Rintanen, Jussi; Fadnis, Saurabh

Generalized 3-Valued Belief States in Conformant Planning

Published in:
PRICAI 2022: Trends in Artificial Intelligence - 19th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2022, Shanghai, China, November 10-13, 2022, Proceedings

DOI:
10.1007/978-3-031-20862-1_8

Published: 01/11/2022

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Rintanen, J., & Fadnis, S. (2022). Generalized 3-Valued Belief States in Conformant Planning. In S. Khanna, J. Cao, Q. Bai, & G. Xu (Eds.), *PRICAI 2022: Trends in Artificial Intelligence - 19th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2022, Shanghai, China, November 10-13, 2022, Proceedings* (pp. 104-117). (Lecture Notes in Computer Science; Vol. 13629 LNCS). Springer. https://doi.org/10.1007/978-3-031-20862-1_8

Generalized 3-Valued Belief States in Conformant Planning

Saurabh Fadnis[0000−0001−9307−281X] and Jussi Rintanen[0000−0001−5983−0074]

Aalto University, Department of Computer Science

Abstract. The high complexity of planning with partial observability has motivated to find compact representations of belief state (sets of states) that reduce their size exponentially, including the 3-valued literal-based approximations by Baral et al. and tag-based approximations by Palacios and Geffner.
We present a generalization of 3-valued literal-based approximations, and an algorithm that analyzes a succinctly represented planning problem to derive a set of formulas the truth of which accurately represents any reachable belief state. This set is not limited to literals and can contain arbitrary formulas. We demonstrate that a factored representation of belief states based on this analysis enables fully automated reduction of conformant planning problems to classical planning, bypassing some of the limitations of earlier approaches.

1 Introduction

In comparison to classical planning, which has a single known initial state and deterministic actions and thus a completely predictable and observable future, more general forms of planning with multiple initial states and incomplete observability require considering sets of possible current states, leading to the notion of *belief states*. In this setting, the knowledge state of an agent is initially incomplete, consisting of multiple states (and not just one), and each action maps the current belief state to a new one, consisting of the new possible current states. This is the reason why limited observability increases the complexity by an exponential in comparison to the fully observable case.

Earlier works have used propositional logic and related NP-complete languages for compact belief space representations in planning under partial observability [2, 19] and full observability [7]. The representations of state sets in these works use sets of *literals*, that is propositional variables and negated propositional variables, which is equivalent to 3-valued valuations in which a state variable can have the value *true*, *false*, or *unknown*. Sets of literals cannot represent arbitrary state sets. For example, the set $\{01, 10, 11\}$ is not representable as a set of literals, and, more generally, any set with *dependencies* between state variables, which is the typical case, cannot be.

Our goal is to provide a method for determining cases in which all relevant state sets can indeed be accurately represented as sets of literals, and when this is not possible, to determine which type of more general representation

is sufficient. Our representation is with sets $T = \{\phi_1, \ldots, \phi_n\}$ of propositional formulas so that any belief state B (state set) is characterized by some $R \subseteq T$ as $B = \{v | v \models \bigwedge R\}$. The component formulas $\phi_1, \ldots, \phi_n$ could be limited to *clauses*, but also unlimited propositional formulas can be used instead. The 3-valued representations [2, 19] can be viewed as a special case, as we could choose the set T to consist of literals x and $\neg x$ for all state variables x.

Hence, belief states can be represented as vectors $(b_1, \ldots, b_n)$, indicating which of the formulas in $T = \{\phi_1, \ldots, \phi_n\}$ hold in the belief state. As this is a bit-vector, different actions are mappings from bit-vectors to bit-vectors, and it is straightforward to turn the conformant planning problem to a standard state-space search problem, solvable for example by classical planners.

The plan of this work is as follows. We will first introduce planning without observability (often known as *conformant planning*), and a novel representation of belief states in terms of subsets of a fixed set of formulas (that we call a *base*). We show how actions can be understood as mappings from valuations of the base to valuations to the base, and we give an algorithm for identifying a base for an arbitrary conformant planning problem. Then we propose a reduction from conformant planning to classical planning, in which each formula in the base is identified with a state variable in classical planning. Both finding a base and deriving the classical planning problem involve worst-case exponential operations, but we show that simple approximation schemes still allow solving many hard conformant planning problems efficiently. We conclude the paper by discussing possible extensions of our work.

2 Preliminaries

Define a problem instance in conformant planning as a tuple $\langle X, I, A, G \rangle$ where

- X is a finite set of *state variables*,
- I is a formula for the *initial states*,
- A is a finite set of formulas over $X \cup \{x' | x \in X\}$ representing *actions*, and
- G is a formula for the *goal states*.

The action representation is the one well known from OBDD and SAT-based planning methods [5], in which the relation between a state and its possible successor states is represented as arbitrary Boolean functions over the state variables $X = \{x_1, \ldots, x_n\}$ and the *next state* variables $X' = \{x'_1, \ldots, x'_n\}$. This is a general representation, to which deterministic and non-deterministic variants of PDDL can be translated.

In Section 5.1 we will also use a representation of actions close to standard modelling languages, in which actions are pairs (p, e) where p is a formula and e (the *effect*) is a set of rules $\phi \triangleright l$, where the literals in l are made true conditional on the formula ϕ being true. If $\phi = \top$, then the literals become true unconditionally (which is the case in the simplest so-called STRIPS actions.)

Example 1 (Sorting Networks). Consider a sorting network problem, in which the initial belief state is the set of all possible states over the state variables

x_1, x_2, x_3, and the three actions are $(\top, (x_1 \wedge \neg x_2) \triangleright (\neg x_1; x_2))$, $(\top, (x_2 \wedge \neg x_3) \triangleright (\neg x_2; x_3))$, and $(\top, (x_1 \wedge \neg x_3) \triangleright (\neg x_1; x_3))$, or equivalently, $\Phi_{12} = (x'_1 \leftrightarrow (x_1 \wedge x_2)) \wedge (x'_2 \leftrightarrow (x_1 \vee x_2)) \wedge (x'_3 \leftrightarrow x_3)$, $\Phi_{23} = (x'_1 \leftrightarrow x_1) \wedge (x'_2 \leftrightarrow (x_2 \wedge x_3)) \wedge (x'_3 \leftrightarrow (x_2 \vee x_3))$. and $\Phi_{13} = (x'_1 \leftrightarrow (x_1 \wedge x_3)) \wedge (x'_2 \leftrightarrow x_2) \wedge (x'_3 \leftrightarrow (x_1 \vee x_3))$. The actions swap the values of two state variables if they are not in increasing order.

Since the initial state is not known and the actions just reorder the unknown values of the state variables, the value of no state variable ever becomes known. The only known thing is the orderings of some state variables.

3 Theory

Given actions and a formula for the initial belief state, our objective is to identify $T = \{\phi_1, \ldots, \phi_n\}$ so that every reachable belief state can be represented as a conjunction of some $R \subseteq T$. We call such a set T a *base*. When a literal-based approximation [2] is sufficient, T is a set of literals. More generally, T consists of arbitrary formulas. For example, we will see that the 3-input sorting network problem can be represented in terms of $T = \{x_1 \rightarrow x_2, x_1 \rightarrow x_3, x_2 \rightarrow x_3\}$.

Definition 1. *Let X be the set of state variables. Then* a transition relation formula *is any formula over $X \cup X'$, where X' consists of "primed" versions x' of state variables $x \in X$ which represent the values of x in the successor state.*

Definition 2. *A transition relation formula Φ is* deterministic *iff there is a logically equivalent formula $\Phi_d = \chi \wedge \bigwedge_{x \in X} x' \leftrightarrow \phi_x$ where χ is a propositional formula over X and each $\phi_x, x \in X$ is a propositional formula over X.*

As is known from BDD-based reachability [5], a formula representing the successors of a given set of states with respect to a transition relation, when the latter two are represented as formulas, can be obtained by using the existential abstraction operation $\exists$ and renaming of variables in X' to the corresponding ones in X, expressed as $[X/X']$.

Definition 3 (Successors). *Given a transition relation formula Φ and a formula ϕ, the* successor *of ϕ w.r.t. Φ (denoted by $succ_\Phi(\phi)$) is $(\exists X.(\phi \wedge \Phi))[X/X']$. For sequences $\Phi_1, \ldots, \Phi_m$ we define $succ_{\Phi_1;\cdots;\Phi_m}(\phi) = succ_{\Phi_m}(\cdots succ_{\Phi_1}(\phi) \cdots)$.*

If the number of formulas in the base T is n, then it would seem that we would have to consider all 2^n different subsets when looking at the possible successor belief states with respect to a given action. We can, however, incompletely and with a complexity reduction from 2^n to n, analyze possible successor belief states for every member of T separately.

Theorem 1. *$succ_\Phi(\alpha \wedge \beta) \models succ_\Phi(\alpha) \wedge succ_\Phi(\beta)$ for any transition relation Φ.*

Proof. We apply the following sequence of equivalences and consequences to each of the variables in X in $\exists X.(\alpha \wedge \beta \wedge \Phi)$, starting from the innermost one, and

resulting in $\exists X.(\alpha \wedge \Phi) \wedge \exists X.(\beta \wedge \Phi)$.

$$\begin{aligned}\exists x.(\alpha \wedge \beta \wedge \Phi) &\equiv (\alpha \wedge \beta \wedge \Phi)[\top/x] \vee (\alpha \wedge \beta \wedge \Phi)[\bot/x] \\ &\equiv (\alpha[\top/x] \wedge \beta[\top/x] \wedge \Phi[\top/x]) \vee (\alpha[\bot/x] \wedge \beta[\bot/x] \wedge \Phi[\bot/x]) \\ &\models ((\alpha[\top/x] \wedge \Phi[\top/x]) \vee (\alpha[\bot/x] \wedge \Phi[\bot/x])) \\ &\quad \wedge ((\beta[\top/x] \wedge \Phi[\top/x]) \vee (\beta[\bot/x] \wedge \Phi[\bot/x])) \\ &\equiv (\exists x.(\alpha \wedge \Phi)) \wedge (\exists x.(\beta \wedge \Phi))\end{aligned}$$

So considering every base formula separately gives *correct* information about successor belief states. But not all information is obtained this way, as the converse of the logical consequence in Theorem 1 does not hold.

Example 2. Consider ϕ_1 that represents the set $\{s_1\}$ and ϕ_2 that represents the set $\{s_2\}$, and Φ that represents the transition relation $\{(s_1, s_3), (s_2, s_3)\}$. Since $\phi_1 \wedge \phi_2 \equiv \bot$, also $\mathrm{succ}_\Phi(\phi_1 \wedge \phi_2) \equiv \bot$. But $\mathrm{succ}_\Phi(\phi_1) \wedge \mathrm{succ}_\Phi(\phi_2)$ represents s_3.

A relation R is *injective* if for all z, whenever xRz and yRz, $x = y$. This means that an action and a successor state determine the predecessor state uniquely. For injective relations the image of conjunction coincides with the conjunction of the images.

Lemma 1. *Let Φ be a transition relation formula that represents an injective relation. Then $succ_\Phi(\phi \wedge \phi') \equiv succ_\Phi(\phi) \wedge succ_\Phi(\phi')$,*

Many actions in standard benchmark problems for classical planning are injective as required in Lemma 1, when restricted to the part of the state space reachable from the initial states, but partially observable problems typically are not. Hence an important problem is the identification of actions and formulas ϕ_1 and ϕ_2 that satisfy $\mathrm{succ}_\Phi(\phi \wedge \phi') \equiv \mathrm{succ}_\Phi(\phi) \wedge \mathrm{succ}_\Phi(\phi')$ even without the action being injective. This is critical for being able to analyze problems efficiently without having to look at all possible combinations of component beliefs.

Nevertheless, in many interesting problems, reasoning about actions is possible even without exhaustive analysis of all combinations of component beliefs.

Example 3. Consider Sorting Networks with three inputs. The shortest plan does compare&swaps for the input pairs $(1,3)$, $(1,2)$ and $(2,3)$, generating the belief states $(x_3 \rightarrow x_1)$, $(x_2 \rightarrow x_1) \wedge (x_3 \rightarrow x_1)$, and $(x_2 \rightarrow x_1) \wedge (x_3 \rightarrow x_1) \wedge (x_3 \rightarrow x_2)$.

Example 4. Consider Sorting Networks with four inputs. The shortest plan consists of compare&swap operations for the input pairs $(1,3), (2,4), (1,2), (3,4)$ and $(2,3)$. The first two actions produce the belief state $(x_3 \rightarrow x_1) \wedge (x_4 \rightarrow x_2)$. After that, the third action, swapping 1 and 2, turns the belief state to

$$(x_3 \rightarrow x_1) \wedge (x_2 \rightarrow x_1) \wedge (x_4 \rightarrow x_1) \wedge ((x_3 \wedge x_4) \rightarrow x_2)$$

that contains $((x_3 \wedge x_4) \rightarrow x_2)$. This implication is only obtained as the image of $(x_3 \rightarrow x_1) \wedge (x_4 \rightarrow x_2)$, and is not obtained from any one $x_i \rightarrow x_j$ alone.

More generally, for the sorting network problems, swap actions create new beliefs from complex combinations of prior beliefs.

While relatively good plans can be found with these implications $x_i \rightarrow x_j$ as the beliefs in a conjunctive belief representation, also for larger numbers of inputs, the smallest plans require increasingly complex beliefs. For example, the sorting network with 20 inputs that has the smallest number of layers has $(x_3 \wedge x_7 \wedge x_{10} \wedge x_{11}) \rightarrow (x_8 \vee x_9 \vee x_{12})$ as one of the intermediate beliefs.

Below we list the maximum clause lengths encountered in the best known (smallest number of layers) sorting networks for up to 20 inputs. Here n is the number of inputs and s is the length of the longest clause in the CNF beliefs.

n	s	n	s	n	s	n	s	n	s	n	s	n	s	n	s	n	s
3	2	4	3	5	3	6	3	7	3	8	3	9	4	10	4	11	4
12	5	13	6	14	5	15	6	16	5	17	5	18	6	19	6	20	7

Many other problems have a far simpler belief space, and it is often enough to look at the components of beliefs one at a time.

Example 5. Consider a rectangular grid, where a robot's position in the East-West direction is indicated by state variables $x_0, \ldots, x_9$, and the location in the North-South direction by state variables $y_0, \ldots, y_9$. The "move north" action is

$$\bigwedge_{i=0}^{9}(x'_i \leftrightarrow x_i) \wedge \bigwedge_{j=1}^{8}(y'_j \leftrightarrow y_{j-1}) \wedge (y'_9 \leftrightarrow (y_9 \vee y_8)) \wedge \neg y'_0$$

with movement at the north wall having no effect. Moves to the other three cardinal directions are analogous. There is a unique initial location for the robot.

$$(\bigvee_{i=0}^{9} x_i) \wedge (\bigvee_{i=0}^{9} y_i) \wedge \bigwedge_{i=0}^{8}\bigwedge_{j=i+1}^{9} \neg(x_i \wedge x_j) \bigwedge_{i=0}^{8}\bigwedge_{j=i+1}^{9} \neg(y_i \wedge y_j)$$

The beliefs in this problem are the conjuncts of the formula for the initial belief state, as well as all sub-intervals of $[0, 9]$ for positions on both X and the Y axes.

$$\{\bigvee_{i=j}^{k} x_i | 0 \leq j \leq k \leq 9\} \cup \{\bigvee_{i=j}^{k} y_i | 0 \leq j \leq k \leq 9\}$$

Reasoning about location can be done independently for X and Y coordinates, one formula at a time.

4 Algorithm for Identifying a Base

We give an algorithm for finding a base T for a conformant planning problem.

1. We start from the initial state description $\phi_1 \wedge \cdots \wedge \phi_n$, where the minimal conjuncts $\phi_1, \ldots, \phi_n$ are taken to be the tentative base T.

2. Pick some action a and a consistent subset $P \subseteq T$, and do the following.
   (a) Compute $\sigma = \mathrm{succ}_{\Phi_a}(\bigwedge_{\phi \in P} \phi)$.
   (b) Make the minimal conjuncts of σ explicit as $\sigma = \psi_1 \wedge \cdots \wedge \psi_m$.
   (c) Add $\psi_1, \cdots, \psi_m$ to T, while eliminating duplicates modulo equivalence.
3. Repeat the previous step until T does not change.

Here we need the *existential abstraction* operation and the *logical equivalence* test. In our implementation – which is discussed later – we have used Ordered Binary Decision Diagrams (OBDD) [4]. Other representations of Boolean functions could be used instead, with different trade-offs between efficiency and size.

The mapping of images σ to conjuncts determines the formulas in the base. The most general solution is to take the conjuncts to be all the *prime implicates* of σ, that is, the minimal clauses logically entailed by σ, but as we will see, something far simpler often works very well in practice.

The number of subsets P of T is exponential in $|T|$, and therefore this computation is in general not feasible. This is exactly as expected, as not all parts of a reduction from the EXPSPACE-complete conformant planning [8, 12] to the PSPACE-complete classical planning [6] can be polynomial time.

However, it turns out that it is often sufficient to limit to subsets $P \subseteq T$ of small cardinality. Often $|P| \leq 1$ is sufficient, so only the empty set and all 1-element subsets of T need to be considered.

The next theorem shows that the general form of our base construction is sufficient to identify a conjunctive decompositions of the belief space in the sense that no matter which action sequence is taken starting in the initial belief state, any reachable belief state can be represented as a conjunction of some subset of formulas in the base.

Theorem 2. *For a formula I and a sequence $\Phi_1, \ldots, \Phi_m$ of transition relation formulas, $\mathrm{succ}_{\Phi_1,\ldots,\Phi_m}(I) \equiv \bigwedge B$ for some $B \subseteq T$.*

Proof. The proof is by induction on the length of the action sequence m, with the claim of the theorem as the induction hypothesis.

Base case $m = 0$: The initial value of T is the conjuncts of the initial state formula, exactly corresponding to the only belief state reachable by not taking any action at all. Hence $\mathrm{succ}_{\epsilon}(I)$ for the empty sequence ϵ is representable in terms of T.

Inductive case $i \geq 1$: By the induction hypothesis, $\mathrm{succ}_{\Phi_1;\cdots;\Phi_{i-1}}(I) \equiv \bigwedge B$ for some $B \subseteq T$. The algorithm goes through all actions, including one with transition relation formula Φ_i, and through all subsets of T, including B. Hence it will compute $\sigma = \mathrm{succ}_{\bigwedge B}(\Phi_i)$, and the conjuncts of σ, however they are identified, will be included in T. Hence $\mathrm{succ}_{\Phi_1;\cdots;\Phi_i}(I) \equiv \bigwedge B$ for some $B \subseteq T$.

Interestingly, the proof shows that – from the completeness point of view – it is not important how the formula σ is split into conjuncts at step (2b) of the algorithm for finding a base. Essentially, splitting σ to a single conjunct as $\sigma = \phi_1$ would simply mean that we enumerate all possible beliefs (formulas) reachable from the initial belief state. In this light, Theorem 2 is not surprising.

The important thing in the algorithm – from the scalability point of view – is the splitting of σ to small conjuncts, so that not every belief state needs to be generated explicitly. Instead, the space of all belief states is conjunctively decomposed to smaller formulas, contained in T, so that any belief state can be represented by some subset $B \subseteq T$. The base T may therefore be exponentially smaller than the set of all belief states reachable from the initial belief state.

Finally, we point out that the algorithm does in general not determine reachability of belief states exactly: actions are considered in belief states (conjunctions of subsets of T) that are not actually reachable from the initial belief state. Hence T may contain formulas that could never be true in a reachable state. This is an obvious source of inefficiency. We comment more on this in Section 6.1.

5 Reduction from Conformant to Classical Planning

We will represent the conformant planning problem as a full-information classical planning problem, with each formula $\phi \in T$ represented by a single state variable x_ϕ. When solving the full-information planning problem, a state s represents the belief state that corresponds to the formula $\bigwedge\{\phi \in T | s \models x_\phi\}$. The set of state variables in the classical planning problem is $X_T = \{x_\phi | \phi \in T\}$.

Additionally, we define the actions, the initial state, and the goal formula.

For every action a of the original (conformant) problem, we define a new action a' that changes the belief state encoded with the state variables in X_T in a way that corresponds to how a changes the belief state.

5.1 Effects

We define $\text{causes}_a^{\phi_1,\dots\phi_n}(\phi)$ as holding if ϕ is one of the conjuncts in $\text{succ}_{\Phi_a}(\phi_1 \wedge \cdots \wedge \phi_n)$. We define $\text{minCauses}_a^{\phi_1,\dots\phi_n}(\phi)$ as holding if

- ϕ is one of the conjuncts in $\text{succ}_{\Phi_a}(\phi_1 \wedge \cdots \wedge \phi_n)$, and
- ϕ is not a conjunct of $\text{succ}_{\Phi_a}(\phi_{i_1} \wedge \cdots \wedge \phi_{i_j})$ for any $\{i_1, \dots, i_j\} \subset \{1, \dots, n\}$.

We iterate over all subsets $\{\phi_1, \dots, \phi_n\}$ of T and all $\phi \in T$, and add the following effects to the action we are constructing.

- If $\text{minCauses}_a^{\phi_1,\dots,\phi_n}(\phi)$ and $\phi_i \not\models \phi$ for all $i \in \{1, \dots, n\}$ then a' has effect $x_{\phi_1} \wedge \cdots \wedge x_{\phi_n} \rhd x_\phi$.
- If $\text{not causes}_a^{\phi}(\phi)$, then a' has effect $(x_\phi \wedge C) \rhd \neg x_\phi$ where C is the conjunction of all $\neg(\phi_{i_1} \wedge \cdots \wedge \phi_{i_k})$ such that $\text{minCauses}_a^{\phi_{i_1},\dots,\phi_{i_k}}(\phi)$.[1]

Again, this computation takes exponential time in the cardinality of T. And, similarly to the computation of a base, this computation can be limited to "small" subsets S of T. For the sorting network problems, for example, classical planning instances that have non-optimal solutions can be produced with $|S| \leq 2$, but for higher number of inputs larger sets S are needed to find optimal solutions.

[1] The left-hand side of this conditional effect can be simplified by replacing all occurrences of ϕ by $\top$, as the effect does something only if ϕ is true when the action is taken. This modification is is needed to maximize Graphplan-style [3] parallelism.

5.2 Preconditions

An action can be taken only if its precondition must be true. For this we need all *minimal consistent subsets* of T from which the precondition follows.

Definition 4. *A set $D \subseteq T$ is* relevant *for a formula χ, if D is consistent, $D \models \chi$, and there is no D' such that $D \subset D'$, D' is consistent, $D' \models \chi$.*

Let $a = \langle \chi, E \rangle$ be an action. Let P be all the sets $D \subseteq T$ relevant for χ. Now the precondition of a' is $\bigvee_{p \in P}(\bigwedge\{x_\phi | \phi \in p\})$.

Clearly, for actions a with the trivial precondition $\top$, the precondition of a' is similarly $\top$. More generally, there may be an exponential number of relevant subsets $D \subseteq T$, so there is no guarantee that this computation is always feasible.

Relevant subsets of T for χ are closely related to *minimal unsatisfiable sets* (MUS) [10, 1]: a relevant subset for ϕ is a MUS of $T \cup \{\neg\phi\}$ that contains $\neg\phi$.

Lemma 2. *Assume $\neg\phi \notin P$. Then $P \subseteq T$ is a relevant set for ϕ if and only if $P \cup \{\neg\phi\}$ is a minimal unsatisfiable set of $T \cup \{\neg\phi\}$.*

Proof. Since $P \cup \{\neg\phi\}$ is unsatisfiable, $P \models \phi$. Since $P \cup \{\neg\phi\}$ is minimal unsatisfiable, we have $P_0 \cup \{\neg\phi\}$ satisfiable and hence $P_0 \not\models \phi$ for all $P_0 \subset P$. Since $P \cup \{\neg\phi\}$ is minimal unsatisfiable, P is satisfiable. Hence by the definition of relevance, P is a relevant set for ϕ.

The computation of minimal inconsistent subsets is expensive, and as before, can be limited to "small" subsets.

5.3 Goals

The goal formula is computed similarly to the preconditions as a disjunction of conjunctions of minimal consistent subsets of T that logically entail the original goal formula G. For goals of the form $G = \gamma_1 \wedge \cdots \wedge \gamma_n$ we can determine the entailing subsets of T separately for each γ_i.

6 Implementation

We have implemented all steps for translating conformant planning to classical planning. The logical operations could be implemented with any class of formulas that can represent any Boolean function, but we chose to use ordered binary decision diagrams OBDDs for three reasons: simplicity, logical simplifications provided by OBDDs canonicity, and constant time equivalence tests.

In our reduction from conformant planning to classical planning there are the three exponential components, which we have approximated by not going through all subsets of formulas, but instead only all "small" subsets of cardinality $\leq n$ for some small n. These three parameters, which limits the cardinalities of these subsets, are used in

1. identifying the base (Section 4),
2. synthesizing the effects of actions (Section 5.1), and
3. synthesizing the formulas for the preconditions and the goal (Section 5.2).

When we use the values 1, 2 and 1 for these three parameters, respectively, we indicate this as the configuration $(1, 2, 1)$.

We first experimented with Sorting Networks, due to their difficulty for existing planners. They are parameterized by the number i of inputs, have i state variables, and yield a base of quadratic size with configuration $(1, 2, 1)$, and a base of cubic size with $(2, 2, 1)$. With $(1, 2, 1)$ we can find non-optimal and not very good solutions until 20 inputs, and better non-optimal solutions not quite as far. This problem is not solvable with the $(1, 1, 1)$ configuration.

Many other benchmark problems are harder than sorting networks in terms of having a far higher number of state variables. However, in many cases this is balanced by them being solvable (even optimally) with the easiest $(1, 1, 1)$ configuration. The number of base formulas is in many cases several hundreds or thousands, and brute force generation of the base in configuration $(2, 1, 1)$ as well as synthesis of actions in configuration $(1, 2, 1)$ become infeasible.

An important part of future work is to utilize structural properties of the problem instances to perform these computations far more efficiently, without having to blindly go through all or most N-element subsets of the base.

6.1 Use of Invariants to Reduce the Base

The use of *invariants*, formulas that hold in all reachable states of a transition system, is common in planning methods that work with partial state representation. In the algorithm in Section 4, invariants help ignore those formulas that are never true in any reachable state, or that are true in all reachable states. This leads to a smaller base. We use a basic algorithm for finding 2-literal invariant clauses [13]. For instance, the formulas $\neg(x_i \wedge x_j)$ in Example 5 are part of every belief state, and therefore the possibility of them being false can be ignored.

7 Experiments

We have done experiments with a collection of standard benchmark problems. Of special interest is Sorting Networks, with complex belief space and complex interactions between beliefs. Results are given in Table 1. We list runtimes, the numbers of actions as well as the number of state variables in the original conformant and in the classical instances. The latter number equals the number of formulas in the base. Palacios & Geffner's [11] T_0 planner uses the K_1 translation by default, but in cases where it does not yield any solutions, we have switched to the K_0 translation, as indicated in the table. We have used the FF [9] and Madagascar [14] planners to solve our PDDL instances. Madagascar constructs *parallel* plans, and an optimality criterion for sorting networks is the number of *layers* of the sorting network, with each layer containing one or more

Table 1. Results for SORTNET. C: *Configuration*; X: *variables in the problem*; Xc: *variables in the translated problem*; A: *actions*; MpC: *Madagascar runtime* ; FF: *FF runtime* ; PG: T_0 *runtime with FF*; OOM: *out of memory*

Instance	C	X	Xc	A	MpC	FF	PG
sort4	(2,2,1)	4	17	6	0.00	0.00	0.07
sort5	(2,2,1)	5	68	10	0.12	0.02	0.53
sort6	(2,2,1)	6	239	15	7.64	2.57	5.73
sort7	(2,2,1)	7	790	21	3006.24	OOM	21.26
sort6	(1,2,1)	6	15	15	0.01	0.00	5.73
sort7	(1,2,1)	7	21	21	0.02	0.00	21.26
sort8	(1,2,1)	8	28	28	0.06	0.00	0.21(K_0)
sort9	(1,2,1)	9	36	36	0.12	0.00	0.38(K_0)
sort10	(1,2,1)	10	45	45	0.26	0.00	0.73(K_0)
sort15	(1,2,1)	15	105	105	2.47	0.04	15.92(K_0)
sort18	(1,2,1)	18	153	153	7.94	0.17	121.71(K_0)

compare&swap actions so that each input is only sorted by at most one of the actions. However, it turned out that although pairs of compare&swaps like on $(1,3)$ and on $(2,4)$ do not interfere when the state variables are the input values, the actions after our translation do interfere, as they impact and depend on the same beliefs $x_i \rightarrow x_j$, and hence Madagascar cannot benefit from the parallelism.

All sorting network problems are solvable with the configuration $(1,2,1)$, by looking at the joint images of *pairs* of beliefs of the form $x_i \rightarrow x_j$, but this is insufficient to find optimal solutions (see Example 4). The generation of the PDDL in these cases is fast, less than 10 seconds even for large instances. On these problems we are quite competitive with T_0. As pointed out earlier, optimal solutions e.g. with 20 inputs seem to require the configuration $(7,7,1)$, which leads to quite large PDDL representations.

With the configuration $(2,2,1)$ also formulas $x_i \wedge x_j \rightarrow x_k$ are included in the base, and this allows (in principle) optimal solutions to be found until at least 8 inputs, as discussed earlier. While our experiments did not use optimal planners, the configuration $(2,2,1)$ still allows us to find better sub-optimal plans than what can be found with configuration $(1,2,1)$. But, as the number of formulas in the base is cubic in the number of inputs, and not quadratic, the PDDL translation is far bigger, and the planners do not scale up as far as with the $(1,2,1)$ configuration. Also, the runtimes for generating the PDDL grows very quickly with the increasing number of inputs.

For the rest of the benchmark problems the situation is quite different, as the configuration $(1,1,1)$ is always sufficient. The scalability of our approach is only limited by the size of the base, as we only have to look at each formula in the base in isolation at each stage of the translation process. Data on a collection of standard benchmarks similar to that used by Palacios and Geffner [11] are given in Tables 2 and 3. Our runtimes in comparison to Palacios & Geffner's T_0 [11] are in some cases comparable, and in many cases clearly behind, for example in

Table 2. Runtimes of a number of benchmark problems

Instance	X	Xc	A	MpC	FF	PG
corners-square-p40	80	1722	4	1.69	0.30	0.53
corners-square-p84	168	7310	4	51.67	5.60	11.71
corners-square-p100	200	10302	4	115.85	11.32	25.14
corners-square-p120	240	14762	4	283.96	27.06	57.05
corners-square-p140	280	29375	4	1492.39	159.50	90.33
corners-square-p200	400	40602	4	3285.06	380.47	485.10
corners-cube-p27	81	1218	6	0.70	0.13	3.80
corners-cube-p52	156	4293	6	13.11	2.31	147.62
corners-cube-p55	165	4788	6	17.37	2.71	226.28
corners-cube-p60	180	5673	6	27.00	4.82	366.48
corners-cube-p75	225	8778	6	75.20	15.01	1463.35
square-center-p24	48	1200	4	0.53	0.11	0.15
square-center-p92	184	17112	4	320.81	80.09	8.99
cube-center-p19	57	1140	6	0.47	0.09	0.15
cube-center-p63	189	12096	6	137.50	25.35	6.33
cube-center-p67	201	13668	6	193.58	38.12	8.01

ring, *safe* and *blocksword*. For the latter two producing the PDDL is slow due to high number of actions and a large base. Notice that the listed runtimes do not include the generation of the PDDL. This time is often substantial. For example, bomb100-100 took 636.9 seconds (10100 actions), bomb20-20 took 0.65 seconds (420 actions), while Sortnet with 9 inputs and configuration (1,2,1) took 0.24 seconds (36 actions). The time is dominated by image computation, which we believe can be substantially sped up, especially when actions are simple. Planner by To et al. [18] is often comparable to that of Palacios and Geffner, but in many cases scale up further in the benchmark series.

8 Related Work

Baral, Kreinovich and Trejo [2] investigate 3-valued belief state representations, in which state variables are *true*, *false*, or *unknown*. This form of incompleteness is equivalent to representing belief states as sets (conjunctions) of literals. Baral et al. demonstrate how many types of interesting problems are efficiently solvable with this type of representation, and that the complexity is substantially reduced, down to PSPACE, which is the same as with classical planning.

Palacios and Geffner [11] propose an approach to conformant planning that is based on dependencies of state variable values on the initial values of some other state variables. Their literals KL/t could be viewed as implications $t \rightarrow KL$, and the *merge*s, inferring KL from $\bigwedge_{t \in T} KL/t \rightarrow KL$ as, as a form of logical deduction, analysis by cases. Their planner can in general solve more of the standard benchmark problems on conformant planning than ours, but our planner outperforms it with the sorting network problems, because Palacios and Geffner's

Table 3. Results from a number of benchmark problems

Instance	X	Xc	A	MpC	FF	PG
comm-p10	69	314	59	0.11	0.00	0.05
comm-p15	99	454	84	0.26	0.00	0.05
comm-p20	245	1130	208	11.94	0.02	0.16
bomb20-20	40	60	420	0.00	0.00	0.05
bomb100-5	105	110	505	0.06	0.00	0.21
bomb100-60	160	220	6060	0.12	0.05	1.04
bomb100-100	200	300	10100	0.24	1.39	2.40
coins-p10	34	200	40	0.41	0.00	0.1
coins-p12	76	1866	88	10340.96	0.09	0.1
coins-p16	86	2020	110	TO	0.19	0.09
coins-p18	86	2020	110	TO	0.18	0.06
coins-p20	86	2020	110	TO	0.17	0.07
uts-p1	5	41	4	0.00	0.00	0.01
uts-p2	9	892	16	2.65	0.10	0.01
uts-p3	13	11354	36	5004.71	104.45	0.03
logistics-p2-2-2	20	48	30	0.00	0.00	0.02
logistics-p4-3-3	69	201	156	0.03	0.00	0.03
uts-l01	5	41	4	0.00	0.00	0.01
uts-l02	9	882	10	1.52	0.08	0.02
safe-p5	6	78	5	0.00	0.00	0.00
safe-p10	11	2102	10	29.09	0.70	0.01

method leads to exponentially large classical planning problems in this case. Further, Palacios&Geffner limit to deterministic actions, whereas our work covers arbitrary actions, including non-deterministic ones.

To et al. [15] used DNF as a belief state representation, then turned to prime implicates [16] and CNF [17], demonstrating different trade-offs. In these works, belief states are sets of formulas, not valuations of propositional variables like in our work, and no reduction to the classical planning problem is considered.

9 Conclusion

We have investigated the representation of belief states as vectors of truth values. This representation attempts to lower the complexity of belief space planning by replacing the combinatorially far harder notion of formulas by much easier states. We have shown our methods to be useful even when strict limits are imposed on how thoroughly an approximate belief space representation is created. These limits risk losing completeness. An important topic for further research is obtaining completeness guarantees even under these size limits. Future work also includes generalizing the results to *partial observability*. Observations help increase the accuracy of the beliefs. In this case we would expect to be able to similarly often achieve an exponential complexity reduction.

References

1. Bailey, J., Stuckey, P.J.: Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In: Practical Aspects of Declarative Languages; 7th International Symposium, PADL 2005, Long Beach, CA, USA, January 10-11, 2005. Proceedings. pp. 174–186. No. 3360 in Lecture Notes in Computer Science, Springer-Verlag (2005)
2. Baral, C., Kreinovich, V., Trejo, R.: Computational complexity of planning and approximate planning in the presence of incompleteness. Artificial Intelligence **122**(1), 241–267 (2000)
3. Blum, A.L., Furst, M.L.: Fast planning through planning graph analysis. Artificial Intelligence **90**(1-2), 281–300 (1997)
4. Bryant, R.E.: Symbolic Boolean manipulation with ordered binary decision diagrams. ACM Computing Surveys **24**(3), 293–318 (1992)
5. Burch, J.R., Clarke, E.M., Long, D.E., MacMillan, K.L., Dill, D.L.: Symbolic model checking for sequential circuit verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **13**(4), 401–424 (1994)
6. Bylander, T.: The computational complexity of propositional STRIPS planning. Artificial Intelligence **69**(1-2), 165–204 (1994)
7. Geffner, T., Geffner, H.: Compact policies for non-deterministic fully observable planning as sat. In: ICAPS 2018. Proceedings of the Twenty-Eighth International Conference on Automated Planning and Scheduling. pp. 88–96. AAAI Press (2018)
8. Haslum, P., Jonsson, P.: Some results on the complexity of planning with incomplete information. In: Recent Advances in AI Planning. 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999. Proceedings. pp. 308–318. No. 1809 in Lecture Notes in Artificial Intelligence, Springer-Verlag (2000)
9. Hoffmann, J., Nebel, B.: The FF planning system: fast plan generation through heuristic search. Journal of Artificial Intelligence Research **14**, 253–302 (2001)
10. Liffiton, M.H., Sakallah, K.A.: On finding all minimally unsatisfiable subformulas. In: Theory and Applications of Satisfiability Testing; 8th International Conference, SAT 2005, St Andrews, UK, June 19-23, 2005. Proceedings. pp. 173–186. No. 3569 in Lecture Notes in Computer Science, Springer-Verlag (2005)
11. Palacios, H., Geffner, H.: Compiling uncertainty away in conformant planning problems with bounded width. Journal of Artificial Intelligence Research **35**, 623–675 (2009)
12. Rintanen, J.: Complexity of planning with partial observability. In: ICAPS 2004. Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling. pp. 345–354. AAAI Press (2004)
13. Rintanen, J.: Regression for classical and nondeterministic planning. In: ECAI 2008. Proceedings of the 18th European Conference on Artificial Intelligence. pp. 568–571. IOS Press (2008)
14. Rintanen, J.: Planning as satisfiability: heuristics. Artificial Intelligence **193**, 45–86 (2012)
15. To, S., Pontelli, E., Son, T.: A conformant planner with explicit disjunctive representation of belief states. In: Proceedings of the 19th International Conference on Automated Planning and Scheduling. pp. 305–312. AAAI Press (2009)
16. To, S., Son, T., Pontelli, E.: On the use of prime implicates in conformant planning. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1205–1210. AAAI Press (2010)

17. To, S.T., Son, T.C., Pontelli, E.: A new approach to conformant planning using CNF. In: Proceedings of the 20th International Conference on Automated Planning and Scheduling. pp. 169–176. AAAI Press (2010)
18. To, S.T., Son, T.C., Pontelli, E.: A generic approach to planning in the presence of incomplete information: Theory and implementation. Artificial Intelligence **227**, 1–51 (2015)
19. Tu, P.H., Son, T.C., Baral, C.: Reasoning and planning with sensing actions, incomplete information, and static causal laws using answer set programming. Theory and Practice of Logic Programming **7**, 1–74 (2006)