# 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 d–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.

## 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. 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. Current SAT approaches to conformant planning thus follow a generate-and-test strategy [1]: 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): 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 *deterministic Decomposable Negated Normal Form* or *d*–DNNF [2], a form akin to OBDDs [3].

Our scheme for planning is thus based on the following three steps: the planning theory in CNF is first compiled into d-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 [4].

Two recent optimal conformant planners are Rintanen's [5] and Palacios *et al.*'s [4]. The first performs heuristic search in belief space 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. 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.

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 $\text{cond}^k(a) \rightarrow \text{effect}^k(a)$, $k = 1, \ldots, n_a$, where $\text{cond}^k(a)$ and $\text{effect}^k(a)$ are conjunctions of fluent literals. As mentioned above, we assume that actions are deterministic and hence that all uncertainty lies in the initial situation.

In the SAT approach to classical planning, 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 [6]. 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 $a \in O$:

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

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

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

   where the conjunction ranges over the conditions $\text{cond}^k(a)$ associated with effects $\text{effect}^k(a)$ 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.

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.

## 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 $a_i$ (*i.e.*, a full action valuation), let *Init* denote a 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., a possible initial state 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, this will be enough when the theory $T(P)$ is transformed in a suitable way. As a first approximation, consider the problem of finding a *Plan* that complies with

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

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

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

Here $T \,|\, X$ refers to theory $T$ with variables $x$ in $T$ replaced by the value they have in $X$: $\mathsf{true}$ if $x \in X$, and $\mathsf{false}$ if $\neg x \in X$. This operation is known as value substitution or *conditioning* [2].

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 possible $s_0$.

We can modify however the definition of $T(P)'$ in (4) for obtaining a correct SAT formulation of conformant planning. For this we need to *eliminate* or *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 $T$ [7], 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 $\exists f_1 \phi = (\phi \,|\, f_1 = \mathsf{true}) \vee (\phi \,|\, f_1 = \mathsf{false}) = ((a_1 \wedge \mathsf{true}) \vee a_2) \vee ((a_1 \wedge \mathsf{false}) \vee a_2) = (a_1 \vee a_2)$. 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:

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

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 $T_{\mathsf{cf}}(P)$ according to (5), and then feed this theory into a state-of-the-art SAT solver. The crucial point is the generation of $T_{\mathsf{cf}}(P)$ from the original theory $T(P)$: the transformation involves *conditioning* and *conjoining* operations, as well as *projections.* The key operation that is intractable is 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 [2]. 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 $T_{\mathsf{cf}}(P)$ in (5) can be obtained in time and space that is *linear* in the size of the d-DNNF compilation of $T(P)$.

## 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 [2]. The compilation of theories into OBDDs is intractable, but has been found useful in formal verification and more recently in planning [8, 9, 10].

A more recent compilation language is *Deterministic Decomposable Negation Normal Form* (d–DNNF [2]). 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 succinct class of d-DNNFs; in fact, there are OBDDs that are exponentially larger than their equivalent d–DNNFs but not the other way around [2].

### 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. 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 $\vee$. 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 $\alpha_i$.*

Decomposability is the property which makes DNNF tractable: a decomposable NNF formula $\wedge_i \alpha_i$ is indeed satisfiable iff every conjunct $\alpha_i$ is satisfiable, while $\vee_i \alpha_i$ is always satisfiable iff some disjunct $\alpha_i$ 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 [2].

**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 $\alpha_i$ is mutually exclusive.*

Model counting over d-DNNF can be done in time linear in the size of the DAG also by means of a simple bottom-up pass. A theory can be compiled into d–DNNF by applying the expansion $\Delta \equiv (\Delta \,|\, a \wedge a) \vee (\Delta \,|\, \neg a \wedge \neg a)$ recursively [11].

### 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. 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.

## 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 initial situations and goals. Then

1. The theory $T(P)$ is **compiled** into the d–DNNF theory $T_c(P)$
2. From $T_c(P)$, the transformed theory

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

   is obtained by operations that are linear in time and space in the size of the DAG representing $T_c(P)$. 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 $N = 0$ which is increased 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 $T_c(P)$ are important. In particular, it is necessary to compile $T(P)$ into $T_c(P)$ using an ordering of variables that expands on the $Init$ variables first; this is so that the DAG representing the d-DNNF subtheories $T_c(P)|s_0$ for each possible initial state $s_0$, all correspond to (non-necessarily disjoint) fragments of the DAG representing the compiled d-DNNF theory $T_c(P)$. Then the DAG representing the target NNF theory $T_{\mathrm{cf}}(P)$, which is no longer decomposable, is obtained by conjoining these fragments.

## 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 [11], while the SAT solver is `siege_v4` except for very large CNFs that would not load, and where *zChaff* was used instead. We used the same suite of problems as [4] and [5]. Most are challenging problems that emphasize the critical aspects that distinguish conformant from classical planning. **Ring:** A robot can move in $n$ rooms arranged in a circle. The goal is to have all windows closed and locked. **Sorting Networks:** The task is to build a circuit of compare-and-swap gates to sort $n$ boolean variables. **Square-center:** A robot without sensors can move in a grid of $2^n \times 2^n$, and its goal is to get to the middle of the room. For this it must first locate itself into a corner. **Cube-center:** Like the previous one, but in three dimensions. **Blocks:** Refers to the 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.

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 $T_c(P)$ with the time spent in the compilation, and finally the size of the target theory $T_{cf}(P)$ in CNF that is fed to the SAT solver. The first thing to notice is that all the theories considered in [5] compile. Thus, as in [4], *the compilation is not the bottleneck.*

| | | CNF theory | | d–DNNF theory | | | $T_{cf}(P)$ | |
|---|---|---|---|---|---|---|---|---|
| problem | $N^*$ | vars | clauses | nodes | edges | time | vars | 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 $T_c(P)$ and the times spent in the compilation; on the right, the size of the target theories $T_{cf}(P)$ in CNF that are passed to the SAT engine.

Table 2 shows the results of the SAT solver over the transformed theory $T_{cf}(P)$ 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 [4], 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).

The planner is not directly comparable to non-optimal planners like CFF [12], although in many of these problems it has a clear edge. For instance, 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.

| | | | search with horiz $k$ | | | s. with horizon $k-1$ | |
|---|---|---|---|---|---|---|---|
| problem | $N^*$ | $\#S_0$ | time | decisions | #act | time | decisions |
| **serial theories** | | | | | | | |
| ring-r7 | 20 | 15309 | $^\circ$ 2.1 | 2 | 20 | $^\circ$ 0.8 | 0 |
| ring-r8 | 23 | 52488 | > 1.8Gb | | | $^\circ$ 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 | $^\circ$ 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 $^\circ$ indicate that the SAT solver used was *zChaff*, as `siege_v4` could not load $T_{\mathrm{cf}}(P)$ due to its size. Times are in seconds.

## 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. 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 in [12]. When the number of possible initial states $s_0$ is low, rather than getting rid of the fluent variables $f_i$, $i > 0$, by projection as in

$$L \quad = \quad \bigwedge_{s_0 \in Init} \mathsf{project}[\ T(P) \,|\, s_0 \ ; \ Actions\ ] \qquad (6)$$

it may be more convenient to introduce copies of them, one for each possible initial state $s0$, resulting in the different formula

$$L' \quad = \quad \bigwedge_{s_0 \in Init} [\ T(P)^{s_0} \,|\, s_0\ ] \qquad (7)$$

where each $T(P)^{s_0}$ denotes a theory which is like $T(P)$ except that the fluent variables $f_i$, $i > 0$, are replaced by fresh copies $f_i^{s_0}$. Action variables, on the other hand, remain shared among all these theories. It can be shown that models of $L'$ as well as the models of $L$, are in one-to-one correspondence with the conformant plans that solve the problem. The latter approach, which does not require projection or compilation, may work better when the number of possible initial state is low, and collapses to the standard SAT approach to classical planning when all the uncertainty goes away.

In [13] 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 formulated also for QBFs of this form where it would have many elements in common with the "Eliminate and Expand" method of Biere [14], where the elimination is carried out by resolution rather than projection.

**Acknowledgments**

We thank Adnan Darwiche and Blai Bonet for our joint work in [4] that led to this research, and for making their d-DNNF compiler and PDDL to CNF translator, respectively, available to us.

## References

[1] Ferraris, P., Giunchiglia, E.: Planning as satisfiability in nondeterministic domains. In: Proceedings AAAI-2000. (2000) 748–753
[2] Darwiche, A., Marquis, P.: A knowledge compilation map. Journal of Artificial Intelligence Research **17** (2002) 229–264
[3] Bryant, R.E.: Symbolic Boolean manipulation with ordered binary-decision diagrams. ACM Computing Surveys **24** (1992) 293–318
[4] Palacios, H., Bonet, B., Darwiche, A., Geffner, H.: Pruning conformant plans by counting models on compiled d-DNNF representations. In: Proc. of the 15th Int. Conf. on Planning and Scheduling (ICAPS-05), AAAI Press (2005) 141–150
[5] Rintanen, J.: Distance estimates for planning in the discrete belief space. In: Proc. AAAI-04. (2004) 525–530
[6] Kautz, H., Selman, B.: Pushing the envelope: Planning, propositional logic, and stochastic search. In: Proceedings of AAAI-96. (1996) 1194–1201
[7] Lin, F., Reiter, R.: Forget it! In: Working Notes, AAAI Fall Symposium on Relevance, American Association for Artificial Intelligence (1994) 154–159
[8] Giunchiglia, F., Traverso, P.: Planning as model checking. In: Proceedings of ECP-99, Springer (1999)
[9] Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2000)
[10] Cimatti, A., Roveri, M.: Conformant planning via symbolic model checking. Journal of Artificial Intelligence Research **13** (2000) 305–338
[11] Darwiche, A.: New advances in compiling cnf into decomposable negation normal form. In: Proc. ECAI 2004. (2004) 328–332
[12] Brafman, R., Hoffmann, J.: Conformant planning via heuristic forward search: A new approach. In: Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-04). (2004)
[13] Rintanen, J.: Constructing conditional plans by a theorem-prover. Journal of Artificial Intelligence Research **10** (1999) 323–352
[14] Biere, A.: Resolve and expand. In: SAT. (2004)