Mapping Conformant Planning into SAT through Compilation and Projection

Héctor Palacios[1] and Héctor Geffner[2]

[1] Universitat Pompeu Fabra
Paseo de Circunvalación, 8.
Barcelona, Spain
hector.palacios@upf.edu
[2] ICREA & Universitat Pompeu Fabra
Paseo de Circunvalación, 8.
Barcelona, Spain
hector.geffner@upf.edu

Abstract. Conformant planning is a variation of classical AI planning where the initial state is partially known and actions can have non-deterministic effects. While a classical plan must achieve the goal from a given initial state using deterministic actions, a conformant plan must achieve the goal in the presence of uncertainty in the initial state and action effects. Conformant planning is computationally harder than classical planning, and unlike classical planning, cannot be reduced polynomially to SAT (unless P = NP). Current SAT approaches to conformant planning, such as those considered by Giunchiglia and colleagues, thus follow a generate-and-test strategy: the models of the theory are generated one by one using a SAT solver (assuming a given planning horizon), and from each such model, a candidate conformant plan is extracted and tested for validity using another SAT call. This works well when the theory has few candidate plans and models, but otherwise is too inefficient. In this paper we propose a different use of a SAT engine for computing conformant plans where the existence of conformant plans and their extraction is carried out by means of *a single SAT call over a transformed theory.* This transformed theory is obtained by *projecting* the original theory over the action variables. This operation, while intractable, can be done efficiently provided that the original theory is compiled into DNNF (Darwiche 2000), a form akin to OBDDs (Bryant 1992). The experiments that are reported show that the resulting COMPILE-PROJECT-SAT planner is competitive with state-of-the-art optimal conformant planners and improves upon a planner recently reported at ICAPS-05. The techniques apply in direct fashion to the solution of QBFs of the form $\exists x \, \forall y \, \exists z \; \phi$ which are in correspondence to the problem of verifying the existence of conformant plans for a given horizon.

1 Introduction

Conformant planning is a variation of classical AI planning where the initial state is partially known and actions can have non-deterministic effects [1]. While a

classical plan must achieve the goal from a given initial state using deterministic actions, a conformant plan must achieve the goal in the presence of uncertainty in the initial state and action effects. Conformant planning is computationally harder than classical planning, and unlike classical planning, cannot be reduced polynomially to SAT [2]. Current SAT approaches to conformant planning thus follow a generate-and-test strategy [3, 4]: the models of the theory are generated one by one using a SAT solver (assuming a given planning horizon), and from each such model, a candidate conformant plan is extracted and tested for validity using another SAT call. This works well when the theory has few candidate plans and models, but otherwise is too inefficient. In this paper we propose a different use of a SAT engine for computing conformant plans where the existence of conformant plans and their extraction is carried out by means of *a single SAT call over a transformed theory.* This transformed theory is obtained by *projecting* the original theory over the action variables. Projection is the dual of variable elimination (also called forgetting or existential quantification; [5]): the projection of a formula over a subset of its variables is the strongest formula over those variables; e.g., the projection of $((x \wedge y) \vee z)$ over $\{x, z\}$ is $x \vee z$. While projection is intractable, it can be done efficiently provided that the theory is in a certain canonical form such as *Decomposable Negated Normal Form* or DNNF [6], a form akin to OBDDs [7].

Our scheme for planning is thus based on the following three steps: the planning theory in CNF is first compiled into d-DNNF (a suitable subset of DNNF), the compiled theory is then transformed into a new theory over the action variables only, and finally the conformant plan, if there is one, is obtained from this theory by a single invocation of a SAT engine. The experiments that are reported show that this COMPILE-PROJECT-SAT planner is competitive with state-of-the-art optimal conformant planners and improves upon a planner recently reported at ICAPS-05 [8]. The proposed techniques seem to apply also in direct fashion to the solution of *QBFs* of the form $\exists x \forall y \exists z\ \phi$ which are in correspondence to the problem of verifying the existence of conformant plans for a given horizon [9], and are related to the 'resolve' and 'expand' methods for solving QBFs recently proposed in [10].

Two recent optimal conformant planners are Rintanen's [11] and Palacios *et al.*'s [8]. The first performs heuristic search in belief space [12] with a powerful, admissible heuristic obtained by precomputing distances over belief states with at most two states. The second is a branch-and-prune planner that prunes partial plans that cannot comply with some possible initial state. This is achieved by performing model-count operations in linear-time over the d–DNNF representation of the theory [13]. Both schemes assume that all uncertainty lies in the initial situation and that *all actions are deterministic.* In this work, we maintain this simplification which is not critical as non-deterministic effects can be eliminated by adding a polynomial number of hidden fluents. An appealing feature of the new conformant planning scheme is that it is based on the *two off-the-shelf components:* a d–DNNF compiler and a SAT solver. For the experiments

we used the d-DNNF compiler reported in [14] and the SAT solvers reported in [15, 16].

The paper is organized as follows. First we define the conformant planning problem and its formulation in propositional logic. Then we study how to obtain models corresponding to conformant plans. We look then at projection as a logical operation, and at d–DNNFs as a compiled normal form that supports projection in linear time. Finally, we present the complete conformant planning scheme, the experimental results, and some conclusions.

2 Planning and Propositional Logic

Classical planning consists of finding a sequence of actions that transforms a known initial state into a goal given a description of states and actions in terms of a set of variables. Conformant planning is a variation of classical planning where the initial state is partially known and actions can have non-deterministic effects. A conformant plan must achieve the goal for *any* possible initial state and transition.

We consider a language for describing conformant planning problems P given by tuples of the form $\langle F, O, I, G\rangle$ where F stands for the fluent symbols f in the problem, O stands for a set of actions a, and I and G are sets of clauses over the fluents in F encoding the initial and goal situations. In addition, every action a has a precondition $pre(a)$, given by a set of fluent literals, and a list of conditional effects $\mathrm{cond}^k(a) \rightarrow \mathrm{effect}^k(a)$, $k = 1, \ldots, n_a$, where $\mathrm{cond}^k(a)$ and $\mathrm{effect}^k(a)$ are conjunctions of fluent literals. As mentioned above, we assume that actions are deterministic and hence that all uncertainty lies in the initial situation.[3]

In the SAT approach to classical planning [18], the problem of finding a plan for P within N time steps is mapped into a model finding problem over a suitable propositional encoding. In this encoding there are variables x_i for fluents and actions x where i is a temporal index in $[0, N]$ (no action variables x_i are created for $i = N$ though). For a formula B, B_i refers to the formula obtained by replacing each variable x in B by its time-stamped counterpart x_i. The encoding $T(P)$ of a *conformant planning problem* $P = \langle F, I, O, G\rangle$ is obtained as a slight variation of the propositional encoding used for classical planning. For an horizon N, the CNF theory $T(P)$ is given by the following clauses:

1. **Init:** a clause C_0 for each init clause $C \in I$.
2. **Goal:** a clause C_N for each goal clause $C \in G$.
3. **Actions:** For $i = 0, 1, \ldots, N-1$ and $a \in O$:

$$\begin{aligned} a_i &\supset \mathrm{pre}(a)_i && && \text{(preconditions)} \\ \mathrm{cond}^k(a)_i \wedge a_i &\supset \mathrm{effect}^k(a)_{i+1}, && k = 1, \ldots, k_a && \text{(effects)} \end{aligned}$$

[3] As mentioned in [17], uncertainty in action effects can be translated into uncertainty in the initial state by introducing a polynomial number of extra fluents in the problem; e.g. if actions have at most two effects, then it is sufficient to add a 'hidden' fluent for each action and time point, making the action effects deterministic but conditional on the status of the hidden fluents.

4. **Frame:** for $i = 0, 1, \ldots, N-1$, each fluent literal

$$l_i \ \wedge \ \bigwedge_{\mathrm{cond}^k(a)} \neg[\mathrm{cond}^k(a)_i \ \wedge \ a_i] \quad \supset \quad l_{i+1}$$

   where the conjunction ranges over the conditions $\mathrm{cond}^k(a)$ associated with effects $\mathrm{effect}^k(a)$ that support the complement of l.
5. **Exclusion:** $\neg a_i \vee \neg a'_i$ for $i = 0, \ldots, N-1$ if a and a' are incompatible.

The meaning of *Init*, *Goal*, and *Actions* is straightforward. *Frame* expresses the persistence of fluents in the absence of actions that may affect them. Finally *Exclusion* forbids the concurrent execution of actions that are deemed incompatible. For the serial setting, we regard every pair of different actions as incompatible, while in the parallel setting, we regard as incompatible pairs of different actions that interfere with each other (we use a simple notion of interference, simpler actually than the one used in Graphplan [19]: two actions interfere when they affect a common variable; this definition is not critical though).

A conformant planner is optimal when it finds conformant plans for the minimum possible horizon N (makespan). This is achieved by setting the horizon N to 0, and increasing it, one unit at a time, until a plan is found. As in Graphplan, lower bounds could be used to start with higher horizons, but we will not pursue this idea in this paper.

3 Conformant Planning and Models

In classical planning the relation between a problem P and its propositional encoding $T(P)$ is such that the models of $T(P)$ are in one-to-one correspondence with the plans that solve P (for the given horizon). In conformant planning, this correspondence no longer holds: the models of $T(P)$ encode 'optimistic plans', plans that work for *some* initial states and transitions but may fail to work for others, and hence are not conformant. We will see however that it is possible to transform the theory $T(P)$ so that the models of the resulting theory are in correspondence with the conformant plans for P.

Let $Plan$ denote a maximal consistent set of *action* literals a_i, let $Init$ denote the fragment of $T(P)$ encoding the initial situation, and let s_0 refer to a maximal consistent set of fluent literals f_0 compatible with $Init$ (i.e., the possible initial states which we denote as $s_0 \in Init$). Then for a *classical planning* problem P, $Plan$ is a solution if and only if

$$T(P) \ + \ Plan \qquad \text{is satisfiable} \tag{1}$$

For a *conformant problem* P (with deterministic actions only), on the other hand, $Plan$ is a solution if and only if

$$\forall \ s_0 \in Init: \ \ T(P) \ + \ Plan \ + \ s_0 \qquad \text{is satisfiable} \tag{2}$$

In other words, in the conformant setting $Plan$ must work for all possible initial states.

In order to find a *Plan* that complies with (1) it is enough to find a model of $T(P)$, and then set *Plan* to the set of action literals that are true in the model. On the other hand, for finding a *Plan* that complies with (2) this is not enough. As we will show, however, it will be enough to consider the model of a *transformed theory* obtained from $T(P)$.

As a first approximation, consider the problem of finding a *Plan* that complies with

$$T(P)' \ + \ \mathit{Plan} \qquad \text{is satisfiable} \tag{3}$$

where $T(P)'$ is a conjunction that takes into account *all* the initial states

$$T(P)' \quad = \quad \bigwedge_{s_0 \in \mathit{Init}} T(P) \,|\, s_0 \tag{4}$$

Here $T \,|\, X$ refers to theory T with variables x in T replaced by the value they have in X: true if $x \in X$, and false if $\neg x \in X$. This operation is known as value substitution or *conditioning* [6].

If equations (3) and (4) provided a correct formulation of conformant planning, we could obtain a conformant plan by finding a model for $T(P)'$, and extracting *Plan* from the value of the action variables in that model.

The formulation (3-4), however, is *not* correct. The reason is that the theory $T(P)$ contains fluent variables f_i for times $i > 0$ which are neither in *Init* nor in *Plan*. In (2), these variables can take different values for each s_0, while in (3-4), these variables are *forced* to take the *same* value over all s_0.

We can modify however the definition of $T(P)'$ in (4) for obtaining a correct SAT formulation of conformant planning. For this all that is needed is to *forget* the fluent variables f_i, for $i > 0$, from each conjunct $T(P) \,|\, s_0$ in (4).

The forgetting of a set of variables S from a theory T [5], also called elimination or existential quantification, is the dual operation to *Projection* of T over the *rest* of variables V; $V = vars(T) - S$. The projection of T over V, denoted $\mathsf{project}[\, T \,;\, V \,]$, refers to a theory over the variables V whose models are exactly the models of T restricted to those variables. For example, if $\phi = (a_1 \wedge f_1) \vee a_2$ then $\mathsf{project}[\, \phi;\, \{a_1, a_2\} \,] = a_1 \vee a_2$, which can also be understood as

$$\begin{aligned} \mathsf{project}[\, \phi;\, \{a_1, a_2\} \,] \ &= \ \exists f_1 \ \phi \quad = \quad (\phi \,|\, f_1 = \mathsf{true}) \ \vee \ (\phi \,|\, f_1 = \mathsf{false}) \\ &= \ ((a_1 \wedge \mathsf{true}) \vee a_2) \ \vee \ ((a_1 \wedge \mathsf{false}) \vee a_2) \quad = \quad (a_1 \vee a_2) \end{aligned}$$

Getting rid of the fluent variables f_i for $i > 0$ in the conjuncts $T(P) \,|\, s_0$ in (4) simply means to project such formulas over the action variables, as the variables in $T(P) \,|\, s_0$ are either action variables or fluent variables f_i for $i > 0$ (the fluent variables f_i for $i = 0$ have been substituted by the their values in s_0).

The result is that the transformed theory $T(P)'$ becomes:

$$T_{\mathrm{cf}}(P) \quad = \quad \bigwedge_{s_0 \in \mathit{Init}} \mathsf{project}[\, T(P) \,|\, s_0 \ ; \ \mathit{Actions} \,] \tag{5}$$

for which we can prove:

Theorem 1. *The models of $T_{cf}(P)$ in (5) are one-to-one correspondence with the conformant plans for the problem P.*

Equation 5 suggests a simple *scheme* for conformant planning: construct the formula $T_{\mathrm{cf}}(P)$ according to (5), and then feed this theory into a state-of-the-art SAT solver. The crucial point is the generation of $T_{\mathrm{cf}}(P)$ from the original theory $T(P)$: the transformation involves *conditioning* and *conjoining* operations, as well as *projections.* The key operation that is intractable is the projection. It is well known however that projection, like many other intractable boolean transformations, can be performed in linear time provided that the theory is in a suitable compiled form [20]. Of course, the compilation itself may run in exponential time and space, yet this will not be necessarily so on average.

We will actually show that the theory $T_{\mathrm{cf}}(P)$ in (5) can be obtained in time and space that is *linear* in the size of the d-DNNF compilation of $T(P)$ provided that the variables in s_0 are considered first in the compilation. Interestingly, recently [21] reports a related result for QBFs whose matrix is compiled in OBDD form; namely that the QBF becomes tractable if the variable ordering used in the compilation of the matrix is compatible with the ordering of the variables in the prefix. We will say more about this below. From a practical point of view, in the experimental section we will show that the compilation itself is feasible for large and complex theories such as those representing the current benchmarks in optimal conformant planning.

The form of the theory $T_{\mathrm{cf}}(P)$ in (5) is also closely related to the 'resolve' and 'expand' technique for solving QBFs recently proposed by Biere in [10]. Indeed in QBFs of the form $\exists plan\, \forall s_0\, \exists f_i\ \phi$, Biere eliminates the existential f_i variables by resolution, and the universal s_0 variables by expansion. The expansion is similar to the conditioning and conjunction in (5), while the elimination in (5) is done by projection and not resolution. The other difference is that the theory $T(P)$ is first compiled into d-DNNF with a variable ordering that places the universal variables s_0 on top. This ensures that the computation of the target formula $T_{\mathrm{cf}}(P)$ and its size are both linear in the size of the compiled $T_{\mathrm{c}}(P)$ formula. In other words, the transformation of $T(P)$ into $T_{\mathrm{cf}}(P)$ does not blow up if the compilation of $T(P)$ itself does not blow up. Table 1 below shows the close correspondence between the size of the compiled theories and the size of the resulting target theories $T_{\mathrm{cf}}(P)$ over a class of standard benchmarks.

4 Projection and d–DNNF

Knowledge compilation is concerned with the problem of mapping logical theories into suitable target languages that make certain desired operations tractable [22, 23, 24]. For example, the compilation into Ordered Binary Decision Diagrams (OBDDs) renders a large number of operations tractable including model counting [7]. While in all these cases the compilation itself is intractable, its expense may be justified if the operations are used a sufficiently large number of times. Moreover, while the compilation will run in exponential time and space in the worst case, it will not necessarily do so on average. Indeed, the compilation of

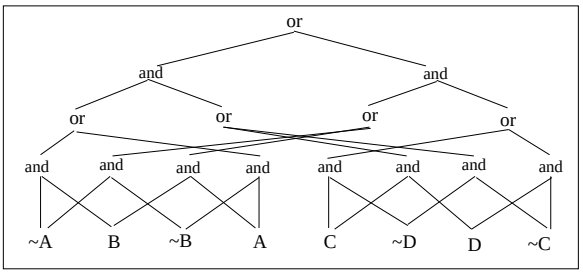

Fig. 1. A negation normal form (NNF) represented as a rooted DAG.

theories into OBDDs has been found useful in formal verification [25] and more recently in planning [26, 27].

A more recent compilation language is *Deterministic Decomposable Negation Normal Form* (d–DNNF [13]). d-DNNFs support a rich set of polynomial time operations and queries; in particular projection and model counting, that are intractable over CNFs, become linear operations over d-DNNFs. OBDDs are a special, less succint class of d-DNNFs; in fact, there are OBDDs that are exponentially larger than their equivalent d–DNNFs but not the other way around [24]. For both languages, OBDDs and d–DNNFs, there are public libraries that support compilation from CNF, along with a variety of queries and transformations.

4.1 Decomposability and Determinism of NNF

A propositional sentence is in Negation Normal Form (NNF) if it is constructed from literals using only conjunctions and disjunctions [28]. A practical representation of NNF sentences is in terms of rooted directed acyclic graphs (DAGs), where each leaf node in the DAG is labeled with a literal, true or false; and each non-leaf (internal) node is labeled with a conjunction $\wedge$ or a disjunction $\vee$; see Fig 1. Decomposable NNFs are defined as follows:

Definition 1 (Darwiche 2001). *A* decomposable negation normal form (DNNF) *is a negation normal form satisfying* decomposition: *for any conjunction $\wedge_i \alpha_i$ in the form, no variable appears in more than one conjunct α_i.*

The NNF in Figure 1 is decomposable. It has ten conjunctions and the conjuncts of each share no variables. Decomposability is the property which makes DNNF tractable: a decomposable NNF formula $\wedge_i \alpha_i$ is indeed satisfiable iff every conjunct α_i is satisfiable, while $\vee_i \alpha_i$ is always satisfiable iff some disjunct α_i is. The satisfiability of a DNNF can thus be tested in linear time by means of a single bottom-up pass over its DAG. A useful subclass of DNNFs closely related to OBDDs is d–DNNF [13]:

Definition 2 (Darwiche 2002). *A* deterministic DNNF (d–DNNF) *is a* DNNF *satisfying* determinism: *for any disjunction $\vee_i \alpha_i$ in the form, every pair of disjuncts α_i is mutually exclusive.*

Compilation into d-DNNF enables model counting in time linear in the size of the DAG also by means of a simple bottom-up pass. In order to compile arbitrary formulas into d-DNNF, consider first the following expansion of a formula Δ:

$$\Delta \quad \equiv \quad (\Delta \,|\, a \,\wedge\, a) \quad \vee \quad (\Delta \,|\, \neg a \,\wedge\, \neg a) \tag{6}$$

where recall that $\Delta|[\neg]a$ stands for the formula Δ conditioned on the literal a $[\neg a]$ (i.e., Δ with the variable a substituted by true or false respectively). If a is the *only* variable in Δ, it is easy to verify that the expression on the right-hand of (6) is in d–DNNF: decomposition holds because the conjuncts of each conjunction share no variables, and determinism holds because the disjuncts are mutually exclusive. When Δ contains many variables, it still can be compiled into d–DNNF by applying the expansion (6) recursively. This expansion is compatible with a number of optimizations such as unit resolution, clause learning, and caching, all of which are implemented in the d–DNNF compiler [14] that we use in the experiments.

4.2 Projection and Conditioning in d–DNNF

For generating a theory equivalent to $T_{\mathrm{cf}}(P)$ in (5), we need to perform three logical transformations: conditioning, conjoining, and projection. Provided that the original theory $T(P)$ is compiled in d–DNNF, all these transformations can be performed in time linear in the size of its DAG representation by a single bottom-up pass [6].[4] The resulting theory $T_{\mathrm{cf}}(P)$, however, is in NNF but not in DNNF due to the added conjunction, and this is why it cannot be checked for consistency in linear time, and has to be fed into a SAT solver (the mapping from NNF to CNF is linear though, adding new 'implied' variables for each of the internal nodes of the DAG).

5 Conformant Planner

Integrating the previous observations, the proposed conformant planner involves the following steps. First, a CNF theory $T(P)$ is obtained from a PDDL-like description of the planning problem extended for representing arbitrary intial situations and goals.[5] Then

[4] Roughly, for conditioning a d–DNNF over a set of literals X, variables are replaced by their 'values' in X: $x \leftarrow \mathsf{true}$ if $x \in X$, and $x \leftarrow \mathsf{false}$ if $\neg x \in X$. Similarly, for projecting a d–DNNF over a set of variables V, all literals involving variables *not* in V are replaced by true. In both cases, the operations at the leaves of the DAG are followed by a bottom-up propagation phase that respects the semantics of the AND and OR connectives. See [6] for details.

[5] A subtlety in the translation from P to the CNF theory $T(P)$ are the frame axioms which may generate an exponential number of clauses. In order to avoid such explosion, each conjunction $c^k(a)_i \wedge a_i$ of a condition $c^k(a)_i$ and the corresponding action a_i, is replaced by a new auxiliary variable z_i and the equivalence $z_i \equiv c^k(a)_i \wedge a_i$

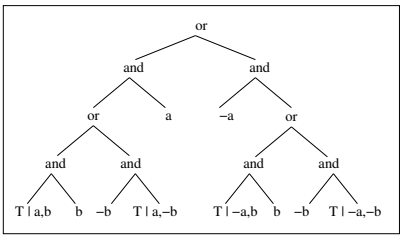

Fig. 2. Compilation of a theory T into d–DNNF by expanding first over variables a and b. Only root node of graphs representing the subtheories $T|X$ for $X \in \{(a,b),(a,\neg b),(\neg a,b),(\neg a,\neg b)\}$ are shown.

1. The theory $T(P)$ is **compiled** into the d–DNNF theory $T_{\mathrm{c}}(P)$
2. From $T_{\mathrm{c}}(P)$, the transformed theory

$$T_{\mathrm{cf}}(P) \quad = \quad \bigwedge_{s_0 \in Init} \mathsf{project}[\, T_{\mathrm{c}}(P) \,|\, s_0 \;;\; Actions \,]$$

   is obtained by operations that are linear in time and space in the size of the DAG representing $T_{\mathrm{c}}(P)$. The resulting theory $T_{\mathrm{cf}}(P)$ is in NNF but due to the added conjunction is *not* decomposable.
3. The NNF theory $T_{\mathrm{cf}}(P)$ is converted into CNF and the **SAT solver** is called upon it.

This sequence of operations is repeated starting from a planning horizon $N = 0$, increasing it by 1 until a solution is found.

Some of the details of the generation of the target theory $T_{\mathrm{cf}}(P)$ from the compiled theory $T_{\mathrm{c}}(P)$ are important. In particular, as we said above, it is necessary to compile $T(P)$ into $T_{\mathrm{c}}(P)$ using an ordering of variables that performs the expansions (6) on the $Init$ variables first; this is so that the DAG representing the d-DNNF subtheories $T_{\mathrm{c}}(P)|s_0$ for each possible initial state s_0, all correspond to (non-necessarily disjoint) fragments of the DAG representing the compiled d-DNNF theory $T_{\mathrm{c}}(P)$. Then the DAG representing the target NNF theory $T_{\mathrm{cf}}(P)$, which is no longer decomposable, is obtained by simply conjoining these fragments.

This can be visualized in Fig. 2 that shows (part of) the DAG for a theory T compiled into d-DNNF by expanding first on the variables a and b. In such a case, it can be seen that the DAG representing the subtheories $T|X$ for each of the possible valuations X of the variables a and b, correspond to subgraphs of this graph (not fully shown). Then, the DAG representation of the NNF theory that results from conjoining all these subtheories $T|X$ can be obtained simply by adding a new AND node to the graph whose children are the nodes representing these subtheories.

is added to the theory. Such auxiliary variables do not affect the compilation into d–DNNF as they are 'implied variables' that are safely projected away when the CNF theories are compiled into d–DNNF.

While d–DNNF compilers do not have to stick completely to the variable ordering chosen as in OBDDs, the compilation into both forms is heavily affected by the order of variables chosen. Moreover, as discussed in [8], it is possible to define an ordering by which the conformant plans can be extracted with no search at all from the compiled theory. However, this variable ordering renders the compilation practically infeasible in most cases. From a theoretical point of view, however, a similar result is established in [21], where QBFs are found to be tractable when the matrix formula is compiled in OBDD form following a variable ordering that is compatible with the ordering of the variables in the prefix.

6 Experimental Results

We performed experiments testing the proposed optimal conformant planner on a Intel/Linux machine running at 2.80GHz with 2Gb of memory. Runs of the d–DNNF compiler and the SAT solver were limited to 2 hours and 1.8Gb of memory. The d–DNNF compiler is Darwiche's `c2d` v2.18 [14], while the SAT solver is `siege_v4` [15] except for very large CNFs that would not load, and where *zChaff* [16] was used instead. We used the same suite of problems as [8] and [11]; some of the problems being originally from [12] and [29]. Most are challenging problems that emphasize the critical aspects that distinguish conformant from classical planning.

- **Ring:** There are n rooms arranged in a circle and a robot that can move clockwise or counter-clockwise, one step a a time. The room features windows that can be closed and locked. Initially, the position of the robot and the status of the windows are not known. The goal is to have all windows closed and locked. The number of initial states is $n \times 3^n$ and the optimal plan has $3n - 1$ steps.
- **Sorting Networks:** The task is to build a circuit made of compare-and-swap gates that maps an input vector of n boolean variables into the corresponding sorted vector. The compare-and-swap action compares two entries in the input vector and swaps their contents if not ordered. The optimal sequential plans minimize the number of gates, while the optimal parallel plans minimize the 'time delay' of the circuit. Only optimal plans for small n are known [30]. The number of initial states is 2^n.
- **Square-center:** A robot without sensors can move in a grid north, south, east, and west, and its goal is to get to the middle of the room. The optimal sequential plan for a grid of size 2^n with an unknown initial location is to do $2^n - 1$ moves in one direction, $2^n - 1$ moves in an orthogonal direction, and then from the resulting corner, $2^n - 2$ moves to the center for a total of $3 \times 2^n - 4$ steps. In the parallel setting, pairs of actions that move the robot in orthogonal directions are allowed. There are 2^{2n} initial states.
- **Cube-center-n:** Like the previous one, but in three dimensions.
- **Blocks:** Refers to blocks-world domain with move-3 actions but in which the initial state is completely unknown. Actions are always applicable but

have an effect only if their normal 'preconditions' are true. The goal is to get a fixed ordered stack with n blocks.

None of the problems feature preconditions, and only sorting, square-center, and cube-center admit parallel solutions (recall that we only allow parallel actions whose effects, ignoring their conditions, affect different variables).

We report compilation and search times. The first is the time taken by the d–DNNF compiler; the second is the time taken by the SAT solver. For the search part, we show the results for both the optimal horizon N^* and N^*-1. The first shows the difficulty of finding conformant plans; the second, the difficulty of proving them optimal.

In Table 1 we show results of the compilation for optimal horizons in the serial setting. The compilation of theories for smaller horizons or parallel formulations is normally less expensive. The table shows the optimal horizon N^* for each problem, the size of the original CNF theory $T(P)$, the size of the DAG representing the compiled theory $T_{\mathrm{c}}(P)$ with the time spent in the compilation, and finally the size of the target theory $T_{\mathrm{cf}}(P)$ in CNF that is fed to the SAT solver.

The first thing to notice is that all the theories considered in [11] compile: most in seconds, few in minutes, and one (ring-r8) in almost half an hour. In other words, for this suite of problems, as noted in [8], *the compilation is not the bottleneck*. On the other hand, as we will see, some theories do compile and yet are not solved by the SAT engine. Note also that as discussed earlier the size of the CNF theories that are fed into the SAT solver is roughly equal to the size of the DAG representing the compiled theories; namely, the number of variables is roughly the number of nodes in the DAG, while the number of clauses is only slightly larger than the number of edges in the DAG. This follows from the way the theory $T(P)$ is compiled (with the $Init$ variables on top), and the way the target theory $T_{\mathrm{cf}}(P)$ in CNF is constructed from it (with new variables representing the internal nodes of the DAG representation).

Table 2 shows the results of the SAT solver over the transformed theory $T_{\mathrm{cf}}(P)$ for both the optimal horizon N^* and N^*-1, and for both the sequential and parallel formulations. While not all problems are solved, the results improve upon those recently reported in [8], solving one additional instance in square-center and sorting. This represents an *order of magnitude* improvement over these domains. In blocks, on the other hand, there is no improvement, while the largest ring instances resulted in very large CNF theories that could not be loaded into Siege but were loaded and solved by zChaff (except for ring-r8 under the optimal planning horizon). Except for the Ring problems, these problems are not trivial, and some have optimal solutions that are not known (like the sorting problems for even small values of n; [30]). The planner fully solves sorting-s7 (in the parallel formulation) that as far as we know is not solved by any other optimal conformant planner including [11].

Interestingly, the Ring problems yield very large theories that are solved easily by the SAT solver when it can actually load the theories. On the other hand, the theories that arise from Square and Cube are relatively much smaller but

problem	N^*	CNF theory vars	CNF theory clauses	d–DNNF theory nodes	d–DNNF theory edges	d–DNNF theory time	$T_{\mathrm{cf}}(P)$ vars	$T_{\mathrm{cf}}(P)$ clauses
ring-r7	20	1081	3683	1008806	2179064	192.2	976203	3105362
ring-r8	23	1404	4814	3887058	8340295	1177.1	3779477	11957085
blocks-b3	9	444	2913	5242	20229	0.3	4667	23683
blocks-b4	26	3036	40732	226967	888847	124.5	223260	1104383
sq-center-e3	20	976	3642	11566	22081	1.1	9664	27956
sq-center-e4	44	4256	16586	90042	174781	47.1	81404	238940
cube-center-c9	33	2700	10350	282916	574791	98.9	276474	839027
cube-center-c11	42	4191	16227	658510	1330313	371.6	647994	1958472
sort-s7	16	1484	6679	115258	283278	12.4	112756	390997
sort-s8	19	2316	12364	363080	895247	77.2	359065	1246236

Table 1. Compilation data for sequential formulation and optimal horizon N^*. On the left, the size of the theories $T(P)$ encoding the conformant planning problems; on the center, the size of the DAGs representing the compiled theories $T_{\mathrm{c}}(P)$ and the time spent in the compilation; on the right, the size of the target theories $T_{\mathrm{cf}}(P)$ in CNF that are passed to the SAT engine.

also much more difficult to solve. Also, as it is well known, parallel formulations scale up better in terms of both compilation and search, and in many problems, finding solutions for the optimal horizon N^* appears much simpler than reporting the lack of solutions for the horizon $N^* - 1$. This may also be in part due to the presence of symmetries in the problem.

The planner is not directly comparable to non-optimal planners like CFF [31], although in many of these problems it has a clear edge. For instance, in [31], CFF times out after 25 minutes on ring-5 and cube-center-5, yet the proposed planner solves the larger ring-7 in a few seconds, and the larger cube-center-9 (parallel case) in a few minutes. On the other hand, in problems that are close to classical planning (as when the uncertainty is small), CFF seems to do much better.

7 Discussion

We presented a COMPILE-PROJECT-SAT scheme for computing optimal conformant plans. The scheme is simple and uses two off-the-shelf components: a d–DNNF compiler and a SAT solver. The results, while preliminary, are encouraging, showing that large and non-trivials problems are solved as well as done by the best optimal conformant planners.

We are currently exploring a variation of the COMPILE-PROJECT-SAT scheme that may be more suitable for dealing with problems that are not that far from classical planning, such as those considered by Brafman and Hoffman in [31]. In such a case, the burden of the compilation does not pay off, and it seems possible to perform the projection using a different mechanism based on the introduction of extra variables. This alternative approach, that preserves the PROJECT-SAT part but skips the compilation, automatically collapses to the standard SAT approach to classical planning when all the uncertainty goes away, and does not depart far from those approaches when the uncertainty is low (namely, the size

problem	N^*	$\#S_0$	search with horiz k time	decisions	#act	s. with horizon $k-1$ time	decisions
serial theories							
ring-r7	20	15309	° 2.1	2	20	° 0.8	0
ring-r8	23	52488	> 1.8Gb			° 2.4	0
blocks-b3	9	13	0.1	1665	9	0.2	3249
blocks-b4	26	73	$> 2h$			$> 2h$	
sq-center-e3	20	64	18.8	52037	20	207.4	207497
sq-center-e4	44	256	5184.4	1096858	44	$> 2h$	
cube-center-c7	24	343	3771.5	578576	24	5574.2	736567
cube-center-c9	33	729	$> 2h$			$> 2h$	
sort-s5	9	32	0.0	352	9	22.0	35053
sort-s6	12	64	40.0	34451	12	$> 2h$	
sort-s7	16	128	3035.6	525256	16	$> 2h$	
sort-s8	19	256	$> 2h$			$> 2h$	
parallel theories							
sq-center-e3	10	64	0.5	2737	20	0.3	1621
sq-center-e4	22	256	423.1	244085	44	1181.5	439532
cube-center-c7	8	343	6.1	4442	24	2.9	1892
cube-center-c9	11	729	114.6	27058	33	156.0	32760
cube-center-c11	14	1331	> 1.8Gb			181.5	13978
sort-s7	6	128	46.1	18932	18	355.4	48264
sort-s8	6	256	° 4256.6	533822	23	$> 2h$	

Table 2. Results for the Search: SAT calls over the tranformed theory $T_{\mathrm{cf}}(P)$ for the optimal horizon N^* (left) and $N^* - 1$ (right), both for sequential and parallel formulations (when they differ). We show the number of initial states, the time spent on the SAT call, the number of decisions made, and the number of actions in the plan found. Entries '$> 2h$' and '> 1.8Gb' mean time or memory exceeded. A signal ° indicate that the SAT solver used was *zChaff*, as `siege_v4` coult not load $T_{\mathrm{cf}}(P)$ due to its size. Times are in seconds.

of the theory that is passed to the SAT solver remains in those cases close to the size of the original encoding of the problem, and not close to the size of the compiled theory as in the approach pursued here).

The decision problem for conformant planning with a polynomial horizon is Σ_2^p-complete [2], which is the complexity of QBFs of the form $\exists x \forall y \exists z\ \phi$. In [32], Rintanen solves conformant planning problems by mapping them into QBFs of the form $\exists plan \forall s_0 \exists f_i\ \phi$. Our scheme for conformant planning can be easily adapted for solving QBFs of this form; indeed, provided that ϕ is compiled in d-DNNF, the existential variables f_i's can be eliminated by projection, the universal variables s_0's can be eliminated by conditioning and conjoning as in (5), and finally the existential *plan* variables can be solved by a single SAT call over the resulting formula. This method for solving QBFs is related to the 'resolve' and 'expand' technique recently proposed by Biere in [10]. Indeed in QBFs of the form $\exists plan \forall s_0 \exists f_i\ \phi$, Biere eliminates the existential f_i variables by resolution, and the universal s_0 variables by an expansion that is similar to the one captured in (5). In the proposed scheme, the existential variables f_i are eliminated by projection rather than resolution, and for that, the theory ϕ is first compiled into d-DNNF with a variable ordering that places the universal variables s_0 on top. This compilation ensures that the projection and the expansion are both done in time and space that is linear in the size of the DAG representing the compiled formula. For other recent works relating QBFs and compilation, see [21]. In the future, we would like to try the proposed methods on QBFs and compare the results with those obtained by existing general QBF solvers. Our preliminary results suggest, however, that the compilation-based approach benefits from the *stratified structure* of planning theories [33], and that its application to general theories may require further refinements in the variable order criterion used in the compilation.

Acknowledgments

We thank Adnan Darwiche and Blai Bonet for our joint work in [8] that led to this research, and for making their d-DNNF compiler and PDDL to CNF translator, respectively, available to us. H. Geffner is partially supported by grant TIC2002-04470-C03-02 from MCyT Spain.

References

[1] Goldman, R.P., Boddy, M.S.: Expressive planning and explicit knowledge. In: Proc. AIPS-1996. (1996)

[2] Turner, H.: Polynomial-length planning spans the polynomial hierarchy. In Flesca, S., Greco, S., Leone, N., Ianni, G., eds.: JELIA. Volume 2424 of Lecture Notes in Computer Science., Springer (2002) 111–124

[3] Ferraris, P., Giunchiglia, E.: Planning as satisfiability in nondeterministic domains. In: Proceedings AAAI-2000. (2000) 748–753

[4] Castellini, C., Giunchiglia, E., Tacchella, A.: Sat-based planning in complex domains: concurrency, constraints and nondeterminism. Artif. Intell. **147** (2003) 85–117

[5] Lin, F., Reiter, R.: Forget it! In: Working Notes, AAAI Fall Symposium on Relevance, American Association for Artificial Intelligence (1994) 154–159
[6] Darwiche, A.: Decomposable negation normal form. J. ACM **48** (2001) 608–647
[7] Bryant, R.E.: Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys **24** (1992) 293–318
[8] Palacios, H., Bonet, B., Darwiche, A., Geffner, H.: Pruning conformant plans by counting models on compiled d-DNNF representations. In: Proc. ICAPS-05.
[9] Rintanen, J.: Constructing conditional plans by a theorem prover. J. of AI Research **10** (1999) 323–352
[10] Biere, A.: Resolve and expand. In: SAT. (2004)
[11] Rintanen, J.: Distance estimates for planning in the discrete belief space. In: Proc. AAAI-04. (2004) 525–530
[12] Bonet, B., Geffner, H.: Planning with incomplete information as heuristic search in belief space. In: Proc. of AIPS-2000, AAAI Press (2000) 52–61
[13] Darwiche, A.: On the tractable counting of theory models and its applications to belief revision and truth maintenance. J. of Applied Non-Classical Logics (2002)
[14] Darwiche, A.: New advances in compiling cnf into decomposable negation normal form. In: Proc. ECAI 2004. (2004) 328–332
[15] Ryan, L.: Siege v4 - sat solver. `http://www.cs.sfu.ca/~loryan/personal/` (2003)
[16] Y.Mahajan, Z. Fu, S.M.: ZChaff2004: An Efficient SAT Solver. Selected Revised Papers Series: Lecture Notes in Computer Science (1994)
[17] Smith, D., Weld, D.: Conformant graphplan. In: Proc. AAAI-98 (1998)
[18] Kautz, H., Selman, B.: Pushing the envelope: Planning, propositional logic, and stochastic search. In: Proc. AAAI-96 (1996)
[19] Blum, A., Furst, M.: Fast planning through planning graph analysis. In: Proc. IJCAI-95 (1995)
[20] Darwiche, A., Marquis, P.: A knowledge compilation map. J. of AI Research **17** (2002) 229–264
[21] Coste-Marquis, S., Berre, D.L., Letombe, F., Marquis, P.: Propositional fragments for knowledge compilation and QBF In: Proc. AAAI-05. (2005)
[22] Selman, B., Kautz, H.: Knowledge compilation and theory approximation. Journal of the ACM **43** (1996) 193–224
[23] Cadoli, M., Donini, F.: A survey on knowledge compilation. AI Communications **10** (1997) 137–150
[24] Darwiche, A., Marquis, P.: A knowledge compilation map. JAIR **17** (2002)
[25] Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)
[26] Giunchiglia, F., Traverso, P.: Planning as model checking. In: Proceedings of ECP-99, Springer (1999)
[27] Cimatti, A., Roveri, M.: Conformant planning via symbolic model checking. JAIR **13** (2000) 305–338
[28] Barwise, J., ed.: Handbook of Mathematical Logic. North-Holland (1977)
[29] Cimatti, A., Roveri, M., Bertoli, P.: Conformant planning via symbolic model checking and heuristic search. Artificial Intelligence **159** (2004) 127–206
[30] Knuth, D.: The Art of Computer Programming, Vol. III: Sorting and Searching. Addison-Wesley (1973)
[31] Brafman, R., Hoffmann, J.: Conformant planning via heuristic forward search: A new approach. In Proc. ICAPS-04. (2004)
[32] Rintanen, J.: Constructing conditional plans by a theorem-prover. JAIR **10** (1999) 323–352
[33] Geffner, H.: Planning graphs and knowledge compilation. In Proc. KR-04, (1994)