\}$
3. merge $\{v_f\}$ with all selected patterns

Evaluation

3 Algorithms:

- $CEGAR^{fadd}$: additive patterns
- $CEGAR^{padd}$: partially additive patterns
- $CEGAR^{nadd}$: disjoint patterns

3 Parameters:

- Initial collection: random goal vs. all goals
- Flaw selection strategy: random flaw vs. LCF
- Blacklist size: 0 (no blacklisting) vs. 20 variables

Coverage

	$CEGAR^{fadd}$	$CEGAR^{padd}$	$CEGAR^{nadd}$
random goal	735 (231)	736 (122)	**757** (153)
all goals	736 (229)	740 (118)	**790** (145)

Coverage

	$CEGAR^{fadd}$	$CEGAR^{padd}$	$CEGAR^{nadd}$
random goal	735 (231)	736 (122)	**757** (153)
all goals	736 (229)	740 (118)	**790** (145)
max of both	750 (240)	724 (129)	**791** (158)

Coverage

	$CEGAR^{fadd}$	$CEGAR^{padd}$	$CEGAR^{nadd}$
random goal	735 (231)	736 (122)	**757** (153)
all goals	736 (229)	740 (118)	**790** (145)
max of both	750 (240)	724 (129)	**791** (158)
random goal & LCF	737 (230)	742 (122)	**757** (153)
all goals & LCF	739 (227)	740 (118)	**790** (143)

Coverage

	$CEGAR^{fadd}$	$CEGAR^{padd}$	$CEGAR^{nadd}$
random goal	735 (231)	736 (122)	**757** (153)
all goals	736 (229)	740 (118)	**790** (145)
max of both	750 (240)	724 (129)	**791** (158)
random goal & LCF	737 (230)	742 (122)	**757** (153)
all goals & LCF	739 (227)	740 (118)	**790** (143)
max of both	748 (239)	721 (129)	**793** (158)

Coverage: Blacklisting

	$CEGAR^{fadd}$	$CEGAR^{padd}$	$CEGAR^{nadd}$
no blacklisting	737 (230)	742 (122)	**790** (143)
blacklisting (20)	743 (34)	**748** (36)	743 (4)

Coverage: Blacklisting

	$CEGAR^{fadd}$	$CEGAR^{padd}$	$CEGAR^{nadd}$
no blacklisting	737 (230)	742 (122)	**790** (143)
blacklisting (20)	743 (34)	**748** (36)	743 (4)
max of both	787 (230)	775 (119)	**811** (146)

iPDB vs CEGAR

iPDB (Haslum et. al., 2007)

- pattern selection using hillclimbing
- heuristic quality is evaluated empirically

iPDB vs CEGAR

iPDB (Haslum et. al., 2007)

- pattern selection using hillclimbing
- heuristic quality is evaluated empirically

Coverage:

- iPDB coverage: 802

iPDB vs CEGAR

iPDB (Haslum et. al., 2007)

- pattern selection using hillclimbing
- heuristic quality is evaluated empirically

Coverage:

- iPDB coverage: 802
- $CEGAR^{nadd}$ without blacklisting: 790

iPDB vs CEGAR

iPDB (Haslum et. al., 2007)

- pattern selection using hillclimbing
- heuristic quality is evaluated empirically

Coverage:

- iPDB coverage: 802
- $CEGAR^{nadd}$ without blacklisting: 790
- $CEGAR^{nadd}$ with+without blacklisting: 811

iPDB vs CEGAR

iPDB (Haslum et. al., 2007)

- pattern selection using hillclimbing
- heuristic quality is evaluated empirically

Coverage:

- iPDB coverage: 802
- $CEGAR^{nadd}$ without blacklisting: 790
- $CEGAR^{nadd}$ with+without blacklisting: 811
- max(iPDB, $CEGAR^{nadd}$): 833

Future Work

- alternative flaw selection strategies

Future Work

- alternative flaw selection strategies
- alternative blacklisting strategies

Future Work

- alternative flaw selection strategies
- alternative blacklisting strategies
  - blacklist variables with the largest domains?

Future Work

- alternative flaw selection strategies
- alternative blacklisting strategies
  - blacklist variables with the largest domains?
- adaptive blacklisting
  - decide automatically if blacklisting is appropriate
  - adjust blacklist size depending on planning task
- cost-partitioning

Conclusion

- $CEGAR^{nadd}$ shows best performance

Conclusion

- *CEGAR*^{nadd} shows best performance
- ...and is competitive with iPDB

Conclusion

- $CEGAR^{nadd}$ shows best performance
- ...and is competitive with iPDB
- $CEGAR^{nadd}$ and iPDB are complementary

Conclusion

- $CEGAR^{nadd}$ shows best performance
- ...and is competitive with iPDB
- $CEGAR^{nadd}$ and iPDB are complementary
- combining a baseline CEGAR algorithm with its blacklisted version gives a significant coverage boost