Relaxed Decision Diagrams for Cost-Optimal Classical Planning

Margarita P. Castro[†], **Chiara Piacentini**[†], **Andre A. Cire**[‡], and **J. Christopher Beck**[†]
[†]Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada, ON M5S 3G8
[‡]Department of Management, University of Toronto Scarborough, Toronto, Canada, ON M1C 1A4

Abstract

We explore the use of multivalued decision diagrams (MDDs) to represent a relaxation of the state-transition graph for classical planning problems. The relaxation exploits the exact state transitions up to a pre-defined memory limit and uses value-accumulating semantics when the limit is reached. Moreover, it provides admissible heuristic values by means of an efficient shortest-path algorithm, which is applied in an A^* algorithm to find cost-optimal plans. We also consider a variant of A^* that takes advantage of feasible solutions extracted by the MDD to reduce the number of states that need to be evaluated. Our experimental evaluation shows that the MDD-based heuristic, despite being computationally more expensive, can be more informative than some state-of-the-art admissible heuristics.

1 Introduction

We present a new admissible heuristic based on a *relaxed multivalued decision diagram* (MDD). A relaxed MDD is a graph of restricted size that over-approximates the set of feasible solutions to a discrete problem. Relaxed MDDs have been largely applied to mathematical programming and discrete optimization, in particular for obtaining optimization bounds for combinatorial and scheduling problems (Hoda, Van Hoeve, and Hooker 2010; Bergman et al. 2016; Kinable, Cire, and van Hoeve 2017).

This paper defines relaxed MDDs for a classical planning task and uses them to compute a novel admissible heuristic to reach a goal node. We explore the relationship between relaxed MDDs and existing techniques to solve classical planning problems, showing that a relaxed MDD is an abstraction of the transition graph for a planning task and that our heuristic dominates the well-known h^{max} heuristic (Bonet and Geffner 2000).

The MDD-based heuristic is used in a variant of A^* inspired by a branch-and-bound tree search. We enhance the A^* search algorithm with a bounding mechanism that reduces the number of states expanded via bounds on plan cost derived from feasible plans extracted from the MDD. The new algorithm is therefore suitable for finding both feasible and optimal plans.

The paper is organized as follows. Section 2 defines a classical planning task and presents related work. Section 3 defines a relaxed MDD for classical planning and Section 4 presents the construction procedure. Section 5 relates relaxed MDDs to transition graphs and compares them to other heuristics in classical planning. Section 6 explains the implementation and our preliminary results are presented in Section 7. Lastly, Section 8 discusses the approach and possible directions for future research.

2 Background

This section presents a formal definition of a cost-optimal classical planning, introduces the notation used in this paper, and reviews work in the classical planning literature that is related to our relaxed MDD approach.

2.1 Cost-Optimal Classical Planning

We consider cost-optimal classical planning tasks with non-zero cost actions using the STRIPS formalism. A planning *task* is a tuple $\Pi = \langle \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where $\mathcal{P}$ is the set of *propositional* variables, $\mathcal{A}$ is the set of actions, $\mathcal{I} \subseteq \mathcal{P}$ is the initial state, and $\mathcal{G} \subseteq \mathcal{P}$ is the set of goal conditions. A state s is defined as a subset of propositional variables, $s \subseteq \mathcal{P}$.

An action $a \in \mathcal{A}$ is a tuple $\langle pre(a), add(a), del(a), c(a) \rangle$, where $pre(a) \subseteq \mathcal{P}$ is the set of preconditions, $add(a) \subseteq \mathcal{P}$ is the set of add effects, $del(a) \subseteq \mathcal{P}$ is the set of delete effects, and $c(a) > 0$ is the action cost. An action a is applicable to a state s if the preconditions are satisfied in s, i.e., $pre(a) \subseteq s$. The application of an action a to a state s produces a successor state s' given by $s' = \phi(a, s) = (s \setminus del(a)) \cup add(a)$.

A solution of a planning task Π is a *plan*, i.e., a sequence of actions such that each action is applicable in its predecessor state and the last state satisfies the goal conditions. Formally, $\pi = (a_0, \dots, a_n)$ is a plan if for each action a_i in π, $pre(a_i) \subseteq \phi(a_{i-1}, \phi(a_{i-2}, \dots \phi(a_0, \mathcal{I})))$, and $\mathcal{G} \subseteq \phi(a_n, \phi(a_{n-1}, \dots \phi(a_0, \mathcal{I}))) = \phi(\pi, \mathcal{I})$.

The cost of a plan π is the sum of all the actions appearing in π, i.e., $c(\pi) = \sum_{i=0}^{n} c(a_i)$. A cost-optimal plan $\hat{\pi}$ is a plan with minimum cost, i.e, $c(\hat{\pi}) \leq c(\pi)$ for any plan π of Π.

Given a planning task Π, we define a delete-free planning task Π^+ where delete effects are ignored. Formally, the delete-free task is given by $\Pi^+ = \langle \mathcal{P}, \mathcal{A}^+, \mathcal{I}, \mathcal{G} \rangle$, where for each $a \in \mathcal{A}$ there is an action $a' \in \mathcal{A}^+$ such that $pre(a') = pre(a)$, $add(a') = add(a)$ and $del(a') = \emptyset$. A *delete relaxation* of a planning task Π refers to its associated delete-free task Π^+.

2.2 Related Work in Planning

Our work is closely related to heuristics based on graphical structures, such as Graphplan (Blum and Furst 1997), red-black relaxed plans (Katz, Hoffmann, and Domshlak 2013), and abstractions (Edelkamp 2001; Helmert et al. 2007). We also discuss the use of decision diagrams for symbolic A^* search in classical planning (Torralba, Linares López, and Borrajo 2016) and the differences with our approach.

Graphplan (Blum and Furst 1997) is a compact data structure for encoding planning problems. It is a directed and layered graph with alternating propositional and action layers, in which nodes represent propositions and actions, respectively. Edges connect a proposition to an action node if the proposition is a precondition of the action, and an action to a proposition node if the proposition belongs to the add or delete effects of the action. Graphplan derives the admissible heuristic h^G by taking the index of the first layer where the goal conditions appear without any mutex relation (Bonet and Geffner 2000). A relaxed version of Graphplan, called the Relaxed Planning Graph (RPG), represents the delete relaxation of a planning task. Relaxed plans can be extracted from the RPG in polynomial time and yield the non-admissible heuristic h^{FF} (Hoffmann and Nebel 2001).

While the delete relaxation provides several other heuristics, e.g., h^{max}, h^{add} (Bonet and Geffner 2001) and $h^{LM\text{-}cut}$ (Helmert and Domshlak 2009), ignoring the delete effects can result in a poor heuristic estimation. Red-black planning heuristics overcome some of the problems of delete relaxation heuristics by dividing the propositional variables into two groups: one that follows the semantics of the delete relaxation and one that takes into account the delete effects of actions (Domshlak, Hoffmann, and Katz 2015). Our relaxed MDD heuristic follows a similar idea in the sense that we partially ignore delete effects, though our approach to doing so is by considering nodes as the union of plan states.

Abstraction-based heuristics are also related to our work. An abstraction maps the search space into a smaller one in which an optimal path from an abstract initial state to an abstract goal state is an admissible heuristic. Different abstraction mappings result in different heuristics, for example pattern database heuristics (Edelkamp 2001) and *merge-and-shrink* (Helmert et al. 2007; Sievers, Wehrle, and Helmert 2014). Our relaxed MDD representation of a planning task can be viewed as an abstraction, as detailed in Section 5.1.

Binary decision diagrams (BDDs) have been used in planning to succinctly represent sets of states (symbolic states). Using this representation, a symbolic version of the A^* search algorithm achieves state-of-the-art performance in cost-optimal classical planning (Torralba, Linares López, and Borrajo 2016). Several admissible heuristics have been proposed to guide the search over the symbolic state-space, e.g., abstraction-based heuristics (Edelkamp, Kissmann, and Torralba 2012; Torralba, López, and Borrajo 2013). In contrast, our approach uses relaxed MDDs to compute admissible heuristics on a standard A^* search algorithm.

Lastly, the planning literature has used edge-value multi-valued decision diagrams (EVMDD) to represent cost functions of planning problem with state-dependent actions costs (Keller et al. 2016; Geißer, Keller, and Mattmüller 2016).

3 Relaxed MDDs for Planning

In this section, we demonstrate the use of relaxed MDDs as a graphical structure to approximate the state-space transition graph. We first define an MDD for classical planning and then extend the definition to relaxed MDDs.

Consider τ as an upper bound on the number of actions in a cost-optimal plan. An MDD for a classical planning task Π is a graphical structure that, starting from the initial state $\mathcal{I}$, represents the set of reachable states after applying at most τ actions. Specifically, an MDD $\mathcal{M} = (\mathcal{N}, \mathcal{E})$ is a layered directed acyclic graph where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node u has a label $\sigma(u)$ that represents a reachable state, i.e., $\sigma(u) \subseteq \mathcal{P}$ is the set of propositions in the state. In particular, the set of nodes is divided into layers $\mathcal{N} = \{\mathcal{N}_0, \mathcal{N}_1, ..., \mathcal{N}_\tau\}$, where layer $\mathcal{N}_0 = \{\mathbf{r}\}$ has a single node, called the root node, and $\sigma(\mathbf{r}) = \mathcal{I}$.

Given an edge $e = (u, v) \in \mathcal{E}$, its tail and head nodes are given by $\rho(e) = u$ and $\kappa(e) = v$, respectively. For a given layer $\mathcal{N}_t$ ($0 \leq t < \tau$), all outgoing edges are directed to a node in layer $\mathcal{N}_{t+1}$, i.e., $\rho(e) \in \mathcal{N}_t$ iff $\kappa(e) \in \mathcal{N}_{t+1}$. Each edge $e \in \mathcal{E}$ has a label $\theta(e)$ that indicates its associated action. Given two nodes $u \in \mathcal{N}_t$ and $v \in \mathcal{N}_{t+1}$ there is an edge $e = (u, v)$ connecting them iff the action associated to the edge, $a = \theta(e)$, is applicable in $\sigma(u)$ and node v represents the successor state, i.e., $pre(a) \subseteq \sigma(u)$ and $\phi(a, \sigma(u)) = \sigma(v)$.

Thus, an MDD for a task Π is a layered state-transition graph. A node $u \in \mathcal{N}_t$ ($0 \leq t \leq \tau$) is associated to a state that can be reached after applying t actions from the initial state $\mathcal{I}$. Specifically, any path $(e_0, ..., e_t)$ in $\mathcal{M}$ from $\mathbf{r}$ to a node $u \in \mathcal{N}_t$ represents a plan $\pi = (\theta(e_0), ..., \theta(e_{t-1}))$ that starts at $\mathcal{I}$ and reaches state $\sigma(u)$.

The construction of such an MDD is, however, impractical. First, the number of reachable states in a planning task Π can grow exponentially with the number of variables. Moreover, the number of actions needed for any cost-optimal plan is unknown, i.e., the minimum number of layers that is required for its construction is also not available in advance.

We define instead *relaxed MDDs*, which are constructed by imposing an additional limit on the number of nodes per layer, i.e., its width $w(\mathcal{M}) := \max\{|\mathcal{N}_t| : 0 \leq t \leq \tau\}$ is bounded by a given parameter $\mathcal{W}$. To enforce this bound, each node in a relaxed MDD represents an approximation of the *union* of one or more states as opposed to a single state. The edges emanating from a node represent all possible actions that can be applied to the union of the states. Two examples of MDDs are depicted in Figure 2 and construction details are presented in Section 4.

3.1 A Relaxed MDD-based Heuristic

Consider a relaxed MDD $\mathcal{M}$ and a node $u \in \mathcal{N}$. Let $\delta^{in}(u)$ and $\delta^{out}(u)$ be the set of edges directed to and emanating from node u, respectively. An edge e is in $\delta^{out}(u)$ if $pre(\theta(e)) \subseteq \sigma(u)$. Then, the proposition label of node u is defined as

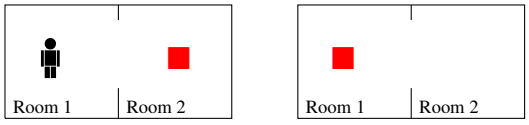
$$\sigma(u) := \bigcup_{e \in \delta^{in}(u)} \phi(\theta(e), \sigma(\rho(e))). \quad (1)$$

Given a planning task Π and a relaxed MDD with width $w(\mathcal{M}) \geq 1$, we can compute the cost to reach each node $u \in \mathcal{N}$ from $\mathbf{r}$, using a shortest path algorithm. Let $\omega^*(u)$ be the minimum cost to reach a node $u \in \mathcal{N}$, with $\omega^*(\mathcal{I}) = 0$. Consider $\mathcal{N}_\mathcal{G} \subseteq \mathcal{N}$ as the set of *goal nodes*, i.e., $u \in \mathcal{N}_\mathcal{G}$ iff $\mathcal{G} \subseteq \sigma(u)$. Then, the relaxed MDD-based heuristic $h^\mathcal{M}$ is given by the minimum cost to reach any goal node:

$$h^\mathcal{M} := \min \{\omega^*(u) : \ u \in \mathcal{N}_\mathcal{G}\}. \tag{2}$$

3.2 Example

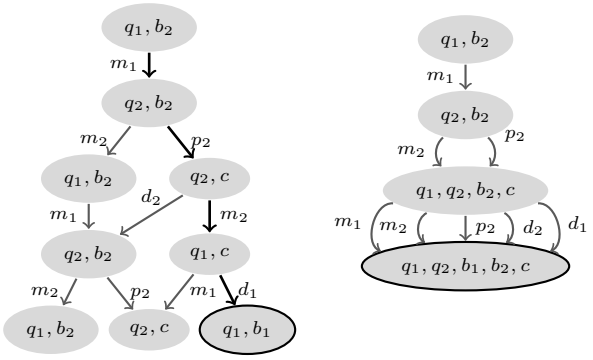
Consider the planning task $\Pi = \langle \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$ depicted in Figure 1. The set of propositions is $\mathcal{P} = \{q_1, q_2, b_1, b_2, c\}$, where q_i indicates if the robot is in room $i \in \{1, 2\}$, b_i if the block is in room i, and c if the robot is carrying the block. The task has six unit cost actions, $\mathcal{A} = \{m_1, m_2, p_1, p_2, d_1, d_2\}$, where m_1 represents moving the robot from room 1 to room 2, m_2 is the opposite move, and for each $i \in \{1, 2\}$, p_i and d_i correspond to picking-up and dropping the block in room i, respectively. The initial state and goal conditions are illustrated in Figures 1a and 1b, respectively.



(a) Initial State, $\mathcal{I} = \{q_1, b_2\}$. (b) Goals, $\mathcal{G} = \{b_1\}$.

Figure 1: Planning domain description.

Figure 2 shows two MDDs for this planning task, with $\tau = 4$. For each MDD, the edge labels correspond to applicable actions and the nodes denote the set of propositions, as described in equation (1). The first MDD (Figure 2a) has one node per reachable state (i.e., it is an *exact* MDD). The node outlined in black represents a goal node and the bold path corresponds to the shortest path with cost $h^\mathcal{M} = 4$.



(a) Exact MDD, $w = 3$. (b) Relaxed MDD, $w = 1$.

Figure 2: MDDs for the example in Section 3.2.

In the second relaxed MDD (Figure 2b), nodes represent the union of one or more states. In this case, the shortest path in the relaxed MDD reaches a goal node after applying 3 actions, i.e., $h^\mathcal{M} = 3$.

As depicted in Figure 2b, width-one relaxed MDDs have a similar structure to relaxed planning graphs (RPG) (Bonet and Geffner 2000). However, the RPG completely ignores delete effects while our relaxed MDD partially considers them. For example, the second node in the relaxed MDD omits proposition q_1, while the RPG would consider it.

4 Relaxed MDD Construction

We present a top-down algorithm to construct a relaxed MDD for a classical planning task Π. Our construction procedure, presented in Algorithm 1, results in a relaxed MDD with width at most $\mathcal{W}$ and with a finite number of layers.

The top-down procedure is as follows. Starting with a single node in the first layer, $\sigma(\mathbf{r}) = \mathcal{I}$, the procedure iteratively constructs one layer at a time by performing three operations. The first operation, UPDATENODES, updates the nodes in a given layer by computing the set of achievable propositions and calculating the cost to reach the node. Moreover, this step updates the heuristic value if a node is a goal node. The second operation, FINDAPPLICABLEACTIONS, finds the set of actions applicable to the nodes and eliminates any action that does not add any information for the heuristic computation. The procedure creates an edge for each action and directs all edges to a single node in the following layer. Lastly, operation SPLITNODES decides how to partition the incoming edges of the new layer to create at most $\mathcal{W}$ nodes.

Algorithm 1 Relaxed MDD construction

```
1: procedure CONSTRUCTMDD(Input: Π, W)
2:     t = 0, h^M = ∞
3:     repeat
4:         UPDATENODES(N_t, h^M)
5:         FINDAPPLICABLEACTIONS(N_t)
6:         t = t + 1
7:         SPLITNODES(N_t, W)
8:     until TERMINATE(N_t)
9:     return h^M
```

In each iteration, the algorithm checks whether or not we should construct a new layer via the TERMINATE procedure. When the construction is completed, Algorithm 1 returns the heuristic value. The following sections explain each of the procedures presented above.

4.1 Updating Nodes

For a given layer $\mathcal{N}_t$, the procedure updates each node $u \in \mathcal{N}_t$ to represent its set of propositions, $\sigma(u)$, and the minimal cost to reach u. The procedure also updates the heuristic value $h^\mathcal{M}$ if we encounter a goal node.

As described in Section 3, each node $u \in \mathcal{N}$ is associated with a label $\sigma(u)$ that corresponds to the set of propositions that are true in at least one state encoded by u. This label is computed by setting $\sigma(\mathbf{r}) = \mathcal{I}$ and applying recursion (1).

Each node $u \in \mathcal{N}_t$ is also associated with a set of labels that represents the minimum cost to reach u. Inspired by the reachability analysis used in h^{max}, we compute the minimum cost to reach each proposition represented in u.

For each $p \in \sigma(u)$, let $\omega(u,p)$ be the cost to reach proposition p in node u. We associate a cost label $\nu(e,p)$ to each incoming edge e that has proposition p in its resulting state, i.e., $p \in \phi(\theta(e), \rho(e))$. Then, $\omega(u,p)$ is calculated by setting $\omega(\mathbf{r},p) = 0$ to all $p \in \mathcal{I}$, and applying the recursion

$$\omega(u,p) := \min\{\nu(e,p) : \ e \in \delta_p^{in}(u)\}, \tag{3}$$

where $\delta_p^{in}(u)$ represents the set of edges in $\delta^{in}(u)$ that have p in their resulting state.

For a given edge e and a proposition $p \in \phi(\theta(e), \rho(e))$, we calculate $\nu(e,p)$ by considering (i) the cost to apply action $\theta(e)$ and (ii) the cost to have p in the resulting state. Let $\nu(e)$ be the minimum cost of applying action $a = \theta(e)$ in node $v = \rho(e)$. We have that $\nu(e)$ is the cost of action a plus the cost of its most expensive precondition on v, i.e.,

$$\nu(e) := c(a) + \max\{\omega(v,q) : q \in pre(a)\}. \tag{4}$$

Then, for each edge $e \in \delta_p^{in}(u)$, we compute $\nu(e,p)$ as the minimum cost to have p. To do so, we identify two cases. If action $a = \theta(e)$ adds proposition p, then the cost to reach p is given by $\nu(e)$. If a does not add p, the cost of p in the tail node $v = \rho(e)$ might be larger than the cost of any precondition of a in v. In that case, we compute $\nu(e,p)$ by considering the cost of the most expensive associated proposition. The edge cost is hence:

$$\nu(e,p) := \begin{cases} \nu(e) & p \in add(a), \\ \max\{\nu(e), c(a) + \omega(v,p)\} & o.w. \end{cases}$$

To summarize, procedure UPDATENODES iterates over all the nodes $u \in \mathcal{N}_t$ and updates labels $\sigma(u)$ and $\omega(u,p)$ for all $p \in \sigma(u)$.

The procedure will also compute a heuristic estimate whenever a node $u \in \mathcal{N}_t$ is a goal node, i.e., $\mathcal{G} \subseteq \sigma(u)$. Given a goal node u, we compute its minimum cost, $\omega^*(u)$, as the cost to reach its most expensive goal proposition, i.e.,

$$\omega^*(u) := \max\{\omega(u,p) : p \in \mathcal{G}\}. \tag{5}$$

Then, we update the heuristic value $h^{\mathcal{M}}$ as

$$h^{\mathcal{M}} = \min\left\{h^{\mathcal{M}},\ \omega^*(u)\right\}. \tag{6}$$

4.2 Applicable and Essential Actions

For a given layer $\mathcal{N}_t$, the FINDAPPLICABLEACTIONS procedure iterates over each node $u \in \mathcal{N}_t$ to find its applicable actions. The procedure eliminates actions that do not contribute to the computation of the heuristic value and creates an edge for each remaining action.

Given a node $u \in \mathcal{N}_t$, let $A(u)$ be the set of its applicable actions, i.e., $A(u) = \{a \in \mathcal{A} : \ pre(a) \subseteq \sigma(u)\}$. This set can be computed, for instance, by iterating over all actions $a \in \mathcal{A}$ and checking if their preconditions are satisfied.

It is possible to identify if an action $a \in A(u)$ will lead to a state that will be part of the heuristic computation. We denote these actions by $\mathcal{M}$-*essential*.

Definition 4.1. Given a relaxed MDD $\mathcal{M}$ and a node $u \in \mathcal{N}_t$, we say that an action $a \in A(u)$ is $\mathcal{M}$-essential if its successor state $v = \phi(a, \sigma(u))$ satisfies all the following conditions:

(i) State v has not been reached before with less cost, i.e., for each node $u' \in \mathcal{N}_{t'}$ ($t' \leq t$) either $v \not\subseteq \sigma(u')$ or $v \subseteq \sigma(u')$ and $\omega(v,p) \leq \omega(u',p)$ for all $p \in v$.

(ii) State v has a minimum cost less than the current heuristic value, i.e., $c(a) + \max\{\omega(u,p) : p \in pre(a)\} < h^{\mathcal{M}}$.

(iii) State v has a minimum cost less than a given incumbent η^*, i.e., $c(a) + \max\{\omega(u,p) : p \in pre(a)\} < \eta^*$.

We develop a set of rules to identify if an action is $\mathcal{M}$-*nonessential*, i.e., it violates at least one of the conditions in Definition 4.1. Consider a node $u \in \mathcal{N}_t$, an applicable action $a \in A(u)$, and its corresponding edge e. Action a is $\mathcal{M}$-nonessential if any of the following rules hold:

Rule 1. The resulting state adds no new propositions and the cost of each proposition does not decrease, i.e., $\forall p \in add(a) : \ p \in \sigma(u) \ \wedge \ \nu(e,p) \geq \omega(u,p)$.

Rule 2. The minimum cost of the resulting state is higher than the current heuristic value, i.e., $\nu(e) \geq h^{\mathcal{M}}$.

Rule 3. The minimum cost of the resulting state is higher than a given incumbent, i.e., $\nu(e) \geq \eta^*$.

Note that Rules 2 and 3 are direct applications of conditions (ii) and (iii) in Definition 4.1. However, Rule 1 is a necessary, but not sufficient, condition to check if a node has been reached before (i.e., condition (i) in Definition 4.1). The main advantage of these rules, in comparison to the conditions in Definition 4.1, is that we can check them in polynomial time during the construction procedure iterating over each edge only once.

Any action $a \in A(u)$ that satisfies one of the above rules is removed from the set of applicable actions, i.e., $A(u) := A(u) \setminus \{a\}$. After we have checked that each remaining applicable action a in node u is not $\mathcal{M}$-nonessential, we generate a new edge e with label $\theta(e) = a$ that emanates from u and points to node u_0 in the next layer.

4.3 Splitting Nodes

The SPLITNODES procedure is similar to the one used for solving sequencing problems in the literature (Andersen et al. 2007). The procedure, shown in Algorithm 2, splits the nodes in a layer $\mathcal{N}_t$ until it reaches the maximum size $\mathcal{W}$ or there is no more splitting needed.

Algorithm 2 Split states procedure

```
1: procedure SPLITNODES(Input: N_t, W)
2:     if h^M = ∞ and t > 10 and W > 1 then
3:         W = W − 1
4:     Q = {p_1, ..., p_|P|} priority queue
5:     while Q.notEmpty() and |N_t| < W do
6:         p = Q.pop()
7:         for u ∈ N_t do
8:             if δ_p^in(u) = ∅ or δ_p^in(u) = δ^in(u) then
9:                 continue
10:            Create node v and N_t = N_t ∪ {v},
11:            redirect arcs using δ^in(v) = δ_p^in(u) and
12:            δ^in(u) = δ^in(u) \ δ_p^in(u).
13:            if |N_t| = W then break
```

Starting with a layer $\mathcal{N}_t = \{u_0\}$ with a single node, the procedure iteratively splits the nodes such that each resulting node represents fewer states. Specifically, the procedure considers a priority queue of propositions. In each iteration, the procedure chooses a proposition p from the queue (line 6). Then, it iterates over all the nodes $u \in \mathcal{N}_t$ to check if there is any node with incoming edges that can be partitioned such that one partition results in a node with p and the other in a node without p (line 8). In such case, we create a new node v where all edges that represent states where p is true are now directed to v (line 10-12). The procedure ends when there are no more propositions in the queue or we have reached the width limit $\mathcal{W}$.

The priority queue Q divides the propositions into three priority levels. The first level corresponds to goal propositions, i.e., any propositions in $\mathcal{G}$. The second level considers landmark propositions (Hoffmann, Porteous, and Sebastia 2004). We use a simple reachability algorithm to identify propositional landmarks. Finally, the last level corresponds to propositions that are not in the previous levels. Inside each level, we rank the propositions in lexicographical order.

In addition, SPLITNODES checks if the heuristic has been updated (line 2), i.e., if a goal node has been reached in a previous layer. If that is not the case, we reduce the maximum width by 1. The width reduction guarantees the termination of the algorithm (Section 4.4). Our implementation starts the width reduction after 10 layers, which gave the best performance in our testing phase. When $\mathcal{W} = 1$, the construction procedure continues ignoring delete effects.

While there are many ways to split nodes that we intend to investigate in the future, this algorithm has two main advantages. First, this splitting procedure guarantees that there are no two nodes in a layer where one node is a subset of the other. This is due to the fact that we start with a single node and that, in each iteration, we separate the edges according to their propositions. The second advantage is that, if $\mathcal{W}$ is big enough, all nodes will represent a single reachable state.

4.4 Termination Condition

The last component of our top-down construction algorithm, TERMINATE, checks whether we need to create a new layer in our relaxed MDD by observing whether or not procedure FINDAPPLICABLEACTIONS created any new edges.

If the planning task Π is solvable (i.e., a goal state is reachable from $\mathcal{I}$), it is sufficient to check if procedure FINDAPPLICABLEACTIONS created a new edge. Specifically, if the task is solvable, we will eventually reach a goal node, which will make $h^{\mathcal{M}} < \infty$. Since $c(a) > 0$ for all $a \in \mathcal{A}$, Rule 2 (Section 4.2) guarantees that there exists a layer $\mathcal{N}_t$ such that all emanating edges have a cost greater than $h^{\mathcal{M}}$. The same is true if we have an upper bound on the cost-optimal plan ($\eta^* < \infty$) and we use Rule 3 to remove $\mathcal{M}$-nonessential actions.

If the planning task is infeasible, the procedure will still terminates due to the width reduction (Section 4.3). Since our implementation ignores delete effects when we reach $\mathcal{W} = 1$, we can guarantee that there exists a layer $\mathcal{N}_t$ in which $A(u) = \emptyset$ for each $u \in \mathcal{N}_t$.

5 Relationship with Existing Techniques

This section explores the relationship of our relaxed MDD-based heuristic with existing approaches. We start by relating the relaxed MDD structure to a transition graph.

Definition 5.1. (Helmert et al. 2007) A *transition graph* is a 5-tuple $\mathcal{T} = \langle \mathcal{S}, \mathcal{L}, \Sigma, s_{\mathcal{I}}, \mathcal{S}_{\mathcal{G}} \rangle$ where $\mathcal{S}$ is a finite set of states, $\mathcal{L}$ is a finite set of transition labels, Σ is a the set of (labeled) transitions $\Sigma \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$, $s_{\mathcal{I}}$ is the initial state, and $\mathcal{S}_{\mathcal{G}}$ is the set of goal states $\mathcal{S}_{\mathcal{G}} \subseteq \mathcal{S}$.

Consider a relaxed MDD $\mathcal{M} = (\mathcal{N}, \mathcal{E})$ and a planning task $\Pi = \langle \mathcal{P}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$. Note that a relaxed MDD is in fact a transition graph. Specifically, we can represent a relaxed MDD as a transition graph $\mathcal{T}(\mathcal{M}) = \langle \mathcal{N}, \mathcal{A}, \mathcal{E}, \mathbf{r}, \mathcal{N}_{\mathcal{G}} \rangle$ where the set of states is given by the nodes in $\mathcal{M}$, the set of action corresponds to the labels, the edges define the transitions, and the initial and goal states are given by $\mathbf{r} \in \mathcal{N}$ and $\mathcal{N}_{\mathcal{G}} \subseteq \mathcal{N}$. In particular, each edge $e \in \mathcal{E}$ is associated with the 3-tuple $\langle \rho(e), \theta(e), \kappa(e) \rangle$, which is an element of $\mathcal{N} \times \mathcal{A} \times \mathcal{N}$.

Helmert et al. (2007) define a transition graph induced by a planning task Π as $\mathcal{T}(\Pi) = \langle \mathcal{S}, \mathcal{A}, \Sigma(\Pi), \mathcal{I}, \mathcal{S}_{\mathcal{G}} \rangle$, where $\mathcal{S}$ is the set of states of a planning task, $\Sigma(\Pi)$ represents the set of valid transitions and $\mathcal{S}_{\mathcal{G}}$ is a subset of states such that $s \in \mathcal{S}_{\mathcal{G}}$ iff $\mathcal{G} \subseteq s$. In particular, any transition $\langle s, a, s' \rangle \in \Sigma(\Pi)$ is such that $pre(a) \subseteq s$ and $s' = \phi(a, s)$.

Consider an unbounded (i.e., $\mathcal{W} = \infty$) relaxed MDD $\mathcal{M}^{\infty} = \{\mathcal{N}^{\infty}, \mathcal{E}^{\infty}\}$. The transition graph given by $\mathcal{M}^{\infty}$, $\mathcal{T}(\mathcal{M}^{\infty})$, is in fact a transition graph induced by the planning task Π. Any path from $\mathbf{r}$ to a node $u \in \mathcal{N}_{\mathcal{G}}$ is a valid plan, and the shortest path represents a cost-optimal plan with cost equal to $h^{\mathcal{M}}$.

5.1 Relaxed MDDs and Abstractions

Definition 5.2. (Helmert et al. 2007) An *abstraction* of a transition graph $\mathcal{T}$ is a pair $\langle \mathcal{T}', \alpha \rangle$ where $\mathcal{T}' = \langle \mathcal{S}', \mathcal{L}, \Sigma', s'_{\mathcal{I}}, \mathcal{S}'_{\mathcal{G}} \rangle$ is a transition graph called the *abstract transition graph* and $\alpha : \mathcal{S} \rightarrow \mathcal{S}'$ is a function called the *abstraction mapping*. Specifically, we have that $\langle \alpha(s), a, \alpha(s') \rangle \in \Sigma'$ for all $\langle s, a, s' \rangle \in \Sigma$, $\alpha(s_{\mathcal{I}}) = s'_{\mathcal{I}}$, and $\alpha(s_{\mathcal{G}}) \in \mathcal{S}'_{\mathcal{G}}$ for all $s_{\mathcal{G}} \in \mathcal{S}_{\mathcal{G}}$.

Abstraction-based heuristics are admissible heuristics calculated as shortest paths on an abstract transition graph. Several works have studied different ways to define abstractions (Edelkamp 2001; Helmert et al. 2007) and how to combine them (Katz and Domshlak 2010).

We now show that an MDD relaxation is equivalent to an abstraction of an unbounded MDD. For theoretical purposes, assume that the construction procedure ignores Rule 2 (Section 4.2) and we have an upper bound on the optimal plan cost, η^*. Note that these two requirements do not affect the heuristic computation over a relaxed MDD.

Proposition 5.1. Consider a classical planning task Π. Let $\mathcal{M}^{\infty} = (\mathcal{N}^{\infty}, \mathcal{E}^{\infty})$ and $\mathcal{M} = (\mathcal{N}, \mathcal{E})$ be two relaxed MDDs constructed using Algorithm 1, where $\mathcal{M}^{\infty}$ has an unbounded width, and $\mathcal{M}$ has a maximum width $1 \leq \mathcal{W} < \infty$. For every node $u \in \mathcal{N}_t^{\infty}$ there exists a node $u' \in \mathcal{N}_{t'}$ ($t' \leq t$) such that

$$\sigma(u) \subseteq \sigma(u') \text{ and } \omega(u, p) \geq \omega(u', p) \ \ \forall p \in \sigma(u). \quad (7)$$

Proof. We prove the above statement by induction on the number of layers of $\mathcal{M}^\infty$. Consider the base case where $t = 0$. By construction we have that $\mathcal{N}_0^\infty = \mathcal{N}_0 = \{\mathbf{r}\}$, where $\sigma(\mathbf{r}) = \mathcal{I}$. Thus, condition (7) is satisfied.

Now consider that (7) is valid for all nodes $u \in \mathcal{N}_t^\infty$, for a given $t \geq 0$. Let $v \in \mathcal{N}_{t+1}^\infty$ and $e \in \delta^{in}(v)$ be any edge directed to v. Take $a = \theta(e)$ and $u = \rho(e)$, i.e., $u \in \mathcal{N}_t^\infty$. By hypothesis, there exists a node $u' \in \mathcal{N}_{t'}$ ($t' \leq t$) such that (7) is satisfied for node u. By construction, a is an applicable action on u'. It might be, however, $\mathcal{M}$-nonessential for u'. If $a \notin A(u')$, then Rule 1 (Section 4.2) has to be true and u' satisfies (7) for node v. If $a \in A(u')$, then there exists a node $v' \in \mathcal{N}_{t'+1}$ such that the edge associated to a directs to it. Note that v' satisfies (7) for v due to (4) and (3). □

A direct result of the above proposition is the admissibility of our relaxed MDD based heuristic.

Theorem 5.1. Given a classical planning task Π and a maximum size $\mathcal{W} \geq 1$, Algorithm 1 computes an admissible heuristic $h^{\mathcal{M}}$.

Proof. Consider a relaxed MDD $\mathcal{M} = (\mathcal{N}, \mathcal{E})$ with $1 \leq \mathcal{W} < \infty$ constructed using Algorithm 1 and an unbounded MDD $\mathcal{M}^\infty = (\mathcal{N}^\infty, \mathcal{E}^\infty)$. From Proposition 5.1, for every goal node $u \in \mathcal{N}_\mathcal{G}^\infty$ there exists a goal node $u' \in \mathcal{N}_\mathcal{G}$ such that

$$\omega(u', p) \leq \omega(u', p) \qquad \forall p \in \mathcal{G},$$

and so $\omega^*(u') \leq \omega^*(u)$. Therefore, $h^{\mathcal{M}} \leq h_\infty^{\mathcal{M}} = h^*$, where h^* is the perfect heuristic. □

We now use Proposition 5.1 to create an abstract mapping from an unbounded MDD to a relaxed one, as shown in Proposition 5.2. In other words, we show that the transition graph defined over a relaxed MDD is an abstract transition graph for a planning task Π.

Proposition 5.2. Consider a planning task Π, a relaxed MDD $\mathcal{M} = (\mathcal{N}, \mathcal{E})$ with maximum width $\mathcal{W} \geq 1$, and the transition graph induced by $\mathcal{M}, \mathcal{T}(\mathcal{M}) = \langle \mathcal{N}, \mathcal{A}, \mathcal{E}, \mathcal{I}, \mathcal{N}_\mathcal{G} \rangle$. There exists an abstraction mapping α such that $\langle \mathcal{T}(\mathcal{M}), \alpha \rangle$ is an abstraction of $\mathcal{T}(\mathcal{M}^\infty)$, where $\mathcal{M}^\infty = (\mathcal{N}^\infty, \mathcal{E}^\infty)$ is an unbounded MDD for Π.

Proof. We define an abstraction mapping $\alpha : \mathcal{N}^\infty \to \mathcal{N}$ recursively over the layers of $\mathcal{M}^\infty$. We start with $\alpha(\mathbf{r}^\infty) = \mathbf{r}$ and assume that we have defined α for all nodes in layer $\mathcal{N}_t^\infty$. For each node $v \in \mathcal{N}_{t+1}^\infty$ take any incoming edge $e \in \delta^{in}(v)$ and its tail $u = \rho(e)$. Consider $u' = \alpha(u) \in \mathcal{N}$. If there exists an edge $e' \in \delta^{out}(u')$ such that $\theta(e) = \theta(e')$, then $\alpha(v) = \rho(e')$, otherwise $\alpha(v) = u'$.

Due to Proposition 5.1, the abstraction mapping α is such that every goal node $u \in \mathcal{N}_\mathcal{G}^\infty$ is mapped to a goal node $u' \in \mathcal{N}_\mathcal{G}$. Moreover, every transition $\langle u, \theta(e), v \rangle$ defined by an edge $e \in \mathcal{E}^\infty$ has a corresponding transition $\langle \alpha(u), \theta(e), \alpha(v) \rangle$ in $\mathcal{T}(\mathcal{M})$. Note that any transition $\langle \alpha(u), \theta(e), \alpha(v) \rangle$ that defines a self loop (i.e., $\alpha(u) = \alpha(v)$) is not explicitly defined by any edge in $\mathcal{M}$. However, we can extend the set of transitions in $\mathcal{T}(\mathcal{M})$ without impacting the heuristic value. Specifically, we can consider $\mathcal{E}' = \mathcal{E} \cup \mathcal{E}_{loops}$, where every edge in $e \in \mathcal{E}_{loops}$ corresponds to an edge that violates Rule 1 (Section 4.2). □

5.2 $h^{\mathcal{M}}$ vs. h^{max}

We now compare our heuristic with the simplest admissible critical path heuristic, h^{max} (Haslum and Geffner 2000). This heuristic computes the minimum cost to reach each proposition from the initial state. Specifically, consider $h(p)$ as the minimum cost to reach $p \in \mathcal{P}$, and $h(a)$ as the minimum cost to use action a. These values are computed recursively using the formula below and setting $h(p) = 0$ for all $p \in \mathcal{I}$, $h(p) = \infty$ for any $p \notin \mathcal{I}$, and $h(a) = \infty$.

$$h(p) := \min_{a \in \mathcal{A}(p)} \{h(p), h(a)\} \qquad \forall p \in \mathcal{P}$$

$$h(a) := c(a) + \max\{h(q) : q \in pre(a)\} \qquad \forall a \in \mathcal{A}$$

Then, the h^{max} heuristic is define as

$$h^{max} := \max\{h(p) : p \in \mathcal{G}\}$$

Proposition 5.3. Consider a classical planning task Π and a relaxed MDD $\mathcal{M} = (\mathcal{N}, \mathcal{E})$ with a maximum size $\mathcal{W} \geq 1$. Then, $h^{\mathcal{M}} \geq h^{max}$.

Proof. First, consider the following statement:

$$h(p) \leq \omega(u, p) \qquad \forall u \in \mathcal{N},\ p \in \sigma(u) \tag{8}$$

We prove (8) by induction over the layers of $\mathcal{M}$. By construction, (8) holds for $\mathcal{N}_0 = \{\mathbf{r}\}$. Now consider that (8) is true for all nodes $u \in \mathcal{N}_t$ and $p \in \sigma(u)$. Consider a node $v \in \mathcal{N}_{t+1}$ and a proposition $p \in \sigma(v)$. By construction, there exists an edge $e \in \delta^{in}(v)$ such that $\nu(e, p) = \omega(v, p)$. Consider action $a = \theta(e)$ and node $u = \rho(e) \in \mathcal{N}_t$. There are two cases, either $p \in add(a)$ or not. If $p \in add(a)$, then

$$\begin{aligned} \nu(e, p) &= c(a) + \max\{\omega(u, q) : q \in pre(a)\} \\ &\geq c(a) + \max\{h(q) : q \in pre(a)\} \geq h(p) \end{aligned}$$

If $p \notin add(a)$, then $p \in \sigma(u)$. Since $u \in \mathcal{N}_t$, we have $h(p) \leq \omega(u, p)$. Then it follows that

$$h(p) + c(a) \leq \omega(u, p) + c(a) \leq \nu(e, p)$$

Therefore, $h(p) \leq v(e, p)$, which proves (8). Since (8) is true, it follows that $h^{\mathcal{M}} \geq h^{max}$. □

6 Implementation

This section presents how we can exploit the graphical structure given by the relaxed MDD to improve the search procedure. Our approach constructs $\mathcal{M}$ in each state s of the search and uses $h^{\mathcal{M}}$ as an admissible heuristic in a modified A^* search algorithm. Specifically, we add a bounding mechanism to A^*, similar to the branch-and-bound algorithm used in Integer Programming (IP) solvers. To do so, we use $\mathcal{M}$ to find feasible plans while computing $h^{\mathcal{M}}$. The following sections explain how we can find a feasible plan using a relaxed MDD and how the cost of this plan enhances the A^* search algorithm.

6.1 Finding Feasible Plans in a Relaxed MDD

Our implementation considers two different procedures to find a feasible plan using the relaxed MDD graphical structure. The first procedure extracts a relaxed plan, denoted by

π^h, with equal cost to the heuristic value and checks its validity. The second approach selects a subset of nodes from the relaxed MDD that represent single states and uses them to find a valid plan π^b. For both procedures, plan extraction and validation occur after the relaxed MDD construction.

Consider a relaxed MDD $\mathcal{M}$ for a state s with at least one minimum cost goal node $u_{\mathcal{G}}$. We follow the edges of $\mathcal{M}$ backward to find a path from $u_{\mathcal{G}}$ to s. The resulting path is a relaxed plan, π^h, that has the same cost as our heuristic $h^{\mathcal{M}}$. If π^h is a valid plan, we create a plan π that is valid for the planning task. Consider $\pi^{\mathcal{I}}$ as the plan from $\mathcal{I}$ to state s given by the search algorithm. Then, we create a feasible plan π concatenating $\pi^{\mathcal{I}}$ and π^h, i.e., $\pi = (\pi^{\mathcal{I}}, \pi^h)$.

The second method allocates a fixed number of nodes, $\mathcal{W}^e \leq \mathcal{W}$, in each layer of $\mathcal{M}$ to participate in the extraction of a valid plan. For each layer $\mathcal{N}_t$, let $\mathcal{N}_t^e$ be a set of nodes that represent a single state (i.e., *exact* nodes), and $\mathcal{N}_t^r$ be a set of node that represent the union approximation of multiple states (i.e., *relaxed* nodes), where $\mathcal{N}_t = \mathcal{N}_t^e \cup \mathcal{N}_t^r$. Specifically, we modify SPLITNODES such that in each layer $\mathcal{N}_t$ we arbitrarily select $\mathcal{W}^e$ edges emanating from nodes $u^e \in \mathcal{N}_t^e$ to be the exact nodes in $\mathcal{N}_{t+1}^e$. If an exact node u^e is a goal node, then we extract a plan taking any path from $\mathbf{r}$ to u^e. Since all parent nodes of an exact node are exact, we can guarantee that the extracted plan is valid. As previously, we generate a valid plan for the planning task by concatenating the extracted plan π^b with $\pi^{\mathcal{I}}$, i.e., $\pi = (\pi^{\mathcal{I}}, \pi^b)$.

While having more exact nodes increases the chances of finding a feasible plan π^b, the heuristic quality can be negatively affected. Since the maximum width does not change, the union approximation of the relaxed nodes is weaker. Hence, we use the second method only to find a first feasible plan.

Note that whenever we find a valid plan π (created with either π^h or π^b), we can use its cost as an upper bound η^* in the construction procedure. Specifically, for a state s in the search, the value of η^* in Rule 3 (Section 4.2) is set to $c(\pi) - c(\pi^{\mathcal{I}})$, where $\pi^{\mathcal{I}}$ is the plan to reach s from $\mathcal{I}$.

6.2 Exploiting Upper Bounds in A^*

To take advantage of the information represented by the MDD, we propose a modified A^* search algorithm that considers the cost of feasible solutions. In particular, our approach is inspired by the branch-and-bound algorithm implemented in IP solvers. Branch-and-bound uses a linear programming (LP) relaxation as an admissible heuristic to guide the search. Whenever the LP relaxation gives an integer solution, the algorithm prunes any node in the search for which the LP relaxation provides a cost greater than the upper bound. Similarly to the branch-and-bound algorithm, our approach uses feasible extracted plans to create an upper bound and prune states in the search space.

We incorporate this idea in A^*, proposing a variant that we call A^*_{BB}. In every expanded state, A^*_{BB} checks the feasibility of a relaxed plan calculated by a relaxed MDD $\mathcal{M}$. The cost of such a valid plan plus the cost of reaching the state is an upper bound on the cost of the optimal solution.

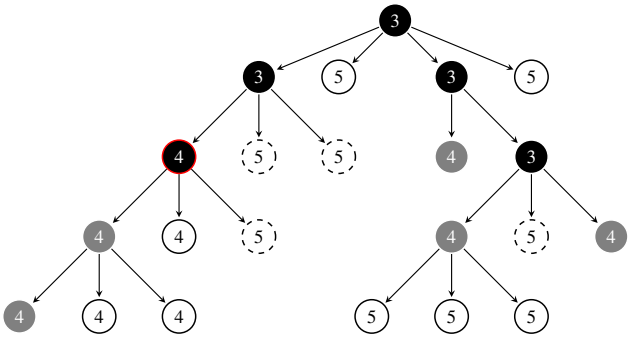


Figure 3: Nodes explored by A^* (in black and gray) and A^*_{BB} (in black). Dashed nodes are states that do not need to be inserted in the search queue by A^*_{BB}. Nodes circled in red are the ones with a feasible plan π^h. Node labels indicate the f-value of a state.

In A^*, states with a f-value[1] strictly greater than the optimal solution are never expanded, thus, the benefit of pruning states with a greater f-value than the upper bound is limited to memory saving during search.

The real advantage of A^*_{BB} arises when $\mathcal{M}$ extracts a feasible plan π^h with the same cost as $h^{\mathcal{M}}$ for a state s. If s is retrieved from the open list, the search terminates and the minimum-cost plan is $\pi = (\pi^{\mathcal{I}}, \pi^h)$. The termination criteria is correct since s has the minimum f-value among the states in the list, i.e., $f = h^{\mathcal{M}} + g$ is a lower bound for the minimum-cost plan. Since $\pi = (\pi^{\mathcal{I}}, \pi^h)$ is a valid plan with cost equal to f, this proves that π is a minimum-cost plan. Notice that A^*_{BB} can be used with any admissible heuristic (consistent or not) that has a plan extraction procedure.

Therefore, A^*_{BB} may avoid expanding states with an f-value equal to the optimal solution, while A^* would need to explore them. Figure 3 shows an example of the difference in states expanded by A^* and A^*_{BB}. Of course, in the worst case, A^*_{BB} will still look at the same number of states as A^*.

7 Preliminary Results

We now present an empirical analysis on the relaxed MDD heuristic using the LPRGP planning system (Coles et al. 2008). We experiment with both A^* and A^*_{BB} algorithms, where ties are broken preferring higher h-values. We consider three variations of relaxed MDDs, where we limit the maximum width to 256, 512 and 1024, respectively. This analysis includes a comparison between our heuristics and h^{max} and the operator counting heuristic h^{oc} (Pommerening et al. 2014). We implemented a STRIPS version of h^{oc} with landmarks and state equation constraints. The LP models are solved using CPLEX v12.7. All experiments are run on a Xeon 3.5GHz processor machine, with a 2 GB memory limit and a 30 minute time limit.

We selected 6 domains with positive action costs, from the last two International Planning Competitions (IPCs): *no-mystery*, *wood-working*, *floortile*, *tetris*, *transport*, and *visit-all*. *No-mystery* and *visit-all* have unary action costs, while

[1]We assume the usual heuristic search notation: $f = h + g$.

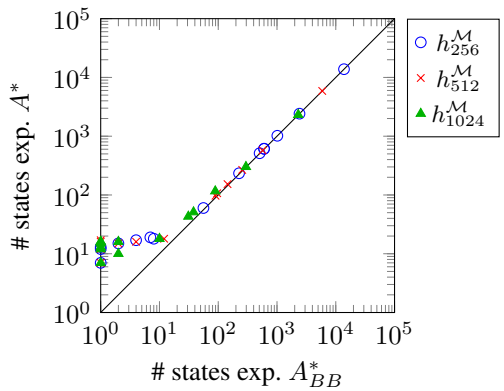


Figure 4: Number of states expanded using A^*_{BB} and A^*.

the other domains feature non-uniform action costs. These domains were chosen due to their range in difficulty and to illustrate the strong and weak aspects of our approach.

7.1 A^* vs. A^*_{BB}

We compare A^* with A^*_{BB} using our relaxed MDD heuristics $h^{\mathcal{M}}_{256}$, $h^{\mathcal{M}}_{512}$ and $h^{\mathcal{M}}_{1024}$ (with $\mathcal{W} \in \{256, 512, 1024\}$, respectively).

Figure 4 shows the number of expanded states (logarithmic scale) for each search algorithm and heuristic. A point in the plot represents an instance and its (x, y) coordinate the number of states expanded by A^*_{BB} and A^*, respectively. Figure 4 shows that A^*_{BB} expands fewer (or equal) number nodes than A^* in all instances, especially for instances that need a small number of expansions to find the cost-optimal plan. In fact, on average A^*_{BB} reduces the number of expanded nodes by 1%, 2% and 6% when using heuristic $h^{\mathcal{M}}_{256}$, $h^{\mathcal{M}}_{512}$ and $h^{\mathcal{M}}_{1024}$, respectively. Similar results are found for number of states evaluated, where A^*_{BB} decreases the number of states evaluated by 1%, 4% and 12% when $h^{\mathcal{M}}_{256}$, $h^{\mathcal{M}}_{512}$ and $h^{\mathcal{M}}_{1024}$ are used, respectively.

As expected, the benefit of using A^*_{BB} is more prominent when using a bigger width. A larger relaxed MDD is more likely to provide valid relaxed plans.

7.2 $h^{\mathcal{M}}$ vs. Existing Techniques

We now compare the performance of our proposed relaxed MDD heuristics against h^{max} and h^{oc}. We use A^*_{BB} as the search algorithm for the $h^{\mathcal{M}}$ heuristics, while we employ A^* for both h^{max} and h^{oc}. Table 1 shows the number of instances that each approach solves to optimality (*# Optimal plans*) and the number of instances for which a relaxed MDD heuristic finds a feasible plan (*# Valid plans*). It should be noted that, due to the nature of A^*, on the problems for which h^{max} and h^{oc} fail to find optimal solutions, they also do not find feasible solutions.

With respect to the number of optimal plans, h^{oc} achieves the best coverage, followed by h^{max} and the $h^{\mathcal{M}}$ heuristics. In particular, $h^{\mathcal{M}}_{256}$ performs best among the relaxed MDD heuristics, finding an optimal plan on 15 instances.

Table 1: Coverage performance.

		# Optimal plans					# Valid plans		
	#	$h^{\mathcal{M}}_{256}$	$h^{\mathcal{M}}_{512}$	$h^{\mathcal{M}}_{1024}$	h^{oc}	h^{max}	$h^{\mathcal{M}}_{256}$	$h^{\mathcal{M}}_{512}$	$h^{\mathcal{M}}_{1024}$
floortile	20	0	0	0	**2**	**2**	19	**20**	**20**
no-mystery	20	8	8	8	**15**	7	10	10	**11**
tetris	20	3	3	3	**13**	5	**15**	14	**15**
transport	20	1	0	0	1	**5**	12	**15**	14
visit-all	20	1	1	1	**6**	0	8	11	**13**
wood-working	20	2	2	2	**5**	2	**19**	**19**	**19**
TOTAL	120	15	14	14	**42**	21	83	89	**92**

To understand these results, Table 2 compares the run time and number of states expanded over the instances that all heuristics solve to optimality. The symbol # indicates the number of instances considered. We can see that the $h^{\mathcal{M}}$ heuristics have the highest average run time. However, we observe an opposite trend in terms of the number of states expanded: all $h^{\mathcal{M}}$ heuristics expand orders of magnitude fewer states than h^{max} and a similar number as h^{oc}. The only exception is *wood-working*, where $h^{\mathcal{M}}$ expands significantly fewer states than h^{oc}.

Table 2: Average run time and states expanded.

		Average run time (sec)				
	#	$h^{\mathcal{M}}_{256}$	$h^{\mathcal{M}}_{512}$	$h^{\mathcal{M}}_{1024}$	h^{oc}	h^{max}
no-mystery	7	20.5	24.8	27.9	**0.6**	49.0
tetris	3	45.3	55.4	69.4	**1.1**	3.1
wood-working	2	307.0	192.1	97.0	**32.2**	223.9
		Average # states expanded				
no-mystery	7	35.6	15.1	**6.4**	45.6	35053.6
tetris	3	360.7	193.7	99.7	**33.0**	6326.0
wood-working	2	553.5	117.0	**20.5**	2238.5	97394.0

While relaxed MDD-based heuristics seem to be highly informative, their computational cost is currently too high to make them competitive with state-of-the-art heuristics.

We also point out the strength of our approach to find valid plans. As shown in Table 1, all relaxed MDD-based heuristics have a high coverage when finding a valid plan. Specifically, our approach has an exceptional performance finding feasible plans in *floortile*, the only domain where none of the $h^{\mathcal{M}}$ heuristics found an optimal solution.

With respect to solution quality, Table 3 shows the mean relative error (MRE) for the best feasible solution found by each MDD-based heuristic. We compute the MRE for instances where all relaxed MDD heuristics found a feasible plan. For a given heuristic and instance, we compute the relative error as $(UB - LB)/UB$, where UB is the best incumbent found the heuristic and LB is the best known lower bound, i.e., either the optimal solution or the best heuristic value in the initial state. The table shows that $h^{\mathcal{M}}_{1024}$, on average, finds the best quality plan. However, on most domains, the feasible plans are still quite far from optimal.

8 Conclusions and Future Works

This work presents a new heuristic to solve cost-optimal classical planning problems based on relaxed multivalued

Table 3: Mean Relative Error for all domains.

Domain	#	$h^{\mathcal{M}}_{256}$	$h^{\mathcal{M}}_{512}$	$h^{\mathcal{M}}_{1024}$
floortile	19	0.61	**0.58**	0.59
no-mystery	10	0.04	0.06	**0.03**
tetris	14	**0.18**	**0.18**	**0.18**
transport	12	0.63	0.59	**0.55**
visit-all	8	**0.31**	0.46	0.46
wood-working	19	0.25	0.25	**0.22**
All instances	82	0.36	0.36	**0.35**

decision diagrams (MDDs), a graphical structure that provides an adjustable approximation of the state-space transition graph. We present an algorithm that constructs relaxed MDDs and calculates an admissible heuristic. Moreover, we show how to exploit the graphical structure to find valid plans and enhance an A^* search algorithm by considering upper bounds. We relate this graphical structure to transition graphs and show that a relaxed MDD is an abstraction of the state transition graph. Moreover, we show that our heuristic is strictly more informative than the h^{max} heuristic.

Preliminary results in a subset of IPC domains show that relaxed MDD heuristics can considerably reduce the number of states expanded during search. However, the effort to compute a relaxed MDD currently makes the approach uncompetitive.

Future directions include an extension of our framework to SAS+ planning and a more in-depth study of the relationship between relaxed MDD and abstractions. In particular, we want to exploit MDDs to represent projections and combine them using a Lagrangian decomposition method (Fisher 2004) similarly to the cost-partition framework (Katz and Domshlak 2010).

References

Andersen, H. R.; Hadzic, T.; Hooker, J. N.; and Tiedemann, P. 2007. A constraint store based on multivalued decision diagrams. In *CP 2007*. Springer. 118–132.

Bergman, D.; Cire, A. A.; van Hoeve, W.-J.; and Hooker, J. N. 2016. *Decision Diagrams for Optimization*. Springer International Publishing, 1 edition.

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial intelligence* 90(1):281–300.

Bonet, B., and Geffner, H. 2000. Planning as Heuristic Search: New Results. *Recent Advances in AI Planning* 1809:360–372.

Bonet, B., and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence* 129(February 2000):5–33.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. A hybrid relaxed planning graph-lp heuristic for numeric planning domains. In *ICAPS 2008*, 52–59.

Domshlak, C.; Hoffmann, J.; and Katz, M. 2015. Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence* 221:73–114.

Edelkamp, S.; Kissmann, P.; and Torralba, Á. 2012. Symbolic a* search with pattern databases and the merge-and-shrink abstraction. In *ECAI 2012*, 306–311.

Edelkamp, S. 2001. Planning with pattern databases. In *ECP 2001*, 13–24.

Fisher, M. L. 2004. The lagrangian relaxation method for solving integer programming problems. *Management science* 50(12):1861–1871.

Geißer, F.; Keller, T.; and Mattmüller, R. 2016. Abstractions for planning with state-dependent action costs. In *ICAPS 2016*, 140–148.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *AIPS 2000*, 140–149.

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

Helmert, M.; Haslum, P.; Hoffmann, J.; et al. 2007. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS 2007*, 176–183.

Hoda, S.; Van Hoeve, W.-J.; and Hooker, J. N. 2010. A systematic approach to MDD-based constraint programming. In *CP 2010*. Springer. 266–280.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *Journal of Artificial Intelligence Research* 22:215–278.

Katz, M., and Domshlak, C. 2010. Optimal admissible composition of abstraction heuristics. *Artificial Intelligence* 174(12-13):767–798.

Katz, M.; Hoffmann, J.; and Domshlak, C. 2013. Red-black relaxed plan heuristics. In *AAAI 2013*, 489–49.

Keller, T.; Pommerening, F.; Seipp, J.; Geißer, F.; and Mattmüller, R. 2016. State-dependent cost partitionings for cartesian abstractions in classical planning. In *IJCAI 2016*, 3161–3169.

Kinable, J.; Cire, A. A.; and van Hoeve, W.-J. 2017. Hybrid optimization methods for time-dependent sequencing problems. *European Journal of Operational Research* 259(3):887 – 897.

Pommerening, F.; Röger, G.; Helmert, M.; and Bonet, B. 2014. LP-based heuristics for cost-optimal planning. In *ICAPS 2014*, 226–234.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In *AAAI 20014*, 2358–2366.

Torralba, Á.; Linares López, C.; and Borrajo, D. 2016. Abstraction heuristics for symbolic bidirectional search. In *IJCAI 2016*, 3272–3278.

Torralba, Á.; López, C. L.; and Borrajo, D. 2013. Symbolic merge-and-shrink for cost-optimal planning. In *IJCAI 2013*.