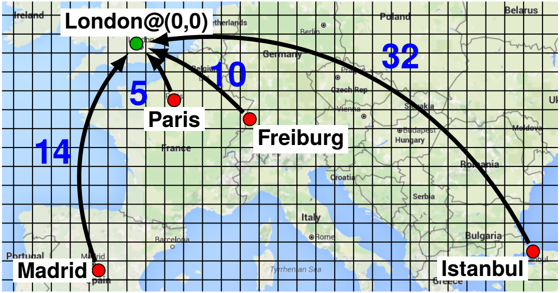
Abstractions for Planning with State-Dependent Action Costs

Florian Geißer   Thomas Keller   Robert Mattmüller

June 15, 2016

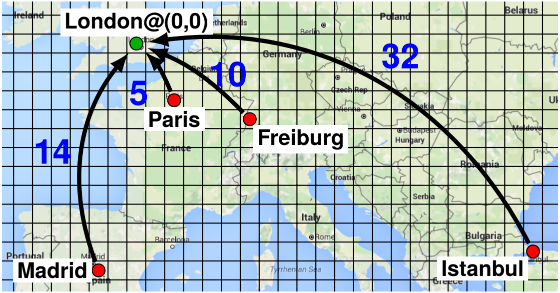
ICAPS 2016, London, UK

What are State-Dependent Action Costs?



UNI FREIBURG

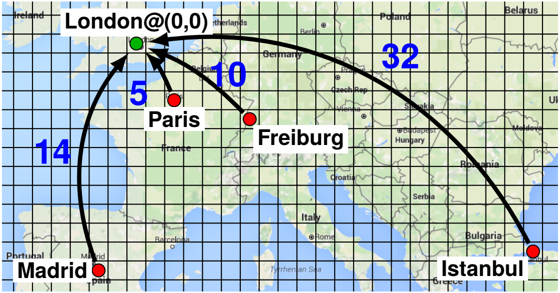
What are State-Dependent Action Costs?



Action costs: unit — constant — state-dependent

$$\textit{cost}(\textit{fly}(Madrid, London)) = 1, \quad \textit{cost}(\textit{fly}(Paris, London)) = 1,$$
$$\textit{cost}(\textit{fly}(Freiburg, London)) = 1, \quad \textit{cost}(\textit{fly}(Istanbul, London)) = 1.$$

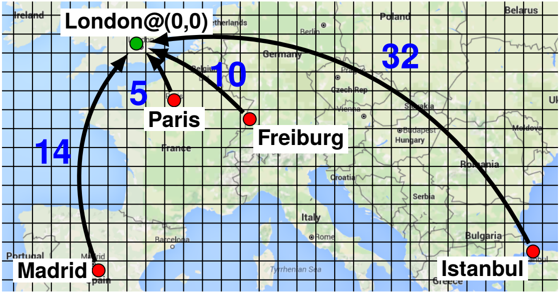
What are State-Dependent Action Costs?



Action costs: unit — constant — state-dependent

$$cost(fly(Madrid, London)) = 14, \quad cost(fly(Paris, London)) = 5,$$
$$cost(fly(Freiburg, London)) = 10, \quad cost(fly(Istanbul, London)) = 32.$$

What are State-Dependent Action Costs?



Action costs: unit — constant — state-dependent

$$cost(flyTo(London)) = |x_{\text{London}} - x_{\text{current}}| + |y_{\text{London}} - y_{\text{current}}|$$
$$= |x_{\text{current}}| + |y_{\text{current}}|.$$

UNI FREIBURG

Why Study State-Dependent Action Costs?

Advantages:

- Structured and “natural”
- Exponentially more compact, fewer redundancies
- Relevant to applications

$\leadsto$ benefits for:

- Human modelers
- Computers/algorithms (exploit structure!)

Handling State-Dependent Action Costs

State of the art:

- Different compilations to constant-cost tasks
- Generalized additive heuristic h^{add}
- Generalized relaxed planning graph to compute h^{add}

Handling State-Dependent Action Costs

State of the art:

- Different compilations to constant-cost tasks
- Generalized additive heuristic h^{add}
- Generalized relaxed planning graph to compute h^{add}

Open questions:

- Optimal planning with state-dependent costs.
- $\rightsquigarrow$ admissible abstraction heuristics
  - abstract transition costs (always/sometimes) efficiently computable?
  - empirical performance?

Edge-Valued Multi-Valued Decision Diagrams

Appropriate data structure to represent action cost functions:

Edge-Valued Multi-Valued Decision Diagrams

Appropriate data structure to represent action cost functions:

Edge-Valued Multi-Valued Decision Diagrams (EVMDDs)

Edge-Valued Multi-Valued Decision Diagrams

Appropriate data structure to represent action cost functions:

Edge-Valued Multi-Valued Decision Diagrams (EVMDDs)

Reasons:

- Follow naturally from desired properties of compilations
- Exhibit additive structure
- Attribute partial costs to facts responsible for them
- Often compact

Edge-Valued Multi-Valued Decision Diagrams

Appropriate data structure to represent action cost functions:

Edge-Valued Multi-Valued Decision Diagrams (EVMDDs)

Reasons:

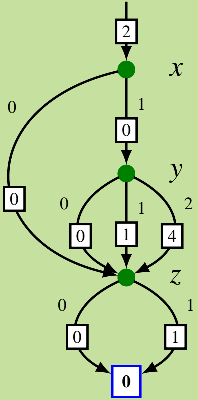
- Follow naturally from desired properties of compilations
- Exhibit additive structure
- Attribute partial costs to facts responsible for them
- Often compact

$\rightsquigarrow$ try to exploit additive structure exhibited by them!

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

$cost_o = xy^2 + z + 2$ $\qquad \mathcal{D}_x = \mathcal{D}_z = \{0,1\},\ \mathcal{D}_y = \{0,1,2\}$

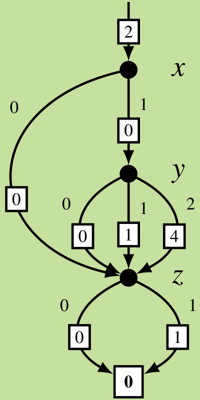


- Directed acyclic graph
- Dangling incoming edge
- Single terminal node **0**
- Decision nodes with:
  - decision variables
  - edge label
  - edge weights

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

$\mathit{cost}_o = xy^2 + z + 2$ $\qquad \mathcal{D}_x = \mathcal{D}_z = \{0,1\},\ \mathcal{D}_y = \{0,1,2\}$



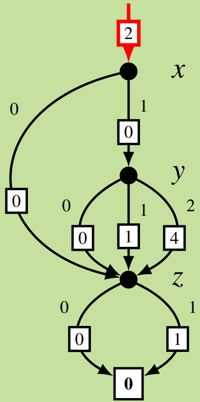
- $s = \{x \mapsto 1,\ y \mapsto 2,\ z \mapsto 0\}$
- $\mathit{cost}_o(s) =$

UNI FREIBURG

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

$cost_o = xy^2 + z + 2$ $\mathcal{D}_x = \mathcal{D}_z = \{0,1\},\ \mathcal{D}_y = \{0,1,2\}$

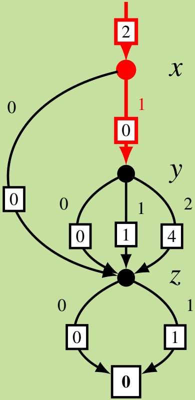


- $s = \{x \mapsto 1,\ y \mapsto 2,\ z \mapsto 0\}$
- $cost_o(s) = 2 +$

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

$cost_o = xy^2 + z + 2$ $\mathcal{D}_x = \mathcal{D}_z = \{0,1\},\ \mathcal{D}_y = \{0,1,2\}$

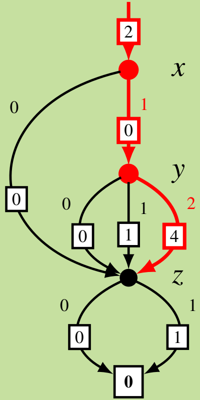


- $s = \{x \mapsto 1,\ y \mapsto 2,\ z \mapsto 0\}$
- $cost_o(s) = 2 + 0 +$

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

$\mathit{cost}_o = xy^2 + z + 2$ $\qquad\qquad$ $\mathcal{D}_x = \mathcal{D}_z = \{0,1\},\ \mathcal{D}_y = \{0,1,2\}$



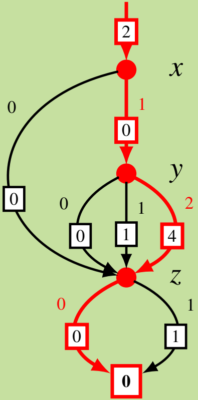
- $s = \{x \mapsto 1,\ y \mapsto 2,\ z \mapsto 0\}$
- $\mathit{cost}_o(s) = 2 + 0 + 4 +$

UNI FREIBURG

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

$\mathit{cost}_o = xy^2 + z + 2$ $\mathcal{D}_x = \mathcal{D}_z = \{0,1\},\ \mathcal{D}_y = \{0,1,2\}$



- $s = \{x \mapsto 1,\ y \mapsto 2,\ z \mapsto 0\}$
- $\mathit{cost}_o(s) = 2 + 0 + 4 + 0 = 6$

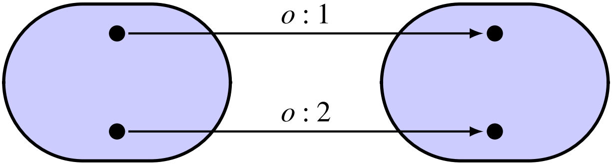
Edge-Valued Multi-Valued Decision Diagrams

Properties of EVMDDs:

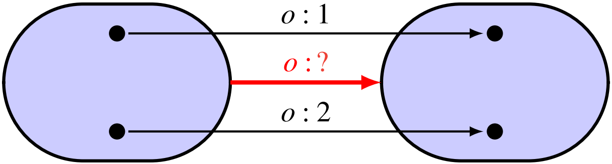
- ✔ Existence
- ✔ Uniqueness/canonicity (if reduced and ordered)
- ✔ Basic arithmetic operations supported

(Lai et al., 1996; Ciardo and Siminiceanu, 2002)

Abstraction Heuristics



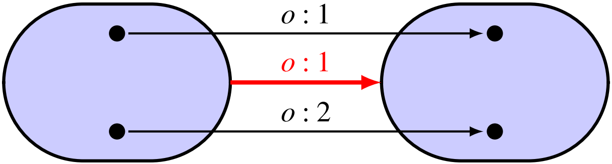
Abstraction Heuristics



Question: What are the abstract action costs?

UNI FREIBURG

Abstraction Heuristics

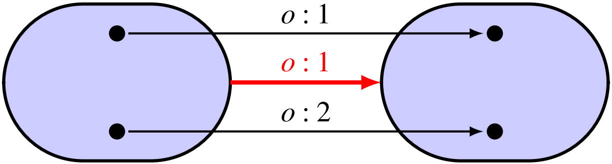


Question: What are the abstract action costs?

Answer: For admissibility, in abstract state s^{abs}, operator o should cost

$$\mathit{cost}_o(s^{abs}) = \min_{\substack{\text{concrete state } s \\ \text{abstracted to } s^{abs}}} \mathit{cost}_o(s).$$

Abstraction Heuristics



Question: What are the abstract action costs?

Answer: For admissibility, in abstract state s^{abs}, operator o should cost

$$cost_o(s^{abs}) = \min_{\substack{\text{concrete state } s \\ \text{abstracted to } s^{abs}}} cost_o(s).$$

Problem: exponentially many states to minimize over

Aim: Compute $cost_o(s^{abs})$ efficiently (given EVMDD for $cost_o(s)$).

Cartesian Abstractions

We will see: this is possible if the abstraction is Cartesian or coarser.

(This includes projections and domain abstractions.)

Cartesian Abstractions

We will see: this is possible if the abstraction is Cartesian or coarser.

(This includes projections and domain abstractions.)

Definition (Cartesian abstraction (Seipp and Helmert, 2013))

A set of states s^{abs} is Cartesian if it is of the form

$$D_1 \times \cdots \times D_n,$$

where $D_i \subseteq \mathcal{D}_i$ for all $i = 1, \ldots, n$.

An abstraction is Cartesian if all its abstract states are Cartesian sets.

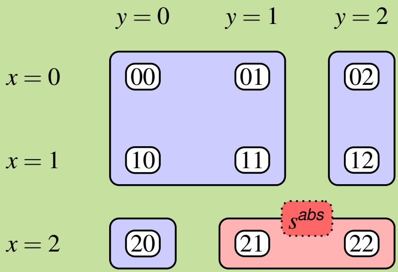
Intuition: In s^{abs}, variables are abstracted independently.

$\rightsquigarrow$ exploit independence when computing abstract costs.

Cartesian Abstractions

Example (Cartesian abstraction)

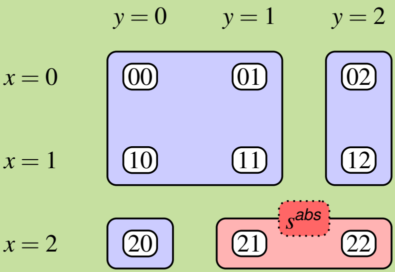
Some Cartesian abstraction over x, y



Cartesian Abstractions

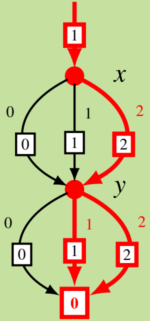
Example (Cartesian abstraction)

Some Cartesian abstraction over x, y



Cost $x+y+1$

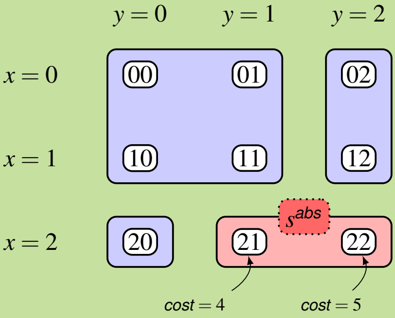
(edges consistent with s^{abs})



Cartesian Abstractions

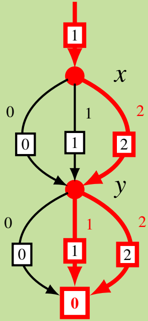
Example (Cartesian abstraction)

Some Cartesian abstraction over x, y



Cost $x + y + 1$

(edges consistent with s^{abs})

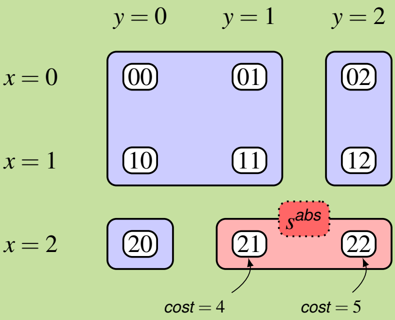


UNI FREIBURG

Cartesian Abstractions

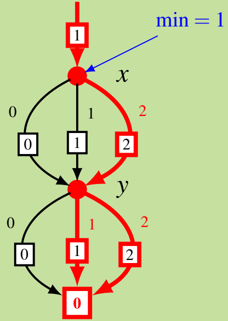
Example (Cartesian abstraction)

Some Cartesian abstraction over x, y



Cost $x + y + 1$

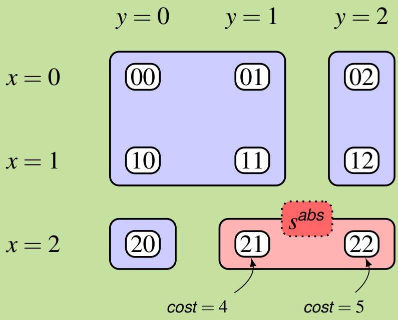
(edges consistent with s^{abs})



Cartesian Abstractions

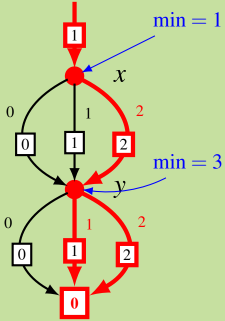
Example (Cartesian abstraction)

Some Cartesian abstraction over x, y



Cost $x+y+1$

(edges consistent with s^{abs})



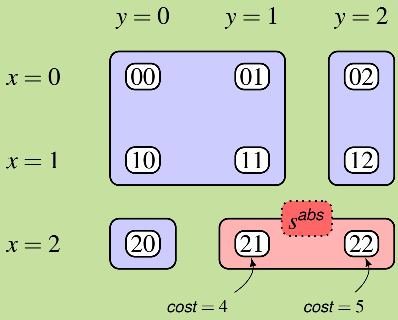
Cartesian Abstractions

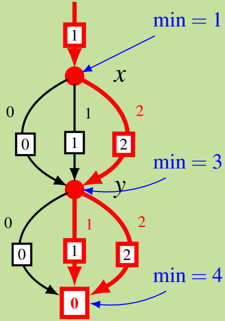
Example (Cartesian abstraction)

Some Cartesian abstraction over x, y

Cost $x + y + 1$

(edges consistent with s^{abs})





UNI FREIBURG

Cartesian Abstractions

What happens here? *or:*

Why does the topsort EVMDD traversal correctly compute $cost_o(s^{abs})$?

1. For each Cartesian state s^{abs} and each variable x, each value $d \in \mathcal{D}_x$ is either consistent with s^{abs} or not.
2. This implies: at all decision nodes associated with variable x, some outgoing edges are enabled, others are disabled.

   This is independent from all other decision nodes/variables.
3. This allows local minimizations over (linearly many) edges instead of global minimization over (exponentially many) paths in the EVMDD when computing minimum costs.

$\rightsquigarrow$ polynomial in EVMDD size!

Cartesian Abstractions

Not Cartesian!

If abstraction is not Cartesian: two variables can be

- independent in the cost function ($\rightsquigarrow$ compact EVMDD), but
- dependent in the abstraction.

$\rightsquigarrow$ cannot consider independent parts of the EVMDD separately.

Cartesian Abstractions

Not Cartesian!

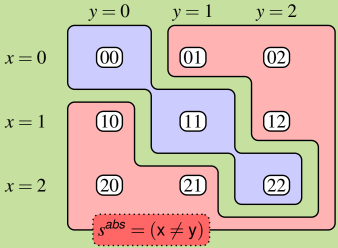
If abstraction is not Cartesian: two variables can be

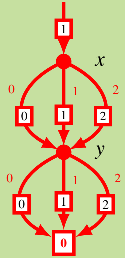
- independent in the cost function ($\rightsquigarrow$ compact EVMDD), but
- dependent in the abstraction.

$\rightsquigarrow$ cannot consider independent parts of the EVMDD separately.

Example (Non-Cartesian abstraction)

cost : $x + y + 1$, *cost*$(s^{abs}) = 2$, local minimization: $1 \rightsquigarrow$ underestimate!

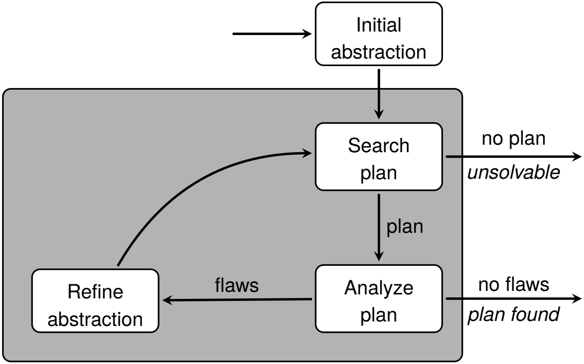




Counterexample-Guided Abstraction Refinement

Wanted: principled way of computing Cartesian abstractions.

⇝ Counterexample-Guided Abstraction Refinement (CEGAR)



Possible flaws in abstract plan:

1. Concrete state does not fit abstract state
   (concrete and abstract traces diverge)
2. Operator not applicable in concrete state
3. Trace completed, but goal not reached

Counterexample-Guided Abstraction Refinement

Cost-Mismatch Flaws

Possible flaws in abstract plan:

1. Concrete state does not fit abstract state (concrete and abstract traces diverge)
2. Operator not applicable in concrete state
3. Trace completed, but goal not reached

Here, we need to consider a further type of flaw:

4. Cost-mismatch flaw: Action more costly in concrete state than in abstract state

Counterexample-Guided Abstraction Refinement

Cost-Mismatch Flaws

Possible flaws in abstract plan:

1. Concrete state does not fit abstract state (concrete and abstract traces diverge)
2. Operator not applicable in concrete state
3. Trace completed, but goal not reached

Here, we need to consider a further type of flaw:

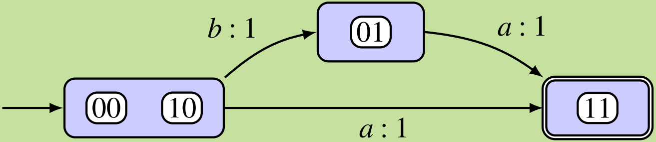
4. Cost-mismatch flaw: Action more costly in concrete state than in abstract state

$\rightsquigarrow$ resolve cost-mismatch flaws with additional refinement.

Counterexample-Guided Abstraction Refinement

Cost-Mismatch Flaws

Example (Cost-mismatch flaw)



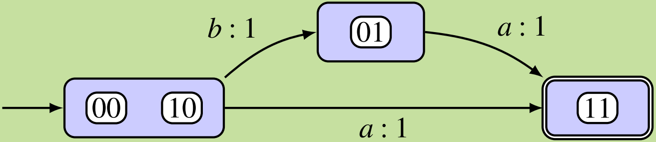
$$a = \langle \top, \quad x \wedge y \rangle, \quad \mathit{cost}_a = 2x + 1 \qquad s_0 = 10$$

$$b = \langle \top, \neg x \wedge y \rangle, \quad \mathit{cost}_b = 1 \qquad s_\star = x \wedge y$$

Counterexample-Guided Abstraction Refinement

Cost-Mismatch Flaws

Example (Cost-mismatch flaw)



$$a = \langle \top, \quad x \wedge y \rangle, \ \ cost_a = 2x + 1 \qquad\qquad s_0 = 10$$

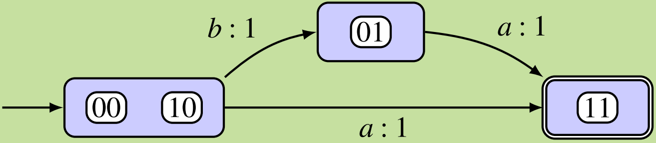
$$b = \langle \top, \neg x \wedge y \rangle, \ \ cost_b = 1 \qquad\qquad s_\star = x \wedge y$$

- Optimal abstract plan: $\langle a \rangle$ (abstract cost 1)

Counterexample-Guided Abstraction Refinement

Cost-Mismatch Flaws

Example (Cost-mismatch flaw)



$$a = \langle \top, \quad x \wedge y \rangle, \ \ cost_a = 2x + 1 \qquad s_0 = 10$$
$$b = \langle \top, \neg x \wedge y \rangle, \ \ cost_b = 1 \qquad s_\star = x \wedge y$$

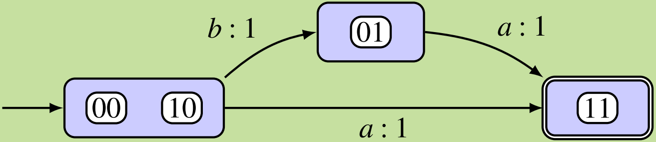
- Optimal abstract plan: $\langle a \rangle$ (abstract cost 1)
- This is also a concrete plan (concrete cost 3)

UNI FREIBURG

Counterexample-Guided Abstraction Refinement

Cost-Mismatch Flaws

Example (Cost-mismatch flaw)



$$a = \langle \top, \quad x \wedge y \rangle, \ \mathit{cost}_a = 2x + 1 \qquad s_0 = 10$$

$$b = \langle \top, \neg x \wedge y \rangle, \ \mathit{cost}_b = 1 \qquad s_\star = x \wedge y$$

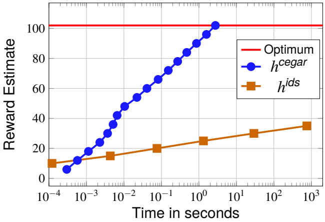
- Optimal abstract plan: $\langle a \rangle$ (abstract cost 1)
- This is also a concrete plan (concrete cost 3)
- But optimal concrete plan: $\langle b, a \rangle$ (concrete and abstract cost 2)

Empirical Evaluation

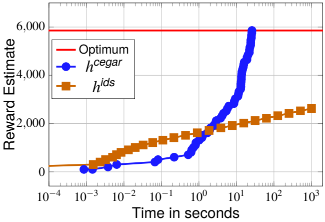
Experiment 1: Compare Anytime Behaviour of h^{cegar} and h^{ids}

Setting: IPPC benchmarks, Prost planner

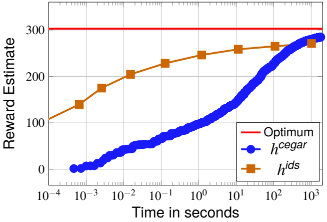
Academic Advising



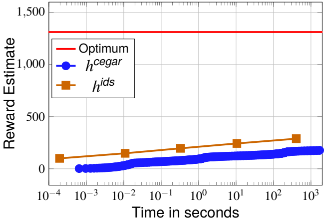
Triangle Tireworld



Tamarisk



Sysadmin

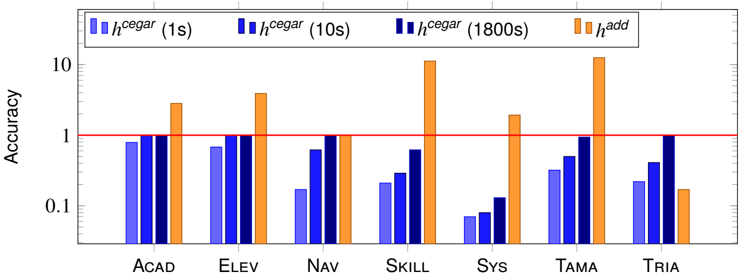


UNI FREIBURG

Empirical Evaluation

Experiment 2: Compare Accuracy of h^{cegar} and h^{add}

Observation/Question: h^{add} neither admissible nor anytime, but possibly more accurate than h^{cegar}? Let's see …



Conclusion:

- h^{cegar} never overestimates.
- h^{cegar} becomes more accurate over time.
- After sufficient time, accuracy of h^{cegar} comparable to that of h^{add}.

Summary

Our motivating challenges were:

- Understand when abstract costs are efficiently computable.
  - ✔ largely understood: if (and only if) abstraction is Cartesian
- Make abstraction heuristics state-dependent-action-cost aware.
  - ✔ done: defined/generalized
    - Cartesian abstractions
    - local EVMDD evaluation
    - generalized CEGAR
- Perform optimal planning with state-dependent action costs.
  - ✔ done: promising empirical results