Planning with State-Dependent Action Costs

ICAPS 2016 Tutorial

Robert Mattmüller Florian Geißer

June 13, 2016

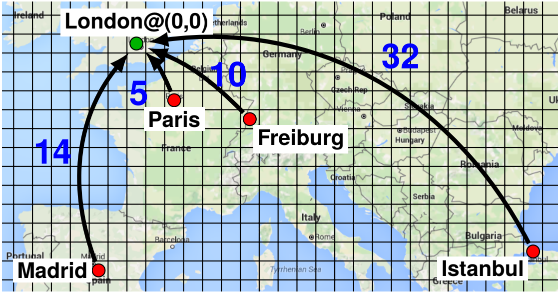
Part I

Theory

Section

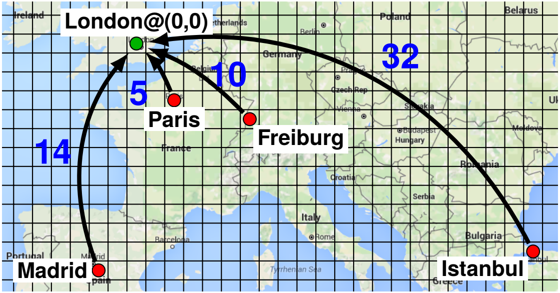
Background

What are State-Dependent Action Costs?



UNI FREIBURG

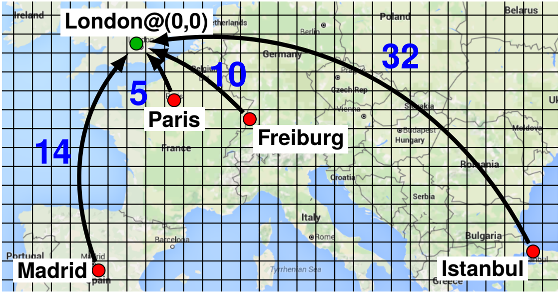
What are State-Dependent Action Costs?



Action costs: unit — constant — state-dependent

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

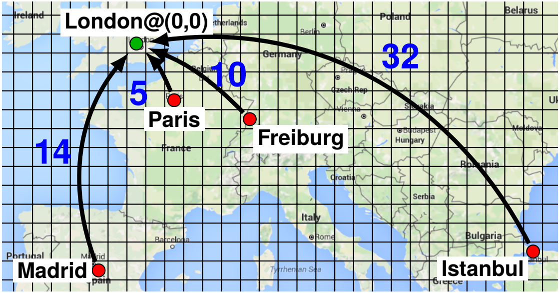
What are State-Dependent Action Costs?



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?

- Human perspective:
  - "natural" and "elegant"
  - modeler-friendly $\rightsquigarrow$ less error-prone?
- Machine perspective:
  - more structured $\rightsquigarrow$ exploit in algorithms?
  - fewer redundancies, exponentially more compact
- Language support:
  - numeric PDDL, PDDL 3
  - RDDL, MDPs (state-dependent rewards!)
- Applications:
  - modeling preferences and soft goals
  - PSR domain

(Abbreviation: SDAC = state-dependent action costs)

Handling State-Dependent Action Costs

Good news:

- Computing g values in forward search still easy.

Challenge:

- But what about SDAC-aware h values?
- Or can we simply compile SDAC away?

This tutorial:

- Proposed answers to these challenges.

Handling State-Dependent Action Costs

Roadmap:

1. Look at compilations.
2. This leads to edge-valued multi-valued decision diagrams (EVMDDs) as data structure to represent cost functions.
3. Based on EVMDDs, formalize and discuss:
   - compilations
   - relaxation heuristics
   - abstraction heuristics

State-Dependent Action Costs

Running Example

Example (Household domain)

Actions:

$$\begin{aligned}\texttt{vacuumFloor} &= \langle \top, \texttt{floorClean} \rangle \\ \texttt{washDishes} &= \langle \top, \texttt{dishesClean} \rangle \\ \texttt{doHousework} &= \langle \top, \texttt{floorClean} \wedge \texttt{dishesClean} \rangle\end{aligned}$$

Cost functions:

$$\begin{aligned}\mathit{cost}_{\texttt{vacuumFloor}} &= [\neg\texttt{floorClean}] \cdot 2 \\ \mathit{cost}_{\texttt{washDishes}} &= [\neg\texttt{dishesClean}] \cdot (1 + 2 \cdot [\neg\texttt{haveDishwasher}]) \\ \mathit{cost}_{\texttt{doHousework}} &= \mathit{cost}_{\texttt{vacuumFloor}} + \mathit{cost}_{\texttt{washDishes}}\end{aligned}$$

State-Dependent Action Costs

Compilations

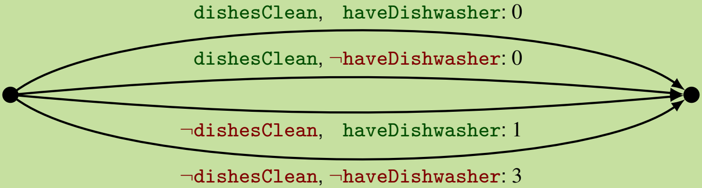
Different ways of compiling SDAC away:

- Compilation I: “Parallel Action Decomposition”
- Compilation II: “Purely Sequential Action Decomposition”
- Compilation III: “EVMDD-Based Action Decomposition”
  (combination of Compilations I and II)

State-Dependent Action Costs

Compilation I: “Parallel Action Decomposition”

Example



$$\begin{aligned}
\mathtt{washDishes}(\ \mathtt{dC}, \ \mathtt{hD}) &= \langle\ \mathtt{dC} \wedge \ \mathtt{hD}, \mathtt{dC}\rangle, \quad \mathit{cost} = 0\\
\mathtt{washDishes}(\ \mathtt{dC}, \neg\mathtt{hD}) &= \langle\ \mathtt{dC} \wedge \neg\mathtt{hD}, \mathtt{dC}\rangle, \quad \mathit{cost} = 0\\
\mathtt{washDishes}(\neg\mathtt{dC}, \ \mathtt{hD}) &= \langle\neg\mathtt{dC} \wedge \ \mathtt{hD}, \mathtt{dC}\rangle, \quad \mathit{cost} = 1\\
\mathtt{washDishes}(\neg\mathtt{dC}, \neg\mathtt{hD}) &= \langle\neg\mathtt{dC} \wedge \neg\mathtt{hD}, \mathtt{dC}\rangle, \quad \mathit{cost} = 3
\end{aligned}$$

State-Dependent Action Costs

Compilation I: “Parallel Action Decomposition”

Compilation I

Transform each action into multiple actions:

- one for each partial state relevant to cost function
- add partial state to precondition
- use cost for partial state as constant cost

Properties:

✓ always possible

✗ exponential blow-up

Question: Exponential blow-up avoidable? $\rightsquigarrow$ Compilation II

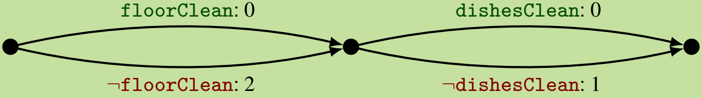
State-Dependent Action Costs

Compilation II: “Purely Sequential Action Decomposition”

Example

Assume we own a dishwasher:

$$cost_{\texttt{doHousework}} = 2 \cdot [\neg\texttt{floorClean}] + [\neg\texttt{dishesClean}]$$



$$\begin{aligned}
\texttt{doHousework}_1(\ \texttt{fC}) &= \langle\ \texttt{fC}, \texttt{fC}\rangle, & cost &= 0\\
\texttt{doHousework}_1(\neg\texttt{fC}) &= \langle\neg\texttt{fC}, \texttt{fC}\rangle, & cost &= 2\\
\texttt{doHousework}_2(\ \texttt{dC}) &= \langle\ \texttt{dC}, \texttt{dC}\rangle, & cost &= 0\\
\texttt{doHousework}_2(\neg\texttt{dC}) &= \langle\neg\texttt{dC}, \texttt{dC}\rangle, & cost &= 1
\end{aligned}$$

State-Dependent Action Costs

Compilation II: “Purely Sequential Action Decomposition”

Compilation II

If costs additively decomposable:

- high-level actions $\approx$ macro actions
- decompose into sequential micro actions

Properties:

- ✔ linear blow-up
- ✗ not always possible
- plan lengths not preserved, costs preserved
- blow-up in search space $\rightsquigarrow$ action ordering!
- attention: all partial effects at end!

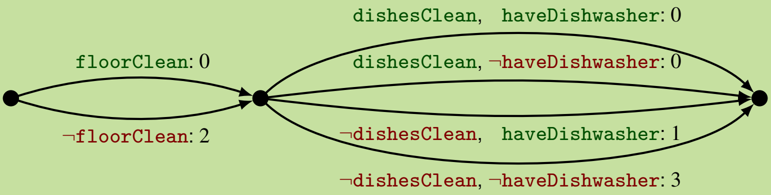
Question: Can this always work (kind of)? $\rightsquigarrow$ Compilation III

State-Dependent Action Costs

Compilation III: “EVMDD-Based Action Decomposition”

Example

$$\mathit{cost}_{\texttt{doHousework}} = [\neg\texttt{floorClean}]\cdot 2 + [\neg\texttt{dishesClean}]\cdot(1 + 2\cdot[\neg\texttt{haveDishwasher}])$$



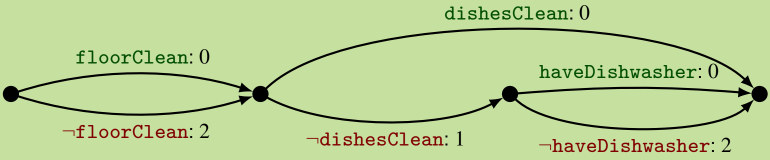
Simplify right-hand part of diagram:

- Branch over single variable at a time.
- Exploit: `haveDishwasher` irrelevant if `dishesClean` is true.

State-Dependent Action Costs

Compilation III: “EVMDD-Based Action Decomposition”

Example (ctd.)



Later:

- Compiled actions
- Auxiliary variables to enforce action ordering

State-Dependent Action Costs

Compilation III: “EVMDD-Based Action Decomposition”

Compilation III

- exploit as much additive decomposability as possible
- multiply out variable domains where inevitable
- Technicalities:
  - fix variable ordering
  - perform Shannon and isomorphism reduction

Properties:

✔ always possible

- worst-case exponential blow-up, but as good as it gets
- plan lengths not preserved, costs preserved
- as before: action ordering, all partial effects at end!

State-Dependent Action Costs

Compilation III: “EVMDD-Based Action Decomposition”

Compilation III provides optimal combination of sequential and parallel action decomposition, given fixed variable ordering.

Question: How to find such decompositions automatically?

Answer: Figure for Compilation III basically a reduced ordered edge-valued multi-valued decision diagram (EVMDD)!

[Lai et al., 1996; Ciardo and Siminiceanu, 2002]

EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

EVMDDs:

- Decision diagrams for arithmetic functions
- Decision nodes with associated decision variables
- Edge weights: partial costs contributed by facts
- Size of EVMDD compact in many “typical” cases

Properties:

✔ satisfy all requirements for Compilation III,
even (almost) uniquely determined by them

✔ already have well-established theory and tool support

✔ detect and exhibit additive structure in arithmetic functions

EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

Consequence:

- represent cost functions as EVMDDs
- exploit additive structure exhibited by them
- draw on theory and tool support for EVMDDs

Two perspectives on EVMDDs:

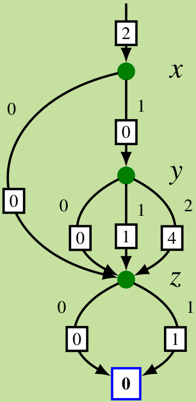
- graphs specifying how to decompose action costs
- data structures encoding action costs
  (used independently from compilations)

EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

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



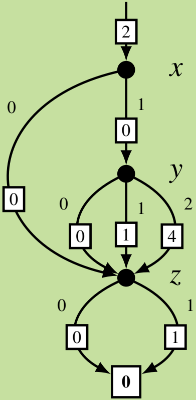
- Directed acyclic graph
- Dangling incoming edge
- Single terminal node **0**
- Decision nodes with:
  - decision variables
  - edge label
  - edge weights

EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

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



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

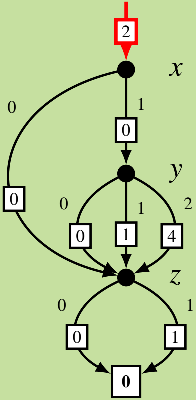
EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

Background
State-Dependent Action Costs
Edge-Valued Multi-Valued Decision Diagrams
Compilation
Relaxations
Abstractions
Summary

Example (EVMDD Evaluation)

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



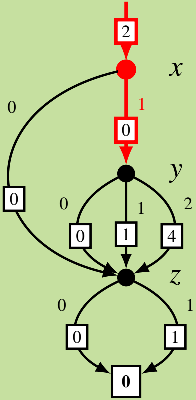
- $s = \{x \mapsto 1,\ y \mapsto 2,\ z \mapsto 0\}$
- $cost_a(s) = 2 +$

EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

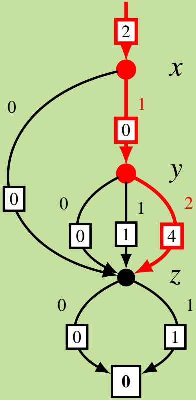
$\mathit{cost}_a = xy^2 + z + 2$ $\qquad \mathcal{D}_x = \mathcal{D}_z = \{0,1\},\ \mathcal{D}_y = \{0,1,2\}$



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

Example (EVMDD Evaluation)

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



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

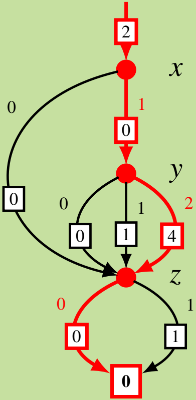
UNI FREIBURG

EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

Example (EVMDD Evaluation)

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



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

UNI FREIBURG

EVMDDs

Edge-Valued Multi-Valued Decision Diagrams

Properties of EVMDDs:

- ✔ Existence for finitely many finite-domain variables
- ✔ Uniqueness/canonicity if reduced and ordered
- ✔ Basic arithmetic operations supported

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

EVMDDs

Arithmetic operations on EVMDDs

Given arithmetic operator $\otimes \in \{+, -, \cdot, \dots\}$, EMVDDs $\mathcal{E}_1$, $\mathcal{E}_2$.
Compute EVMDD $\mathcal{E} = \mathcal{E}_1 \otimes \mathcal{E}_2$.

Implementation: procedure $\texttt{apply}(\otimes, \mathcal{E}_1, \mathcal{E}_2)$:

- Base case: single-node EVMDDs encoding constants
- Inductive case: apply $\otimes$ recursively:
  - push down edge weights
  - recursively apply $\otimes$ to corresponding children
  - pull up excess edge weights from children

Time complexity [Lai et al., 1996]:

- additive operations: product of input EVMDD sizes
- in general: exponential

UNI FREIBURG

Section

Compilation

EVMDD-Based Action Compilation

Example (EVMDD-based action compilation)

Let $a = \langle \mathit{pre}, \mathit{eff} \rangle$, $\mathit{cost}_a = xy^2 + z + 2$.

Auxiliary variables:

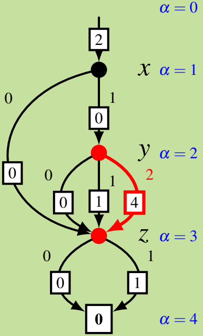
- One semaphore variable σ with $\mathcal{D}_\sigma = \{0, 1\}$ for entire planning task.
- One auxiliary variable $\alpha = \alpha_a$ with $\mathcal{D}_{\alpha_a} = \{0, 1, 2, 3, 4\}$ for action a.

Replace a by new auxiliary actions (similarly for other actions).

UNI FREIBURG

EVMDD-Based Action Compilation

Example (EVMDD-based action compilation, ctd.)



$$a^{\mathit{pre}} = \langle \mathit{pre} \wedge \sigma = 0 \wedge \alpha = 0,\quad \sigma = 1 \wedge \alpha = 1 \rangle, \qquad \mathit{cost} = 2$$

$$a^{1,x=0} = \langle \alpha = 1 \wedge x = 0,\ \alpha = 3 \rangle, \qquad \mathit{cost} = 0$$

$$a^{1,x=1} = \langle \alpha = 1 \wedge x = 1,\ \alpha = 2 \rangle, \qquad \mathit{cost} = 0$$

$$a^{2,y=0} = \langle \alpha = 2 \wedge y = 0,\ \alpha = 3 \rangle, \qquad \mathit{cost} = 0$$

$$a^{2,y=1} = \langle \alpha = 2 \wedge y = 1,\ \alpha = 3 \rangle, \qquad \mathit{cost} = 1$$

$$a^{2,y=2} = \langle \alpha = 2 \wedge y = 2,\ \alpha = 3 \rangle, \qquad \mathit{cost} = 4$$

$$a^{3,z=0} = \langle \alpha = 3 \wedge z = 0,\ \alpha = 4 \rangle, \qquad \mathit{cost} = 0$$

$$a^{3,z=1} = \langle \alpha = 3 \wedge z = 1,\ \alpha = 4 \rangle, \qquad \mathit{cost} = 1$$

$$a^{\mathit{eff}} = \langle \alpha = 4,\ \mathit{eff} \wedge \sigma = 0 \wedge \alpha = 0 \rangle, \qquad \mathit{cost} = 0$$

EVMDD-Based Action Compilation

Let Π be an SDAC-task and Π' the result of EVMDD-based action compilation applied to Π.

Proposition

Π' has only state-independent costs. □

Proposition

Size of Π' is polynomial in size of Π times size of largest EVMDD used in compilation. □

Proposition

Π and Π' admit the same plans (modulo replacement of actions by action sequences). Optimal plan costs are preserved. □

Section

Relaxations

Relaxation Heuristics

We know: Delete-relaxation heuristics informative in classical planning.

Question: Also informative in SDAC planning?

Relaxation Heuristics

Definition (Classical additive heuristic h^{add})

$$h_s^{add}(Facts) = \sum_{fact \in Facts} h_s^{add}(fact)$$

$$h_s^{add}(fact) = \begin{cases} 0 & \text{if } fact \in s \\ \min\limits_{\text{achiever } a \text{ of } fact} [h_s^{add}(pre(a)) + cost_a] & \text{otherwise} \end{cases}$$

Question: How to generalize h^{add} to SDAC?

Relaxations with SDAC

Example

$$a = \langle \top,\ x = 1 \rangle \qquad cost_a = 2 - 2y$$

$$b = \langle \top,\ y = 1 \rangle \qquad cost_b = 1$$

$$s = \{x \mapsto 0, y \mapsto 0\}$$

$$h_s^{add}(y = 1) = 1$$

$$h_s^{add}(x = 1) = ?$$

Relaxations with SDAC

Example

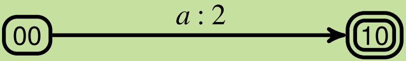
$$a = \langle \top,\ x{=}1 \rangle \qquad \mathit{cost}_a = 2 - 2y$$

$$b = \langle \top,\ y{=}1 \rangle \qquad \mathit{cost}_b = 1$$

$$s = \{x \mapsto 0, y \mapsto 0\}$$

$$h_s^{add}(y{=}1) = 1$$

$$h_s^{add}(x{=}1) = ?$$

- 

Relaxations with SDAC

Example

$$a = \langle \top,\ x{=}1 \rangle \qquad\qquad cost_a = 2 - 2y$$
$$b = \langle \top,\ y{=}1 \rangle \qquad\qquad cost_b = 1$$

$$s = \{x \mapsto 0, y \mapsto 0\}$$
$$h_s^{add}(y{=}1) = 1$$
$$h_s^{add}(x{=}1) = ?$$

- $00 \xrightarrow{a\,:\,2} 10$
- $00 \xrightarrow{b\,:\,1} 01 \xrightarrow{a\,:\,0} 11$ $\Rightarrow$ cheaper!

Relaxations with SDAC

Minimize over all situations where a is applicable.

Definition (Additive heuristic h^{add} for SDAC)

$$h_s^{\mathsf{add}}(fact) = \begin{cases} 0 & \text{if } fact \in s \\ \min\limits_{\text{achiever } a \text{ of } fact} [h_s^{\mathsf{add}}(pre(a)) + \mathit{cost}_a] & \text{otherwise} \end{cases}$$

Relaxations with SDAC

Minimize over all situations where a is applicable.

Definition (Additive heuristic h^{add} for SDAC)

$$h^{add}_s(fact) = \begin{cases} 0 & \text{if } fact \in s \\ \min_{\text{achiever } a \text{ of } fact} [h^{add}_s(pre(a)) + Cost^s_a] & \text{otherwise} \end{cases}$$

$$Cost^s_a = \min_{\hat{s} \in S_a} [cost_a(\hat{s}) + h^{add}_s(\hat{s})]$$

S_a: set of partial states over variables in cost function

$|S_a|$ exponential in number of variables in cost function

Relaxations with SDAC

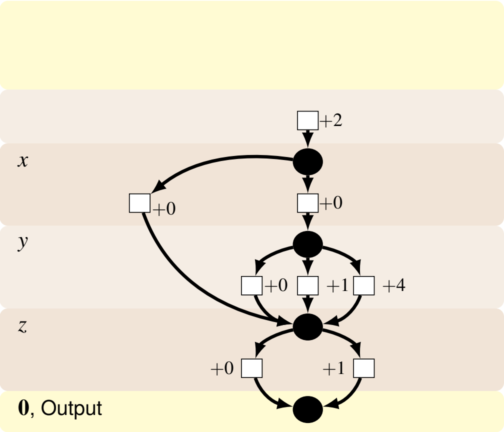
Properties of h^{add} for SDAC:

- Good: classical h^{add} on compiled task $=$ generalized h^{add} on SDAC-task
- Bad: exponential blow-up

Computing h^{add} for SDAC:

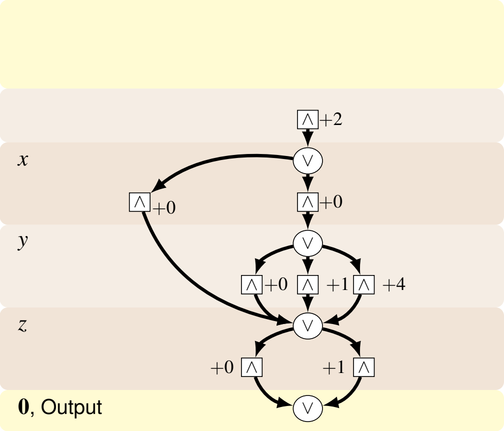
- Option 1: Compute classical h^{add} on compiled task.
- Option 2: Compute $Cost_a^s$ directly.
  - Plug EVMDDs as subgraphs into RPG
  - $\rightsquigarrow$ efficient computation of h^{add}

Option 2: RPG Compilation



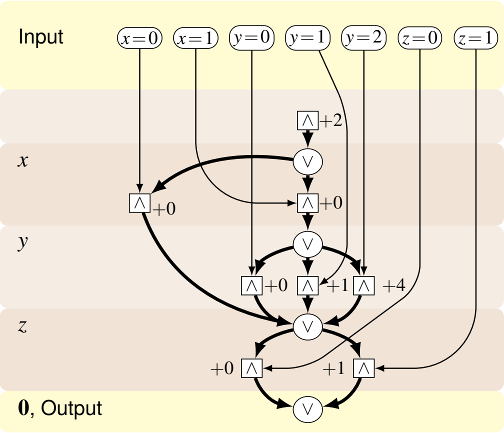
- $cost_a = xy^2 + z + 2$

Option 2: RPG Compilation



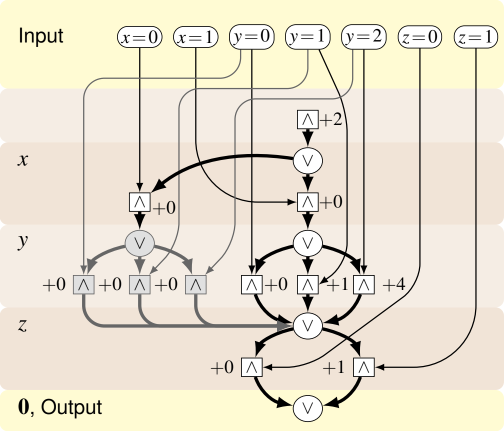
- variable nodes become $\vee$-nodes
- weights become $\wedge$-nodes

Option 2: RPG Compilation



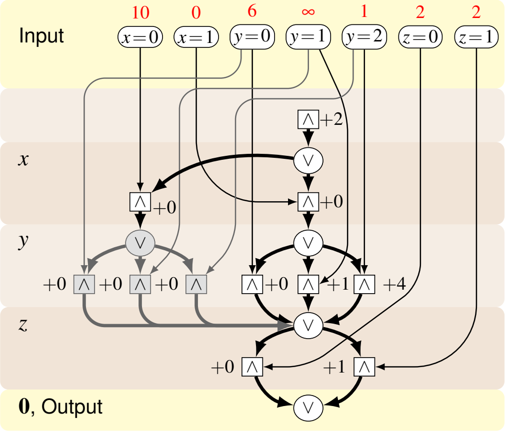
- Augment with input nodes

Option 2: RPG Compilation



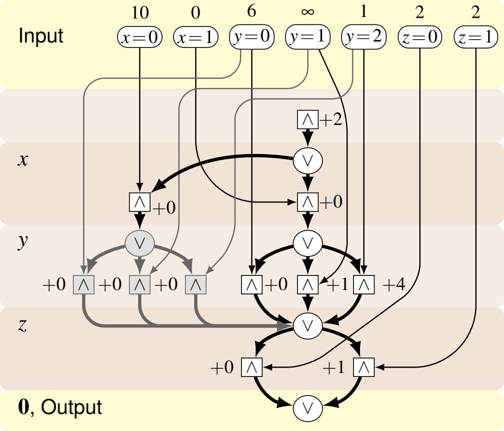
- Ensure complete evaluation

Option 2: Computing $Cost_a^s$



- Insert h^{add} values

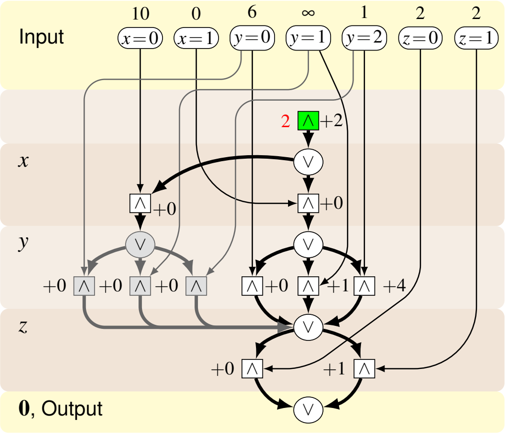
Option 2: Computing $Cost_a^s$



Evaluate nodes:

- $\wedge$: $\sum(\text{parents})$ + weight
- $\vee$: $\min(\text{parents})$

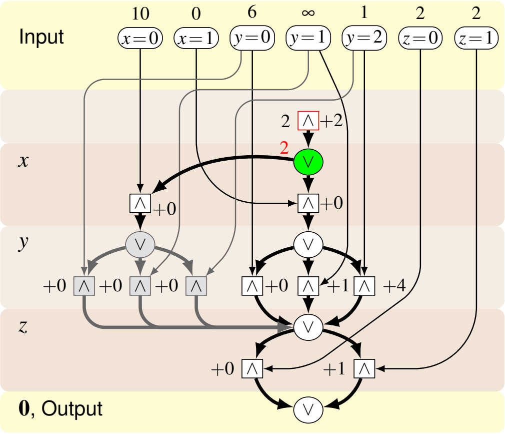
Option 2: Computing $Cost_a^s$



Evaluate nodes:

- $\wedge$: $\sum$(parents) + weight
- $\vee$: $\min$(parents)

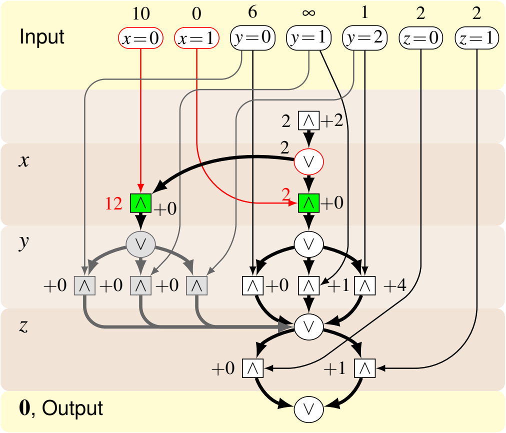
Option 2: Computing $Cost_a^s$



Evaluate nodes:

- $\wedge$: $\sum$(parents) + weight
- $\vee$: $\min$(parents)

Option 2: Computing $Cost_a^s$

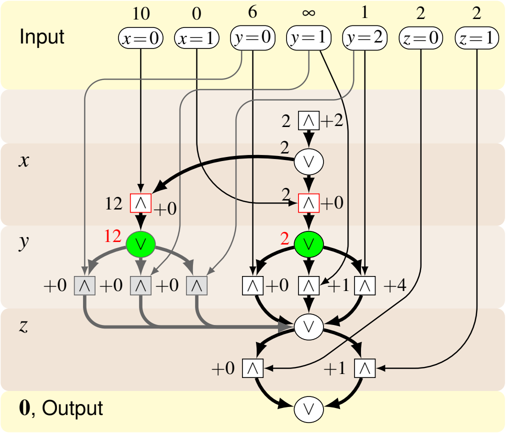


Evaluate nodes:

- $\wedge$: $\sum$(parents) + weight
- $\vee$: $\min$(parents)

UNI FREIBURG

Option 2: Computing $Cost_a^s$



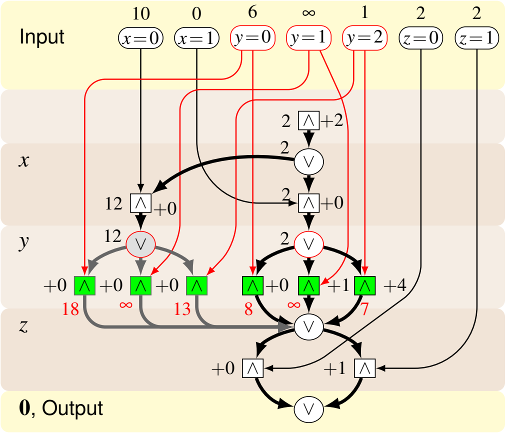
Evaluate nodes:

- $\wedge$: $\sum$(parents) + weight
- $\vee$: $\min$(parents)

Background
Compilation
Relaxations
Relaxed Planning Graph
Abstractions
Summary

UNI FREIBURG

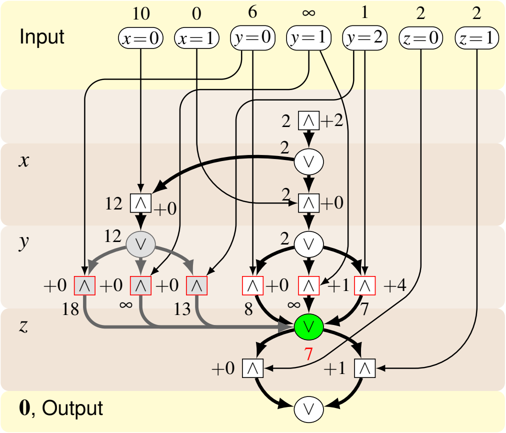
Option 2: Computing $Cost_a^s$



Evaluate nodes:

- $\wedge$: $\sum(\text{parents})$ + weight
- $\vee$: $\min(\text{parents})$

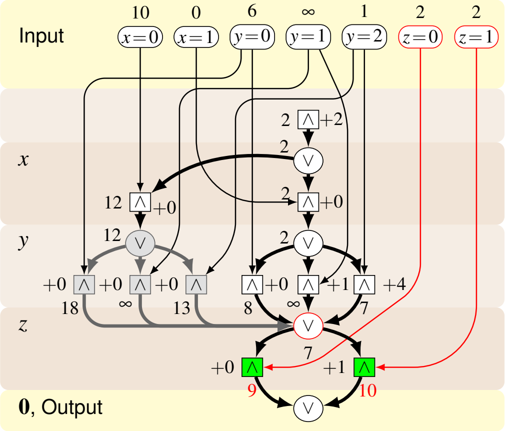
Option 2: Computing $Cost_a^s$



Evaluate nodes:

- $\wedge$: $\sum$(parents) + weight
- $\vee$: $\min$(parents)

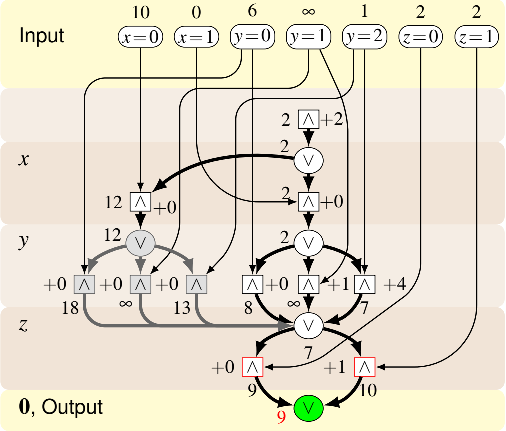
Option 2: Computing $Cost_a^s$



Evaluate nodes:

- $\wedge$: $\sum(\text{parents})$ + weight
- $\vee$: $\min(\text{parents})$

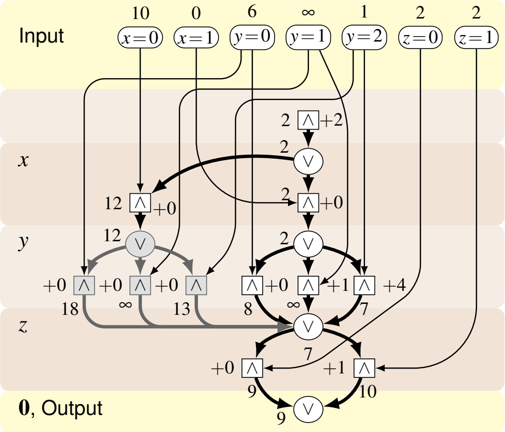
Option 2: Computing $Cost_a^s$



Evaluate nodes:

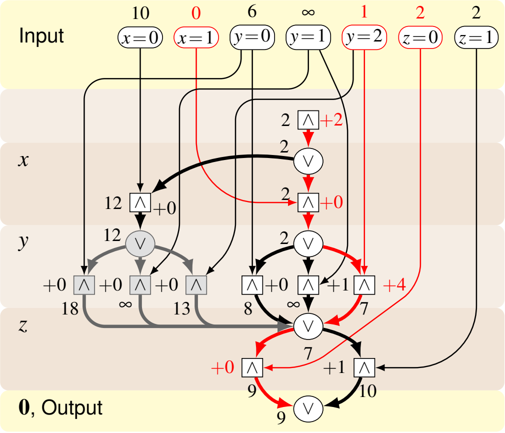
- $\wedge$: $\sum(\text{parents})$ + weight
- $\vee$: $\min(\text{parents})$

Option 2: Computing $Cost_a^s$



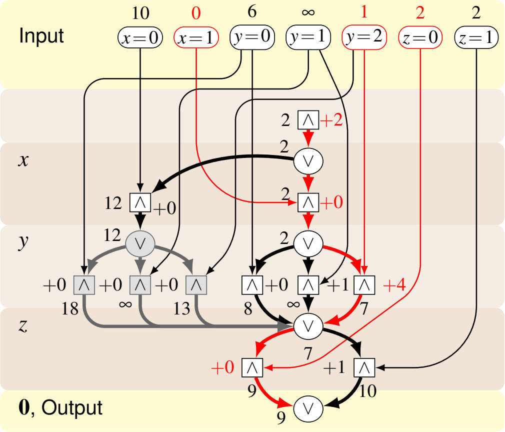
- $Cost_a^s = \min_{\hat{s} \in S_a} [cost_a(\hat{s}) + h_s^{add}(\hat{s})]$

Option 2: Computing $Cost_a^s$



- $Cost_a^s = \min_{\hat{s} \in S_a} [cost_a(\hat{s}) + h_s^{add}(\hat{s})]$
- $cost_a = xy^2 + z + 2$
- $\hat{s} = \{x \mapsto 1, y \mapsto 2, z \mapsto 0\}$

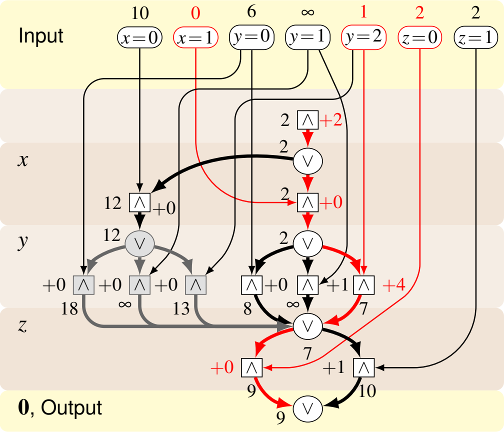
Option 2: Computing $Cost_a^s$



- $Cost_a^s = \min_{\hat{s} \in S_a} [cost_a(\hat{s}) + h_s^{add}(\hat{s})]$
- $cost_a = xy^2 + z + 2$
- $\hat{s} = \{x \mapsto 1, y \mapsto 2, z \mapsto 0\}$
- $cost_a(\hat{s}) = 1 \cdot 2^2 + 0 + 2 = 6$

  $= 2 + 0 + 4 + 0$
- $h_s^{add}(\hat{s}) = 0 + 1 + 2 = 3$

Option 2: Computing $Cost_a^s$



- $Cost_a^s = \min_{\hat{s}\in S_a}[cost_a(\hat{s}) + h_s^{add}(\hat{s})]$
- $cost_a = xy^2 + z + 2$
- $\hat{s} = \{x \mapsto 1, y \mapsto 2, z \mapsto 0\}$
- $cost_a(\hat{s}) = 1 \cdot 2^2 + 0 + 2 = 6$

  $= 2 + 0 + 4 + 0$
- $h_s^{add}(\hat{s}) = 0 + 1 + 2 = 3$
- $Cost_a^s = 6 + 3 = 9$

Additive Heuristic

RPG compilation:

- RPG subgraph in each layer for each action
- Connect subgraphs with precondition graphs
- Link outputs to next proposition layer

- Good: classical h^{add} on compiled task $=$
  generalized h^{add} on SDAC-task $=$
  cost value computed using RPG compilation

Section

Abstractions

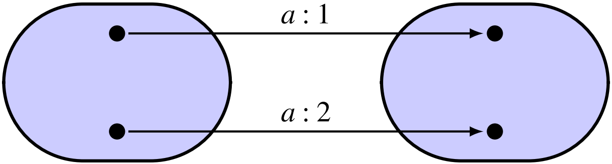
Abstraction Heuristics

Question: Why consider abstraction heuristics?

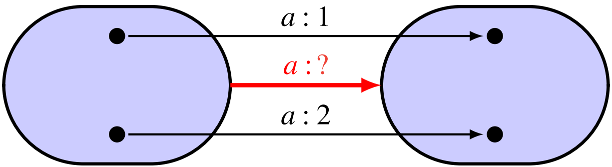
Answer:

- admissibility
- $\rightsquigarrow$ optimality

Abstraction Heuristics



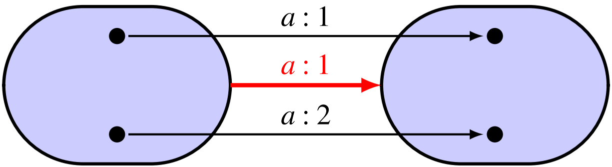
Abstraction Heuristics



Question: What are the abstract action costs?

UNI FREIBURG

Abstraction Heuristics

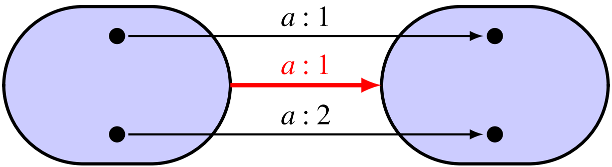


Question: What are the abstract action costs?

Answer: For admissibility, abstract cost of a should be

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

Abstraction Heuristics



Question: What are the abstract action costs?

Answer: For admissibility, abstract cost of a should be

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

Problem: exponentially many states in minimization

Aim: Compute $\mathit{cost}_a(s^{\text{abs}})$ efficiently (given EVMDD for $\mathit{cost}_a(s)$).

Cartesian Abstractions

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

(Includes projections and domain abstractions.)

Cartesian Abstractions

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

(Includes projections and domain abstractions.)

Definition (Cartesian abstraction)

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 abstract states are Cartesian sets.

[Seipp and Helmert, 2013]

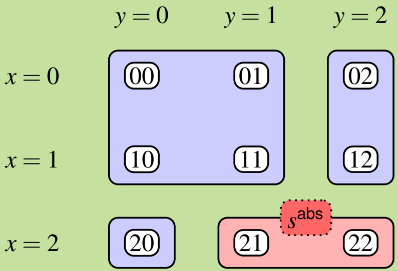
Intuition: Variables are abstracted independently.

$\rightsquigarrow$ exploit independence when computing abstract costs!

Cartesian Abstractions

Example (Cartesian abstraction)

Cartesian abstraction over x, y



UNI FREIBURG

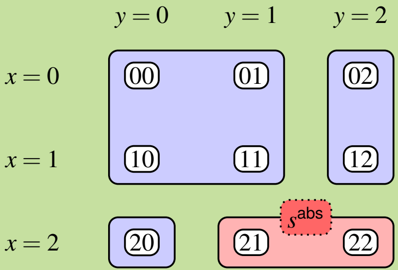
Cartesian Abstractions

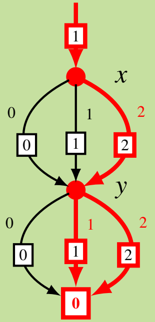
Example (Cartesian abstraction)

Cartesian abstraction over x, y

Cost $x + y + 1$

(edges consistent with s^{abs})

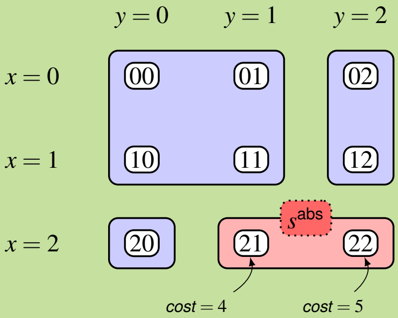




Cartesian Abstractions

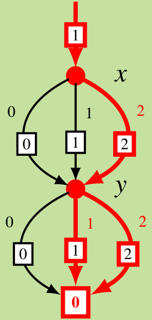
Example (Cartesian abstraction)

Cartesian abstraction over x, y



Cost $x + y + 1$

(edges consistent with s^{abs})



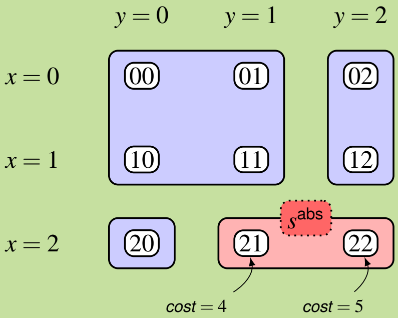
Cartesian Abstractions

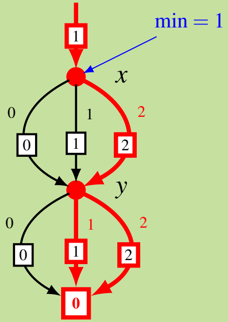
Example (Cartesian abstraction)

Cartesian abstraction over x, y

Cost $x + y + 1$

(edges consistent with s^{abs})





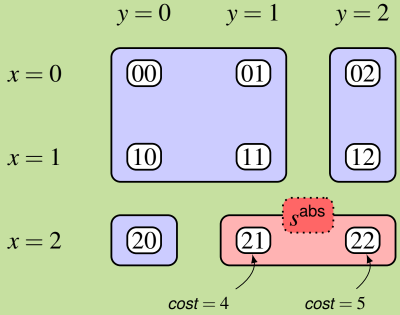
Cartesian Abstractions

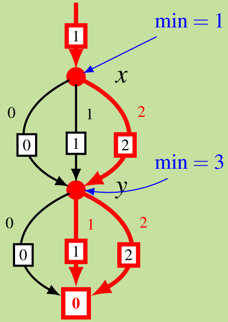
Example (Cartesian abstraction)

Cartesian abstraction over x, y

Cost $x+y+1$

(edges consistent with s^{abs})





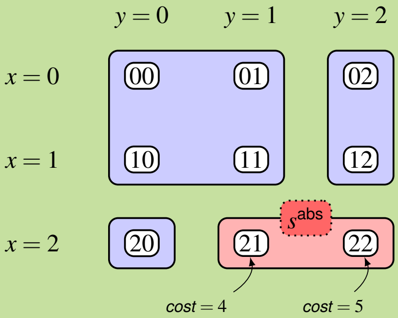
Cartesian Abstractions

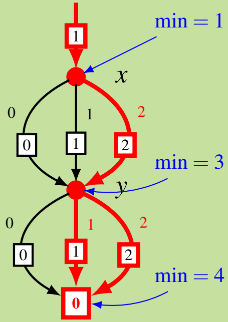
Example (Cartesian abstraction)

Cartesian abstraction over x, y

Cost $x+y+1$

(edges consistent with s^{abs})





Cartesian Abstractions

What happens here? *or:*

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

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 not Cartesian: two variables can be

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

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

Cartesian Abstractions

Not Cartesian!

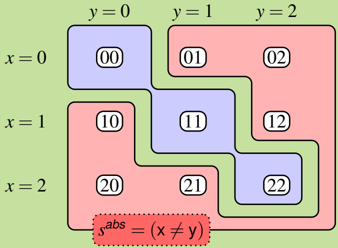
If abstraction not Cartesian: two variables can be

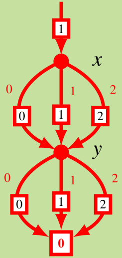
- independent in cost function ($\rightsquigarrow$ compact EVMDD), but
- dependent in abstraction.

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

Example (Non-Cartesian abstraction)

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



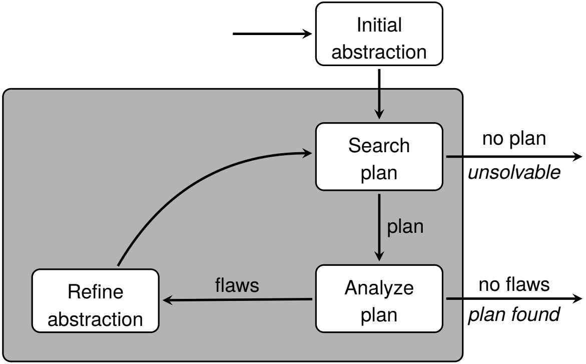


UNI FREIBURG

Counterexample-Guided Abstraction Refinement

Wanted: principled way of computing Cartesian abstractions.

⇝ Counterexample-Guided Abstraction Refinement (CEGAR)



Counterexample-Guided Abstraction Refinement

Possible flaws in abstract plan:

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

Counterexample-Guided Abstraction Refinement

Possible flaws in abstract plan:

1. Concrete state does not fit abstract state (concrete and abstract traces diverge)
2. Action 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

Possible flaws in abstract plan:

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

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

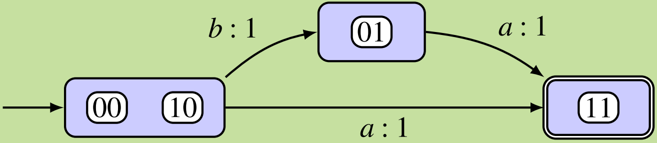
4. Cost-mismatch flaw: Action more costly in concrete state than in abstract state

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

UNI FREIBURG

Counterexample-Guided Abstraction Refinement

Example (Cost-mismatch flaw)

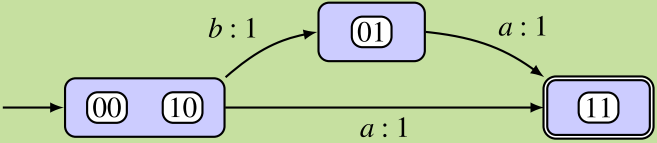


$$a = \langle \top, \quad x \wedge y \rangle, \quad \mathit{cost}_a = 2x + 1 \qquad s_0 = 10$$
$$b = \langle \top, \neg x \wedge y \rangle, \quad \mathit{cost}_b = 1 \qquad s_\star = x \wedge y$$

Background
Compilation
Relaxations
Abstractions
Cartesian Abstractions
Counterexample-Guided Abstraction Refinement
Summary

Counterexample-Guided Abstraction Refinement

Example (Cost-mismatch flaw)

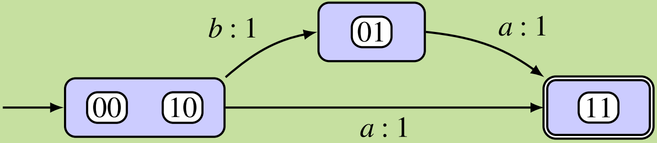


$$a = \langle \top, \quad x \wedge y \rangle, \quad \mathit{cost}_a = 2x + 1 \qquad s_0 = 10$$
$$b = \langle \top, \neg x \wedge y \rangle, \quad \mathit{cost}_b = 1 \qquad s_\star = x \wedge y$$

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

Counterexample-Guided Abstraction Refinement

Example (Cost-mismatch flaw)

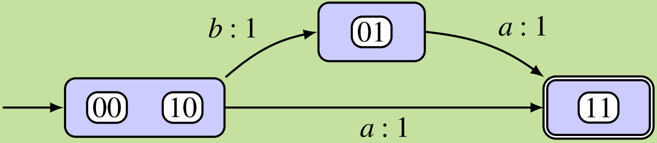


$$a = \langle \top, \quad x \wedge y \rangle, \quad \mathit{cost}_a = 2x + 1 \qquad s_0 = 10$$
$$b = \langle \top, \neg x \wedge y \rangle, \quad \mathit{cost}_b = 1 \qquad s_\star = x \wedge y$$

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

Counterexample-Guided Abstraction Refinement

Example (Cost-mismatch flaw)



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

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

Section

Summary

Summary

Summary: EVMDDs

- compact representation of cost functions
- exhibit additive structure

Recall: motivating challenges

- compiling SDAC away $\rightsquigarrow$ solved!
  - EVMDD-based action compilation
  - preserves h^{add} and h^{abs}
- SDAC-aware h values $\rightsquigarrow$ possible!
  - h^{add}
  - RPG embedding
  - Cartesian abstraction heuristics

Future Work

Future Work:

- Other delete-relaxation heuristics such as h^{FF}
- Static and dynamic EVMDD variable orders

Part II

Practice

Section

Libraries

EVMDD Libraries

MEDDLY

- MEDDLY: Multi-terminal and Edge-valued
  Decision Diagram LibrarY
- Authors: Junaid Babar and Andrew Miner
- Language: C++
- License: open source (LGPLv3)
- Advantages:
  - many different types of decision diagrams
  - mature and efficient
- Disadvantages:
  - documentation
- Code: `http://meddly.sourceforge.net`

EVMDD Libraries

pyevmdd

- pyevmdd: EVMDD library for Python
- Authors: RM and FG
- Language: Python
- License: open source (GPLv3)
- Disadvantages:
  - restricted to EVMDDs
  - neither mature nor optimized
- Purpose: our EVMDD playground
- Code: `https://github.com/robertmattmueller/pyevmdd`
- Documentation: `http://pyevmdd.readthedocs.io/en/latest/`

Section

PDDL

PDDL Representation

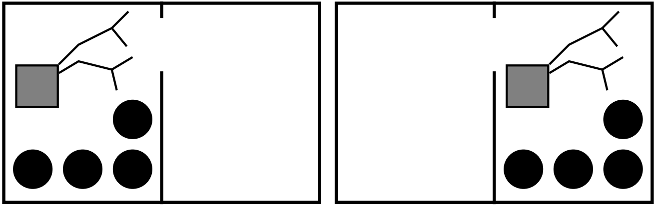
Usual way of representing costs in PDDL:

- effects `(increase (total-cost) (<expression>))`
- metric `(minimize (total-cost))`

Custom syntax:

- Besides `:parameters`, `:precondition`, and `:effect`, actions may have field
- `:cost (<expression>)`

Gripper

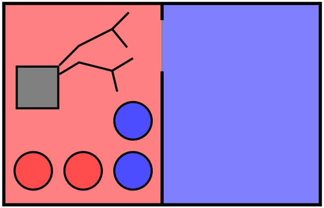


initial state goal state

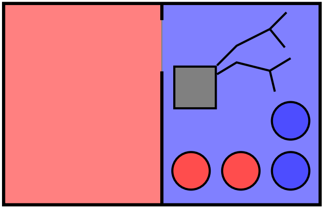
Libraries

PDDL

Colored Gripper



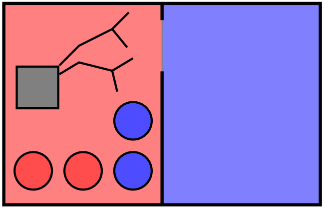
initial state



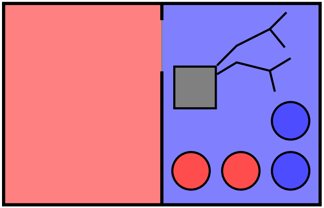
goal state

- Colored rooms and balls
- Cost of move increases if ball color differs from its room color
- Goal did not change!

Colored Gripper



initial state



goal state

- Colored rooms and balls
- Cost of move increases if ball color differs from its room color
- Goal did not change!

$$cost(move) = \sum_{\text{ROOM}} \sum_{\text{BALL}} (at(\text{BALL}, \text{ROOM}) \wedge (red(\text{BALL})) \wedge (blue(\text{ROOM}))$$
$$+ \sum_{\text{ROOM}} \sum_{\text{BALL}} (at(\text{BALL}, \text{ROOM}) \wedge (blue(\text{BALL})) \wedge (red(\text{ROOM}))$$

EVMDD-Based Action Compilation

Example (EVMDD-based action compilation)

Let $a = \langle \mathit{pre}, \mathit{eff} \rangle$, $\mathit{cost}_a = xy^2 + z + 2$.

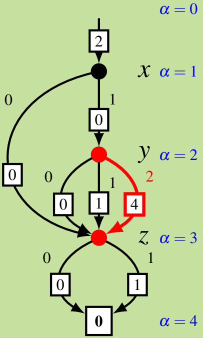
Auxiliary variables:

- One semaphore variable σ with $\mathcal{D}_\sigma = \{0, 1\}$ for entire planning task.
- One auxiliary variable $\alpha = \alpha_a$ with $\mathcal{D}_{\alpha_a} = \{0, 1, 2, 3, 4\}$ for action a.

Replace a by new auxiliary actions (similarly for other actions).

EVMDD-Based Action Compilation

Example (EVMDD-based action compilation, ctd.)



$$
\begin{aligned}
a^{pre} &= \langle pre \wedge \sigma = 0 \wedge \alpha = 0, \\
&\qquad \sigma = 1 \wedge \alpha = 1\rangle, && cost = 2 \\
a^{1,x=0} &= \langle \alpha = 1 \wedge x = 0,\ \alpha = 3\rangle, && cost = 0 \\
a^{1,x=1} &= \langle \alpha = 1 \wedge x = 1,\ \alpha = 2\rangle, && cost = 0 \\
a^{2,y=0} &= \langle \alpha = 2 \wedge y = 0,\ \alpha = 3\rangle, && cost = 0 \\
a^{2,y=1} &= \langle \alpha = 2 \wedge y = 1,\ \alpha = 3\rangle, && cost = 1 \\
a^{2,y=2} &= \langle \alpha = 2 \wedge y = 2,\ \alpha = 3\rangle, && cost = 4 \\
a^{3,z=0} &= \langle \alpha = 3 \wedge z = 0,\ \alpha = 4\rangle, && cost = 0 \\
a^{3,z=1} &= \langle \alpha = 3 \wedge z = 1,\ \alpha = 4\rangle, && cost = 1 \\
a^{eff} &= \langle \alpha = 4,\ eff \wedge \sigma = 0 \wedge \alpha = 0\rangle, && cost = 0
\end{aligned}
$$

EVMDD-Based Action Compilation Tool

- Disclaimer:
  - Not completely functional
  - Still some bugs
- Uses pyevmdd
- Language: Python
- License: open source
- Code: `https://github.com/robertmattmueller/sdac-compiler`

Part III

Acknowledgements

Acknowledgements

Acknowledgements:

- Christian Muise, for taking the time to get our compiler running in the cloud.
- Erik Wacker, for working on the compiler.
- Thomas Keller, for doing all the reasearch behind this tutorial with us.

UNI FREIBURG

Part IV

References

References I

Blai Bonet and Hector Geffner.
Planning as heuristic search: New results.
In **Proc. ECP**, pages 359–371, 1999.

Blai Bonet, Gábor Loerincs, and Hector Geffner.
A robust and fast action selection mechanism for planning.
In **Proc. AAAI**, pages 714–719, 1997.

Junaid Badar and Andrew Miner.
MEDDLY: Multi-terminal and Edge-valued Decision Diagram LibrarY.
`http://meddly.sourceforge.net/`, 2011.

References II

Thomas Ball, Andreas Podelski, and Sriram K. Rajamani.
Boolean and Cartesian abstraction for model checking C programs.
In **Proc. TACAS**, pages 268–283, 2001.

Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith.
Counterexample-guided abstraction refinement.
In **Proc. CAV**, pages 154–169, 2000.

Gianfranco Ciardo and Radu Siminiceanu.
Using edge-valued decision diagrams for symbolic generation of shortest paths.
In **Proc. FMCAD**, pages 256–273, 2002.

References III

Florian Geißer, Thomas Keller, and Robert Mattmüller.
Delete relaxations for planning with state-dependent action costs.
In **Proc. IJCAI**, pages 1573–1579, 2015.

Florian Geißer, Thomas Keller, and Robert Mattmüller.
Abstractions for planning with state-dependent action costs.
In **Proc. ICAPS**, 2016.

Franc Ivankovic, Patrik Haslum, Sylvie Thiébaux, Vikas Shivashankar, and Dana S. Nau.
Optimal planning with global numerical state constraints.
In **Proc. ICAPS**, pages 145–153, 2014.

References IV

Thomas Keller, Florian Pommerening, Jendrik Seipp, Florian Geißer, and Robert Mattmüller.
State-dependent cost partitionings for Cartesian abstractions in classical planning.
In **Proc. IJCAI**, 2016.
To appear.

Yung-Te Lai, Massoud Pedram, and Sarma B. K. Vrudhula.
Formal verification using edge-valued binary decision diagrams.
IEEE Transactions on Computers, 45(2):247–255, 1996.

Jendrik Seipp and Malte Helmert.
Counterexample-guided Cartesian abstraction refinement.
In **Proc. ICAPS**, pages 347–351, 2013.