Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams

David Speck and **Florian Geißer** and **Robert Mattmüller**
University of Freiburg, Germany
{speckd, geisserf, mattmuel}@informatik.uni-freiburg.de

Abstract

Symbolic representations have attracted significant attention in optimal planning. Binary Decision Diagrams (BDDs) form the basis for symbolic search algorithms. Closely related are Algebraic Decision Diagrams (ADDs), used to represent heuristic functions. Also, progress was made in dealing with models that take state-dependent action costs into account. Here, costs are represented as Edge-valued Multi-valued Decision Diagrams (EVMDDs), which can be exponentially more compact than the corresponding ADD representation. However, they were not yet considered for symbolic planning.

In this work, we study EVMDD-based symbolic search for optimal planning. We define EVMDD-based representations of state sets and transition relations, and show how to compute the necessary operations required for EVMDD-A*. This EVMDD-based version of symbolic A* generalizes its BDD variant, and allows to solve planning tasks with state-dependent action costs. We prove theoretically that our approach is sound, complete and optimal. Additionally, we present an empirical analysis comparing EVMDD-A* to BDD-A* and explicit A* search. Our results underscore the usefulness of symbolic approaches and the feasibility of dealing with models that go beyond unit costs.

Introduction

The motivation for this paper comes from two related sources. First, symbolic search has proven to be a useful approach to optimal classical planning, as the results of the sequential optimal track of the International Planning Competition (IPC) 2014 show. Usually, Binary Decision Diagrams (BDDs) (Bryant 1986) are used as the underlying symbolic data structure (Edelkamp and Reffel 1998; Edelkamp and Helmert 2001; Torralba et al. 2014a). The second source of motivation is the observation that there are alternative data structures that have advantages over BDDs that we attempt to exploit in this paper.

In particular, we address the following three shortcomings of BDDs: (a) They only allow branching over binary variables, which necessitates representing a single multi-valued variable from the planning task by several binary BDD variables. While this is more of a nuisance than a big deal, it does complicate, e. g., the task of finding good variable orderings. Luckily, this issue is trivial to address by allowing decision diagrams to branch over multi-valued variables. (b) BDDs only allow two output values. This is a slightly more serious issue, since it necessitates complicated techniques like bucketing of the open list into subsets with identical g-values. While this already happens for constant-cost tasks, we expect an even greater fragmentation into diverse, possibly often small, buckets for tasks with varying and state-dependent costs (Geißer, Keller, and Mattmüller 2015). Again, in principle, it is easy to work around this issue by allowing more than two output values. This is exactly what multi-terminal decision diagrams, such as Algebraic Decision Diagrams (ADDs), do (Bahar et al. 1997). However, (c) ADDs can be exponentially larger than equivalent representations that do not carry the output values at their terminal nodes, but in the form of edge values, such as Edge-valued Binary (or: Multi-valued) Decision Diagrams (EVBDDs/EVMDDs) (Lai, Pedram, and Vrudhula 1996; Ciardo and Siminiceanu 2002). For example, the function $f : \{0,1\}^{n+1} \rightarrow \{0,\dots,2^{n+1}-1\}$ with $f(x_0,\dots,x_n) = \sum_{i=0}^{n} 2^i x_i$ has an exponentially large ADD representation for every possible variable ordering, since f takes 2^{n+1} different values, which means that an ADD representing f needs 2^{n+1} different terminal nodes. An EVBDD representation of f, on the other hand, will be linear in size for every possible variable ordering, as f is completely additively separable, and the EVBDD can handle the contributions by all variables independently (Roux and Siminiceanu 2010).

In this paper, we want to address the question whether the theoretical advantages of EVMDDs over BDDs can be put to practical use in optimal symbolic planning, and if so, under which conditions. We first define which EVMDD operations are required, and show that EVMDDs not only represent state sets, but also the costs required to reach these states. We define how transition relations, and image or preimage operations can be represented with EVMDDs. On top of this, we build a symbolic version of the A* algorithm, EVMDD-A*, which we empirically compare to BDD-A* and explicit-state A*. While BDD-A* is superior on many unit-cost tasks, EVMDD-A* performs particularly well in problems with state-dependent costs.

Preliminaries

Planning Tasks

We consider planning tasks with state-dependent action costs, but point out that classical, unit-cost tasks, are an important special case.

Definition 1. *A* planning task *is a tuple* $\Pi = (\mathcal{V}, A, s_0, s_\star, (c_a)_{a \in A})$ *consisting of the following components:* $\mathcal{V} = \{v_1, \dots, v_n\}$ *is a finite set of* state variables, *each with an associated finite domain* $\mathcal{D}_v = \{0, \dots, |\mathcal{D}_v| - 1\}$*. A* fact *is a pair* (v, d)*, alternatively written as* $v \doteq d$*, where* $v \in \mathcal{V}$ *and* $d \in \mathcal{D}_v$*, and a* partial variable assignment s *over* $\mathcal{V}$ *is a consistent set of facts. If* s *assigns a value to each* $v \in \mathcal{V}$*,* s *is called a* state*. Let* $\mathcal{S}$ *denote the set of states of* Π*. States and partial variable assignments are functions which map variables to values, i.e.* $s(v)$ *returns the value of variable* v *in state* s *(analogous for partial variable assignments).* A *is a set of* actions*, where an action is a pair* $a = \langle \text{pre}, \text{eff} \rangle$ *of partial variable assignments (or: sets of facts), called* preconditions *and* effects*. By* $\text{pre}(a)$ *we refer to the precondition of* a*, by* $\text{prevars}(a)$ *we refer to the corresponding variables. Similarly,* $\text{eff}(a)$ *and* $\text{effvars}(a)$ *refers to the effect of* a*, and the variables, respectively. The state* $s_0 \in \mathcal{S}$ *is called the* initial state*, and the partial state* $s_\star$ *specifies the* goal *condition. Each action* $a \in A$ *has an associated* cost function $c_a : \mathcal{S} \rightarrow \mathbb{N}$ *that assigns the application cost of* a *to all states.*

An action a is applicable in state s iff $\text{pre} \subseteq s$. Applying a to s yields the state $\hat{s}$ with $\hat{s}(v) = \text{eff}(a)(v)$ where $\text{eff}(a)(v)$ is defined, and $\hat{s}(v) = s(v)$ otherwise. We write $s[a]$ for $\hat{s}$. A state s is a goal state iff $s_\star \subseteq s$. We denote the set of goal states by $S_\star$. Let $\pi = \langle a_0, \dots, a_{n-1} \rangle$ be a sequence of actions from A. We call π *applicable* in s_0 if there exist states $s_1, \dots, s_n$ such that a_i is applicable in s_i and $s_{i+1} = s_i[a_i]$ for all $i = 0, \dots, n-1$. We call π a *plan* for Π if it is applicable in s_0 and if $s_n \in S_\star$. The *cost* of plan π is the sum of action costs along the induced state sequence, i.e., $cost(\pi) = \sum_{i=0}^{n-1} c_{a_i}(s_i)$.

Symbolic Search Planning

State space search is a state-of-the-art approach to finding (optimal) plans. While explicit state space search operates on individual states, symbolic search, originally introduced in the area of model checking (McMillan 1993), operates on sets of states by performing operations on a representation of their *characteristic function*. Given a set of states $S \subseteq \mathcal{S}$, its characteristic function $f_S : \mathcal{S} \rightarrow \{0, \infty\}$ represents whether $s \in S$ (then $f_S(s) = 0$) or $s \notin S$ (then $f_S(s) = \infty$).[1]

Actions are represented by so-called *transition relations* (TRs). The TR T_a of action a is the binary relation over $\mathcal{S}$ consisting of all pairs (s, t) where a is applicable in s and $s[a] = t$. Just like sets of states can be represented by their characteristic function, so can TRs. However, since TRs talk about two states at once, we need two sets of variables: the *source set* is represented by unprimed variables $v \in \mathcal{V}$. The *target set* is represented by primed variables v': for each $v \in \mathcal{V}$ we add a primed version v' with $\mathcal{D}_v = \mathcal{D}_{v'}$. Conceptually, the source set encodes the action precondition and the target set its effects. Given a set of states S and a TR T_a, the *image* operation $\text{image}(f_S, T_a)$ produces a representation of the set of successor states that can be reached from S with a.

The most prominent representation of characteristic functions and transition relations in symbolic planning are (reduced and ordered) *Binary Decision Diagrams* (BDDs) (Bryant 1986). A BDD is a directed acyclic graph with a single root node and two terminal nodes, the 0-sink and the 1-sink. Internal nodes correspond to binary[2] variables, and each node has two successors: the *low edge* represents that the current variable is false, while the *high edge* represents that the current variable is true. Evaluation of a function then corresponds to traversal of the BDD according to the assignment of the variables. Fig. 1a shows the BDD for the characteristic function $f_{\{s \in \mathcal{S} \mid s(x)=1 \text{ and } s(y)=0\}}$. For planning tasks where actions do not have unit costs, we are interested in the cost required to reach a set of states. For constant action costs, it is common to represent multiple actions with the same cost i by one (or more) transition relation(s) (i.e. one or more BDDs) T_i. During search, open and closed list are also represented as a list of BDDs, one BDD for each g-value (Edelkamp and Kissmann 2009).

For planning tasks with state-dependent action costs, it is not clear how to represent transition relations with BDDs, as the resulting successor state set may have different costs for each state. Hansen, Zhou, and Feng (2002) use another form of decision diagrams: *Algebraic Decision Diagrams* (Bahar et al. 1997) represent functions of the form $f : \mathcal{S} \rightarrow \mathbb{N} \cup \{\infty\}$ (again, multi-valued variables have binary encodings). Informally, while BDDs only have two sink nodes, ADDs have one sink node for each evaluation of f. ADDs are closely related to BDDs, and they can be transformed to a sequence of BDDs in polynomial time with at most polynomial overhead (Torralba 2015). However, in the work of Hansen, Zhou, and Feng (2002) additional improvements would be desirable, such as: (a) supporting zero action costs, (b) providing a solution for reconstructing plans after finding a goal state and (c) an empirical evaluation regarding actual planning. In the following, we will show how to realize these improvements with EVMDDs.

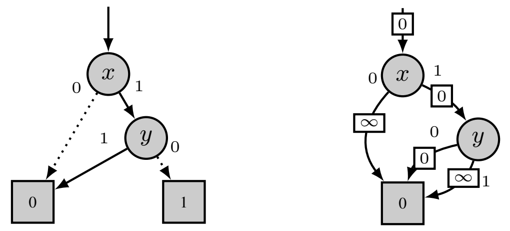
Edge-Valued Multi-Valued Decision Diagrams

Edge-valued Multi-valued Decision Diagrams (EVMDDs) (Ciardo and Siminiceanu 2002) represent functions of the form $f : \mathcal{S} \rightarrow \mathbb{N} \cup \{\infty\}$ and have gained attention in planning with state-dependent action costs (Geißer, Keller, and Mattmüller 2015; 2016; Mattmüller et al. 2018).

Definition 2. *An* EVMDD *over* $\mathcal{V}$ *is a tuple* $\mathcal{E} = \langle \kappa, \mathbf{f} \rangle$*, where* $\kappa \in \mathbb{Z}$ *is a constant and* $\mathbf{f}$ *is a directed acyclic graph consisting of two types of nodes: (i) there is a single* terminal node *denoted by* 0*. (ii) A* nonterminal node $\mathbf{v}$ *is a tuple* $(v, \chi_0, \dots, \chi_k, w_0, \dots, w_k)$ *where* $v \in \mathcal{V}$ *is a variable,* $k = |\mathcal{D}_v| - 1$*, children* $\chi_0, \dots, \chi_k$ *are terminal or*

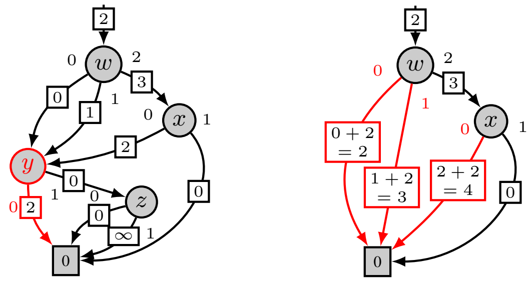
[1]For us, f_S maps to 0 or ∞, because we will interpret its values as *costs*. In a propositional setting, when dealing with BDDs, we would let f_S map to *truth values* 1 if $s \in S$, and 0 if $s \notin S$, instead.

[2]Each finite-domain variable $v \in \mathcal{V}$ can be represented by $\lceil \log_2 |\mathcal{D}_v| \rceil$ binary variables.



(a) BDD representing S. (b) EVMDD $\mathcal{E}_S$ representing S.

Figure 1: Representations of $S = \{s|\, s(x) = 1 \wedge s(y) = 0\}$.



(a) $\mathcal{E}$ before restricting y to 0. (b) $\mathcal{E}|_{y=0}$ (weights not normed).

Figure 2: Visualization of the restrict operator for EVMDDs.

nonterminal nodes of $\mathcal{E}$, and $w_0, \dots, w_k \in \mathbb{N} \cup \{\infty\}$ s.t. $\min_{i=0,\dots,k} w_i = 0$ are the weights assigned to the edges to the children. The weight *of an edge from* $\mathbf{v}$ *to child* χ_i *is* w_i.

By $\mathbf{f}$ we also refer to the root node of $\mathcal{E}$. Edges of $\mathcal{E}$ between parent and child nodes are implicit in the definition of the nonterminal nodes of $\mathcal{E}$. The following definition specifies the arithmetic function denoted by a given EVMDD.

Definition 3. *An EVMDD $\mathcal{E} = \langle \kappa, \mathbf{f} \rangle$ denotes the arithmetic function $\kappa + f$ where f is the function denoted by* $\mathbf{f}$. *The terminal node 0 denotes the constant function 0, and $(v, \chi_0, \dots, \chi_k, w_0, \dots, w_k)$ denotes the arithmetic function over $\mathcal{S}$ given by $f(s) = f_{s(v)}(s) + w_{s(v)}$, where $f_{s(v)}$ is the arithmetic function denoted by child $\chi_{s(v)}$. We write $\mathcal{E}(s)$ for $\kappa + f(s)$.*

For fixed variable orders, reduced and ordered EVMDDs (Def. 4) are unique (Lai, Pedram, and Vrudhula 1996).

Definition 4. *An EVMDD $\mathcal{E}$ is* reduced *if (i) there is no internal node $(v, \chi_0, \dots, \chi_k, 0, \dots, 0)$ with $\chi_0 = \dots = \chi_k$, and (ii) there are no two nonterminal nodes n and n' such that $n = n'$. An EVMDD is* ordered *if it satisfies the requirement that variables on each path from root to sink always appear in the same order.*

From now on we only talk about reduced and ordered EVMDDs and assume a fixed variable order. In the graphical representation of an EVMDD $\mathcal{E} = \langle \kappa, \mathbf{f} \rangle$, $\mathbf{f}$ is represented by a rooted directed acyclic graph and κ by a dangling incoming edge to the root node of $\mathbf{f}$. The terminal node is depicted by a rectangular node labeled 0. Edge labels d are written next to the edges, edge weights w_d in boxes on the edges. In the following, we define the EVMDD representation of the characteristic function for a set of states.

Definition 5. *Let $S \subseteq \mathcal{S}$ and f_S its characteristic function. We say $\mathcal{E}_S$ represents f_S and write $s \in \mathcal{E}_S$, iff $\mathcal{E}_S(s) \neq \infty$.*

Fig. 1b shows the EVMDD for the characteristic function $f_{\{s \in \mathcal{S} \,|\, s(x)=1 \text{ and } s(y)=0\}}$. Since we are interested in performing operations on characteristic functions we consider plus, minus (Lai, Pedram, and Vrudhula 1996) and the extension of union and intersection for EVMDDs (Ciardo and Siminiceanu 2002). Additionally, we rename variables.

Definition 6. *Given functions $f, g : \mathcal{S} \to \mathbb{N} \cup \{\infty\}$.* Plus *$(+)$ and* minus *$(-)$ are defined as usual: $\mathcal{E}_f + \mathcal{E}_g = \mathcal{E}_{f+g}$ and $\mathcal{E}_f - \mathcal{E}_g = \mathcal{E}_{f-g}$.* IntersectionMax *($\overset{\max}{\wedge}$) and* unionMin *($\overset{\min}{\vee}$) are defined as follows: $\mathcal{E}_f \overset{\max}{\wedge} \mathcal{E}_g = \mathcal{E}_{\max(f,g)}$, and $\mathcal{E}_f \overset{\min}{\vee} \mathcal{E}_g = \mathcal{E}_{\min(f,g)}$. The operator $[\mathcal{V}' \leftrightarrow \mathcal{V}]$ renames the primed variables into unprimed ones and vice versa.*

Intuitively, for $S, \hat{S} \subseteq \mathcal{S}$, the union $\mathcal{E}_S \overset{\min}{\vee} \mathcal{E}_{\hat{S}}$ assigns ∞ to s iff $s \notin S \cup \hat{S}$. Regarding symbolic planning, we require additional operators, which have been generalized for ADDs (Hansen, Zhou, and Feng 2002), but were not yet introduced for EVMDDs.

Additional EVMDD Operations

We define the additional operations: *existential quantification*, *restrict*, *preserve minimum* and *complement*. Implementation of these operations is realized by specialization of the *apply* procedure (Lai, Pedram, and Vrudhula 1996).

Existential Least-Cost Quantification. Existential quantification for a propositional formula ψ is realized by the expression $\exists v \psi := \psi_{|v=0} \vee \psi_{|v=1}$, where $\psi_{|v=i}$ *restricts* the value of v to i. Hansen, Zhou, and Feng (2002) extend this concept to ADDs, which they call *existential least-cost quantification*. Given an arithmetic function f, the existential least-cost quantification over a variable v, written as $\exists^{\text{LC}}_v(f)$, corresponds to the minimal cost of f for all values of v. We generalize this concept to multi-valued domains. Furthermore, if V is a set of variables, $\exists^{\text{LC}}_V(f)$ performs least-cost quantification for all variables $v \in V$:

$$\exists^{\text{LC}}_v(\mathcal{E}) := \mathcal{E}|_{v=0} \overset{\min}{\vee} \mathcal{E}|_{v=1} \overset{\min}{\vee} \cdots \overset{\min}{\vee} \mathcal{E}|_{v=|\mathcal{D}(v)|-1} \tag{1}$$

$$\exists^{\text{LC}}_V(\mathcal{E}) := \exists^{\text{LC}}_{v_1}(\exists^{\text{LC}}_{v_2}(\cdots(\exists^{\text{LC}}_{v_{|V|}}(\mathcal{E}))\cdots)) \tag{2}$$

Computing existential least-cost quantification over variable v requires $|\mathcal{D}(v)|$ calls of the *restrict* operator, and $\lceil \log_2(|\mathcal{D}(v)|) \rceil$ calls of applying *UnionMin*. Generalizing the *restrict* operator for EVMDDs is straightforward: Applying $\mathcal{E}|_{v=i}$ to a node $\mathbf{v} = (v, \chi_0, \dots, \chi_k, w_0, \dots, w_k)$ replaces $\mathbf{v}$ with χ_i. The weight w_i is pulled up and added to the corresponding edge weight connecting $\mathbf{v}$ with its parent. Fig. 2 depicts EVMDDs $\mathcal{E}$ and $\mathcal{E}|_{y=0}$.

Preserve-min. In EVMDD-based planning, the preserve-min operation can be used to extract states with lowest cost from an EVMDD. Given $f : \mathcal{S} \to \mathbb{N} \cup \{\infty\}$, we define $\mathrm{pmin}(f) : \mathcal{S} \to \mathbb{N} \cup \{\infty\}$ with $\mathrm{pmin}(f)(s) = f(s)$ if $f(s) = \min_{s \in \mathcal{S}} f(s)$ and $\mathrm{pmin}(f)(s) = \infty$, otherwise.

Complement. For arithmetic functions, no traditional complement exists; we define the complement of function $f : \mathcal{S} \to \mathbb{N} \cup \{\infty\}$ as the function $\neg f(s) = 0$, if $f(s) = \infty$ and $\neg f(s) = \infty$, otherwise. Note that this definition of the complement is not self inverse.

EVMDD-based Planning

In BDD-based symbolic planning, each BDD represents a set of states, and multiple BDDs are required to encode at what cost the states are reachable. With EVMDDs, we may encode the information about costs in the same diagram that encodes reachability. Consider Fig. 4b. The EVMDD represents the set of states $S = \{s | s(x) = 1\}$, since all other states are mapped to ∞. At the same time, the EVMDD encodes the cost of these states: $s_1 = \{x \doteq 1, y \doteq 0\}$ has a cost of 3 while $s_2 = \{x \doteq 1, y \doteq 1\}$ has a cost of 8. In the following, we introduce formal definitions which form the basis for a symbolic EVMDD variant of the A* algorithm.

Transition Relation. Recall that a transition relation for action a is defined over the *source* and *target* set, represented by unprimed variables $\mathcal{V}$ and primed variables $\mathcal{V}'$. Unlike BDDs, transition relations represented by EVMDDs also encode the cost of the action, and can represent actions with state-dependent action costs. The domain of cost function c_a is extended by primed variables $\mathcal{V}'$, and for any state t' defined over $\mathcal{V}'$ we have $c_a(s \cup t') = c_a(s)$, for all $s \in \mathcal{S}$.

Definition 7. *The* ***transition relation (TR)*** *for an action $a \in A$ is defined as follows.*

$$T_a := \bigwedge\nolimits^{\max}_{\langle v,d\rangle \in \mathrm{pre}(a)} \mathcal{E}_{\{(s,t')|s(v)=d\}} \quad (3)$$

$$\overset{\max}{\wedge} \bigwedge\nolimits^{\max}_{\langle v,d\rangle \in \mathrm{eff}(a)} \mathcal{E}_{\{(s,t')|t'(v')=d\}} \quad (4)$$

$$\overset{\max}{\wedge} \bigwedge\nolimits^{\max}_{\mathcal{V}\setminus \mathrm{effvars}(a)} \mathcal{E}_{\{(s,t')|s(v)=t'(v')\}} \quad (5)$$

$$\overset{\max}{\wedge} \mathcal{E}_{c_a} \quad (6)$$

Intuitively, construction of the transition relation consists of the intersection (i.e. intersectionMax for EVMDDs) of different sets of states. Term 3 encodes the precondition: all facts of the precondition are encoded by the EVMDD of their characteristic function over the unprimed variables. Term 4 encodes the effects, but over primed variables. Additionally, we have to encode that variables not affected by the action keep their former values. Term 5 encodes the so called *frame axioms*. Up to now, states are either assigned a cost of 0 or a cost of ∞. Finally, we "add" the action cost to the representation (Term 6); states that were mapped to 0 are now mapped to the cost of the action, other states still map to ∞. Fig. 3 shows the transition relation for an action $a = \langle x \doteq 0, x \doteq 1\rangle$ with $c_a = 5y + 1$. Note that it is in

$a = \langle x \doteq 0, x :\doteq 1\rangle$

$c_a = 5y + 1$

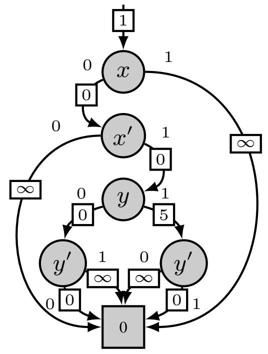


Figure 3: EVMDD T_a representing action a with cost c_a.

general possible to represent multiple actions $A' \subseteq A$ with a *single* transition relation $T = \bigvee^{\min}_{a \in A'} T_a$.

Recall that we write $s \in \mathcal{E}$ iff $\mathcal{E}(s) \neq \infty$. Given state s, s' is its corresponding primed version, with $s(v) = s'(v')$ for all $v \in \mathcal{V}$. Let s be a state over $\mathcal{V}$ and t' a state over $\mathcal{V}'$, then (s, t') is the state over $\mathcal{V} \cup \mathcal{V}'$ with $(s, t')(v) = s(v)$ and $(s, t')(v') = t'(v')$ for $v \in \mathcal{V}$, $v' \in \mathcal{V}'$. In the following, we show that T_a correctly encodes a.[3]

Lemma 1. *Let (s, t') be an arbitrary state over $\mathcal{V} \cup \mathcal{V}'$. For any action a it holds that $(s, t') \in T_a$ iff a is applicable in s and $t = s[a]$.* □

Lemma 2. *Let $(s, t') \in T_a$. Then $T_a(s, t') = c_a(s)$.*

Proof. The intermediate EVMDD T'_a of Terms (3) to (5) of Def. 7 contains only states with 0 or infinite cost (Def. 5). Since $(s, t') \in T_a$, we have $T'_a(s, t') = 0$. Then, $T_a(s, t') = (T'_a \overset{\max}{\wedge} \mathcal{E}_{c_a})(s, t') = \max(T'_a(s, t'), c_a(s, t')) = \max(0, c_a(s, t')) = c_a(s, t') = c_a(s)$. □

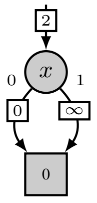
Image. The image operation takes a symbolic state and a transition relation, and computes the symbolic successor state. It consists of a sequence of basic algebraic operations. We also define the preimage, required for backward and bidirectional search algorithms.

Definition 8. *The* image *and* preimage *operations for EVMDDs are defined as follows:*

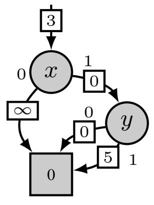
$$\mathrm{image}(\mathcal{E}, T) := (\exists^{\mathrm{LC}}_{\mathcal{V}}(\mathcal{E} + T))[\mathcal{V}' \leftrightarrow \mathcal{V}]$$
$$\mathrm{preimage}(\mathcal{E}, T) := \exists^{\mathrm{LC}}_{\mathcal{V}'}(\mathcal{E}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T)$$

The image operation consists of three steps: First, $\mathcal{E} + T$ encodes the action application. States where the precondition does not hold will be mapped to infinite costs; the action cost is added for the other states, and effects are applied (still encoded over $\mathcal{V}'$). In the second step we apply $\exists^{\mathrm{LC}}_{\mathcal{V}}$, i.e. we fix the unprimed variables with their minimum cost. This intermediate representation consists only of primed variables. Finally, we have to rename the primed variables, such that the symbolic successor state is again defined over $\mathcal{V}$, instead

[3]Full proofs of all Lemmas and Theorems are available as a technical report (Speck, Geißer, and Mattmüller 2018).



(a) A symbolic state represented as EVMDD $\mathcal{E}$ with the corresponding cost a state is reachable with.



(b) The resulting symbolic state represented by EVMDD $\hat{\mathcal{E}} = \mathrm{image}(\mathcal{E}, T_a)$. T_a is shown in Fig. 3.

Figure 4: Visualization of the image operator for EVMDDs.

of $\mathcal{V}'$ ($[\mathcal{V}' \leftrightarrow \mathcal{V}]$). Figs. 3 and 4 show an example of the image operator, computing the image of symbolic state S (Fig. 4a) with transition relation T_a (Fig. 3), resulting in Fig. 4b. In the following we show the correctness of the operator, i.e. it computes successor states and preserves minimal costs. Theorems 1 and 2 follow from Lemmas 1 and 2 and the definitions of T, $\exists^{\mathrm{LC}}_{\mathcal{V}}$ and $[\mathcal{V}' \leftrightarrow \mathcal{V}]$.

Theorem 1. *Let t be an arbitrary state over $\mathcal{V}$. Then $t \in \mathrm{image}(\mathcal{E}, T_a)$ iff there exists a state $s \in \mathcal{E}$ such that a is applicable in s and $t = s[a]$.* □

Theorem 2. *Let $\hat{\mathcal{E}} = \mathrm{image}(\mathcal{E}, T_a)$. Then $\hat{\mathcal{E}}(t) = \min_s(\mathcal{E}(s) + c_a(s))$ for all states $t \in \hat{\mathcal{E}}$.* □

Given an EVMDD $\hat{\mathcal{E}}$ representing states $\hat{S}$, and given transition relation T_a, the preimage operator computes the set of predecessors S, with $s[a] \in \hat{S}$ for $s \in S$. However, the costs associated with S are slightly different than one might expect: $\mathrm{preimage}(\hat{\mathcal{E}}, T_a)(s) = c_a(s) + \hat{\mathcal{E}}(s[a])$, i.e. the cost of $t = s[a]$ plus the cost of applying a in s. This will be required for backward search algorithms, where we start with the set of goal states, associated with a cost of 0. Then the preimage operator computes the minimal cost of reaching a goal state. The following two theorems can be derived similarly to Theorems 1 and 2.

Theorem 3. *Let s be an arbitrary state over $\mathcal{V}$. Then $s \in \mathrm{preimage}(\hat{\mathcal{E}}, T_a)$ iff there exists a state $t \in \hat{\mathcal{E}}$ such that a is applicable in s and $t = s[a]$.* □

Theorem 4. *Let $\mathcal{E} = \mathrm{preimage}(\hat{\mathcal{E}}, T_a)$. For any state $s \in \mathcal{E}$ it holds that $\mathcal{E}(s) = \hat{\mathcal{E}}(s[a]) + c_a(s)$.* □

EVMDD-based A*

A* search (Hart, Nilsson, and Raphael 1968) is a best-first search algorithm used to generate optimal plans. Beginning from the initial state, it expands states according to their f-value, until a goal state is found, where $f(s) = g(s) + h(s)$ for state $s \in \mathcal{S}$. The g-value $g(s)$ corresponds to the cost necessary to reach s, and the heuristic value $h(s)$ estimates the cost to reach the goal from state s. Heuristic h is called *consistent* if (1) $h(s) \leq h(s[a]) + c_a(s)$ for all $s \in \mathcal{S}$ and $a \in A$ where $s[a]$ is defined, and (2) $h(s) = 0$ for all $s \in S_\star$. If h is consistent, then A* is guaranteed to expand states only once and to generate an optimal plan. Typically, already expanded states are stored in the *closed list* (*Closed*), states that are queued for expansion are stored in the *open list* (*Open*).

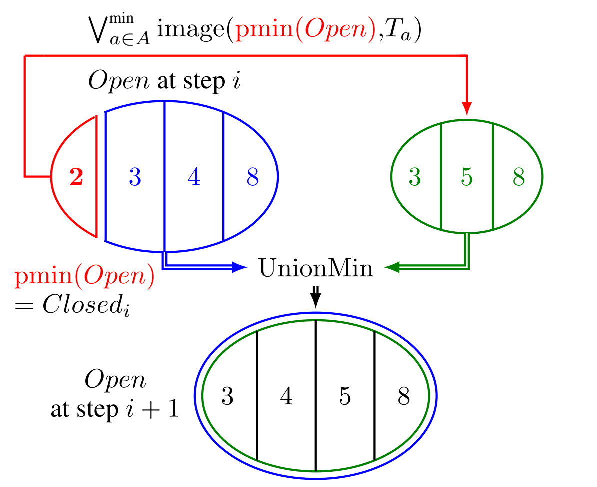


Figure 5: One iteration step of the EVMDD-A* algorithm.

In EVMDD-A* the EVMDD variant of A* we expand not a single state, but the *set of states* with the lowest f-value. For this, *Closed* and *Open* are represented as EVMDDs. If $s \in Open$, then there exists a path from s_0 to s with cost $Open(s)$, otherwise $Open(s) = \infty$. The closed list is divided into multiple EVMDDs, $Closed_i$, where each EVMDD corresponds to a single expansion step i. This is required for plan reconstruction. To understand EVMDD-A* without plan reconstruction, one may think of a single closed list, containing all states which have already been explored together with their g-values. In general, if h is represented as an EVMDD $\mathcal{E}_h$ (cf. Conclusion), computing $f(s)$ is a simple addition of two EVMDDs.

EVMDD-A*. Before we explain the algorithm in detail, we give a high-level description of a single iteration, depicted in Fig. 5. The top left part depicts the open list at step i. It contains states with cost 2, 3, 4 and 8, i.e. the EVMDD *Open* maps the corresponding states to their cost values, and all other states to ∞. In each iteration, we extract the set of states with the current cheapest cost, here 2, from the open list by applying the preserve-min and the image operator. The resulting set of states is depicted to the right. Finally, we merge the remaining states of the open list (i.e. states that were not yet expanded) with the newly generated set, resulting in the open list at step $i + 1$. The picture omits that we do not add already expanded states to the open list (these are included in the closed list). The algorithm iterates until a goal is reached or the complete state space is explored.

In detail, EVMDD-A* (Alg. 1) works as follows: first, the open list is initialized with the EVMDD representing the initial state (line 2). The algorithm continues as long as any state in the open list has finite cost with respect to its f-value (line 4). In line 5, the EVMDD with minimal f-values

Algorithm 1: Pseudocode for EVMDD-A*.

```
Function EVMDD-$A^\star(\Pi)$
    $Open \leftarrow \mathcal{E}_{\{s_0\}}$
    $i \leftarrow 0$
    while $(Open + \mathcal{E}_h) \neq \infty$ do
        $\mathcal{E} \leftarrow \text{pmin}(Open + \mathcal{E}_h) - \mathcal{E}_h$
        $Closed_i \leftarrow \mathcal{E}$
        if $\mathcal{E} \overset{\max}{\wedge} \mathcal{E}_{S_\star} \neq \infty$ then
            return $\text{ConstPlan}(\Pi, \mathcal{E} \overset{\max}{\wedge} \mathcal{E}_{S_\star}, Closed_{0:i})$
        $Open \leftarrow Open \overset{\min}{\vee} \bigvee^{\min}_{a \in A} \text{image}(\mathcal{E}, T_a)$
        $Open \leftarrow Open \overset{\max}{\wedge} \neg(\bigvee^{\min}_{j=0:i} Closed_j)$
        $i \leftarrow i + 1$
    return "no plan"
```

is computed; $\mathcal{E}_h$ is subtracted again, such that $\mathcal{E}$ only represents the g-values of all states with minimal f-value.[4] These states are stored in the closed list for iteration i (line 6), and are expanded next. If any state of $\mathcal{E}$ is a goal state, the plan is reconstructed and returned (line 7/8) with Alg. 2. Otherwise, the states represented with $\mathcal{E}$ are expanded, by calculating the image for every action $a \in A$. The newly explored states are merged with the open list; if a state was already represented by $Open$, only the minimal cost is considered (line 9). Finally, already explored states (contained in $Closed$) are removed from $Open$ by setting their cost to ∞ (line 10).

Plan reconstruction. For explicit A* search, constructing the plan after a goal was found is easy, since each state keeps track of its predecessor state. For symbolic planning, regardless of the data structure, we have to do more work. Function ConstPlan (Alg. 2) is a backward greedy search with perfect heuristic $h^\star = g^\star$, which is obtained by $Closed$. Its runtime is negligible with respect to the actual search. Its input is, besides Π, the EVMDD $\mathcal{E}_\star$ representing all goal states found in iteration i (with the same cost) and all closed lists up to iteration i. First, an arbitrary found goal state is chosen and the empty plan is initialized (lines 2&3). In each step, the algorithm iterates over all actions, until the initial state is reached. For each action a, the EVMDD representing the predecessor states of s_t (line 6) is computed. In line 7, these predecessor states are associated with their g-value (represented by $Closed$). Afterwards, the cost of a is added to the g-value and only the cheapest states are preserved in $\mathcal{E}_{min}$ (line 8). The condition in line 9 checks that the cost of these remaining states corresponds to the g-value of s_t. If this is not the case, a was not part of the optimal plan and the next action has to be checked. Otherwise, a predecessor state is assigned to s_t (line 11), i is set to the (unique) iteration step at which s_t was expanded (line 12), and a is prepended to the current plan π (line 13).

In the following, we show that, given a consistent heuristic, EVMDD-A* is sound, complete, and optimal.

[4]We define $x - \infty = x$ for $x \in \mathbb{N} \cup \{\infty\}$.

Algorithm 2: Pseudocode for Plan Reconstruction.

```
Function $\text{ConstPlan}(\Pi, \mathcal{E}_\star, Closed_{0:i})$
    $s_t \leftarrow \text{selectAnyState}(\mathcal{E}_\star)$
    $\pi \leftarrow [\,]$
    while $s_t \neq s_0$ do
        foreach $a \in A$ do
            $\mathcal{E} \leftarrow \mathcal{E}_{\{s \mid s[a]=s_t,\text{ where } s[a] \text{ is defined}\}}$
            $\mathcal{E} \leftarrow \mathcal{E} \overset{\max}{\wedge} (\bigvee^{\min}_{j=0:i-1} Closed_j)$
            $\mathcal{E}_{min} \leftarrow \text{pmin}(\mathcal{E} + \mathcal{E}_{c_a})$
            if $\min(\mathcal{E}_{min}) \neq Closed_i(s_t)$ then
                continue
            $s_t \leftarrow \text{selectAnyState}(\mathcal{E}_{min})$
            $i \leftarrow$ the unique $i$ with $Closed_i(s_t) \neq \infty$
            $\pi \leftarrow a\ ;\ \pi$
            break
    return $\pi$
```

Theorem 5. *Given a consistent heuristic h, EVMDD-A* is sound and complete.*

Proof sketch. Follows from Theorem 1 and consistency of h. States reachable from s_0 are expanded until a goal state s is found or all states have been expanded. Plan reconstruction only considers states on the closed list leading to s. □

Theorem 6. *Given a consistent heuristic h, EVMDD-A* is optimal.*

Proof sketch. A state s is expanded iff there exists no state with smaller f-value in the open list. By Theorem 2 and consistency of h, the cost of s at expansion is the cheapest cost with which state s is reachable from s_0. Therefore, once a goal state is expanded, its cost is the cheapest cost reaching the goal from s_0. □

Bidirectional search. Most successful symbolic planners nowadays use symbolic bidirectional search (Torralba et al. 2014a), which combines forward and backward search. We give a brief summary how bidirectional search can be realized with EVMDDs. Backward search is implemented by the preimage operator (Def. 8). For that purpose, EVMDD-A* replaces the initial state with the goal formula $s_\star$. Plan reconstruction has to be modified accordingly, in particular, c_a has to be added to $\mathcal{E}_{\{s_t\}}$, since the cost depends on the state the action is applied to. For bidirectional search, we have separate open and closed lists for forward ($Open_{\text{fwd}}$, $Closed_{\text{fwd}}$) and backward ($Open_{\text{bwd}}$, $Closed_{\text{bwd}}$) search. A search step consists either of a backward or a forward search step (and modifies the respective open and closed lists). If a state of the current search is expanded and was already contained in the closed list of the search in the opposite direction, a goal path is found. Its cost is determined by adding the respective EVMDDs. As such a goal path is not necessary optimal, search has to continue, until it is proven that there is no cheaper goal path. Let $\mathcal{E}_{\text{fwd}}$ be the EVMDD representing the currently expanded states in forward direction.

A collision of the search fronts can be detected by computing $\mathcal{E}_{\text{bd}} = \mathcal{E}_{\text{fwd}} + Closed_{\text{bwd}}$. If there exists a state $s \in \mathcal{E}_{\text{bd}}$, a collision was found and $\mathcal{E}_{\text{bd}}(s)$ corresponds to the cost of the detected goal path. To prove that the currently best goal path with cost c is optimal, it is sufficient to check if $c \leq \min(Open_{\text{fwd}}) + \min(Open_{\text{bwd}})$, as this proves that no cheaper collision (goal path) is possible. Finally, plan reconstruction is executed for both directions and the returned plans are combined.

Empirical Evaluation

This section describes the technical aspects of our new planning system (SYMPLE) in detail and evaluates its performance against state-of-the-art optimal planning systems. We conduct two types of experiments: first, we evaluate its performance regarding optimal planning with constant action costs by analyzing the IPC-2014 benchmark set (Vallati et al. 2015). The second part of the evaluation considers optimal planning with state-dependent action costs. Regardless of the benchmark set, all experiments have the usual time limit of 30 minutes and a 4 GB memory limit.

The SYMPLE Planning System

SYMPLE is based on the Fast Downward Planning System (Helmert 2006).[5] The preprocessing is taken from SYMBA, winner of the IPC-2014 (Torralba et al. 2014a). This includes GAMER's (Kissmann, Edelkamp, and Hoffmann 2014) SAS$^+$ encoding, its h^2 invariant computation and pruning of spurious actions (Alcázar and Torralba 2015); we set a time limit of 300 seconds for this part of the preprocessing. Additionally, we took advantage of GAMER's and SYMBA's variable ordering algorithm (Kissmann and Edelkamp 2011), which plays a crucial role in symbolic planning based on decision diagrams. Regarding the representation of actions, the SYMPLE planning system combines as many actions as possible into a transition relation, until the representation exceeds 100k nodes. This size limitation also holds for invariants, which are joined together and separately represented as EVMDDs (similar to BDDs). We evaluate EVMDD-A* (Alg. 1) with progressive and bidirectional search using the blind heuristic. For the latter, in order to decide if a forward search step or a backward search step appears to be more promising, we compare the runtime of the last forward step to the runtime of the last backward step. Conditional effects are encoded by extending the transition relations (Kissmann, Edelkamp, and Hoffmann 2014). The underlying library for EVMDD operations is an extended version of MEDDLY-0.14 (Babar and Miner 2010). The extension consists of the implementation of the operations described in the Preliminaries section. Finally, we extended MEDDLY to support EVMDDs with infinite costs.

Planning with Constant Action Costs

Table 1 shows the performance of SYMPLE in comparison to Fast Downward with $A^\star_{\text{blind}}$, $A^\star_{\text{lmcut}}$ (Helmert and Domshlak 2009) and SYMBA2 on the IPC-2014 benchmark set.

Domain (#Tasks)	$A^\star_{\text{blind}}$	$A^\star_{\text{lmcut}}$	GAMER	cGAMER	SYMBA2	**SYMPLE (prog. / bd.)**
BARMAN (14)	0	0	3	**6**	**6**	0 / 0
CAVEDIVING (20)	6	3	∗(3)	∗	**7**	6 / **7**
CHILDSNACK (20)	0	0	2	1	**4**	0 / 0
CITYCAR (20)	9	0	**18**	**18**	**18**	8 / 8
FLOORTILE (20)	8	17	13	**20**	**20**	14 / 14
GED (20)	15	15	∗	∗	**19**	13 / 15
HIKING (20)	8	8	14	**15**	**15**	11 / 12
MAINTENANCE (5)	**5**	**5**	∗	∗	**5**	**5** / **5**
OPENSTACKS (20)	2	2	16	19	**20**	15 / 15
PARKING (20)	0	**3**	0	**3**	**3**	0 / 0
TETRIS (17)	7	9	3	**11**	10	3 / 3
TIDYBOT (20)	0	**13**	0	**13**	11	0 / 0
TRANSPORT (20)	4	6	6	8	**9**	4 / 6
VISITALL (20)	3	5	5	**6**	**6**	4 / **6**
TOTAL COV. (256)	67	86	83	120	**153**	83 / 91

Table 1: Coverage of planning systems for the IPC-2014 benchmark set. SYMPLE is evaluated with progression (prog.) and bidirectional (bd.) search. (C)GAMER: results from IPC-2014 (∗ indicates parsing errors).

All planners use the the same translation and preprocessing procedure described above. Although SYMPLE was developed with a focus on domains containing actions with state-dependent costs, its performance is notable. SYMPLE outperforms $A^\star_{\text{blind}}$, and performs similar to $A^\star_{\text{lmcut}}$ and the IPC-2014 results of the symbolic planner GAMER and CGAMER (Torralba et al. 2014b). Note that GAMER and CGAMER had a parsing bug which led to fewer solved instances (CAVEDIVING, GED and MAINTENANCE). The difference between SYMPLE and SYMBA2 can be traced back to multiple reasons. While both planning systems use symbolic state representation, SYMBA2 is clearly much more sophisticated by integrating additional techniques: e.g. abstraction heuristics (Torralba, López, and Borrajo 2016) and e-deletion (Torralba et al. 2017), which have not yet been taken into account in SYMPLE. In addition, SYMBA2 is based on the CUDD (Somenzi 2017) library to perform BDD operations, while EVMDD operations in SYMPLE are based on MEDDLY. To the authors' best knowledge, MEDDLY is currently the only decision diagram library supporting EVMDDs. Unfortunately, there is no comparison of MEDDLY with other decision diagram libraries due to lack of functionality (Dijk et al. 2015). We performed some small-scale experiments using BDDs, which indicate that CUDD is superior to MEDDLY in speed and memory regarding basic operations (union and intersection), roughly by a factor of two. The main advantage of SYMBA2 over SYMPLE is the image operation which is the bottleneck in symbolic planning (Torralba 2015). While SYMPLE uses a basic implementation, SYMBA2 applies the relational product (Burch et al. 1994) to compute the image, which is more efficient. Concluding, using EVMDDs can lead to unnecessary overhead for solving unit cost planning tasks where Boolean representations are sufficient to distinguish between reached and unreached states. Half of the do-

[5]Available online: `https://gkigit.informatik.uni-freiburg.de/dspeck/symple`

Domain (#Tasks)	$A^{\star}_{\text{blind}}$	SYMBA2 (exp / cost)	**SYMPLE (prog. / bd.)**
ASTERIX (30)	9	26 / **30**	**30** / **30**
C-GRIPPER (30)	7	3 / 10	9 / **11**
GR-PEG-08 (30)	**29**	0 / 25	27 / 27
GR-PEG-11 (20)	**19**	0 / 15	17 / 18
SDAC-OS-08 (30)	6	**19** / 9	15 / 15
SDAC-OS-11 (20)	3	**20** / 10	**20** / **20**
SDAC-OS-14 (20)	0	**17** / 0	7 / 7
TSP-MH (30)	**19**	1 / 16	15 / 15
TOTAL COV. (21)	92	86 / 115	140 / **143**

Table 2: Comparison of different planner systems using benchmarks containing state-dependent action costs. SYMPLE is evaluated with prog. and bd. search. For SYMBA2, we compare two compilations to tasks with constant action cost: exponential (exp) and cost-based (cost).

mains (7 out of 14) included in the IPC-2014 benchmark are unit cost tasks.

Planning with State-Dependent Action Costs

Table 2 summarizes our experimental results for domains containing actions with state-dependent costs. We analyzed 210 tasks of eight different domains. Due to lack of domains with state-dependent action costs we created a new benchmark set[6], which contains six modified domains from former IPCs and two novel domains. It follows a short description of each domain.

In ASTERIX, the task is to collect an Edelweiss, located on top of a mountain. The cost to climb the mountain depends on its slope. Additionally, it can be necessary to "knock out" some Romans. COLORED GRIPPER is based on the GRIPPER domain. In this version, balls and rooms are either red or blue. Initially, all balls are located in the blue room and each *move* action is penalized by the number of balls located in a room with different color. GREEDY-PEGSOL is a modified version of the PEGSOL domain. The only difference is that the cost of action *end-move* is state-dependent: its cost is equal the number of remaining pegs on the board. SDAC-OPENSTACKS is based on the IPC domain OPENSTACKS. Unlike its constant cost version, the plan cost depends on the number of open stacks in every time step. TSP-MH is a version of the traveling salesman problem with the Manhattan distance metric. Every task consists of a 256×256 grid with an increasing number of randomly placed cities. There is a *visit-city* action for every city. Its cost depends on the current location of the salesman.

Unfortunately, there are not many planning systems supporting actions with state-dependent costs – especially in optimal planning. EVMDD-$A^{\star}$ is a novel approach to natively support state-dependent action costs in optimal planning. To evaluate our approach for state-dependent action costs, we compare SYMPLE, Fast Downward with $A^{\star}_{\text{blind}}$, and the IPC-2014 winner SYMBA2. For $A^{\star}_{\text{blind}}$, we represent action costs as EVMDDs with MEDDLY. For SYMBA2, we provide two compilations to tasks with constant action costs: the exponential compilation generates an action for every assignment of variables contained in the corresponding cost function, with cost according to evaluation of the cost function under this assignment. The cost-based compilation is based on the representation of the cost functions as EVMDDs. It generates sub-actions, corresponding to the edges in the EVMDD, and subsequent applications of sub-actions reflect the evaluation of the cost function (Geißer, Keller, and Mattmüller 2015). Note that the first compilation is exponential in the number of cost function variables, and therefore often not feasible. For example, in GREEDY-PEGSOL the cost function of action *end-move* depends on all pegs, resulting in 2^{32} possible assignments and 2^{32} newly generated actions (obviously not feasible). Again, all planning systems apply the same preprocessing; however, since compilation results in an increased number of variables and actions, SYMBA2 requires more preprocessing time.

Table 2 shows that SYMPLE outperforms the other approaches. Interestingly, on domains such as GREEDY-PEGSOL and TSP-MH, the explicit search approach $A^{\star}_{\text{blind}}$ performs best, whereas in combinatorially more challenging problems like ASTERIX or SDAC-OPENSTACKS, symbolic approaches work best. SYMBA only outperforms SYMPLE in the OPENSTACKS domains. The reason for this is that all cost functions in these domains only depend on a single variable, therefore the exponential encoding only has a small blow-up and is compensated by SYMBA's better performance. The good results of $A^{\star}_{\text{blind}}$ on the PEGSOL domains is due to the greedy character of optimal plans (greedily removing as many pegs in order to generate long moves). Overall, the results show that SYMPLE is superior regarding state-dependent action costs and indicate that EVMDD-$A^{\star}$ is a promising step to generalize planning and natively support actions with unit, constant and state-dependent costs.

Conclusion

In this work, we applied Edge-valued Multi-valued Decision Diagrams to symbolic planning, which is a novel approach. We defined EVMDD-based representations of state sets, transition relations and showed how to compute the necessary operations required for EVMDD-$A^{\star}$, a sound, complete and optimal symbolic search algorithm. While the empirical evaluation showed that BDD-$A^{\star}$ is superior on many tasks with unit-costs, EVMDD-$A^{\star}$ outperforms other approaches in domains with state-dependent costs.

For future work, we plan to represent heuristics, such as *Merge-and-Shrink* abstraction (Helmert et al. 2014), as EVMDDs, similar to how it is done for ADDs (Torralba, López, and Borrajo 2013). Contrary to explicit-state search, applying ADD-based heuristic representations to symbolic planning does not always pay off. We want to investigate if the structural advantage of EVMDDs carries over to the representation of heuristic functions, and if it is thus possible to reinforce heuristic symbolic search. Furthermore, we will revisit the implementation of the image operation for EVMDDs by drawing on results for efficient implementation of the relational product for BDDs (Burch et al. 1994).

[6]Available online: `https://gkigit.informatik.uni-freiburg.de/dspeck/SDAC-Benchmarks`

Acknowledgments

We sincerely thank the anonymous reviewers for their insightful and detailed comments.

References

Alcázar, V., and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In *Proc. ICAPS*, 2–6.

Babar, J., and Miner, A. 2010. MEDDLY: Multi-terminal and Edge-Valued Decision Diagram Library. In *Proc. ICQES*, 195–196.

Bahar, R. I.; Frohm, E. A.; Gaona, C. M.; Hachtel, G. D.; Macii, E.; Pardo, A.; and Somenzi, F. 1997. Algebraic Decision Diagrams and Their Applications. *Formal Methods in System Design* 10(2–3):171–206.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers* 35(8):677–691.

Burch, J. R.; Clarke, E. M.; Long, D. E.; McMillan, K. L.; and Dill, D. L. 1994. Symbolic Model Checking for Sequential Circuit Verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13(4):401–424.

Ciardo, G., and Siminiceanu, R. 2002. Using Edge-Valued Decision Diagrams for Symbolic Generation of Shortest Paths. In *Proc. FMCAD*, 256–273.

Dijk, T. v.; Hahn, E. M.; Jansen, D. N.; Li, Y.; Neele, T.; Stoelinga, M.; Turrini, A.; and Zhang, L. 2015. A Comparative Study of BDD Packages for Probabilistic Symbolic Model Checking. In *Proc. SETTA*, 35–51.

Edelkamp, S., and Helmert, M. 2001. MIPS: The Model-Checking Integrated Planning System. *AI Magazine* 22(3):67–72.

Edelkamp, S., and Kissmann, P. 2009. Optimal Symbolic Planning with Action Costs and Preferences. In *Proc. IJCAI*, 1690–1695.

Edelkamp, S., and Reffel, F. 1998. OBDDs in Heuristic Search. In *Proc. KI*, 81–92.

Geißer, F.; Keller, T.; and Mattmüller, R. 2015. Delete Relaxations for Planning with State-Dependent Action Costs. In *Proc. IJCAI*, 1573–1579.

Geißer, F.; Keller, T.; and Mattmüller, R. 2016. Abstractions for Planning with State-Dependent Action Costs. In *Proc. ICAPS*, 140–148.

Hansen, E.; Zhou, R.; and Feng, Z. 2002. Symbolic Heuristic Search Using Decision Diagrams. In *Proc. SARA*, 83–98.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics* 4(2):100–107.

Helmert, M., and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: Whats the Difference Anyway? In *Proc. ICAPS*, 162–169.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the ACM* 61:16:1–16:63.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26:191–246.

Kissmann, P., and Edelkamp, S. 2011. Improving Cost-Optimal Domain-Independent Symbolic Planning. In *Proc. AAAI*.

Kissmann, P.; Edelkamp, S.; and Hoffmann, J. 2014. Gamer and Dynamic-Gamer: Symbolic Search at IPC 2014. In *Proc. IPC 2014*, 77–84.

Lai, Y.; Pedram, M.; and Vrudhula, S. B. K. 1996. Formal Verification Using Edge-Valued Binary Decision Diagrams. *IEEE Transactions on Computers* 45(2):247–255.

Mattmüller, R.; Geißer, F.; Wright, B.; and Nebel, B. 2018. On the Relationship Between State-Dependent Action Costs and Conditional Effects in Planning. In *Proc. AAAI*.

McMillan, K. L. 1993. *Symbolic Model Checking*. Kluwer Academic Publishers.

Roux, P., and Siminiceanu, R. I. 2010. Model-Checking with Edge-Valued Decision Diagrams. In *Proc. NFM*.

Somenzi, F. 2017. CUDD: CU decision diagram package - release 2.5.0.

Speck, D.; Geißer, F.; and Mattmüller, R. 2018. Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams - Detailed Proofs. In *Technical Report: University of Freiburg, Faculty of Engineering*, number 284.

Torralba, Á.; Alcázar, V.; Borrajo, D.; Kissmann, P.; and Edelkamp, S. 2014a. SymBA*: A Symbolic Bidirectional A* Planner. In *Proc. IPC 2014*, 105–109.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2014b. cGamer: Constrained Gamer. In *Proc. IPC 2014*, 84–86.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient symbolic search for cost-optimal planning. *Artificial Intelligence* 242:52–79.

Torralba, Á.; López, C. L.; and Borrajo, D. 2013. Symbolic Merge-and-Shrink for Cost-Optimal Planning. In *Proc. IJCAI*, 2394–2400.

Torralba, Á.; López, C. L.; and Borrajo, D. 2016. Abstraction Heuristics for Symbolic Bidirectional Search. In *Proc. IJCAI*, 3272–3278.

Torralba, Á. 2015. *Symbolic Search and Abstraction Heuristics for Cost-Optimal Planning*. Ph.D. Dissertation, Universidad Carlos III de Madrid.

Vallati, M.; Chrpa, L.; Grześ, M.; McCluskey, T. L.; Roberts, M.; and Sanner, S. 2015. The 2014 International Planning Competition: Progress and Trends. *AI Magazine* 36(3):90–98.

Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams - Detailed Proofs

David Speck and **Florian Geißer** and **Robert Mattmüller**
University of Freiburg, Germany
{speckd, geisserf, mattmuel}@informatik.uni-freiburg.de

Abstract

This report contains the proof of correctness, soundness and optimality for EVMDD-A* presented in the paper *Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams* (Speck, Geißer, and Mattmüller 2018).

1 Transition Relation

Lemma 1. *Let (s,t') be an arbitrary state over $\mathcal{V} \cup \mathcal{V}'$. For any action a it holds that $(s,t') \in T_a$ iff a is applicable in s and $t = s[a]$.*

Proof. Let T'_a be the intermediate EVMDD of Terms (3) to (5). By construction of T'_a: a state $(s,t') \in T'_a$ iff a is applicable in s and $t = s[a]$. Furthermore, it holds that $(s,t') \in \mathcal{E}_{c_a}$ for all $(s,t') \in \mathcal{V} \cup \mathcal{V}'$ (Def. 1). Thus, $(s,t') \in T_a$ iff $(s,t') \in (T'_a \overset{\max}{\wedge} \mathcal{E}_{c_a})$ iff $(s,t') \in T'_a$ iff a is applicable in s and $t = s[a]$. □

Lemma 2. *Let $(s,t') \in T_a$. Then $T_a(s,t') = c_a(s)$.*

Proof. The intermediate EVMDD T'_a of Terms (3) to (5) contains only states with 0 or infinite cost (Def. 4 & Def. 5). Since $(s,t') \in T_a$, it holds that $T'_a(s,t') = 0$. Then, $T_a(s,t') = (T'_a \overset{\max}{\wedge} \mathcal{E}_{c_a})(s,t') = \max(T'_a(s,t'), c_a(s,t')) = \max(0, c_a(s,t')) = c_a(s,t') = c_a(s)$. □

2 Image

Note that we sometimes use "min" instead of $\overset{\min}{\vee}$. This simplifies the notations. If "min" is used for partial functions, we mean $\overset{\min}{\vee}$.

Theorem 1. *Let t be an arbitrary state over $\mathcal{V}$. Then $t \in \text{image}(\mathcal{E}, T_a)$ iff there exists a state $s \in \mathcal{E}$ such that a is applicable in s and $t = s[a]$.*

Proof.

$$\begin{aligned}
& t \in \text{image}(\mathcal{E}, T_a) && \\
& \Leftrightarrow t \in (\exists^{\text{LC}}_{\mathcal{V}}(\mathcal{E} + T_a))[\mathcal{V}' \leftrightarrow \mathcal{V}] && \text{(Definition 7)} \\
& \Leftrightarrow t' \in \exists^{\text{LC}}_{\mathcal{V}}(\mathcal{E} + T_a) && \text{(Substitution Lemma)} \\
& \Leftrightarrow t' \in \exists^{\text{LC}}_{v_1,\ldots,v_n}(\mathcal{E} + T_a) && \text{(Definition } \exists^{\text{LC}}) \\
& \Leftrightarrow \exists s : (s, t') \in (\mathcal{E} + T_a) && \text{(Transformation)} \\
& \Leftrightarrow \exists s : (s, t') \in \mathcal{E} \text{ and } (s, t') \in T_a && \text{(Definition 4)} \\
& \Leftrightarrow \exists s : s \in \mathcal{E} \text{ and } (s, t') \in T_a && \text{(Transformation)} \\
& \Leftrightarrow \exists s : s \in \mathcal{E} \text{ and } a \text{ is applicable in } s \text{ and } t = s[a] && \text{(Lemma 1)} \\
& \Leftrightarrow \text{there exists a state } s \in \mathcal{E} \text{ s.t. } a \text{ is applicable in } s && \text{(Transformation)} \\
& \qquad \text{and } t = s[a] && \square
\end{aligned}$$

From Theorem 1, Lemma 1 and Lemma 2 follows Corollary 1 which will be used to prove Theorem 2.

Corollary 1. *Let t be an arbitrary state over $\mathcal{V}$ with $t \in \text{image}(\mathcal{E}, T_a)$. Then there exists a state $s \in \mathcal{E}$ such that $(s, t') \in T_a$.*

Proof. By definition $t \in \text{image}(\mathcal{E}, T_a)$. Thus, by Theorem 1 there is a state $s \in \mathcal{E}$ such that a is applicable in s and $t = s[a]$. It follows that there exists a state $s \in \mathcal{E}$ such that $(s, t') \in T_a$ (Lemma 1). $\square$

Theorem 2. *Let $\hat{\mathcal{E}} = \text{image}(\mathcal{E}, T_a)$. Then $\hat{\mathcal{E}}(t) = \min_s(\mathcal{E}(s) + c_a(s))$ for all states $t \in \hat{\mathcal{E}}$.*

Proof.

$$\begin{aligned}
\hat{\mathcal{E}}(t) &= (\text{image}(\mathcal{E}, T_a))(t) && \\
&= ((\exists^{\text{LC}}_{\mathcal{V}}(\mathcal{E} + T_a))[\mathcal{V}' \leftrightarrow \mathcal{V}])(t) && \text{(Definition 7)} \\
&= (\exists^{\text{LC}}_{\mathcal{V}}(\mathcal{E} + T_a))(t') && \text{(Substitution Lemma)} \\
&= (\exists^{\text{LC}}_{v_1,\ldots,v_n}(\mathcal{E} + T_a))(t') && \text{(Definition } \exists^{\text{LC}}) \\
&= (\min_{v_1,\ldots,v_n}(\mathcal{E} + T_a))(t') && \text{(Definition } \exists^{\text{LC}}) \\
&= (\min_s(\mathcal{E} + T_a))(t') && \text{(Transformation)} \\
&= (\min_s(\mathcal{E}(s, *) + T_a(s, *)))(t') && \text{(Transformation)} \\
&= \min_s(\mathcal{E}(s, t') + T_a(s, t')) && \text{(Transformation)} \\
&= \min_s(\mathcal{E}(s) + T_a(s, t')) && \text{(Transformation)} \\
&= \min_s(\mathcal{E}(s) + c_a(s)) && \text{(Corollary 1 + Lemma 2)}
\end{aligned}$$

$\square$

3 Preimage

Theorem 3. *Let s be an arbitrary state over $\mathcal{V}$. Then $s \in \text{preimage}(\hat{\mathcal{E}}, T_a)$ iff there exists a state $t \in \hat{\mathcal{E}}$ such that a is applicable in s and $t = s[a]$.*

Proof.

$$\begin{aligned}
& s \in \text{preimage}(\hat{\mathcal{E}}, T_a) && \\
\Leftrightarrow\ & s \in \exists^{\text{LC}}_{\mathcal{V}'}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T_a) && \text{(Definition 7)}\\
\Leftrightarrow\ & s \in \exists^{\text{LC}}_{v'_1,\ldots,v'_n}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T_a) && \text{(Definition } \exists^{\text{LC}})\\
\Leftrightarrow\ & \exists t : (s, t') \in (\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T_a) && \text{(Transformation)}\\
\Leftrightarrow\ & \exists t : (s, t') \in \hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] \text{ and } (s, t') \in T_a && \text{(Transformation)}\\
\Leftrightarrow\ & \exists t : (t, s') \in \hat{\mathcal{E}} \text{ and } (s, t') \in T_a && \text{(Substitution Lemma)}\\
\Leftrightarrow\ & \exists t : t \in \hat{\mathcal{E}} \text{ and } (s, t') \in T_a && \text{(Transformation)}\\
\Leftrightarrow\ & \exists t : t \in \hat{\mathcal{E}} \text{ and } a \text{ is applicable in } s \text{ and } t = s[a] && \text{(Lemma 1)}\\
\Leftrightarrow\ & \text{there exists a state } t \in \hat{\mathcal{E}} \text{ s.t. } a \text{ is applicable in } s && \text{(Transformation)}\\
& \quad \text{and } t = s[a] && \square
\end{aligned}$$

From Theorem 3, Lemma 1 and Lemma 2 follows Corollary 2 which will be used to prove Theorem 4.

Corollary 2. *Let s be an arbitrary state over $\mathcal{V}$ with $s \in \text{preimage}(\hat{\mathcal{E}}, T_a)$. Then there exists a state $t \in \hat{\mathcal{E}}$ such that $(s, t') \in T_a$.*

Proof. By definition $s \in \text{preimage}(\hat{\mathcal{E}}, T_a)$. Thus, by Theorem 3 there is a state $t \in \hat{\mathcal{E}}$ such that a is applicable in s and $t = s[a]$. It follows that there exists a state $t \in \hat{\mathcal{E}}$ such that $(s, t') \in T_a$ (Lemma 1). $\square$

Theorem 4. *Let $\mathcal{E} = \text{preimage}(\hat{\mathcal{E}}, T_a)$. For any state $s \in \mathcal{E}$ it holds that $\mathcal{E}(s) = \hat{\mathcal{E}}(s[a]) + c_a(s)$.*

Proof.

$$\begin{aligned}
\mathcal{E}(s) &= (\text{preimage}(\hat{\mathcal{E}}, T_a))(s) & \\
&= (\exists^{\text{LC}}_{\mathcal{V}'}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T_a))(s) & \text{(Definition 7)} \\
&= (\exists^{\text{LC}}_{v'_1,\dots,v'_n}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T_a))(s) & \text{(Definition } \exists^{\text{LC}}) \\
&= (\min_{v'_1,\dots,v'_n}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T_a))(s) & \text{(Definition } \exists^{\text{LC}}) \\
&= (\min_{t'}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'] + T_a))(s) & \text{(Transformation)} \\
&= (\min_{t}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'](*, t') + T_a(*, t')))(s) & \text{(Transformation)} \\
&= \min_{t}(\hat{\mathcal{E}}[\mathcal{V} \leftrightarrow \mathcal{V}'](s, t') + T_a(s, t')) & \text{(Transformation)} \\
&= \min_{t}(\hat{\mathcal{E}}(t, s') + T_a(s, t')) & \text{(Substitution Lemma)} \\
&= \min_{t}(\hat{\mathcal{E}}(t) + T_a(s, t')) & \text{(Transformation)} \\
&= \min_{s[a]}(\hat{\mathcal{E}}(s[a]) + c_a(s)) & \text{(Cor. 2 + Lem. 2 + Thm. 3)} \\
&= \hat{\mathcal{E}}(s[a]) + c_a(s) & \text{(Definition 1)}
\end{aligned}$$

□

4 EVMDD-A*

Lemma 3. *Let Π be a planning task and h be a consistent heuristic. EVMDD-A* expands states in the same order and with the same g-values as A* with FIFO tie-breaking rule.*

Proof. Let S_f be all states with minimum f-value of an open list *Open*. Recall that in A* the tie-breaking between different states with minimum f-value in *Open* can be arbitrary. Let's assume the tie-breaking rule is "first in first out (FIFO)". The difference between EVMDD-A* and A* is that EVMDD-A* expands all states of S_f at once while A* iteratively ($|S_f|$ iterations) extracts these states. It is not possible that any other state is expanded before the $|S_f|$ iterations are finished, because h is consistent and therefore all newly generated successors have at least the f-value of all states in S_f.

- **Goal check.** Any ordering of expanding states in S_f is possible in A*. Thus, it is equivalent to first check if any state in S_f is a goal state.
- **Closed list.** Any ordering of expanding states in S_f is possible in A*. Thus, it is equivalent to first add all states S_f to the closed list and then expand all states S_f.
- **Open list.** By Theorem 1, in EVMDD-A*, all successors of S_f are generated and added to the open list if they are not contained in the closed list.

This is equivalent to adding them iteratively to *Open*. By Theorem 2 the cost of a successor $\hat{s}$ is the minimum cost with which $\hat{s}$ is reachable from any state in S_f applying action a. In line 9 (Algorithm 1), the minimum cost is taken from the current cost of $\hat{s}$ in *Open* or the minimum cost with which $\hat{s}$ is reachable from S_f applying any actions $a \in A$. Thus, the cost of a state $\hat{s}$ in *Open* is only updated iff it is reachable with lower cost from any expanded state in S_f. Again, this is equivalent to A* after $|S_f|$ iterations.

Therefore, EVMDD-A* and A* expand nodes in the same order and with the same g-values. □

Lemma 4. *Let* Π *be a planning task and* h *be a consistent heuristic. EVMDD-A** *returns "no plan" iff A** *returns "no plan".*

Proof. In EVMDD-A*, "no plan" is returned iff the open list is empty. By Lemma 3, the open list in EVMDD-A* is found empty iff the open list in A* is found empty. □

Lemma 5. *Let* Π *be a planning task and* h *be a consistent heuristic. If a plan exists for* Π*, EVMDD-A** *returns the same plan as A** *with FIFO tie-breaking rule.*

Proof. EVMDD-A* expands states in the same order and with the same g-values as A* (Lemma 3). Heuristic h is consistent, therefore all states in the closed list have minimum g-values g^*, i.e. the minimum cost with which they can be reached from s_0. ConstPlan is a version of backward greedy search with perfect heuristic $h^* = g^*$ where the g^*-values are stored in the closed list. Thus, ConstPlan and therefore EVMDD-A* returns an optimal plan from s_0 to any goal state expanded in EVMDD-A*. EVMDD-A* expands the same goal state as A* (Lemma 3). Thus, EVMDD-A* returns a plan iff A* returns a plan and EVMDD-A* returns the same plan as A* (if a plan exists). □

Theorem 5 & 6. *EVMDD-A** *is complete, sound and optimal for consistent heuristics.*

Proof. Let Π be a planning task and h be a consistent heuristic. EVMDD-A* returns "no plan" iff A* returns "no plan" (Lemma 4). If a plan exists for Π, EVMDD-A* returns the same plan as A* (Lemma 5). EVMDD-A* is complete, sound and optimal for consistent heuristics because A* is it too. □

References

Speck, D.; Geißer, F.; and Mattmüller, R. 2018. Symbolic Planning with Edge-Valued Multi-Valued Decision Diagrams. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS).* Accepted.