Partial Implicit Unfolding in the Davis-Putnam Procedure for Quantified Boolean Formulae

Jussi Rintanen

Albert-Ludwigs-Universität Freiburg, Institut für Informatik
Georges-Köhler-Allee, 79110 Freiburg im Breisgau
Germany

Abstract. Quantified Boolean formulae offer a means of representing many propositional formula exponentially more compactly than propositional logic. Recent work on automating reasoning with QBF has concentrated on extending the Davis-Putnam procedure to handle QBF. Although the resulting procedures make it possible to evaluate QBF that could not be efficiently reduced to propositional logic (requiring worst-case exponential space), its efficiency often lags much behind the reductive approach when the reduction is possible. We attribute this inefficiency to the fact that many of the unit resolution steps possible in the reduced (propositional logic) formula are not performed in the corresponding QBF. To combine the conciseness of the QBF representation and the stronger inferences available in the unquantified representation, we introduce a stronger propagation algorithm for QBF which could be seen as partially unfolding the universal quantification. The algorithm runs in worst-case exponential time, like the reduction of QBF to propositional logic, but needs only polynomial space. By restricting the algorithm the exponential behavior can be avoided while still preserving many of the useful inferences.

1 Introduction

Quantified Boolean formulae are a generalization of the satisfiability problem of the propositional logic that allows a more concise representation of many classes of formulae. The additional conciseness lifts the complexity of evaluating QBF to PSPACE-complete, which is in strong contrast to the NP-completeness of propositional satisfiability. However, the connection between the two problems is close, and not surprisingly some of the recent procedures for evaluating QBF [3, 20] are extensions of the Davis-Putnam procedure [4]. An alternative solution technique is to reduce a QBF to an unquantified propositional formula, and to test its truth by a conventional satisfiability algorithm. The drawback of this reductive approach is that the size of the propositional formula is worst-case exponential in the size of the QBF, which usually makes it impractical for all but the simplest QBF.

A problem with the extensions of the Davis-Putnam procedure to QBF is that many of the unit resolution steps that would be possible with the reduced formula do not take place. For restricted types of QBF, when the number of quantifier alternations is small, this problem has been partially overcome [20].

In this paper we attempt to provide a more general solution to this problem. The solution departs from earlier work on evaluating QBF in that the binary search algorithm is combined with a propagation algorithm that runs in *exponential time* in the size of the QBF. In general, algorithms for intractable problems have a restricted number of sources of non-polynomial behavior, and it is not a priori clear that using an exponential time subprocedure is sensible. Therefore, we present techniques for avoiding the exponentiality of the propagation algorithm to make the algorithm more practical. Our hypothesis is that the exponential reduction in problem size, due to the use of QBF instead of an equivalent unquantified formula, justifies a more expensive propagation algorithm. We also believe that in some cases even an exponential time propagation algorithm could be justified. The algorithm can be viewed as a conventional unit propagation algorithm for a QBF representation of unquantified clause sets.

The structure of the paper is as follows. In Sect. 4 we discuss the computational problem in detail by giving practically motivated examples of QBF that are very difficult for the current QBF algorithms. In Sect. 5 we outline the propagation algorithm, and in Sect. 6 we propose improvements and give a restricted variant of the algorithm that runs in polynomial time. Sect. 7 gives a preliminary experimental analysis of the algorithm, and Sect. 8 discusses related work.

2 Preliminaries

Quantified Boolean formulae are of the form $q_1 x_1 \cdots q_n x_n \phi$ where ϕ is a propositional formula and the prefix consists of universal $\forall$ and existential $\exists$ quantifiers q_i and the propositional variables x_i occurring in ϕ. Define $\phi[\psi/x]$ as the formula obtained from ϕ by replacing occurrences[1] of the propositional variable x by the formula ψ. The truth of formulae is defined recursively as follows. The truth of a formula that does not contain variables, that is, that consists of connectives and the constants true $\top$ and false $\bot$, is defined by the truth-tables for the connectives. A formula $\exists x \phi$ is true if and only if $\phi[\top/x]$ or $\phi[\bot/x]$ is true. A formula $\forall x \phi$ is true if and only if $\phi[\top/x]$ and $\phi[\bot/x]$ are true. Examples of true formulae are $\forall x \exists y(x \leftrightarrow y)$ and $\exists x \exists y(x \wedge y)$. The formulae $\exists x \forall y(x \leftrightarrow y)$ and $\forall x \forall y(x \vee y)$ are false. Changing the order of two consecutive variables quantified by the same quantifier does not affect the truth-value of the formula. It is often useful to ignore the ordering of consecutive variables and view each quantifier as quantifying a set of formulae, for example $\exists x_1 x_2 \forall y_1 y_2 \phi$.

3 The Extension of the Davis-Putnam Procedure to QBF

We have designed and implemented an algorithm that determines the truth-value of quantified Boolean formulae [20]. The Davis-Putnam procedure [4] is a

[1] We assume that nested quantifiers do not quantify the same variable.

special case of the algorithm. The main differences are that instead of only or-nodes, the search tree for quantified Boolean formulae contains also and-nodes that correspond to universally quantified variables, and that the order of the variables in the prefix constrains the order in which the variables generate a search tree. The algorithm takes as input formulae in which all quantifiers are in front of the formulae and the body is in conjunctive normal form.

The main procedure of the algorithm sketched in Fig. 1 takes three parameters.[2] The variable e is true if the first quantifier in the prefix of the formula is $\exists$. The sequence $\langle V_1, \ldots, V_n \rangle$ represents the prefix. For example, if the prefix is $\exists x_1 \exists x_2 \forall x_3 \exists x_4$, then $V_1 = \{x_1, x_2\}, V_2 = \{x_3\}$ and $V_3 = \{x_4\}$. The set C consists of clauses $\{l_1, \ldots, l_n\}$ where $n \geq 0$ and l_i are literals. The empty clause $\emptyset$ is false.

PROCEDURE decide($e, \langle V_1, V_2, \ldots, V_n \rangle, C$)
BEGIN
    $C :=$ unit(C);
    IF $\emptyset \in C$ *THEN RETURN* false;
    IF $n = 0$ *THEN RETURN* true;
    remove from V_1 all variables occurring in a unit clause in C;
    IF $V_1 = \emptyset$ *THEN*
        RETURN decide(not $e, \langle V_2, \ldots, V_n \rangle, C$);
    $x :=$ a member of V_1;
    $V_1 := V_1 \backslash \{x\}$;
    IF e *THEN*
        IF decide($e, \langle V_1, \ldots, V_n \rangle, C \cup \{\{x\}\}$)
        THEN RETURN true;
    ELSE
        IF not decide($e, \langle V_1, \ldots, V_n \rangle, C \cup \{\{x\}\}$)
        THEN RETURN false;
    RETURN decide($e, \langle V_1, \ldots, V_n \rangle, C \cup \{\{\neg x\}\}$)
END

Fig. 1. The extension of the Davis-Putnam procedure to QBF

The subprocedure *unit* performs simplification by unit resolution and unit subsumption; unit(S) is defined as the fixpoint of F under a set S of clauses.

$$\begin{aligned} F(C) = &\{c \backslash \{l\} | c \in C, \{\bar{l}\} \in C, l \in c\} \\ &\cup \{c \in C | l \notin c \text{ and } \bar{l} \notin c \text{ for all } \{l\} \in C\} \\ &\cup \{\{l\} \in C\} \end{aligned}$$

[2] The algorithm is simplified because we just want to indicate what the main differences to the Davis-Putnam procedure are. For example, we do not require that the variable x does not have a truth-value when it is branched on.

4 Motivating Examples

First we give simple examples illustrating which unit propagations are not performed by the Davis-Putnam QBF procedure of Section 3, and then we show a practical example of a class of formulae that are equivalent to exponentially bigger propositional formulae, and for which the lack of unit propagations makes even small formulae very difficult.

Example 1. Consider the QBF $\exists x \forall y \exists z(y \rightarrow z, z \rightarrow x)$. By considering the case when the universally quantified variable y gets the value true, one sees that also x has to be true. That this kind of reasoning may speed up evaluation of QBF considerably is shown by Rintanen [20].

The above line of reasoning often allows inferring some of the truth-values of the outermost variables in the Davis-Putnam QBF procedure. For QBF with prefix $\exists\forall\exists$, considering all valuations of the universal variables allows performing all the desired unit propagation steps.[3] However, when the prefix contains more than one block of universal variables, this is not the case.

Example 2. Consider the QBF $\exists a \forall b \exists x_1 x_2 \forall y \exists z_1 z_2((y \rightarrow z_1) \wedge (z_1 \rightarrow x_1) \wedge ((\neg y \wedge x_1) \rightarrow z_2) \wedge (z_2 \rightarrow x_2) \wedge ((x_1 \wedge x_2 \wedge b) \rightarrow a))$. Unlike in Example 1 where (repeatedly) choosing truth-values for all universal variables and then performing unit propagation yielded all desired values for the outermost variables, in this example the same strategy does not suffice. The problem is that two valuations, respectively assigning $b = \top, y = \top$ and $b = \top, y = \bot$, are needed, and neither of these alone allows inferring a. First one uses the first assignment and infers x_1, exchanges $y = \top$ to $y = \bot$, and only then can one infer a with x_2. After using the first assignment, one in general cannot preserve the values obtained for x_1 and x_2, because these could depend on the choice $b = \top$, to which we have not committed to.

The propagation pattern present in Example 2 could be made still more intricate. In $\exists T \forall U \exists X \forall Y \Phi$ we could be forced to repeatedly alternate between valuations v_1 and v_2 of the universal variables Y in order to infer more and more values for the existential variables X, keeping part of the valuation (for the outermost universal variables U) fixed. The second example shows that the hierarchical propagation structure could be vital for solving naturally occurring QBF.

Example 3. Consider the following formula that represents the existence of transition sequences of length 2^n between two states [21].

$$\exists S S' (\mathrm{reach}_n(S, S') \wedge I \wedge G) \tag{1}$$

Here I and G are the formulae describing the initial and goal states respectively expressed in terms of variables from sets S and S'. Here $\mathrm{reach}_i(S, S')$ means that

[3] Of course, performing this computation that is exponential in the number of universal variables may in practise be too expensive to be useful.

a state represented in terms of variables from S' can be reached with 2^i steps from a state represented in terms of variables from S. It is recursively defined as follows.

$$
\begin{aligned}
\mathrm{reach}_0(S,S') &\stackrel{\mathrm{def}}{\equiv} R(S,S') \\
\mathrm{reach}_{i+1}(S,S') &\stackrel{\mathrm{def}}{\equiv} \exists T \forall c \exists T_1 \exists T_2 (\mathrm{reach}_i(T_1,T_2) \\
&\quad \wedge (c \rightarrow (T_1 = S \wedge T_2 = T)) \\
&\quad \wedge (\neg c \rightarrow (T_1 = T \wedge T_2 = S')))
\end{aligned}
$$

Here R is the one-step transition relation on two sets of variables that respectively represent the state variables for the predecessor and the successor states. The sets T and S consist of propositional variables, and $S = T$ for $S = \{s_1, \ldots, s_n\}$ and $T = \{t_1, \ldots, t_n\}$ means $(s_1 \leftrightarrow t_1) \wedge \cdots \wedge (s_n \leftrightarrow t_n)$. The idea of the definition of $\mathrm{reach}_{i+1}(S,S')$ is that the variables T describe a state halfway between S and S', and the two values for the variable c correspond to two reachability tests, one between S and T, and the other between T and S'. This is very close to the PSPACE membership proof of s-t reachability of graphs represented in terms of state variables [15, 14, 2].

If we eliminate all universal variables from Formula 1, we see that it is essentially a concise $O(\log t)$ space ($t = 2^n$) representation of

$$I_0 \wedge R(S_0,S_1) \wedge R(S_1,S_2) \wedge \cdots \wedge R(S_{t-1},S_t) \wedge G_t \tag{2}$$

with only one occurrence of the transition relation R. Now, there are many instances of Formula 2 (especially if the estimated transition sequence length 2^i is "low") in which unit propagation immediately yields many state variable values [9, 18]. However, for the corresponding Formula 1 none of this takes place. The Davis-Putnam QBF procedure performs an exhaustive search through the valuations of all the variables but the innermost ones. This makes even small reachability problems (a couple of dozen state variables and transition sequence length 4) practically unsolvable on the Davis-Putnam QBF procedure, while the corresponding Formula 2 would be solved immediately by any reasonable satisfiability algorithm.

Example 4. When the transition relation R is the implication $s_0 \rightarrow s_1$, we obtain the following formula for reachability of length 4.

$$
\begin{array}{ll}
 & \exists ab \\
\exists x_1 \forall c_1 \exists s_1 \exists t_1 & \\
\exists x_2 \forall c_2 \exists s_2 \exists t_2 & (a \wedge \\
 & c_1 \rightarrow ((a \leftrightarrow s_1) \wedge (x_1 \leftrightarrow t_1)) \wedge \\
 & \neg c_1 \rightarrow ((x_1 \leftrightarrow s_1) \wedge (b \leftrightarrow t_1)) \wedge \\
 & c_2 \rightarrow ((s_1 \leftrightarrow s_2) \wedge (x_2 \leftrightarrow t_2)) \wedge \\
 & \neg c_2 \rightarrow ((x_2 \leftrightarrow s_2) \wedge (t_1 \leftrightarrow t_2)) \wedge \\
 & (s_2 \rightarrow t_2))
\end{array}
$$

This formula is equivalent to (with respect to a and b)

$$a \wedge (a \rightarrow s_1) \wedge (s_1 \rightarrow s_2) \wedge (s_2 \rightarrow s_3) \wedge (s_3 \rightarrow b),$$

and by unit resolution one can directly infer that b has to be true.

5 The New Unit Propagation Algorithm

Let u be the first universal variable in the prefix of a QBF Φ, and let $x_1, \ldots, x_n$ be the existential variables in the prefix *after* u. Then u can be eliminated from Φ by producing the QBF $\Phi_u = \Phi[x'_1/x_1, \ldots, x'_n/x_n, \top/u] \wedge \Phi[\bot/u]$. If $u_1, \ldots, u_m$ are the universal variables in Φ, then Φ is true if and only if $\Phi_{u_1,\ldots,u_m}$ is true. This latter formula is the one obtained by eliminating the universal variables from Φ in the given order that agrees with their order in the prefix. Now $\Phi_{u_1,\ldots,u_m}$ contains existential variables only, and it – when ignoring the quantifiers – can be viewed as a normal satisfiability problem in the propositional logic. Because $\Phi_{u_1,\ldots,u_m}$ can have a size exponential in the size of Φ, this reduction is usually not a practical way of evaluating QBF.

The algorithm we propose could be viewed as partially unfolding the QBF: at any point of time only one of the 2^m conjuncts of $\Phi_{u_1,\ldots,u_m}$ is produced, call it χ, which corresponds to a single valuation of all the universal variables. Such formulae χ typically share (existential) variables, because existential variables get renamed in the reduction only when they follow a universal variable in the prefix. We would like to perform unit resolution on these formulae χ so that truth-values obtained for shared variables would be propagated also to other formulae χ' obtained by partially unfolding Φ.

This idea leads to the hierarchical propagation algorithm in Fig. 2 that does not explicitly produce the formulae $\Phi_{u_1,\ldots,u_m}$. It takes the following parameters.

- $Q = \langle V_1, \ldots, V_n \rangle$ is a sequence of sets of variables that represents the quantifiers of the QBF, where V_{2i+1} for $i \geq 0$ are universally quantified and V_{2i} for $i \geq 1$ are existentially quantified, and
- C is the body of the formula (a set of clauses).

```
PROCEDURE propagate(⟨V1, V2, V3, ..., Vn⟩, C)
IF n = 0 THEN RETURN unit(C);
again:
FOR EACH valuation v of V1 DO
   C' := propagate(⟨V3, ..., Vn⟩, C ∪ v);
   IF {p} ∈ C'\C or {¬p} ∈ C'\C for some p ∈ P
                        where P = V\(V1 ∪ ··· ∪ Vn)
   THEN
      BEGIN
         C := C ∪ (C' ∩ ({{p}|p ∈ P} ∪ {{¬p}|p ∈ P}));
         GOTO again;
      END
END
RETURN C;
```

Fig. 2. The hierarchical unit propagation algorithm

Valuations v of V_1 above are sets of unit clauses with exactly one occurrence of every variable in V_1. The set V consists of all variables occurring in the

QBF. The existential variables V_{2i} in the prefix Q are used by the propagation algorithm only as far as conventional unit propagation produces them.

The algorithm runs in exponential time on the size of the QBF because the number of valuations may be exponential. However, it needs only polynomial space. This is because at any given point of time only one valuation of the universal variables and the values inferred for the existential variables need to be stored explicitly.

6 Improvements

The algorithm can be improved by taking into account properties of the body C of the formula, and by preventing the worst-case exponential running time. First, we give stricter conditions on the selection of valuations v based on the possibilities of performing unit resolution steps. This often leads to a big reduction in the runtime but does not eliminate the exponential time worst-case behavior. Second, in Sect. 6.1 we consider further restrictions that lead to a polynomial runtime.

Example 5. Consider a clause set C in which only the clause $u_1 \vee u_2 \vee x$ contains less than two existential variables, namely the variable x. The only possibility of performing unit resolution is to assign both u_1 and u_2 false.

Therefore, only such values should be assigned to the universal values that contribute to producing a unit clause. It would be possible to take the usefulness criterion further. The new unit clause that is obtained should have a complementary occurrence in a clause with less than 3 literals: otherwise the unit clause would be the only one that is produced, and therefore often not very useful.

6.1 Restrictions Leading to Polynomial Runtime

Producing unit clauses from two clauses $u \vee \phi_1$ and $\neg u \vee \phi_2$ is complementary. This is the reason why the worst-case runtime of the algorithm is exponential: otherwise it would suffice to choose one valuation for the universal variables so that all possible unit clauses are produced.

Example 6. Consider a clause set that includes the clauses $(u_1 \vee x_1), (\neg u_1 \vee x'_1), (u_2 \vee x_2), (\neg u_2 \vee x'_2), \cdots, (u_n \vee x_n), (\neg u_n \vee x'_n)$. One can obtain 2^n different sets of unit clauses by assigning 2^n different combinations of truth-values to the universal variables $u_1, \ldots, u_n$.

The exponential behavior of the propagation algorithm can be avoided by refraining from trying out all valuations of the universal variables. A reasonable strategy would be to try only enough valuations so that each unit clause (but not necessarily every combination of unit clauses) is obtained once. Of course, this restriction means that correspondence between the unit resolution steps available in the unquantified propositional formula and in the corresponding QBF is lost.

Depending on the QBF in question, this could be a big loss or not. For the $O(\log n)$ QBF encoding of the s-t reachability problem one may be forced to try out an exponential number of combinations of unit clauses to obtain all inference steps.

7 An Implementation of the Algorithm

We have implemented the propagation algorithm as a variant of the QBF solver described by us earlier [20]. The new propagation algorithm replaces the less general inversion and sampling techniques. The solver heavily uses a general form of the failed literal rule for reducing the size of the search tree: see Li and Anbulagan's work on the sat00 implementation of the Davis-Putnam procedure [12]. From the other techniques described by Rintanen [20] only the splitting of the clause set to disjoint subsets with no shared variables is used.

The new propagation algorithm is used at every node of the search tree, just like the standard unit propagation algorithm. The implementation consists of two mutually recursive subprocedures, the first traversing through all the valuations of a block of universal quantifiers, and the second keeping track of the existential variables that have been inferred. The polynomial time behavior described in Sect. 6.1 is achieved by labeling every clause that has been made a unit clause, and refraining from trying a truth-value (true or false) for a universal variable if it would not help producing a unit clause that has not already been labeled. This way the number of valuations tried is at most as high as the number of clauses with one existential literal and one or more universal literals.

7.1 Structured Formulae

We evaluated the algorithm on problems from AI planning that are encoded like Example 3. They solve a small blocks' world problem with 4 blocks, the Towers of Hanoi with 3 disks, and the well-known bw-large.a and bw-large.b blocks world problems. We list the runtimes on our implementations of the basic Davis-Putnam QBF procedure, with the inversion/sampling techniques from [20], and with the new propagation algorithm, in Table 1. The runs were on a 360MHz Sun Sparc. We terminated each run that lasted for more than one hour. Even the best runtimes presented here are worse than the runtimes of conventional satisfiability algorithms on the reduced formulae. A bigger set of test runs is reported in Table 2. These QBF include ones representing planning under incomplete information [19], some randomly generated problems, and the encoding of long chains of implications as in Example 4. On some of the problems the stronger propagation algorithms slow down QBF evaluation because only very few or no new literals can be inferred, and running the algorithms is relatively expensive.

7.2 Random Formulae

Contrary to what was reported by us earlier for the less general unfolding techniques [20], our implementation of the new propagation algorithm improves the

Table 1. Runtimes of the QBF algorithm on QBF from AI planning. The last QBF for each problem is true, the preceding ones are false.

problem	path length	prefix	vars	clauses	runtime in seconds DP/QBF	DP+inversion	DP+new alg.
BLOCKS4	2	∃∀∃	149	1183	> 3600	0.03	0.04
BLOCKS4	4	∃∀∃∀∃	210	1505	> 3600	1527.55	3.20
BLOCKS4	8	∃∀∃∀∃∀∃	271	1827	> 3600	> 3600	4.22
HANOI3	2	∃∀∃	709	16599	> 3600	0.56	0.54
HANOI3	4	∃∀∃∀∃	962	17553	> 3600	> 3600	23.08
HANOI3	8	∃∀∃∀∃∀∃	1215	18507	> 3600	> 3600	> 3600
bw-large.a	2	∃∀∃	1099	62916	> 3600	0.89	1.13
bw-large.a	4	∃∀∃∀∃	1370	65688	> 3600	> 3600	256.75
bw-large.b	2	∃∀∃	1871	178890	> 3600	1.74	2.38
bw-large.b	4	∃∀∃∀∃	2268	183741	> 3600	> 3600	15.65
bw-large.b	8	∃∀∃∀∃∀∃	2665	188592	> 3600	> 3600	> 3600

Davis-Putnam QBF procedure runtimes on difficult randomly generated problems substantially. This is because of the stricter and more goal-directed criteria for selecting truth-values for universal variables.

Tables 3 and 4 show the runtimes of our QBF solver on random QBF (model A of Gent and Walsh [6]) respectively without and with the new propagation algorithm. The times reported are runtimes (in milliseconds) of 150 variable ∃∀∃ formulae with varying numbers of universal variables and clauses/variables ratios. The runtimes are averages on 1000 formulae. The percentage of universal variables (rows in the tables) varies from 0 to 53.3, and the number of existential variables before and after the universal variables is the same. The clauses/variables ratio varies from 1 to 4 (the columns in the tables.) The propagation algorithm produced each unit clause at least once, but did not produce all combinations of unit clauses. The ratios of the runtimes with and without the new propagation algorithm are shown in Table 5. In the phase transition region on the most difficult QBF (see [20]) the new propagation algorithm speeds up the evaluation by a factor of ten. On easier formulae, especially those containing a high number of universal variables, the algorithm slows down the evaluation, up to a factor of five. On bigger formulae and with more quantifiers the speed-ups are much bigger.

8 Related Work

Early work on quantified Boolean formulae include the polynomial time algorithm by Aspvall et al. [1] for quantified 2-literal clauses, and the polynomial time decision algorithm for quantified Horn clauses by Kleine Büning et al. [10]. Kleine Büning et al. also define a resolution rule for QBF.

Cadoli et al. [3] extended the Davis-Putnam procedure to handle quantified Boolean formulae. Their algorithm is similar to the one in Sect. 3, first defined in [20], but is based on two mutually recursive procedures that respectively

Table 2. Comparison of the runtimes of the basic Davis-Putnam QBF procedure, the same with the new propagation algorithm restricted to polynomial runtime, and the unrestricted new propagation algorithm, on a number of QBF.

	basic DP/QBF		new algo. $O(p(n))$		new algo. $O(2^n)$	
problem	runtime	tree size	runtime	tree size	runtime	tree size
BLOCKS3i.4.4.qcnf	> 10 min.	-	0.20	0	0.17	0
BLOCKS3i.5.3.qcnf	> 10 min.	-	94.10	378	101.80	378
BLOCKS3i.5.4.qcnf	> 10 min.	-	9.19	50	9.80	50
BLOCKS3ii.4.3.qcnf	18.53	1015	0.15	0	0.11	0
BLOCKS3ii.5.2.qcnf	345.98	10021	0.19	0	0.18	0
BLOCKS3ii.5.3.qcnf	> 10 min.	-	1.49	29	1.44	29
BLOCKS3iii.4.qcnf	20.35	1728	0.05	0	0.03	0
BLOCKS3iii.5.qcnf	> 10 min.	-	0.73	26	0.70	26
BLOCKS4i.6.4.qcnf	> 10 min.	-	8.96	0	9.00	0
BLOCKS4i.7.3.qcnf	> 10 min.	-	> 10 min.	-	> 10 min.	-
BLOCKS4i.7.4.qcnf	> 10 min.	-	> 10 min.	-	> 10 min.	-
BLOCKS4ii.6.3.qcnf	> 10 min.	-	8.78	0	8.85	0
BLOCKS4ii.7.2.qcnf	> 10 min.	-	15.21	0	15.12	0
BLOCKS4ii.7.3.qcnf	> 10 min.	-	453.03	190	455.85	190
BLOCKS4iii.6.qcnf	> 10 min.	-	4.39	0	4.45	0
BLOCKS4iii.7.qcnf	> 10 min.	-	230.21	178	218.21	178
CHAIN12v.13.qcnf	0.25	36	0.43	12	73.67	12
CHAIN13v.14.qcnf	0.31	39	0.55	13	182.61	13
CHAIN14v.15.qcnf	0.38	42	0.74	14	449.17	14
CHAIN15v.16.qcnf	0.52	45	0.92	15	> 10 min.	-
CHAIN16v.17.qcnf	0.64	48	1.13	16	> 10 min.	-
CHAIN17v.18.qcnf	0.74	51	1.43	17	> 10 min.	-
CHAIN18v.19.qcnf	0.91	54	1.74	18	> 10 min.	-
CHAIN19v.20.qcnf	1.07	57	2.15	19	> 10 min.	-
CHAIN20v.21.qcnf	1.23	60	2.68	20	> 10 min.	-
CHAIN21v.22.qcnf	0.51	63	2.12	21	> 10 min.	-
CHAIN22v.23.qcnf	0.71	66	2.57	22	> 10 min.	-
CHAIN23v.24.qcnf	1.05	69	3.18	23	> 10 min.	-
TOILET10.1.iv.19.qcnf	> 10 min.	-	> 10 min.	-	> 10 min.	-
TOILET10.1.iv.20.qcnf	1.54	19	5.84	18	5.83	18
TOILET16.1.iv.31.qcnf	> 10 min.	-	> 10 min.	-	> 10 min.	-
TOILET16.1.iv.32.qcnf	10.83	31	61.51	30	62.22	30
TOILET2.1.iv.3.qcnf	0.01	3	0.01	0	0.01	0
TOILET2.1.iv.4.qcnf	0.00	3	0.00	2	0.03	2
TOILET6.1.iv.11.qcnf	554.59	111102	46.63	1658	46.52	1658
TOILET6.1.iv.12.qcnf	0.20	11	0.51	10	0.53	10
TOILET7.1.iv.13.qcnf	> 10 min.	-	659.26	17851	> 10 min.	-
TOILET7.1.iv.14.qcnf	0.42	13	1.58	12	1.61	12
R3CNF_150_3_15_2.50_0.T.qcnf	0.82	549	0.05	5	0.05	5
R3CNF_150_3_15_2.50_1.F.qcnf	5.75	3388	0.59	24	0.67	24
R3CNF_150_3_15_2.50_2.T.qcnf	1.54	859	0.13	9	0.15	9
R3CNF_150_3_15_2.50_3.T.qcnf	0.38	300	0.06	6	0.09	6
R3CNF_150_3_15_2.50_4.T.qcnf	2.26	1394	0.32	17	0.34	17
R3CNF_150_3_15_2.50_5.T.qcnf	0.69	522	0.18	9	0.23	11
R3CNF_150_3_15_2.50_6.F.qcnf	14.59	11165	0.92	31	1.14	22
R3CNF_150_3_15_2.50_7.F.qcnf	11.79	7460	1.58	51	1.96	41
R3CNF_150_3_15_2.50_8.F.qcnf	19.34	9865	0.87	34	0.88	33
R3CNF_150_3_15_2.50_9.T.qcnf	0.66	423	0.11	7	0.11	7
impl02.qcnf	0.01	11	0.01	1	0.00	0
impl04.qcnf	0.02	91	0.02	14	0.01	0
impl06.qcnf	0.12	563	0.09	117	0.01	0
impl08.qcnf	0.90	3249	0.96	713	0.05	0
impl10.qcnf	5.62	18435	13.21	4097	0.27	0
impl12.qcnf	25.92	104193	19.59	23223	1.32	0
impl14.qcnf	112.71	588383	118.66	131225	7.21	0
impl16.qcnf	595.66	3322021	650.82	740999	33.72	0
impl18.qcnf	> 10 min.	-	> 10 min.	-	126.96	0
impl20.qcnf	> 10 min.	-	> 10 min.	-	579.23	0

Table 3. Runtimes the basic QBF solver on 150 variable ∃∀∃ QBF. The columns correspond to an increasing clauses-to-variables ratio, and the rows correspond to an increasing percentage of universal variables.

	1.00	1.50	2.00	2.50	3.00	3.50	4.00
0.0%	5.14	8.80	18.62	27.11	34.16	51.22	313.54
5.3%	5.19	9.43	152.52	1874.86	5679.83	4225.16	988.06
10.7%	5.29	10.97	847.85	6815.47	3922.47	1141.70	450.39
16.0%	5.60	16.67	1369.13	1985.74	869.00	402.73	232.92
21.3%	5.60	28.63	491.05	438.56	281.99	207.00	134.24
26.7%	5.84	43.45	149.33	140.81	120.16	98.20	86.07
32.0%	6.27	39.42	63.22	64.31	65.86	60.30	59.77
37.3%	7.09	23.06	33.59	36.49	40.14	43.78	47.44
42.7%	8.07	15.91	21.08	26.02	30.12	32.94	37.32
48.0%	8.70	15.52	20.65	21.10	25.49	27.60	29.01
53.3%	6.98	9.68	12.56	15.54	18.98	21.61	25.08

Table 4. Runtimes on 150 variable QBF with the new propagation algorithm

	1.00	1.50	2.00	2.50	3.00	3.50	4.00
0.0%	5.20	9.06	19.20	28.35	36.26	54.76	392.60
5.3%	5.82	12.00	80.59	107.53	369.88	439.06	153.00
10.7%	5.97	14.26	80.64	382.74	240.94	122.04	84.91
16.0%	6.00	19.93	195.35	228.58	131.70	126.41	85.20
21.3%	6.86	34.37	181.10	173.56	140.23	95.50	88.40
26.7%	7.19	56.05	164.50	160.94	117.94	100.16	87.78
32.0%	7.87	74.94	199.48	141.58	106.85	97.14	96.59
37.3%	10.03	84.98	155.40	119.27	97.54	94.36	101.35
42.7%	12.81	75.78	109.68	94.22	84.99	88.74	118.92
48.0%	17.88	67.29	84.90	73.21	78.43	88.52	106.42
53.3%	14.18	39.02	51.43	62.12	67.55	82.78	100.19

Table 5. The ratio between the runtimes in Tables 4 and 3

	1.00	1.50	2.00	2.50	3.00	3.50	4.00
0.0%	1.011	1.029	1.031	1.045	1.061	1.069	1.252
5.3%	1.121	1.272	0.528	0.057	0.065	0.103	0.154
10.7%	1.128	1.299	0.095	0.056	0.061	0.106	0.188
16.0%	1.071	1.195	0.142	0.115	0.151	0.313	0.365
21.3%	1.225	1.200	0.368	0.395	0.497	0.461	0.658
26.7%	1.231	1.289	1.101	1.142	0.981	1.019	1.019
32.0%	1.255	1.901	3.155	2.201	1.622	1.610	1.616
37.3%	1.414	3.685	4.626	3.268	2.429	2.155	2.136
42.7%	1.587	4.763	5.203	3.621	2.821	2.693	3.186
48.0%	2.055	4.335	4.111	3.469	3.076	3.207	3.668
53.3%	2.031	4.030	4.094	3.997	3.559	3.830	3.994

handle the existential and the universal variables. Cadoli et al. also give a pure literal rule for universal variables, and propose a test that detects the truth of a QBF by performing a satisfiability test of the QBF with the universal variables ignored. Giunchiglia et al. [8] generalize backjumping [17] to QBF so that also universal variables can be jumped over. Backjumping speeds up the evaluation of some classes of randomly generated QBF substantially. Also Letz [11] discusses backjumping, as well as other QBF extensions of techniques that have been used in implementations of the Davis-Putnam procedure. The techniques discussed by Cadoli et al., Giunchiglia et al. and Letz in general improve the runtimes, backjumping on certain randomly generated problems substantially, but on the kind of structured problems discussed in Sect. 7.1 the implementations of all these algorithms have the same extremely high runtimes as our implementation without the stronger propagation algorithm.

Plaisted et al. [16] have presented a decision procedure for QBF that is not based on the Davis-Putnam procedure. The procedure recursively eliminates variables from a formula by repeatedly replacing a subformulae by another that allows same valuations for the variables that occur also outside the subformula but does not contain the variables that occur only in the original subformula. No comparison of the algorithm to other algorithms for evaluating QBF has been carried out, because only an implementation of the algorithm for the unquantified propositional case exists.

Work directly related to the new propagation algorithm has been presented earlier by us and other authors. A restricted form of partial unfolding was first considered by Rintanen [20]. The technique presented there is capable of obtaining all unit resolution steps only when there is one group of universal variables; that is, when the prefix is $\exists\forall\exists$. The technique is applicable to longer prefixes, but in those cases it is incomplete. A variant of the technique was presented by Feldmann et al. [5], but experiments by Giunchiglia et al. [8] suggest that it is not a proper improvement over Rintanen's original proposal.

The unit resolution problem is not restricted to the Davis-Putnam QBF procedure: it is also present in algorithms for other problems, most notably in algorithms for stochastic satisfiability [13]. The propagation algorithm could be also applied in algorithms for QBF that are not in CNF. In this context an important question is detecting – from the non-clausal formula – the possibilities of performing inference steps corresponding to unit resolution. Once this question is answered, application of the propagation algorithm with its improvements is straightforward.

The new propagation algorithm for QBF could be contrasted to the work by Ginsberg and Parkes [7] which generalizes the Davis-Putnam procedure to schematically represented propositional formulae. Ginsberg and Parkes consider quantification over constant symbols, and restrict to universal quantification. Both algorithms process conventional propositional formulae that are represented compactly (exponential size reduction), and an intractable subproblem emerges because of the exponential reduction in problem size. In Ginsberg and Parkes' algorithm clauses and propositional variables are represented schematically, for example $p(a, b)$ where a and b are members of a fixed set of constants, and the computational problem to be solved is to find out – given a partial valuation and a set of schematically represented clauses – whether there are unit clauses with the literal $p(a, b)$ or $\neg p(a, b)$. Note that Ginsberg and Parkes represent only the clause set implicitly; all of the parametric propositional variables are represented explicitly. In the QBF case, the clauses and existential variables are parameterized by the universal variables that occur in the prefix before the relevant existential variables. Our algorithm performs unit resolution with the implicitly represented unquantified clause set. We do not represent the parameterized variables explicitly (there is an exponential number of them), and only infer truth-values for the current outermost existential variables.

9 Conclusions

We have presented a propagation algorithm for QBF that takes advantage of the possibility of partially unfolding the QBF, that is, making explicit part of the propositional formula that would be obtained by eliminating the universal variables from the QBF. The algorithm tries to infer truth-values for the outermost existential variables in the QBF, thereby reducing the need for exhaustive case-analysis on those variables. The algorithm may need exponential time because the fully unfolded propositional formula can have a size exponential in the size of the QBF. We discussed improvements to the algorithm and restrictions that make the algorithm run in polynomial time.

We investigated the behavior of the algorithm on a narrow class of formulae that was part of the initial motivation for studying the problem, and on these formulae the algorithm is a differentiating factor between practically unsolvable and easily solvable QBFs. Whether the algorithm is useful on more general classes of QBF remains to be seen. To investigate the topic further we would need QBF with three or more alternations of quantifiers in the prefix. For QBF with prefix $\exists\forall\exists$ the algorithm works like a technique earlier proposed by us [20]. We believe that on many QBF with a longer prefix the hierarchical propagation algorithm substantially reduces the need for exhaustive search. However, the overhead of the algorithm is relatively high, and when the reduction in search space does not take place, the propagation algorithm slows down QBF evaluation.

There are some areas in QBF implementations that potentially benefit from observing the presence of the new propagation algorithm. For example, branching heuristics should prefer variables that increase the number of clauses that contain only one existential variable (and possibly some universal variables.) The branching heuristics of our current implementation just count the number of new one and two literal clauses. However, for most of the QBF obtained by translation from the planning problems discussed in this paper, this would not appear to make a difference, because there are few clauses that contain universal variables and more than one existential variable.

The ideas behind the paper point to possible improvements to the Davis-Putnam QBF procedure. A general problem with the procedure is that branching variables have to be selected from the variables quantified by the outermost quantifier, and therefore the choice of the variables is much less flexible than in the unquantified case. The problem is that – at a given state of the search process – none of the values chosen for the outermost variables might immediately constrain the values of the remaining variables, which leads to blind and exhaustive search. The idea of viewing the QBF as explicitly standing for an unquantified propositional formulae suggests that branching variables could be inner variables when they are viewed as being parameterized by the values of the preceding universal variables. It could be the case that assigning a truth-value to some of the inner variables could constrain the other variables considerably, which would reduce the search space. However, because the number of parametric variables can be exponential in the number of variables in the QBF, it is not clear how and why this would lead to more efficient evaluation of QBF.

References

[1] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear time algorithm for testing the truth of certain quantified Boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979. Erratum in 14(4):195, June 1982.

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

[3] M. Cadoli, A. Giovanardi, and M. Schaerf. An algorithm to evaluate quantified Boolean formulae. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98) and the Tenth Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 262–267. AAAI Press, July 1998.

[4] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[5] Rainer Feldmann, Burkhard Monien, and Stefan Schamberger. A distributed algorithm to evaluate quantified Boolean formulae. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000) and the Twelfth Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, pages 285–290. AAAI Press, 2000.

[6] Ian Gent and Toby Walsh. Beyond NP: the QSAT phase transition. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) and the Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, pages 648–653. AAAI Press / The MIT Press, 1999.

[7] Matthew L. Ginsberg and Andrew J. Parkes. Satisfiability algorithms and finite quantification. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*, pages 690–701. Morgan Kaufmann Publishers, 2000.

[8] Enrico Giunchiglia, Massimo Narizzano, and Armando Tacchella. An analysis of backjumping and trivial truth in quantified Boolean formulas satisfiability. In F. Esposito, editor, *AI*AI 2001: Advances in Artificial Intelligence. 7th Congress of the Italian Association for Artificial Intelligence, Proceedings*, number 2175 in Lecture Notes in Computer Science, pages 111–122. Springer-Verlag, 2001.

[9] Henry Kautz and Bart Selman. Pushing the envelope: planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial*

Intelligence Conference, pages 1194–1201, Menlo Park, California, August 1996. AAAI Press.

[10] Hans Kleine Büning, Marek Karpinski, and Andreas Flögel. Resolution for quantified Boolean formulas. *Information and Computation*, 117:12–18, 1995.

[11] Reinhold Letz. Advances in decision procedures for quantified Boolean formulas. working notes of the IJCAR 2001 workshop on Theory and Applications of Quantified Boolean Formulas, 2001.

[12] Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, pages 366–371, Nagoya, Japan, August 1997.

[13] Michael L. Littman. Initial experiments in stochastic satisfiability. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99) and the Eleventh Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, pages 667–672. AAAI Press / The MIT Press, 1999.

[14] Antonio Lozano and José L. Balcázar. The complexity of graph problems for succinctly represented graphs. In Manfred Nagl, editor, *Graph-Theoretic Concepts in Computer Science, 15th International Workshop, WG'89*, number 411 in Lecture Notes in Computer Science, pages 277–286, Castle Rolduc, The Netherlands, 1990. Springer-Verlag.

[15] Christos H. Papadimitriou and Mihalis Yannakakis. A note on succinct representations of graphs. *Information and Control*, 71:181–185, 1986.

[16] David A. Plaisted, Armin Biere, and Yunshan Zhu. A satisfiability procedure for quantified Boolean formulae, 2001. unpublished.

[17] Patrick Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3):268–299, 1993.

[18] Jussi Rintanen. A planning algorithm not based on directional search. In A. G. Cohn, L. K. Schubert, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Sixth International Conference (KR '98)*, pages 617–624. Morgan Kaufmann Publishers, June 1998.

[19] Jussi Rintanen. Constructing conditional plans by a theorem-prover. *Journal of Artificial Intelligence Research*, 10:323–352, 1999.

[20] Jussi Rintanen. Improvements to the evaluation of quantified Boolean formulae. In Thomas Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1192–1197. Morgan Kaufmann Publishers, August 1999.

[21] Roberto Sebastiani. Personal communication, Breckenridge, Colorado, April 2000.