

Principles of AI Planning

10. Digression: Planning with State-Dependent Action Costs

Albert-Ludwigs-Universität Freiburg



Bernhard Nebel and Robert Mattmüller

December 2nd, 2015

1 Motivation



- State-Dependent Action Costs
- Running Example
- Cost Formalism
- State-Dependent Action Costs in Forward Search

- Motivation
- State-Dependent Action Costs
- Running Example
- Cost Formalism
- Forward Search
- Delete
- Relaxations
- Conclusion

December 2nd, 2015

B. Nebel, R. Mattmüller – AI Planning

3 / 43

Why State-Dependent Action Costs?



- State-dependent action costs allow a higher-level representation of operators.
- More natural representation of many actions:
 - `drive_from_to(Basel, Freiburg)` at cost 70,
 - `drive_from_to(Stuttgart, Freiburg)` at cost 200,
 - `drive_from_to(Frankfurt, Freiburg)` at cost 270 vs.
 - `drive_to(Freiburg)` at cost 70 if in Basel, 200 if in Stuttgart, and 270 if in Frankfurt.
- Leads to fewer parameters in action schemas and to fewer actions.

- Motivation
- State-Dependent Action Costs
- Running Example
- Cost Formalism
- Forward Search
- Delete
- Relaxations
- Conclusion

December 2nd, 2015

B. Nebel, R. Mattmüller – AI Planning

4 / 43

Why State-Dependent Action Costs?



- In classical planning: actions have **unit costs**.
 - Each action o costs 1.
- Simple extension: actions have **constant costs**.
 - Each action o costs some $c_o \in \mathbb{N}$.
 - Example: Driving between two cities costs amount proportional to distance.
 - Still easy to handle algorithmically, e.g. when computing g and h values.
- Further extension: actions have **state-dependent costs**.
 - Each action o has **cost function** $c_o : S \rightarrow \mathbb{N}$.
 - Example: Driving to a destination city costs amount proportional to distance, depending on the current city.
 - Computing g values in forward search still easy, but definition and computation of forward-cost heuristics like h_{add} or h_{max} unclear.

- Motivation
- State-Dependent Action Costs
- Running Example
- Cost Formalism
- Forward Search
- Delete
- Relaxations
- Conclusion

December 2nd, 2015

B. Nebel, R. Mattmüller – AI Planning

5 / 43

State-Dependent Action Costs

Running Example



For the rest of this chapter, we consider the following running example.

Example

Action `clean` cleans all dishes that are currently dirty. Its cost depends on how many pieces are dirty.

There are three pieces:

- one plate (costs 1 unit to clean)
- one mug (costs 2 units to clean), and
- one bowl (costs 4 units to clean).

(Question: How much can applying action `clean` cost?)

- Motivation
- State-Dependent Action Costs
- Running Example
- Cost Formalism
- Forward Search
- Delete Relaxations
- Conclusion

Planning With State-Dependent Action Costs



Definition

A **planning task with state-dependent action costs** or **SDAC planning task** is a tuple $\Pi = \langle \Theta, (c_o)_{o \in O} \rangle$ where

- Θ is an **SAS⁺ planning task** with operators O and
- $c_o : S \rightarrow \mathbb{N}$ is the **cost function** of o for each $o \in O$.

Definitions of plans etc. stay as before. A plan is **optimal** if it minimizes the sum of action costs from start to goal.

- Motivation
- State-Dependent Action Costs
- Running Example
- Cost Formalism
- Forward Search
- Delete Relaxations
- Conclusion

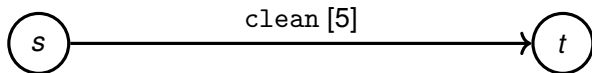
Planning With State-Dependent Action Costs

Forward Search



Example

Assume in state s , the plate^[1] and the bowl^[4] are dirty and the mug^[2] is clean. Then when expanding s , we know that the action `clean` costs 5 units there.



This means:

- We can easily compute g values in forward search.
- But what about h values?

- Motivation
- State-Dependent Action Costs
- Running Example
- Cost Formalism
- Forward Search
- Delete Relaxations
- Conclusion

2 State-Dependent Action Costs and Delete Relaxations



- Delete Relaxations in SAS⁺
- Action Costs in Relaxed States
- Additive Decomposition of Action Costs
- Edge-Valued Multi-Valued Decision Diagrams
- EVMDD-Based Action Compilation
- The Additive Heuristic
- RPG Compilation

- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDDs
- Action Compilation
- The Heuristic
- RPG Compilation
- Conclusion

- Assume we want to compute the **additive heuristic** h_{add} in a task with state-dependent action costs.
- But what does an action o cost in a relaxed state s^+ ?
- And how to compute that cost?

Delete relaxation in SAS⁺ tasks works as follows:

- Operators are already in effect normal form.
- We do not need to impose a positive normal form, because all conditions are conjunctions of facts, and facts are just variable-value pairs and in that sense always positive.
- Hence $o^+ = o$ for any operator o , and $\Pi^+ = \Pi$.
- For simplicity, we identify relaxed states s^+ with their on-sets $on(s^+)$.
- Then, a relaxed state s^+ is a set of facts (v, d) with $v \in V$ and $d \in \mathcal{D}_v$ including at least one fact (v, d) for each $v \in V$ (but possibly more than one, which is what makes it a **relaxed** state).

- A relaxed operator o is applicable in a relaxed state s^+ if all precondition facts of o are contained in s^+ .
- Relaxed states **accumulate** facts reached so far.
- Applying a relaxed operator o to a relaxed state s^+ adds to s^+ those facts made true by o .

Relaxed plans, dominance, monotonicity etc. as before. The above definition generalizes the one for propositional tasks.

Example

Our cleaning task contains the facts

- (plate, **clean**) and (plate, **dirty**),
- (mug, **clean**) and (mug, **dirty**), and
- (bowl, **clean**) and (bowl, **dirty**).

We abbreviate them as pc , pd , mc , md , bc , and bd .

Then, informally, $c_{\text{clean}} = pd + 2 \cdot md + 4 \cdot bd$.

A bit more formally,

$$c_{\text{clean}}(s) = s(pd) + 2 \cdot s(md) + 4 \cdot s(bd),$$

where $s(\text{fact}) = 1$ if $\text{fact} \in s$, and $s(\text{fact}) = 0$, otherwise.

Action Costs in Relaxed States

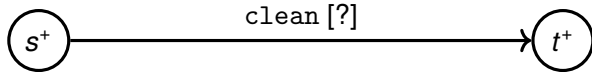


Example

Assume s^+ is the relaxed state with

$$s^+ = \{pc, pd, md, bc, bd\}.$$

Then, what should `clean` cost in s^+ ?



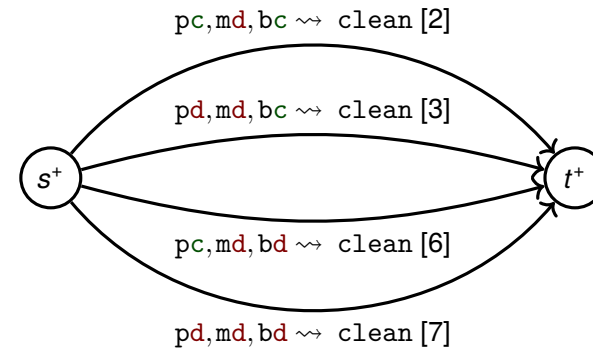
- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

Action Costs in Relaxed States



Idea: We should assume the cheapest way of applying o^+ in s^+ to guarantee admissibility of h^+ .
(Allow at least the behavior of the unrelaxed setting.)

Example



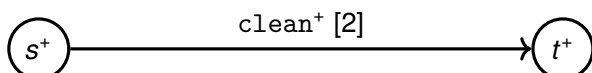
- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

Action Costs in Relaxed States



Idea: We should assume the cheapest way of applying o^+ in s^+ to guarantee admissibility of h^+ .
(Allow at least the behavior of the unrelaxed setting.)

Example



- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

Action Costs in Relaxed States



Definition

Let V be a set of FDR variables, $s : V \rightarrow \bigcup_{v \in V} \mathcal{D}_v$ an unrelaxed state over V , and $s^+ \subseteq \{(v, d) \mid v \in V, d \in \mathcal{D}_v\}$ a relaxed state over V . We call s **consistent** with s^+ if $\{(v, s(v)) \mid v \in V\} \subseteq s^+$.

Definition

Let $o \in O$ be an action with cost function c_o , and s^+ a relaxed state. Then the **relaxed cost** of o in s^+ is defined as

$$c_o(s^+) = \min_{s \in S \text{ consistent with } s^+} c_o(s).$$

(Question: How many states s are consistent with s^+ ?)

- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

Action Costs in Relaxed States

Problem with this definition: There are generally exponentially many states s consistent with s^+ to minimize over.

Example

There are exponentially many subsets of pieces that can be dirty, leading to exponentially many different cost values of exponentially many states consistent with s^+ .

Central question: Can we still do this minimization efficiently?

- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{has}
- RPG Compilation
- Conclusion

Additive Decomposition of Action Costs

Answer: Kind of, if we assume we can **additively decompose the cost function by variables**.

Example

The function $c_{\text{clean}} = p\text{d} + 2 \cdot m\text{d} + 4 \cdot b\text{d}$ is a sum of independent costs depending on a single fact each.

- Fact $p\text{d}$ contributes cost 1.
- Fact $m\text{d}$ contributes cost 2.
- Fact $b\text{d}$ contributes cost 4.

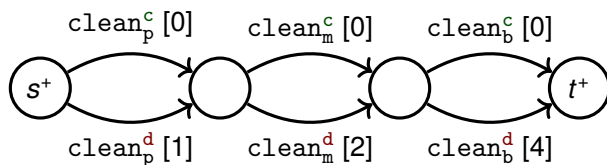
This suggests that, when computing $c_{\text{clean}}(s^+)$, we should assume that we can clean all pieces individually, in sequence.

- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{has}
- RPG Compilation
- Conclusion

Additive Decomposition of Action Costs

Example

- Action $\text{clean}_{\text{piece}}^c$ cleans piece, costs 0, and has the additional precondition that piece is already clean (c).
- Action $\text{clean}_{\text{piece}}^d$ cleans piece, costs the actual cleaning cost of piece (1, 2 or 4, depending on the piece), and has the additional precondition that piece is still dirty (d).

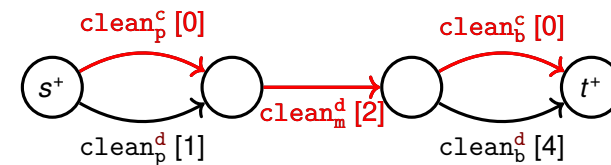


- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{has}
- RPG Compilation
- Conclusion

Additive Decomposition of Action Costs

Example

In s^+ , all facts but $m\text{c}$ are contained, i.e., all actions but clean_m^c are applicable.



Hence, we may choose the cheapest viable path from s^+ to t^+ to determine $c_{\text{clean}}(s^+) = 0 + 2 + 0$.

- Motivation
- Delete Relaxations
- SAS⁺
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{has}
- RPG Compilation
- Conclusion

Additive Decomposition of Action Costs



Remark: When there are such inter-variable independencies in the cost function, the additive decomposition is up to exponentially more efficient than minimizing over all states s consistent with s^+ . (Why?)

Question: Does this additive decomposition of the cost function always work?

Answer: Unfortunately not.

Counterexample:

- Variables x and y with domains $\mathcal{D}_x = \mathcal{D}_y = \{0, \dots, 10\}$.
- Action o with cost function $c_o = |x - 5| \cdot |y - 5|$.
- Need to consider all combinations of x and y and whether they are consistent with s^+ .

- Motivation
- Delete Relaxations
- SAS*
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

Edge-Valued Multi-Valued Decision Diagrams (EVMDs)



Idea: We can still try to exploit additive independencies, where they exist, and multiply out variable domains, otherwise.

Observation/good news: There is a data structure for the encoding of arithmetic functions that does exactly that: **edge-valued multi-valued decision diagrams (EVMDs)!**

- Motivation
- Delete Relaxations
- SAS*
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

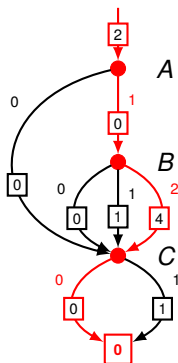
EVMDs

Cost Computation for Unrelaxed States



Example

Arithmetic function $c_o = AB^2 + C + 2$ with $\mathcal{D}_A = \mathcal{D}_C = \{0, 1\}$, $\mathcal{D}_B = \{0, 1, 2\}$.



- Directed acyclic graph
- Dangling incoming edge
- Single terminal node **0**
- Variable nodes with:
 - edge label
 - edge weights
- We see: C independent from rest, B only matters if $A \neq 0$.

- Motivation
- Delete Relaxations
- SAS*
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

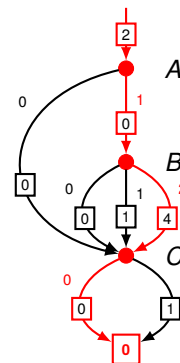
EVMDs

Cost Computation for Unrelaxed States



Example

Arithmetic function $c_o = AB^2 + C + 2$ with $\mathcal{D}_A = \mathcal{D}_C = \{0, 1\}$, $\mathcal{D}_B = \{0, 1, 2\}$.

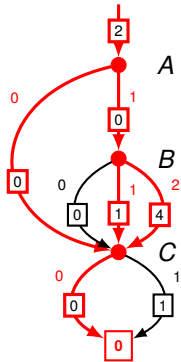


- $s : A=1, B=2, C=0$
- $c_a(s) = 2 + 0 + 4 + 0 = 6$

- Motivation
- Delete Relaxations
- SAS*
- Costs in Relaxed States
- Additive Cost Decomposition
- EVMDs
- Action Compilation
- π_{SAS}
- RPG Compilation
- Conclusion

Example

Relaxed state $s^+ = \{(A, 0), (A, 1), (B, 1), (B, 2), (C, 0)\}$.



- Computing $c_o(s^+) =$ minimizing over $c_o(s)$ for all s consistent with $s^+ =$ minimizing over all start-end-paths in EVMDD following only edges consistent with s^+ .
- **Observation:** Minimization over exponentially many paths can be replaced by **top-sort traversal of EVMDD**, minimizing over incoming arcs consistent with s^+ at all nodes!

Motivation

Delete Relaxations

SAS*

Costs in Relaxed States

Additive Cost Decomposition

EVMDDs

Action Compilation

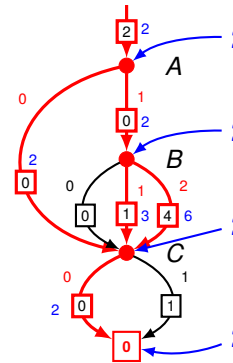
τ_{pass}

RPG Compilation

Conclusion

Example

Relaxed state $s^+ = \{(A, 0), (A, 1), (B, 1), (B, 2), (C, 0)\}$.



- $c_o(s^+) = 2$
- Cost-minimizing s consistent with s^+ : $s(A) = s(C) = 0, s(B) \in \{1, 2\}$.

Motivation

Delete Relaxations

SAS*

Costs in Relaxed States

Additive Cost Decomposition

EVMDDs

Action Compilation

τ_{pass}

RPG Compilation

Conclusion

Edge-Valued Multi-Valued Decision Diagrams (EVMDDs)

Theorem

A top-sort traversal of the EVMDD for c_o , adding edge weights and minimizing over incoming arcs consistent with s^+ at all nodes, computes $c_o(s^+)$ and takes time in the order of the size of the EVMDD. □

Good news: Size of the EVMDD is small for many well-behaved cost functions. (**Question:** For example?)

Bad news: In the worst case, the size of the EVMDD itself is exponential in the number of state variables on which the cost function depends.

So? This is as good as it gets. In the worst case, we cannot avoid minimizing over exponentially many unrelaxed states. But often, we can.

Motivation

Delete Relaxations

SAS*

Costs in Relaxed States

Additive Cost Decomposition

EVMDDs

Action Compilation

τ_{pass}

RPG Compilation

Conclusion

EVMDD-Based Action Compilation

Okay, but this does not give us any heuristic values yet!

- But there is one thing we can already do: Remember that in the example we broke down the clean action into three micro actions with separate, independent costs to clean the individual pieces?
- We can generalize this idea for arbitrary EVMDDs: each edge in the EVMDD becomes a new micro action with constant cost corresponding to the edge constraint, precondition that we are currently at its start EVMDD node, and effect that we are currently at its target EVMDD node.

Motivation

Delete Relaxations

SAS*

Costs in Relaxed States

Additive Cost Decomposition

EVMDDs

Action Compilation

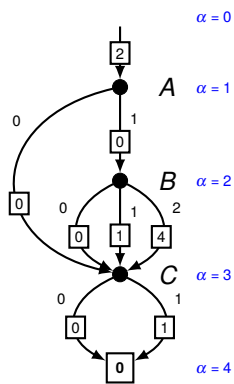
τ_{pass}

RPG Compilation

Conclusion

Example

Assume $o = \langle \chi, e \rangle$, with $c_o = AB^2 + C + 2$. Use auxiliary variable α with $\mathcal{D}_\alpha = \{0, 1, 2, 3, 4\}$. Replace o by new auxiliary actions:



- $o^\chi = \langle \chi \wedge \alpha = 0, \alpha = 1 \rangle$, cost 2
- $o^{1,3,A=0} = \langle \alpha = 1 \wedge A = 0, \alpha = 3 \rangle$, cost 0
- $o^{1,2,A=1} = \langle \alpha = 1 \wedge A = 1, \alpha = 2 \rangle$, cost 0
- $o^{2,3,B=0} = \langle \alpha = 2 \wedge B = 0, \alpha = 3 \rangle$, cost 0
- $o^{2,3,B=1} = \langle \alpha = 2 \wedge B = 1, \alpha = 3 \rangle$, cost 1
- $o^{2,3,B=2} = \langle \alpha = 2 \wedge B = 2, \alpha = 3 \rangle$, cost 4
- $o^{3,4,C=0} = \langle \alpha = 3 \wedge C = 0, \alpha = 4 \rangle$, cost 0
- $o^{3,4,C=1} = \langle \alpha = 3 \wedge C = 1, \alpha = 4 \rangle$, cost 1
- $o^e = \langle \alpha = 4, e \wedge \alpha = 0 \rangle$, cost 0

Motivation
Delete Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 τ_{base}
RPG Compilation
Conclusion

We call the planning task obtained from an SDAC task as sketched above the **EVMDD-based action compilation**.

You may wonder if the EVMDD-based action compilation is always possible. The answer is yes!

- **Existence:** Each action cost function over finitely many finite-domain state variables can be encoded as an EVMDD.
- **Canonicity:** For a fixed variable ordering, reduced ordered EVMDDs are unique (cf. theory of Binary Decision Diagrams, BDDs).
- **Basic Operations:** EVMDDs support basic arithmetic operations such as addition, subtraction, multiplication.

Remark: We may even use different variable orders for cost functions of different actions.

Motivation
Delete Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 τ_{base}
RPG Compilation
Conclusion

Theorem

The EVMDD-based action compilation has a size that is polynomial in the size of the original SDAC planning task times the size of the largest EVMDD used in the compilation.

Theorem

The EVMDD-based action compilation is a planning task with constant (state-independent) action costs.

Theorem

The EVMDD-based action compilation of an SDAC planning task is equivalent to the original task in the sense that they admit the same plans (modulo replacement of SDAC actions by action sequences in the compilation). Moreover, optimal plan costs are preserved.

Motivation
Delete Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 τ_{base}
RPG Compilation
Conclusion

Recall the usual definition of h_{add} for classical planning:

Definition

$$h_{\text{add}}^s(\text{Facts}) = \sum_{\text{fact} \in \text{Facts}} h_{\text{add}}^s(\text{fact})$$

$$h_{\text{add}}^s(\text{fact}) = \begin{cases} 0 & \text{if } \text{fact} \in s \\ \min_{\text{achiever } o = \langle \chi, e \rangle \text{ of } \text{fact}} [h_{\text{add}}^s(\chi) + c_o] & \text{otherwise} \end{cases}$$

Motivation
Delete Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 τ_{base}
RPG Compilation
Conclusion

Example

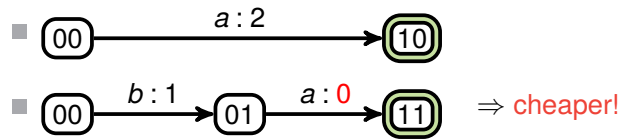
$$a = \langle \top, A := 1 \rangle \quad c_a = 2 - 2 \cdot B$$

$$b = \langle \top, B := 1 \rangle \quad c_b = 1$$

$$s : A=0 \wedge B=0$$

$$h_{\text{add}}^s(B=1) = 1$$

$$h_{\text{add}}^s(A=1) = ?$$



Motivation
Delete
Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 h_{add}
RPG Compilation
Conclusion

Minimize over all situations where a is applicable (WLOG we assume, for simplicity, that no variable occurs both in the cost function and the precondition of the same action):

Definition

$$h_{\text{add}}^s(\text{fact}) = \begin{cases} 0 & \text{if } \text{fact} \in s \\ \min_{\text{achiever } o=\langle \chi, e \rangle \text{ of } \text{fact}} [h_{\text{add}}^s(\chi) + C_s^o] & \text{otherwise} \end{cases}$$

$$C_s^o = \min_{\hat{s} \in S_o} [c_o(\hat{s}) + h_{\text{add}}^s(\hat{s})]$$

S_o : set of valuations of variables in cost function

Motivation
Delete
Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 h_{add}
RPG Compilation
Conclusion

Example

$$a = \langle \top, A := 1 \rangle \quad c_a = 2 - 2 \cdot B$$

$$b = \langle \top, B := 1 \rangle \quad c_b = 1$$

$$s : A=0 \wedge B=0$$

$$h_{\text{add}}^s(B=1) = 1$$

$$h_{\text{add}}^s(A=1) = 1$$

- $C_s^a = \min_{\hat{s} \in S_a} [c_a(\hat{s}) + h_{\text{add}}^s(\hat{s})]$
- $\hat{s}_0 : B=0 \longrightarrow c_a(B=0) + h_{\text{add}}(B=0) = 2 + 0 = 2$
- $\hat{s}_1 : B=1 \longrightarrow c_a(B=1) + h_{\text{add}}(B=1) = 0 + 1 = 1$

Motivation
Delete
Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 h_{add}
RPG Compilation
Conclusion

Theorem

Let Π be an SDAC planning task, let Π' be an EVMDD-based action compilation of Π , and let s be a state of Π . Then the classical h_{add} heuristic in Π' gives the same value for s as the generalization of h_{add} to SDAC tasks defined above gives for s in Π . □

Motivation
Delete
Relaxations
SAS*
Costs in Relaxed States
Additive Cost Decomposition
EVMDDs
Action Compilation
 h_{add}
RPG Compilation
Conclusion

3 Conclusion



Motivation
Delete
Relaxations
Conclusion

SDAC Planning and EVMDDs

Conclusion



Motivation
Delete
Relaxations
Conclusion

Summary:

- State-dependent actions costs practically relevant.
- EVMDDs exhibit and exploit structure in cost functions.
- Graph-based representations of arithmetic functions.
- Edge values express partial cost contributed by facts.
- Size of EVMDD is **compact** in many “typical” cases.
- Can be used to compile tasks with state-dependent costs to tasks with state-independent costs.
- Alternatively, can be embedded into the RPG to compute forward-cost heuristics directly.
- For h_{add} , both approaches give the same heuristic values.

SDAC Planning and EVMDDs

Conclusion



Motivation
Delete
Relaxations
Conclusion

Future Work:

- Investigation of other delete-relaxation heuristics for tasks with state-dependent action costs.
- Investigation of abstraction heuristics in SDAC setting (*work in progress*).
- Investigation of static and dynamic EVMDD variable orders (static variable orders: *work in progress*).
- Better integration of SDAC in PDDL (*work in progress*).
- Tool support (*work in progress*).
- Benchmarks.

SDAC Planning and EVMDDs

References



Motivation
Delete
Relaxations
Conclusion

- G. Ciardo and R. Siminiceanu, “Using edge-valued decision diagrams for symbolic generation of shortest paths,” in **Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2002)**, pp. 256–273, 2002.
- F. Geißer, T. Keller, and R. Mattmüller, “Delete relaxations for planning with state-dependent action costs,” in **Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)**, pp. 1573–1579, 2015.