Compact Tree Encodings for Planning as QBF

Olivier Gasquet, Dominique Longin, Frédéric Maris, Pierre Régnier, Maël Valais

doi: 10.4114/intartif.vol21iss62pp103-113

Compact Tree Encodings for Planning as QBF

Olivier Gasquet, Dominique Longin, Frédéric Maris, Pierre Régnier, Maël Valais
IRIT – University of Toulouse, Toulouse, France
{gasquet,longin,maris,regnier,valais}@irit.fr

Abstract Considerable improvements in the technology and performance of SAT solvers has made their use possible for the resolution of various problems in artificial intelligence, and among them that of generating plans. Recently, promising Quantified Boolean Formula (QBF) solvers have been developed and we may expect that in a near future they become as efficient as SAT solvers. So, it is interesting to use QBF language that allows us to produce more compact encodings. We present in this article a translation from STRIPS planning problems into quantified propositional formulas. We introduce two new Compact Tree Encodings: CTE-EFA based on Explanatory frame axioms, and CTE-OPEN based on causal links. Then we compare both of them to CTE-NOOP based on No-op Actions proposed in [3]. In terms of execution time over benchmark problems, CTE-EFA and CTE-OPEN always performed better than CTE-NOOP.

Keywords: Planning, Quantified Boolean Formula, Encodings

1 Introduction

An algorithmic approach for plans synthesis is automated compilation (i.e., transformation) of planning problems. In the Satplan planner [12], a planning problem is transformed into a propositional formula whose models, corresponding to solution plans, can be found using a SAT solver. The SAT approach searches for a solution-plan of fixed length k. In case of failure to find such a plan, this length is increased before restarting the search for a solution. In the classical framework, the complexity of finding a solution to any problem is PSPACE-hard, but the search for a fixed-size solution becomes NP-hard [2]. This compilation approach directly benefits from improvements in SAT solvers[1]. The most obvious example is the planner Blackbox [14, 15] (and its successors Satplan'04 [11] and Satplan'06 [16]). These planners won the optimal (in the number of plan steps) planning track of the International Planning Competitions[2] IPC-2004 and IPC-2006. This was unexpected because these planners were essentially updates of Blackbox and did not include any real novelty: improved performance was mainly due to progresses in the underlying SAT solver.

Numerous improvements of this original approach have been proposed since then, in particular via the development of more compact and efficient encodings: [13, 7, 18, 19, 22, 23, 24, 28].

Following these works, numerous other similar techniques for encoding planning problems have been developed: Linear Programming (LP) [30], Constraint Satisfaction Problems (CSP) [6], SAT Modulo Theories (SMT) [29, 20, 27]. More recently, a Quantified Boolean Formulas (QBF) approach had been proposed by [26, 3]. Currently SAT solvers outperform QBF solvers and the SAT approach is the most effective because SAT solvers and encodings have been greatly improved since 1992. However, over the past decade, there has been a growing interest in the QBF approach. The competitive evaluation of QBF solvers QBFEVAL[3] is now a joint event with the international SAT conference and QBF solvers improve regularly. QBFEVAL'16 had more participants than ever and QBF-related papers represented 27% of all papers published at SAT'16. Some promising techniques have been adapted

[1] http://www.satcompetition.org/
[2] http://www.icaps-conference.org/index.php/Main/Competitions
[3] http://www.qbflib.org/index_eval.php

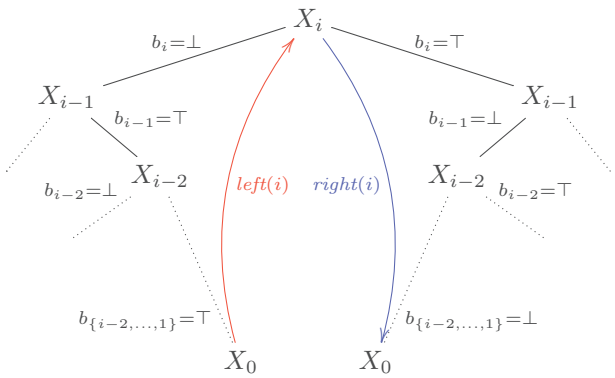

Figure 1: Both possible transitions in a CTE following the branching structure of a QBF: $X_0 \to X_i$ (from leaf to node on the left) and $X_i \to X_0$ (from node to leaf on the right). Note that i refers to any level (except for the leaf layer), not only the root.

to QBF solving such as counterexample guided abstraction refinement (CEGAR) [4, 10, 9, 21]. For comparable SAT / QBF encodings, the QBF approach also have the advantage to generate more compact formulas [3]. Even if the QBF approach is not as efficient as the SAT approach, it deserves the interest of the community.

Our paper shows that beyond the implementation of solvers, further work must be done to improve the encodings. In particular, we introduce two new QBF Compact Tree Encodings of STRIPS planning problems: CTE-EFA based on Explanatory frame axioms, and CTE-OPEN based on causal links. Then we compare both of them to [3] where CTE-NOOP based on No-op Actions is proposed. In terms of execution time over benchmark problems, CTE-EFA and CTE-OPEN always performed better than CTE-NOOP.

2 Planning as QBF

In [3], two different approaches of planning as QBF have been proposed: Flat Encoding, that was first introduced by [25] as an approach to general reachability, and Compact Tree Encoding (CTE). In [3], authors showed that Compact Tree Encodings outperform Flat Encodings. Both these planning encodings make use of the branching structure of the QBF to reuse a single set of clauses that describes a single step in the plan. The two assignments inside each universal variable represent the first and second half of the plan split around that branch. The assignments to each existential set represent action choices within a single step.

2.1 Preliminary Definitions

Let $\mathcal{F}$ be a finite set of *fluents* (atomic propositions). A STRIPS *planning problem* is a tuple $\langle I, \mathcal{A}, G\rangle$ where $I \subseteq \mathcal{F}$ is the set of initial fluents, $G \subseteq \mathcal{F}$ is the set of goal fluents and $\mathcal{A}$ is the set of actions. An action $a \in \mathcal{A}$ is a tuple $\langle Pre(a), Add(a), Del(a)\rangle$ where

- $Pre(a) \subseteq \mathcal{F}$ is the set of fluents required to be true in order to execute a,
- $Add(a) \subseteq \mathcal{F}$ and $Del(a) \subseteq \mathcal{F}$ are the sets of fluents respectively added and removed by the action a.

All QBF encodings studied in this paper use propositional variables for actions. The Compact Tree Encoding proposed in [3] is based on the *planning graph* introduced in [1] and uses additional no-op actions as frame axioms. We denote it by CTE-NOOP. Considering every action as a propositional variable, we define a set of propositional variables X, given by $X = \mathcal{A} \cup \{noop_f \mid f \in \mathcal{F}\}$.

In a CTE formula, we want to select two consecutive steps in order to define transitions (Figure 1). For each depth i of the tree, X_i denotes a copy of the set of variables X.

For CTE-NOOP, there exists a single variable $a_i \in X_i$ for each action and a single variable $noop_{f,i} \in X_i$ (no-op action) for each fluent used to determine a transition in the plan. At a same depth i, the value of these variables depends on the node (corresponding to a step in the plan) selected by the values of upper universal branching variables $b_{i+1} \ldots b_{depth}$. More details can be find in the slides[4].

An upper bound on the plan length is $2^{k+1} - 1$, where k is the number of alternations of quantifiers in the quantified boolean formula associated with the planning problem. In the case of CTE, k is also the compact tree

[4] `https://www.irit.fr/~Frederic.Maris/documents/coplas2018/slides.pdf`

depth. The number of possible states for a given planning problem is bounded by $2^{|\mathcal{F}|}$. Then, the existence of a plan can be determined using a linear QBF encoding with at most $k =\mid \mathcal{F} \mid$.

In the sequel, we propose two new encodings of planning problems into QBF. The first, denoted by CTE-OPEN, is based on causal links (plan-space). It has been introduced by [19] but needs to be adapted using additional variables for open conditions. The second, denoted by CTE-EFA, is based on explanatory frame axioms (state-space) first introduced by [12] and uses variables for fluents as well as for actions.

2.2 Causal Link Encoding: CTE-OPEN

The plan-space encodings of [19] cannot be directly adapted to the CTE. All these encodings refer to three indexed (not necessarily consecutive) steps of the plan. This is not possible in a CTE because each rule can refer to only one branch of the tree. To overcome this problem, it would be possible to duplicate the tree by adding, for each branching variable b_i, two more branching variables b'_i and b''_i, and for each node X_i, two node copies X'_i and X''_i, and equivalence rules $\bigwedge_{x_i \in X_i} \big((x_i \leftrightarrow x'_i) \wedge (x_i \leftrightarrow x''_i)\big)$. Unfortunately, this would increase the branching factor unnecessarily. So, we propose a new plan-space encoding which allows us to only refer to consecutive steps in the plan.

For every fluent $f \in \mathcal{F}$, we create a propositional variable $open_f$ to express that f holds in some previous step and must be protected at least until the current step. In Figure 2, the fluent f is an *open condition* in step S_i, entailing that either $f \in I$ or an action a' which adds f is executed in a previous step S_{i-k}. Open conditions are propagated backwards until the initial state or some step in which they are added by an action.

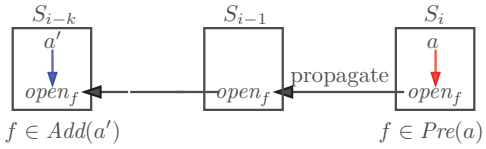

Figure 2: Causal link: a'produces f for a.

We define the set of "open" variables, denoted as Δ, as $\Delta = \{open_f \mid f \in \mathcal{F}\}$. Considering every action as a propositional variable, we define a set of propositional variables X, given by $X = \mathcal{A} \cup \Delta$.

Quantifiers For each depth i of the tree, X_i denotes a copy of the set of variables X. It exists a single variable $a_i \in X_i$ for each action used to determine last transition in the plan and a single variable $open_{f,i} \in X_i$ for each fluent used to determine if f is an open condition. At a same depth i, the value of these variables depends on the node (corresponding to a step in the plan) selected by the values of upper universal branching variables $b_{i+1} \dots b_{depth}$.

$$\begin{array}{l} \exists_{a \in \mathcal{A}}\, a_{depth}.\ \exists_{f \in \mathcal{F}}\, open_{f,depth}.\forall b_{depth}. \\ \exists_{a \in \mathcal{A}}\, a_{depth-1}.\ \exists_{f \in \mathcal{F}}\, open_{f,depth-1}.\forall b_{depth-1}. \\ \qquad \dots \\ \exists_{a \in \mathcal{A}}\, a_1.\ \exists_{f \in \mathcal{F}}\, open_{f,1}.\forall b_1.\ \exists_{a \in \mathcal{A}}\, a_0.\ \exists_{f \in \mathcal{F}}\, open_{f,0}. \end{array}$$

In the following, a *node* now refers to a non-leaf node (i.e., an inner node) and *depth* is the depth of the tree. The predecessor of a node at level i is the rightmost leaf of the left subtree. The successor of a node at level i is the leftmost leaf of the right subtree. In order to select these transitions, we introduce the leaf-to-node operator $left(i)$ defined as:

$$left(i) \equiv \neg b_i \wedge \bigwedge_{j=1}^{i-1} b_j.$$

Symmetrically, we introduce the node-to-leaf operator $right(i)$ defined as:

$$right(i) \equiv b_i \wedge \bigwedge_{j=1}^{i-1} \neg b_j.$$

Open conditions If an action a is executed in a step of the plan, then each precondition of a must be an open condition at this step (i.e., a causal link is required for this precondition).

$$\bigwedge_{i=0}^{depth} \bigwedge_{a \in \mathcal{A}} \left(a_i \Rightarrow \bigwedge_{f \in Pre(a)} open_{f,i} \right)$$

In the last plan step leading to the goal (i.e. the rightmost leaf of the tree), all the goal fluents must be either open conditions or added by actions executed in this step.

$$\bigwedge_{i=1}^{depth} b_i \Rightarrow \bigwedge_{f \in G} \left(open_{f,0} \vee \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a)}} a_0 \right)$$

Propagate and close No conditions should remain open in the first plan step (i.e. the leftmost leaf of the tree) if it is not provided in the initial state.

$$\bigwedge_{i=1}^{depth} \neg b_i \Rightarrow \bigwedge_{f \in \mathcal{F} \setminus I} \neg open_{f,0}$$

Any open condition in a step must either remain open or be added (closed) by an action in the previous step.

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,i} \wedge left(i) \big) \Rightarrow \left(open_{f,0} \vee \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a)}} a_0 \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,0} \wedge right(i) \big) \Rightarrow \left(open_{f,i} \vee \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a)}} a_i \right) \right)$$

Protect open conditions An open condition in a given step cannot be removed in the previous step. This guarantees not to break any causal link in the plan.

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,i} \wedge left(i) \big) \Rightarrow \bigwedge_{\substack{a \in \mathcal{A} \\ f \in Del(a)}} \neg a_0 \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left(\big(open_{f,0} \wedge right(i) \big) \Rightarrow \bigwedge_{\substack{a \in \mathcal{A} \\ f \in Del(a)}} \neg a_i \right)$$

Prevent negative interactions In a given step, if an action removes a fluent which is needed or added by another action, then these two actions cannot be both executed in this step.

$$\bigwedge_{i=0}^{depth} \bigwedge_{a \in \mathcal{A}} \bigwedge_{f \in (Add(a) \cup Pre(a))} \bigwedge_{\substack{a' \in \mathcal{A} \\ a \neq a' \\ f \in Del(a')}} \left(\neg a_i \vee \neg a'_i \right)$$

2.3 State-Space Encoding: CTE-EFA

In this encoding, we define the set of propositional variables as $X = \mathcal{A} \cup \mathcal{F}$. Each step is now defined by a transition (as in CTE-OPEN) as well as the resulting state (valuation of the fluents in $\mathcal{F}$). The formula is an adaptation to the CTE of the well known state-space SAT encoding rules based on explanatory frame axioms of [12].

Quantifiers At each depth i of the tree, it exists a single variable a_i for each action used to determine last transition in the plan and a single variable f_i for each fluent used to determine the state. At a same depth i , the values of these variables depend on the node (corresponding to a transition in the plan and the resulting state) selected by the values of upper universal branching variables $b_{i+1} \dots b_{depth}$.

$$\begin{array}{c} \exists\limits_{a\in\mathcal{A}} a_{depth}. \exists\limits_{f\in\mathcal{F}} f_{depth}. \forall b_{depth}. \\ \exists\limits_{a\in\mathcal{A}} a_{depth-1}. \exists\limits_{f\in\mathcal{F}} f_{depth-1}. \forall b_{depth-1}. \\ \dots \\ \exists\limits_{a\in\mathcal{A}} a_1. \exists\limits_{f\in\mathcal{F}} f_1. \forall b_1. \exists\limits_{a\in\mathcal{A}} a_0. \exists\limits_{f\in\mathcal{F}} f_0. \end{array}$$

Goal In the state after the last plan transition (i.e. the rightmost leaf of the tree), all goal fluents must be achieved.

$$\bigwedge_{i=1}^{depth} b_i \Rightarrow \bigwedge_{f\in G} f_0$$

Conditions and effects of actions If an action a is executed in a transition of the plan, then each effect of a occurs in the resulting state and each condition of a is required in the previous state.

$$\bigwedge_{i=0}^{depth} \bigwedge_{a\in\mathcal{A}} \left(a_i \Rightarrow \left(\bigwedge_{f\in Add(a)} f_i \right) \wedge \left(\bigwedge_{f\in Del(a)} \neg f_i \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{a\in\mathcal{A}} \left(a_i \wedge \mathit{left}(i) \Rightarrow \bigwedge_{f\in Pre(\mathcal{A})} f_0 \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{a\in\mathcal{A}} \left(a_0 \wedge \mathit{right}(i) \Rightarrow \bigwedge_{f\in Pre(\mathcal{A})} f_i \right)$$

Moreover, an action which do not have all conditions in initial state cannot be executed in the first plan transition (i.e. the leftmost leaf of the tree):

$$\bigwedge_{i=1}^{depth} \neg b_i \Rightarrow \bigwedge_{\substack{a\in\mathcal{A} \\ Pre(a)\not\subset I}} \neg a_0$$

Explanatory frame axioms If the value of a fluent changes between two consecutive states, then an action which produces this change is executed in the plan transition between these states.

$$\bigwedge_{i=1}^{depth} \bigwedge_{f\in\mathcal{F}} \left((\neg f_0 \wedge f_i \wedge \mathit{left}(i)) \Rightarrow \left(\bigvee_{\substack{a\in\mathcal{A} \\ f\in Add(a)}} a_i \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f\in\mathcal{F}} \left((\neg f_i \wedge f_0 \wedge \mathit{right}(i)) \Rightarrow \left(\bigvee_{\substack{a\in\mathcal{A} \\ f\in Add(a)}} a_0 \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f\in\mathcal{F}} \left((f_0 \wedge \neg f_i \wedge \mathit{left}(i)) \Rightarrow \left(\bigvee_{\substack{a\in\mathcal{A} \\ f\in Del(a)}} a_i \right) \right)$$

$$\bigwedge_{i=1}^{depth} \bigwedge_{f \in \mathcal{F}} \left((f_i \wedge \neg f_0 \wedge right(i)) \Rightarrow \left(\bigvee_{\substack{a \in \mathcal{A} \\ f \in Del(a)}} a_0 \right) \right)$$

An extra rule is also required to describe explanatory frame axioms for the first plan transition from initial state (i.e. the leftmost leaf of the tree):

$$\bigwedge_{f \in \mathcal{F} \setminus I} \left(\left(f_0 \wedge \bigwedge_{i=1}^{depth} \neg b_i \right) \Rightarrow \bigvee_{\substack{a \in \mathcal{A} \\ f \in Add(a) \\ Pre(a) \subset I}} a_0 \right)$$

$$\bigwedge_{f \in I} \left(\left(\neg f_0 \wedge \bigwedge_{i=1}^{depth} \neg b_i \right) \Rightarrow \bigvee_{\substack{a \in \mathcal{A} \\ f \in Del(a) \\ Pre(a) \subset I}} a_0 \right)$$

Prevent negative interactions Unlike in CTE-NOOP and CTE-OPEN, contradictory effects are already disallowed by previous rules (effects of actions). Then, this rule only need to prevent interactions between conditions and deletes of actions. If an action removes a fluent which is needed by another action, then these two actions cannot be both executed in a same plan transition.

$$\bigwedge_{i=0}^{depth} \bigwedge_{a \in \mathcal{A}} \bigwedge_{f \in Pre(a)} \bigwedge_{\substack{a' \in \mathcal{A} \\ a \neq a' \\ f \in Del(a')}} \left(\neg a_i \vee \neg a'_i \right)$$

3 Experimental Trials

To compare these three encodings on a same basis we used our translator TouIST[5] [5] that can use several QBF solvers. We ran all available STRIPS IPC benchmarks (1 through 8, except for the 7th which was not available and authors did not answer) on an Intel Xeon CPU E7-8890 v4 @ 2.20GHz, 512 GB of RAM. The domains tested include Gripper, Logistics, Mystery, Blocks, Elevator, Depots, DriverLog, ZenoTravel, FreeCell, Airport, Pipesworld-NoTankage, Pipesworld-Tankage, PSR, Satellite, OpenStacks, Pathways, Rovers, Storage, TPP, Trucks, ChildSnack, Hiking, VisitAll and the non-IPC Ferry.

We tried to consider as many QBF solvers as possible using the QBFEval 2017 as a reference. Qute (version of 2017-07-09, based on dependency learning QCDCL) and CaQE (version of 2017-07-08, based on CEGAR clausal abstraction) were not able to give a valuation for the outer existential quantifier. AIGSolve and Qell weren't available for download. GhostQ was skipped (but we should have included it). DepQBF (version 6.03 of 2017-08-02, based on "generalized Q-resolution", described in [17]) and RAReQS (version 1.1 of 2013-05-07, based on a CEGAR approach, detailed in [8]) were the only solvers left. RAReQS was consistently twice as fast as DepQBF, we thus dismissed DepQBF and only shown results for RAReQS. Finally, we did not apply any QBF preprocessor (e.g., Bloqqer).

[5] `https://www.irit.fr/touist`

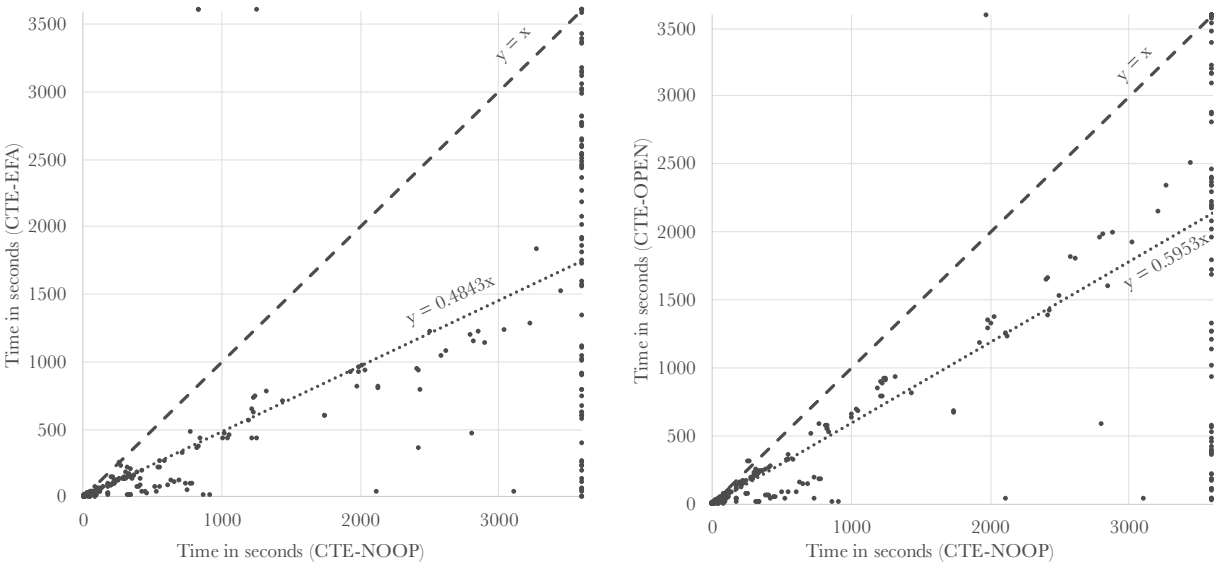

Figure 3: Plan decision time (EFA/OPEN vs NOOP).

Encoding	Solved problems	Decision Time	Literals	Clauses	Transitions-over-nodes ratio
CTE-NOOP	412 over 2112 (20%)	0%	0%	0%	30%
CTE-EFA	463 over 2112 (22%)	-55%	-26%	+15%	47%
CTE-OPEN	445 over 2112 (21%)	-41%	-2%	-28%	17%

Table 1: Comparison of the presented encodings across 65 STRIPS domains from IPC 1 through 8 (IPC 7 excepted) with a total of 2112 problems. Decision time, literals count, clauses count and the transitions-over-nodes ratio are averages. The transitions-over-nodes ratio measures the quantity (in average) of transition-based constraints over node-based constraints.

We ran these benchmarks using our new encodings CTE-EFA and CTE-OPEN as well as the state-of-the-art compact tree encoding (CTE-NOOP). We compared them two-by-two by considering the time needed to prove the existence of a plan (decision time, Figure 3) and the overall time required to obtain a plan (extraction time, Figure 5). The "decision" step consists of launching incrementally the QBF solver on a CTE of increasing depth until the solver returns true or reaches the upper bound (total number of fluents). The "extraction" step consists of one solver launch per node of the tree in order to retrieve the plan. Each experiment had a 60 minutes[6] timeout for searching the plan and 60 minutes for extracting it. The benchmark results are available as an Excel file[7].

The results show that our encodings CTE-EFA and CTE-OPEN are more efficient than CTE-NOOP both in plan existence as well as in plan extraction. CTE-EFA by a factor of 2.1 (1/0.4843) and CTE-OPEN by a factor of 1.7 (1/0.5953). Also, the comparison between CTE-EFA and CTE-OPEN (Figure 4, Figure 6) consistently shows that CTE-EFA outperforms CTE-OPEN by a factor of 1.4 (1/0.7266). Table 1 gives a summary of the benchmark results.

Contrary to what happens with flat encodings, the gain over CTE-NOOP cannot be explained by the difference in quantifier alternations as the depth is the same in the three encodings. However, the way actions are represented in these encodings may explain this difference.

4 Discussion

In order to identify the source of these improvements, we propose two hypothesis:

[6]The grounding step (i.e., action instantiation) is not included in the elapsed time.

[7]`https://www.irit.fr/~Frederic.Maris/documents/coplas2018/results.xls`

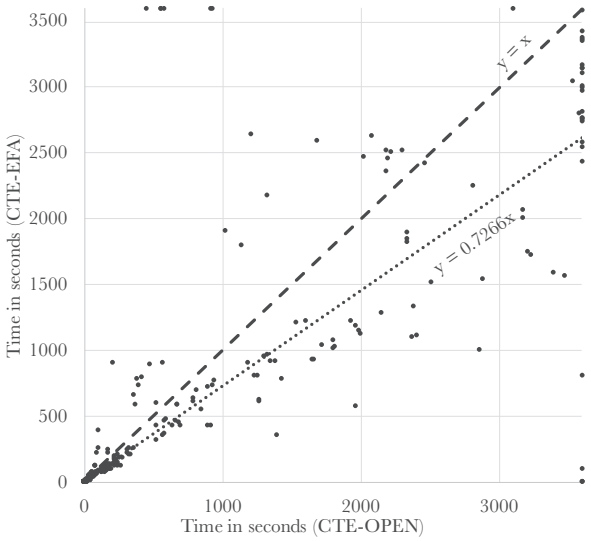

Figure 4: Plan decision time (EFA vs OPEN).

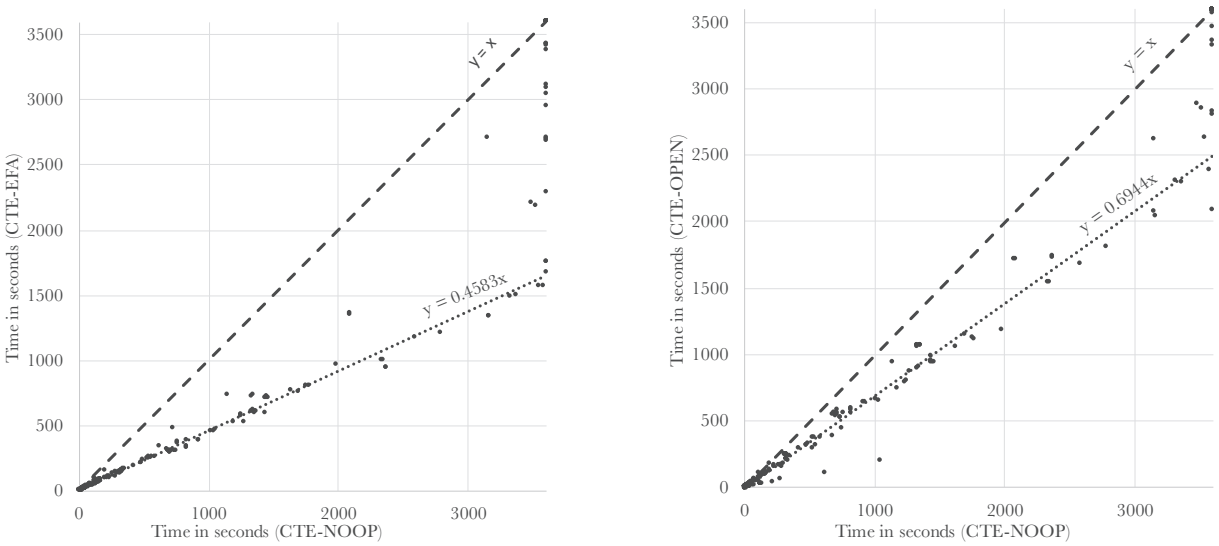

Figure 5: Plan extraction time (EFA/OPEN vs NOOP).

Hypothesis 1 "The performance gain is correlated to a decrease in the number of clauses and/or literals across encodings". Although the size of the problem is known to be noticeably non-correlated with its hardness in SAT, we wondered if we could see the same non-correlation. As shown in Table 1, we do not observe any clear tendency: CTE-EFA tends to have a slightly higher number of clauses (+15%) than CTE-NOOP although having less variables (-26%). CTE-OPEN has the same number of literals and much less clauses than CTE-NOOP, but resulting in a lower performance gain (-41%) than CTE-EFA (-55%). This non-correlation leads us to reject this hypothesis.

Hypothesis 2 "The performance gain is due to a difference in the number of transition-based constraints compared to the number of node-based constraints". Intuitively, one can think that a lower ratio of transition-based constraints over node-based constraints would ease the solving process: in node-based constraints, a clause has the same context[8] across the whole QBF expansion. In branch constraints, the corresponding

[8]Context and expansion are defined in [3]. Intuitively, the expansion is a tree representing the QBF and a context is a leaf in that tree.

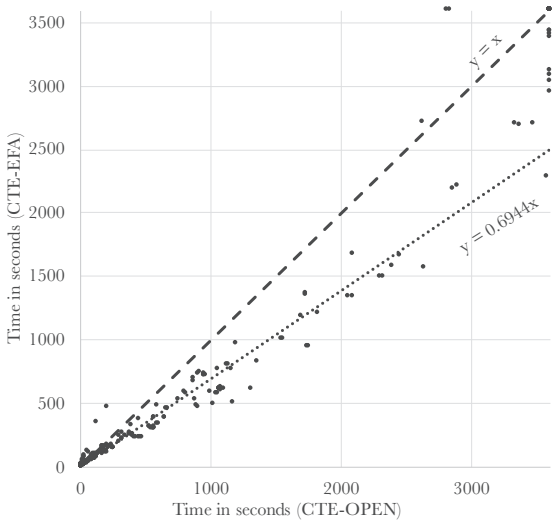

Figure 6: Plan extraction time (EFA vs OPEN).

clause has different contexts depending on the selected branch. The idea is that clauses based on different contexts slow the solver down. As displayed in Table 1, this hypothesis does not appear to be correct experimentally: although CTE-OPEN shows a lower transitions-to-nodes ration, it does not lead to the best performance gain. On the contrary, CTE-EFA has a poorer ratio, although being the most efficient compared to CTE-NOOP. We thus refute this hypothesis as we did not see any noticeable correlation supporting it, although we noticed a slight tendency where the decrease of time and of number of clauses were correlated.

Through these hypotheses, we tried to understand the causes of these enhancements. None of these hypothesis proved to be useful.

5 Conclusion

We have proposed two new QBF Compact Tree Encodings: CTE-OPEN based on causal links (plan-space) and CTE-EFA based on explanatory frame axioms (state-space). We compared these encodings with the state-of-the-art QBF encoding CTE-NOOP. In average over all available STRIPS IPC benchmarks, CTE-EFA performed twice as fast as CTE-NOOP (respectively 1.7 times faster for CTE-OPEN).

Through experiments, we refuted the two hypotheses we had formulated in order to explain the causes of this enhancement: neither the difference in the number of literals and clauses nor the transitions-to-nodes ratio between the three encodings allow us to draw conclusions.

Although it is fair to say that the work we are presenting lacks explanations on the reasons for the gain in performance, we think that this paper aims at showing the interest of systematically studying the statistical properties of the various action representations in encodings in order to understand the ontological choices related to these action representations.

Furthermore, it is noticeable that the performance ranking of the various action representations of SAT encodings (e.g., No-op performs better than EFA) is different in QBF (as we showed, EFA performs better than No-op in the CTE encoding). It would be interesting to study more broadly the methods used in SAT for encoding actions and see how their QBF counterpart behave.

References

[1] A. Blum and M. Furst. Fast planning through planning-graphs analysis. *Artificial Intelligence*, 90(1-2):281–300, 1997.

[2] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artif. Intell.*, 69(1-2):165–204, 1994.

[3] Michael Cashmore, Maria Fox, and Enrico Giunchiglia. Planning as quantified boolean formula. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI 2012 - 20th European Conference on Artificial Intelligence. Including Prestigious Applications of Artificial Intelligence (PAIS-2012) System Demonstrations Track, Montpellier, France, August 27-31 , 2012*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 217–222. IOS Press, 2012.

[4] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[5] Alexis Comte, Olivier Gasquet, Abdelwahab Heba, Olivier Lezaud, Frederic Maris, Khaled Skander-Ben-Slimane, and Mael Valais. Twist your logic with touist. *CoRR*, abs/1507.03663, 2015.

[6] Minh Binh Do and Subbarao Kambhampati. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artif. Intell.*, 132(2):151–182, 2001.

[7] M. Ernst, T. Millstein, and D.S. Weld. Automatic SAT-compilation of planning problems. In *Proc. IJCAI-97*, 1997.

[8] Mikolás Janota, William Klieber, João Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. In Alessandro Cimatti and Roberto Sebastiani, editors, *Theory and Applications of Satisfiability Testing - SAT 2012 - 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings*, volume 7317 of *Lecture Notes in Computer Science*, pages 114–128. Springer, 2012.

[9] Mikolás Janota, William Klieber, Joao Marques-Silva, and Edmund M. Clarke. Solving QBF with counterexample guided refinement. *Artif. Intell.*, 234:1–25, 2016.

[10] Mikolás Janota and Joao Marques-Silva. Solving QBF by clause selection. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 325–331. AAAI Press, 2015.

[11] H. Kautz. Satplan04: Planning as satisfiability. In *Abstracts of the 4th International Planning Competition, IPC-04*, 2004.

[12] H. Kautz and B. Selman. Planning as satisfiability. In *ECAI-92*, pages 359–363, 1992.

[13] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic and stochastic search. In *Proc. AAAI-96*, pages 1194–1201, 1996.

[14] H. Kautz and B. Selman. BLACKBOX: A new approach to the application of theorem proving to problem solving. In *Proceedings of AIPS-98 Workshop on Planning as Combinatorial Search*, 1998.

[15] H. Kautz and B. Selman. Unifying SAT-based and Graph-based planning. In *Proc. IJCAI-99*, pages 318–325, 1999.

[16] H. Kautz, B. Selman, and J. Hoffmann. Satplan04: Planning as satisfiability. In *Abstracts of the 5th International Planning Competition, IPC-06*, 2006.

[17] Florian Lonsing and Uwe Egly. Depqbf 6.0: A search-based QBF solver beyond traditional QCDCL. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 371–384. Springer, 2017.

[18] A. Mali and S. Kambhampati. Refinement-based planning as satisfiability. In *Proc. Workshop planning as combinatorial search, AIPS-98*, 1998.

[19] A. Mali and S. Kambhampati. On the utility of plan-space (causal) encodings. In *Proc. AAAI-99*, pages 557–563, 1999.

[20] Frederic Maris and Pierre Régnier. TLP-GP: new results on temporally-expressive planning benchmarks. In *(ICTAI 2008), Vol. 1*, pages 507–514, 2008.

[21] Markus N. Rabe and Leander Tentrup. CAQE: A certifying QBF solver. In Roope Kaivola and Thomas Wahl, editors, *Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27-30, 2015.*, pages 136–143. IEEE, 2015.

[22] J. Rintanen. Symmetry reduction for sat representations of transition systems. In *Proc. ICAPS-03*, 2003.

[23] J. Rintanen, K. Heljanko, and Niemelä. Parallel encodings of classical planning as satisfiability. In *Proc. European Conference on Logics in Artificial Intelligence, JELIA-04*, 2004.

[24] J. Rintanen, K. Heljanko, and Niemelä. Planning as satisfiability: parallel plans and algorithms for plan search. *Artificial Intelligence*, 170(1213):1031–1080, 2006.

[25] Jussi Rintanen. Partial implicit unfolding in the davis-putnam procedure for quantified boolean formulae. In *Logic for Programming, Artificial Intelligence, and Reasoning, 8th International Conference, LPAR 2001, Havana, Cuba, December 3-7, 2001, Proceedings*, pages 362–376, 2001.

[26] Jussi Rintanen. Asymptotically optimal encodings of conformant planning in QBF. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 1045–1050, 2007.

[27] Jussi Rintanen. Discretization of temporal models with application to planning with SMT. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 3349–3355, 2015.

[28] Jussi Rintanen, Bernhard Nebel, J. Christopher Beck, and Eric A. Hansen, editors. *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*. AAAI, 2008.

[29] Ji-Ae Shin and Ernest Davis. Processes and continuous change in a sat-based planner. *Artif. Intell.*, 166(1-2):194–253, 2005.

[30] Steven A. Wolfman and Daniel S. Weld. The LPSAT engine & its application to resource planning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, pages 310–317, 1999.