Message passing for quantified Boolean formulas

Pan Zhang[1]**, Abolfazl Ramezanpour**[1]**, Lenka Zdeborová**[2] **and Riccardo Zecchina**[1]

[1] Politecnico di Torino, Corso Duca degli Abruzzi 24, I-10129 Torino, Italy
[2] Institut de Physique Theorique, IPhT, CEA Saclay, and URA 2306, CNRS, 91191 Gif-sur-Yvette, France
E-mail: july.lzu@gmail.com, abolfazl.ramezanpour@polito.it, lenka.zdeborova@gmail.com and riccardo.zecchina@polito.it

Abstract. We introduce two types of message passing algorithms for quantified Boolean formulas (QBF). The first type is a message passing based heuristics that can prove unsatisfiability of the QBF by assigning the universal variables in such a way that the remaining formula is unsatisfiable. In the second type, we use message passing to guide branching heuristics of a Davis–Putnam–Logemann–Loveland (DPLL) complete solver. Numerical experiments show that on random QBFs our branching heuristics give robust exponential efficiency gain with respect to state-of-the-art solvers. We also manage to solve some previously unsolved benchmarks from the QBFLIB library. Apart from this, our study sheds light on using message passing in small systems and as subroutines in complete solvers.

Keywords: analysis of algorithms, heuristics, message-passing algorithms, typical-case computational complexity

J. Stat. Mech. (2012) P05025

Contents

1. Introduction

Satisfiability of quantified Boolean formulas (QBF) is a generalization of the Boolean satisfiability problem (SAT) where universal quantifiers are added to the existential ones. QBF are useful in modeling practical problems harder than NP (set of decision problems that can be verified in polynomial time by a non-deterministic Turing machine), e.g. planning, verification or combinatorial game playing. Algorithmic complexity of QBF ranges in the polynomial hierarchy up to PSPACE (set of decision problems that can be solved by a Turing machine using a polynomial amount of space).

Message passing algorithms are used in a wide range of algorithmically hard problems, from constraint satisfaction problems such as the satisfiability problem [1] to gene regulation network reconstruction [2], error correcting codes [3], or compressed sensing [4]. Message passing algorithms can be very efficient for large random systems where complete algorithms cannot be applied. However, on small systems sizes or for structured SAT problems the use of message passing has been so far very limited. For instance there have been attempts to solve small satisfiability problems by using message passing in a complete algorithm [5]. However, according to these result it seems that message passing is helpful only for satisfiable random formulas, where complete algorithms are less efficient than stochastic local search algorithms.

In this paper we introduce two types of message passing algorithms for quantified Boolean formulas, one as a heuristic scheme and another as a subroutine for a complete solver. We mainly consider QBF with two alternations $\forall X \exists Y \Phi$ with X corresponding to the set of universal variables, Y corresponding to the set of existential variables, and Φ being a CNF formula. Such QBF are said to be satisfiable if for all configurations of the universal variables, there exists an assignment of the existential variables that satisfies the formula. Examples for the case with more alternations are presented in section 3.

Solving a QBF is in general more difficult than solving a SAT formula. In the case of random SAT formulas, the state-of-art SAT solvers e.g. kcnfs [6] or march [7] can solve hard satisfiable 3-SAT instances with 600 variables, while the state-of-art QBF solvers, e.g. QuBE7.2 [8], can solve hard random QBF with number of variables only up to 60. The

intuitive reason for this difference is that a much larger search space needs to be explored in QBF. Another reason is that, whereas for SAT formulas there are several good decision heuristics based on e.g. look-ahead [9] or backbone-search [10], for QBF efficient decision heuristics are missing. Here we introduce belief propagation based decision heuristics that provide considerable speed up to the state of the art QBF solvers. A similar attempt has been done in [11] where, however, the use of survey propagation instead of belief propagation was unfortunate as we discuss briefly later.

For the SAT problem, besides complete algorithms based on Davis–Putnam–Logemann–Loveland (DPLL), see e.g. chapters 2, 3, and 4 of [12], or resolution, there are also many efficient heuristic algorithms, see e.g. chapter 4 of [12]. However, to the best of our knowledge, all algorithms proposed for QBF were complete except one [13]. Here we introduce a new message passing based heuristics for proving unsatisfiability that improves on the algorithm of [13].

2. Definitions and formulas

In this section we remind ourselves of the definition of random QBF that we use as benchmarks for our algorithms [14, 15]. We also remember the standard formulas for belief propagation and survey propagation update rules [1, 16, 17].

To express QBF with t alternations, we use $Q_1V_1Q_2V_2\cdots Q_tV_t\Phi$ where Q_n denotes quantifier $\exists$ or $\forall$ at nth alternation, V_n denotes the set of variables at nth alternation and Φ denotes the set of clauses. We evaluate satisfiability of QBF in the following way. If one is able to find an assignment of the universal variables for which no solution exists, the QBF is said to be unsatisfiable, otherwise it is satisfiable. For instance, $\forall X\exists Y\Phi$ denotes QBF with two alternations, and the set of universal X and existential Y variables. This formula is satisfiable if for every assignment of X, one can find an assignment for Y such that Φ is satisfied. We will use the notation $N_u \equiv |X|$, $N_e \equiv |Y|$, M for the number of clauses in Φ, $\alpha_e \equiv M/N_e$ and $\alpha_u \equiv M/N_u$.

Several models for random QBF were proposed. In this paper, we consider the (L, K) model [15] and the model-B [14]. In model-B, each clause in Φ has U universal variables and $V > 0$ existential variables that are selected randomly from the whole set of universal (respectively existential) variables. The (L, K) model is a special case of model-B formula with two alternations, which specifies a formula $\forall X\exists Y\Phi$ where each clause in Φ contains $L + K$ variables, L from X and K from Y.

In message passing algorithms, belief propagation (BP) [16] and survey propagation (SP) [1], we define $\{\psi_i^s\}$ to be the marginal probability that variable i takes assignment s among all the solutions (BP) or among all the solution clusters (SP). In BP s has two possible values $+$ or $-$, with $\psi_i^+ + \psi_i^- = 1$, in SP s has three possible values $+$, $*$ and $-$ with $\psi_i^+ + \psi_i^* + \psi_i^- = 1$. We say that a variable is biased if $\psi_i^+ \neq \psi_i^-$, the larger the difference the larger bias the variable has. If $\psi_i^+ > \psi_i^-$ we define the bias of i to be ψ_i^+, and if $\psi_i^+ < \psi_i^-$ we define the bias of i to be ψ_i^-.

Let us define ∂_+i as the set of clauses to which i belongs non-negated, and ∂_-i be the set of clauses to which i belongs negated. Then the set of clauses to which variable i belongs can be written as $\partial i = \{a\} \cup \mathcal{S}_{ia} \cup \mathcal{U}_{ia}$ where: (a) if i is not negated in a then $\mathcal{S}_{ia} = \partial_+i \setminus a$, $\mathcal{U}_{ia} = \partial_-i$ and (b) if i is negated in a then $\mathcal{S}_{ia} = \partial_-i \setminus a$, $\mathcal{U}_{ia} = \partial_+i$.

The BP marginals are computed as

$$\psi_i^+ = \frac{\prod_{b \in \partial_- i} u_{b \to i} \prod_{b \in \partial_+ i} (1 - u_{b \to i})}{\prod_{b \in \partial_+ i} u_{b \to i} \prod_{b \in \partial_- i} (1 - u_{b \to i}) + \prod_{b \in \partial_- i} u_{b \to i} \prod_{b \in \partial_+ i} (1 - u_{b \to i})}, \tag{1}$$

where messages $u_{b \to i}$ are a fixed point of the following iterative equations

$$\psi_{i \to a} = \frac{\prod_{b \in \mathcal{S}_{ia}} u_{b \to i} \prod_{b \in \mathcal{U}_{ia}} (1 - u_{b \to i})}{\prod_{b \in \mathcal{S}_{ia}} u_{b \to i} \prod_{b \in \mathcal{U}_{ia}} (1 - u_{b \to i}) + \prod_{b \in \mathcal{U}_{ia}} u_{b \to i} \prod_{b \in \mathcal{S}_{ia}} (1 - u_{b \to i})},$$
$$u_{b \to i} = \frac{1 - \prod_{j \in \partial b \setminus i} \psi_{j \to b}}{2 - \prod_{j \in \partial b \setminus i} \psi_{j \to b}}. \tag{2}$$

Iteration equations of SP are written as:

$$\psi_{i \to a}^U = \frac{1}{C_{i \to a}} \left[1 - \prod_{b \in \mathcal{U}_{ia}} u_{b \to i} \right] \prod_{b \in \mathcal{S}_{ia}} u_{b \to i},$$
$$\psi_{i \to a}^S = \frac{1}{C_{i \to a}} \left[1 - \prod_{b \in \mathcal{S}_{ia}} u_{b \to i} \right] \prod_{b \in \mathcal{U}_{ia}} u_{b \to i},$$
$$\psi_{i \to a}^* = \frac{1}{C_{i \to a}} \prod_{b \in \partial i \setminus a} u_{b \to i}, \tag{3}$$
$$u_{a \to i} = 1 - \prod_{j \in \partial a \setminus i} \psi_{j \to a}^U,$$

where $\psi_{i \to a}^U$ ($\psi_{i \to a}^S$) denotes $\psi_{i \to a}^+$ ($\psi_{i \to a}^-$) if i is negated in a and $\psi_{i \to a}^-$ ($\psi_{i \to a}^+$) if i is not negated in a. $C_{i \to a}$ is a normalization constant ensuring $\psi_{i \to a}^+ + \psi_{i \to a}^* + \psi_{i \to a}^- = 1$. SP marginals are computed by

$$\psi_i^+ = \frac{1}{C_i} \left[1 - \prod_{b \in \partial_+ i} u_{b \to i} \right] \prod_{b \in \partial_- i} u_{b \to i},$$
$$\psi_i^- = \frac{1}{C_i} \left[1 - \prod_{b \in \partial_- i} u_{b \to i} \right] \prod_{b \in \partial_+ i} u_{b \to i}, \tag{4}$$
$$\psi_i^* = \frac{1}{C_i} \prod_{b \in \partial i} u_{b \to i},$$

where C_i is again a normalization constant.

3. Heuristic algorithm for proving unsatisfiability

Proving unsatisfiability for the two-level QBF can be done by finding an assignment of the universal variables that leaves the existential part of the formula unsatisfiable. One strategy is to find a configuration of the universal variables that leaves the largest possible number of clauses unsatisfied; see e.g. WalkMinQBF [13]. This, however, leads to heuristic algorithms that do not use in any way the existential part of the QBF. Here we suggest and
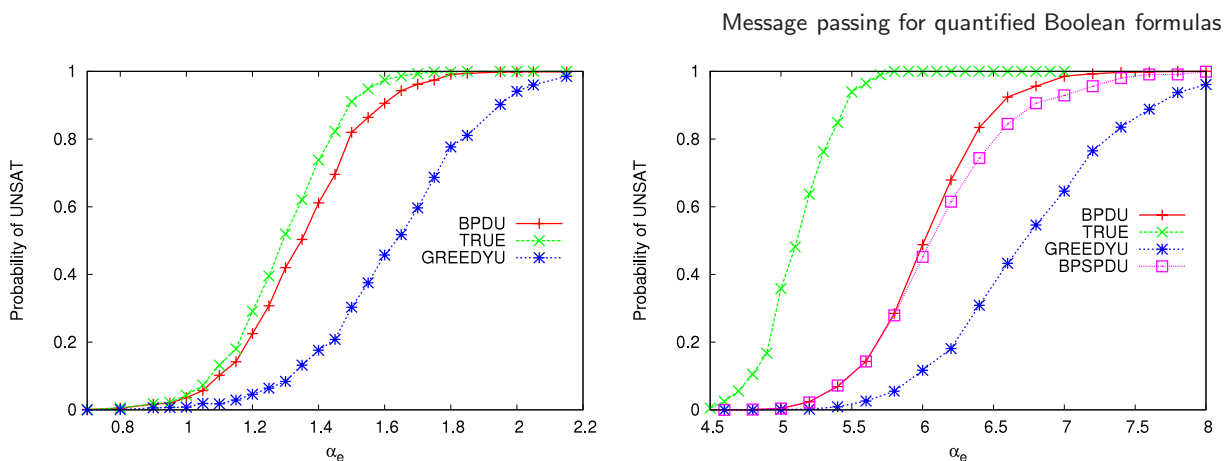
Figure 1. Ratio of unsatisfiable formulas found by the various algorithms discussed in the text for random $(1,2)$ QBF instances with $N_u = N_e = 200$ (left) and $(1,3)$ QBF instances with $N_u = N_e = 50$ (right) as a function of $\alpha_e = M/N_e$, Data are averaged over 1000 instances.

test a belief propagation decimation heuristics for proving unsatisfiability that takes into account the whole formula and outperforms significantly the previously known heuristics.

Our belief propagation decimation heuristics (BPDU) for proving unsatisfiability of QBF works as follows: input is the QBF formula $\forall X \exists Y \Phi$. We disregard for a moment the quantifiers in the QBF formula and run randomly initialized BP on the whole formula (i.e. with both universal and existential variables) until convergence or until the maximum number of allowed iterations $t_{\max}$ is achieved (typically we use $t_{\max} = 300$ in our algorithms). We select the most biased universal variable and fix it against the direction of the bias. We repeat the above steps until all the universal variables are assigned. Then we run a complete SAT solver (in our case kcnfs [6, 10]) on the formula consisting of the existential variables and clauses that were not satisfied by any of the universal variables. If this remaining formula is unsatisfiable then we have proved unsatisfiability of the QBF. If the remaining formula is satisfiable then the algorithm outputs 'unknown', since there might be another configuration of the universal variables that would leave the remaining formula unsatisfiable.

A variation on the above heuristic algorithm is to run survey propagation instead of belief propagation whenever survey propagation converges to a nontrivial fixed point. In what follows we call this variation BPSPDU.

To evaluate the performance of the BPDU and BPSPDU heuristics we first apply them on random $(1,2)$ and $(1,3)$ (which stand for (L,K) instances) QBF instances with varying numbers of clauses (more clauses make the formulas less likely to be satisfiable), see figure 1. For both cases, a complete solver (e.g. QuBe7) can solve every formula hence we can compare the fraction of unsatisfiable formulas found by heuristics to the true fraction. We also show the fraction of unsatisfiable formulas found by the 'greedy' strategy, which fixes the universal variables in order to allow the largest possible number of unsatisfied clauses. In the case of $(1,K)$ QBF this is easy as every universal variable just needs to be set positive if it appears negated in more clauses than non-negated, and vice versa. In figure 1 we see that the BPDU and BPSPDU heuristics perform much better than the greedy strategy, and in the case of $(1,2)$ QBF the performance is not too far from the optimal.
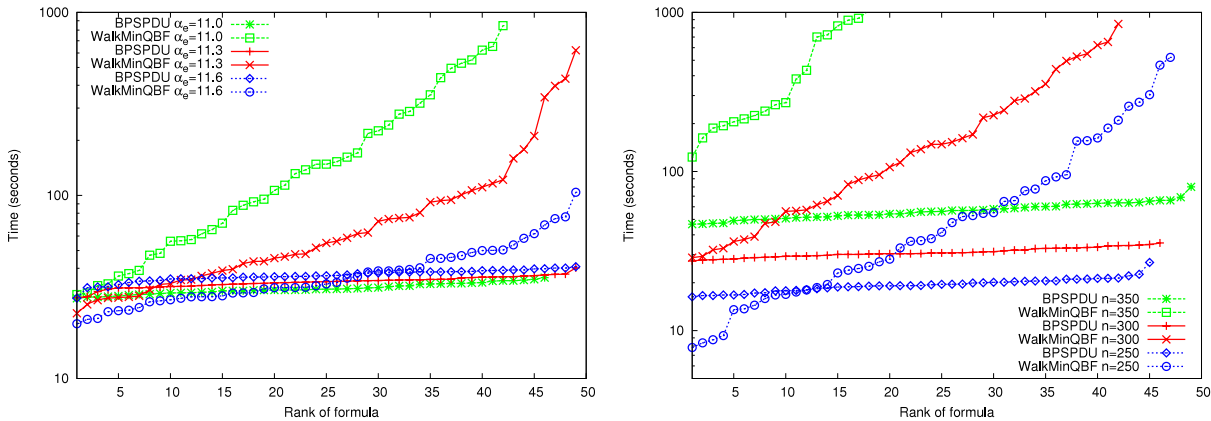
Figure 2. Running time of the BPSPDU heuristics compared to WalkMinQBF on 50 random $(2,3)$ QBF benchmarks with the same number of variables $N_e = N_u = 300$ but different α_e values (left), and the same $\alpha_e = 11.0$ but different number of variable $n = N_e = N_u$ (right).

In figure 2 we compare the performance of the BPSPDU and WalkminQBF heuristics for random $(2,3)$ instances. The WalkMinQBF aims at setting the universal variables in order to maximize the number of unsatisfied clauses, and then evaluates the remaining SAT formula with a complete SAT solver. Note that the complete SAT solver used by WalkminQBF and BPSPDU is the same, kcnfs [6, 10], so the difference in performance comes only from the quality of the universal configuration given by the two heuristics.

As the figure shows, formulas with larger number of clauses are easier for WalkMinQBF, and the running time of the algorithm has larger fluctuations compared to that of BPSPDU. The difference between the two algorithms is clearer for larger problem instances; with $N_e = N_u = 350$, BPSPDU solves 49 out of the 50 instances, and WalkMinQBF solves only 17 of them within 1000 seconds. The above results indicate that the universal configuration suggested by BPSPDU is much better than the one suggested by WalkMinQBF.

BPDU and BPSPDU cannot be applied to more general multi-alternation QBF directly, but note that a multi-alteration QBF can be transformed to a 2-alternation QBF by changing the order of universal and existential variables. For example, given a 4-alternation QBF $\mathcal{F} = \forall X_1 \exists X_2 \forall X_3 \exists X_4 \Phi$, we can arrive at $\mathcal{F}' = \forall X_1 X_3 \exists X_2 X_4$ by switching the order of X_2 and X_3. One can prove that if $\mathcal{F}'$ is unsatisfiable, then $\mathcal{F}$ is unsatisfiable. The heuristic algorithm can be used to prove the unsatisfiability of $\mathcal{F}'$, and so that of $\mathcal{F}$. We did not test the idea of generalization of heuristic solver by switching quantifier order, we leave this to future work.

4. Message passing to guide QBF complete solvers

When the BPDU or the BPSPDU algorithms introduced in section 6 output 'unknown', there might be another configuration of the universal variables that makes the formula unsatisfiable. Given that we were fixing the universal variables starting with the most biased one, it might be a good strategy to backtrack on the variables fixed in the later

stages. In this section we extend this idea into a complete DPLL-style solver, which is using message passing to decide on which variables to branch next.

DPLL-style algorithms are the most efficient complete solvers for SAT and QBF, they search the whole configurational space by backtracking. The difference between DPLL for SAT and for QBF is that in QBF DPLL does backtracking on existential variable when it encounters a contradiction, and does backtracking on universal variable when it encounters a solution, see e.g. [18] for details. Besides the basic DPLL backtracking procedure, there are several components in modern SAT and QBF solvers that lead to exponential speeding up, among the important ones are decision heuristics, unit-clause propagation, non-chronological back-jumping, conflict and solution driven clause learning [18, 20]. Our contribution concerns the *decision heuristics* which is used in order to decide which variable will be used in the next branch and which sign of the variable should be checked first. Decision heuristics guide DPLL to the more relevant branches and keeps it away from irrelevant branches.

Here we propose a decision heuristics that uses information coming from the result of belief propagation (that was iterated until convergence or for t_{max} steps on the whole formula ignoring the quantifiers). We propose to start branching with the more biased universal variables and assign them first the less probable values. For the existential variables we start branching also on the more biased ones, but assign them the more probable values. The motivation is that this will speed up the search of a universal configuration that will leave the existential part of the QBF unsatisfiable, and the search for a solution on the existential part. In particular we propose two decision heuristics.

In the BPH decision heuristics we simply order the variables according to their bias, starting with the most biased one, and assign them first the value opposite to the bias for the universal variables and according to the bias for the existential ones.

In the BPDH decision heuristics we run BP and choose the most biased unassigned variables which belong to the highest quantifier order. If this variable is universal we assign it the value opposite to its bias, if this variable is existential we assign it according to its bias. We repeat BP on the simplified formula. Finally the BPDH heuristics will branch variables in the same order as they were encountered in this procedure. Most of the good decision heuristics for SAT and QBF solvers are dynamic, which means that the branching sequence is being updated during the run. Our BPH and BPDH decision heuristics are computationally heavier than the other efficient decision heuristics e.g. VSIDS, MOMs or failed-literal-detection [9], so we use it only once at the very beginning of the DPLL run.

We report performance of DPLL with our BPH and BPDH decision heuristics on two levels. The first level is using BPH and BPDH in the pure DPLL (no features such as conflict and solution driving back-jumping and clause learning included). The second level is using BPH in a state-of-art QBF solver QuBE7.2, which is one of the fastest known solvers today.

In figure 3 we plot the number of solutions and the number of conflicts encountered in DPLL using a pure DPLL algorithm and those encountered in DPLL with BPDH and BPH decision heuristics on random $(1, 3)$ QBF formulas. By pure DPLL we mean with no back-jumping nor clause learning, and the default decision heuristics is VSIDS (variable state independent decaying sum) [19, 20], which is based on dynamic statistics of literal count as a score to order literals (in the case of pure DPLL, no learned clause contributes to the literal count). Since pure DPLL is very CPU-time demanding, we use formulas
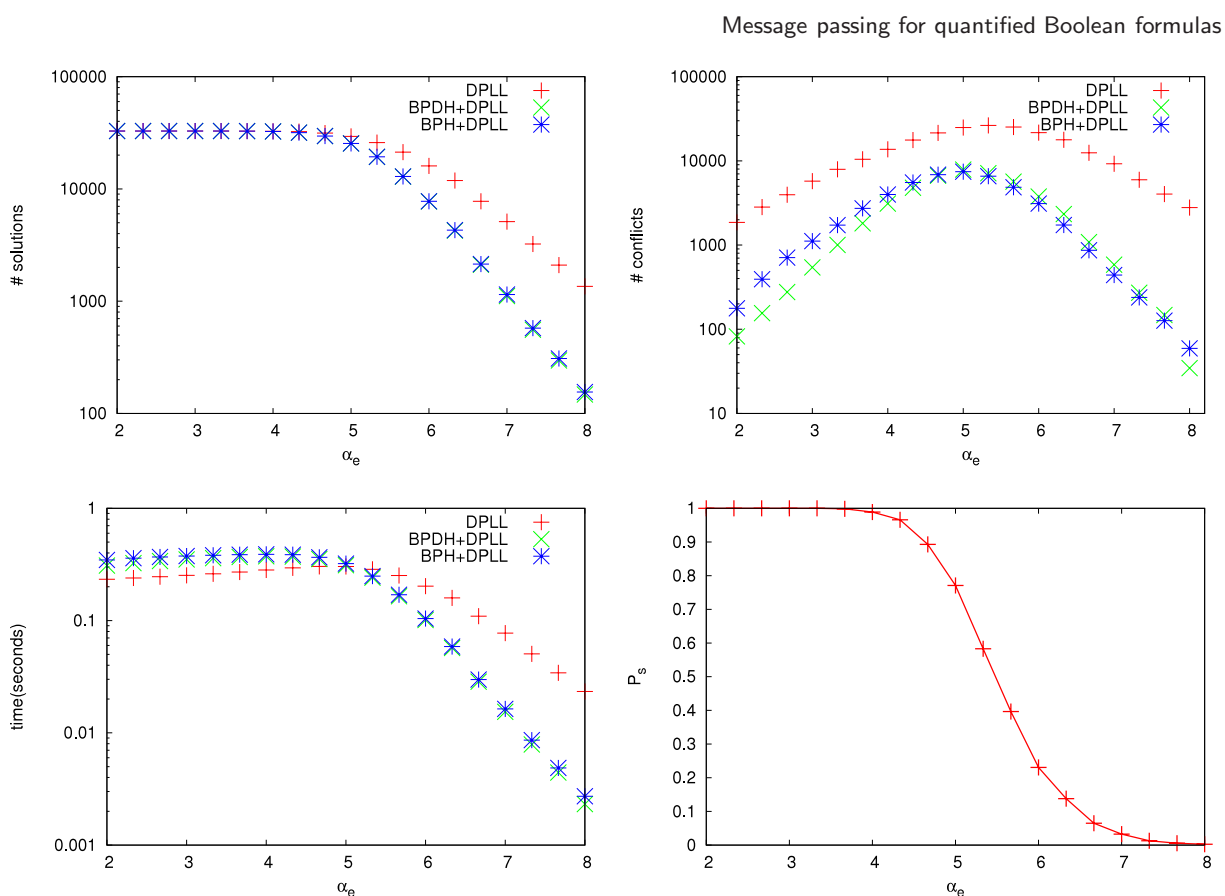
Figure 3. Number of solutions, number of conflicts, running time and ratio of satisfiable formulas P_s, as resulting from pure DPLL for solving random $(1,3)$ formulas, data are averaged over $10\,000$ instances. $N_u = 15$ and $N_e = 15$.

with only $N_u = N_e = 15$ variables. We also plot the ratio of satisfiable formulas P_s, and we can see that when almost all formulas are satisfiable, the average number of solutions encountered in DPLL is always 2^{15}, because to prove the satisfiability of a formula, pure DPLL has to scan all the 2^{N_u} universal configurations. When P_s becomes smaller than one, DPLL with BPDH and BPH encounters a much smaller number of solutions than the pure DPLL. Figure 3 shows that in the whole range of parameters the number of conflicts encountered by DPLL with BPDH and BPH is always much smaller than DPLL with VSIDS decision heuristics. The fewer solutions and conflicts encountered, the smaller is the search tree explored by the algorithm.

A better way to extract the full power of BPH and BPDH in a DPLL-based search is to use solution and conflict driven back-jumping and clause learning. In clause learning, reasons for solutions and conflicts are analyzed and stored as learned clauses in order to implement non-chronological back-jumping to more relevant branches of the search tree and to avoid the encounter of the same solutions or conflicts in the future search. With clause learning, DPLL does not have to explore 2^{N_u} satisfiable leaves of the search tree, and good decision heuristics could lead to a smaller number of solutions. We applied clause learning to the pure DPLL with and without BPH. Results show that with BPH and BPDH, both search tree size and running time are exponentially smaller than without BPH for both unsatisfiable and satisfiable formulas.
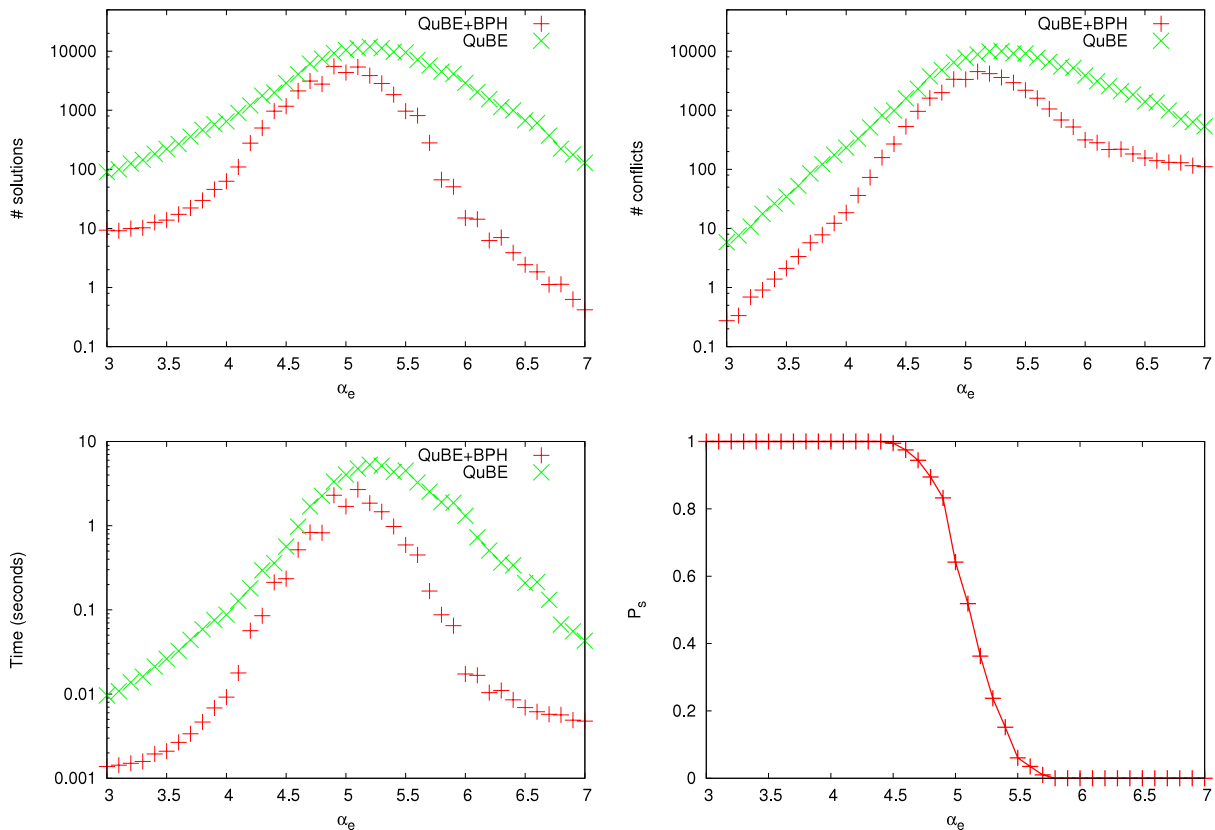
Figure 4. Number of conflicts, solutions and time used by DPLL of QuBE7.2 and QuBE7.2 + BPH in solving random $(1, 3)$ formulas, data are averaged over 200 instances. $N_u = 50$ and $N_e = 50$.

As a next step we implemented message passing decision heuristics in a state-of-the-art solver; we chose QuBE7.2, which uses solution and conflict driven clause learning, as one of the fastest known solver's today (DepQBF [21] won the last evaluation in 2010, but the latest QuBE we are using comes after DepQBF, so it is not clear to us which one is faster), and replaced the decision heuristics in QuBE7.2 by BPH. We have also tried using BPH in other state-of-the-art solvers and found quantitatively similar results.

Our results are presented in figure 4. The power of QuBE enables us to reach larger formulas than we used in figure 3, so our experiments are carried out on random $(1, 3)$ formulas with $N_e = N_u = 50$. From the figures we can see that BPH considerably reduces the size of the search tree as well as the computation time exponentially for a whole range of α_e, the improvement in performance is relatively small only close to the transition region. Figure 5 shows the computational time reduction with the system size. We see that with the same time limit, BPH enables QuBE to solve larger formulas.

Ideas similar to ours have already been explored in [11], where the authors studied an algorithm named HSPQBF that uses SP as the decision heuristics in a QBF solver Quaffle [18]. We see that using BP as the decision heuristics is more stable than SP because SP has a narrow region of working parameter. Recall that in random SAT formulas with the number of variables going to infinity, BP reports correct marginals with constraint density ranging from zero to the condensation transition point [23]. However, SP always has a
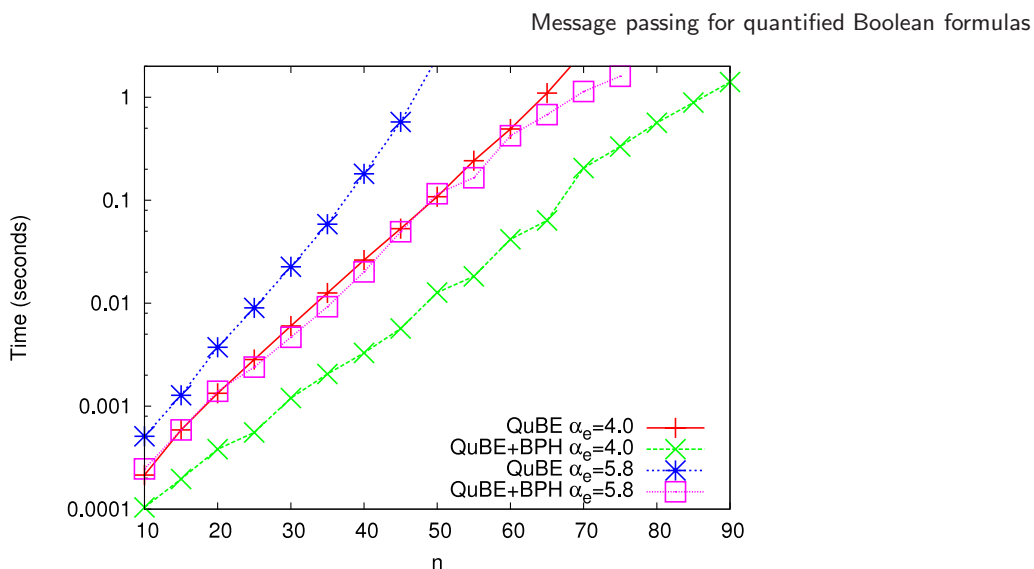
9

Figure 5. Time used by DPLL of QuBE7.2 and QuBE7.2 + BPH in solving random $(1,3)$ formulas with fixed α_e values and a different number of variables n $(N_u = N_e = n)$. Data are averaged over 2000 instances. As shown in figure 4, with $\alpha_e = 4.0$, most formulas are satisfiable and with $\alpha_e = 5.8$, most formulas are unsatisfiable.
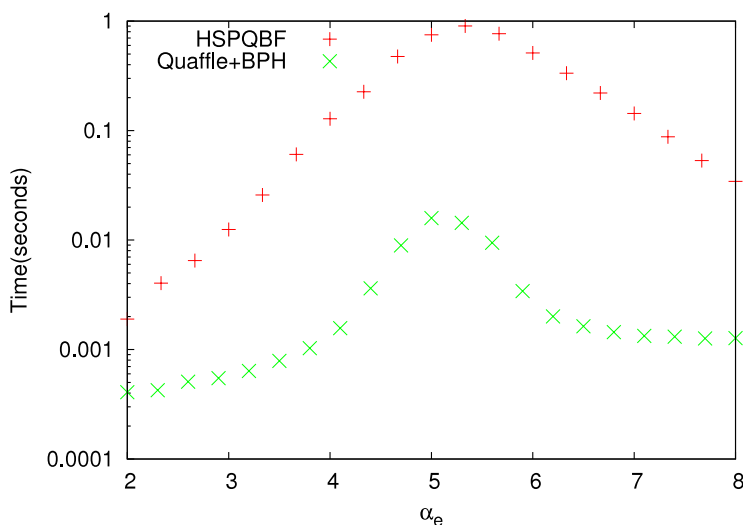

Figure 6. Time used by Quaffle+BPH and by HSPQBF [11] in solving random $(1,3)$ formulas, data are averaged over 2000 instances. $N_u = 30$ and $N_e = 30$.

trivial solution (zero messages) when the constraint density is smaller than a value that lies relatively close to the SAT–UNSAT transition point [1, 24]. Moreover, hard QBF instances are often located in the region where the SAT formula created by ignoring the quantifiers is easy and SP often has a trivial solution, but BP works well. We cross-checked these intuitions using BPH to guide Quaffle, and compared the running time of DPLL in solving random $(1,3)$ instances by BPH guided Quaffle and HSPQBF in figure 6. The figure indicates that in the whole range of α_e, Quaffle with BPH gives better results than HSPQBF. We also checked other types of random formulas and similar results were obtained.
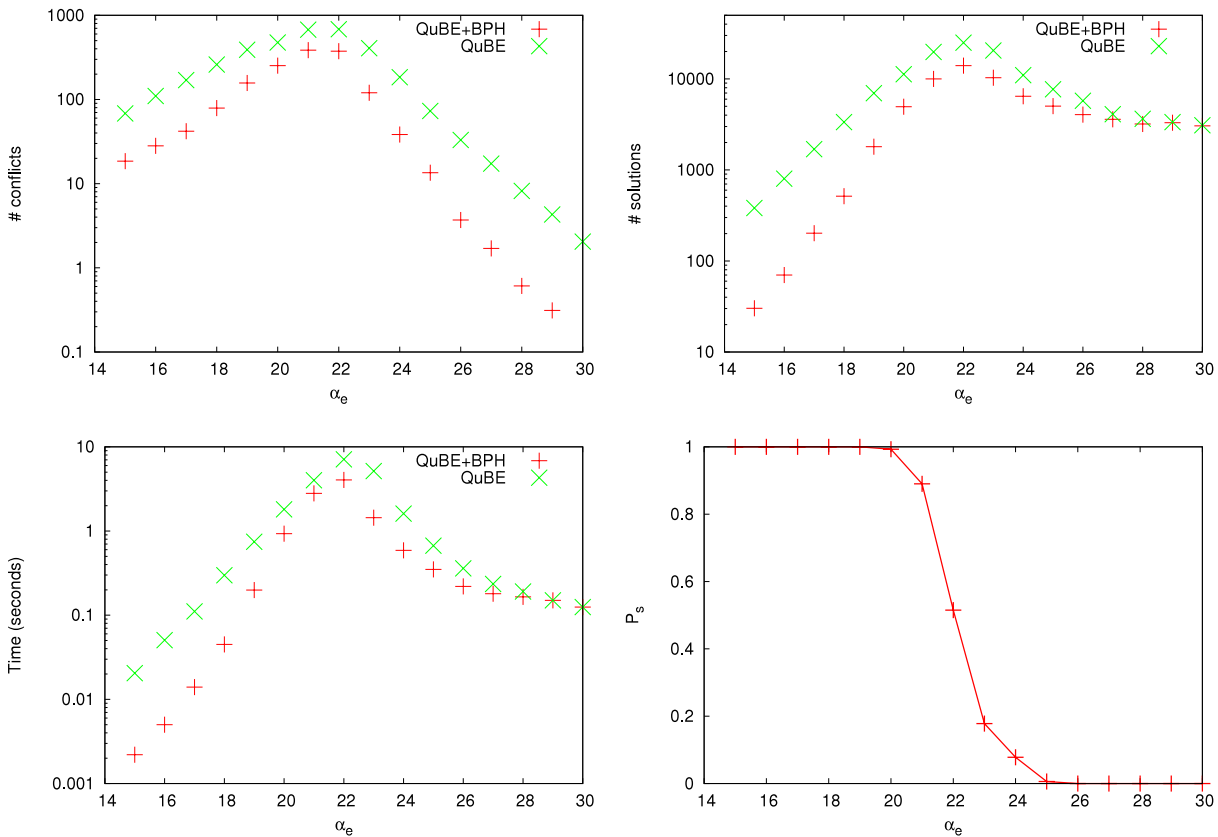
Figure 7. Number of conflicts, solutions and time used by DPLL of QuBE7.2 in solving model-B formulas, data are averaged over 200 instances. Formulas have four alternations and $n = 20$ variables in each alternation. Each clause in the formula contains four existential variables that are selected randomly from all the existential variables (i.e. from the second- and fourth-level variables) and one universal variable that is selected randomly from all the universal variables (i.e. from the first and third level variables). α_e in figure corresponds to the number of clauses divided by n. In generating formula, we use a formula generator downloaded from QBFLIB [22].

Unlike BPDH, BPH works naturally on QBF with multiple alternations. To test performance of BPH in general QBF, we tested model-B formulas with four alternations, the results are plotted in figure 7. As the figure shows, for small α_e BPH improves the performance considerably. However, it gives only small performance improvement for large α_e when all formulas are unsatisfiable.

5. Performance of BPH on structured formulas

In contrast with random formulas, BP usually does not give accurate information about the solution space on structured formulas because of the existence of many short loops. Hence we do not expect BP to improve complete solvers in solving every structured formula. We tested some structured benchmarks from QBFLIB [22]; some of the results

Table 1. Running time (in seconds) of QuBE7.2 and QuBE7.2 + BPH in solving structured benchmarks.

Name of instance	QuBE	QuBE + BPH	Name of instance	QuBE	QuBE + BPH
ii8c1-50	>600	14.21	ev-pr-8x8-15-7-0-1-2-lg	0.023	>600
ncf_16_128_2_edau.8	>600	0.27	flipflop-12-c	0.71	>600
ncf_16_64_8_euad.4	>600	4.01	lut4_3_fAND	0.25	>600
x170.5	520.44	63.42	cf_2_9x9_w_	0.18	539.49
ncf_8_32_4_euad.4	301.86	0.14	ncf_4_32_4_u.9	35.69	565.35
szymanski-5-s	153.00	19.31	c2_BMC_p2_k8	95.49	207.25
k_d4_n-5	408.15	83.94	toilet_a_10_01.15	7.45	510.84
ncf_4_32_8_edau.3	349.78	0.25	cf_3_9x9_d_	0.17	480.27
x170.19	340.05	13.07	connect_9x8_8_D	0.08	115.79
k_grz_p-13	190.66	24.99	stmt21_252_267	3.93	142.49
connect_5x4_3_R	97.51	38.35	k_ph_n-20	7.65	462.44
BLOCKS4i.6.4	49.21	29.87	toilet_a_10_01.11	0.02	416.16
C880.blif_0.10_1.00 _0_1_out_exact	93.21	44.45	x115.4	201.46	302.64
CHAIN18v.19	346.39	288.12	stmt21_143_314	92.61	92.75

are listed in table 1. We can see from the table that in some formulas, BPH increases the performance of QuBE7.2 while in other formulas, BPH decreases the performance. Remarkably, some instances e.g. *ncf_16_64_8_edau.8* problem and *ii8c1-50* problem, which have not been solved by other solvers (in previous QBF evaluations), can be solved by QubE7.2 + BPH in a few seconds.

6. Conclusion and discussion

In this paper we developed heuristic and complete algorithms for QBF based on message passing. Our heuristic algorithms BPDU and BPSPDU use message passing to find a universal assignment that evaluates to an unsatisfiable remaining formula, thus proving the unsatisfiability of 2-alternation QBF. On random formulas, our heuristic algorithms outperform WalkminQBF. Our complete algorithm is based on the DPLL process, which searches the whole configurational space by backtracking more efficiently, using message passing branching heuristics BPDH and BPH. Our algorithms usually give exponential efficiency gain on random QBF and in some cases they provide large improvement also on structured QBF and solve some previously unsolved benchmarks in QBFLIB. The algorithms described above can be downloaded from [25].

In our understanding, the power of BPH comes from the inference of solution-space structure of the underlying system by message passing. If the information is accurate, as shown in random problems [1, 23, 24], with it one is able to drive the search process to a certain target, e.g. to find a solution of a SAT formula [1] or to reduce the size of the search tree in solving QBF as studied in this paper. Our results should encourage further investigation of the use of message passing as heuristic solvers or as guides for heuristics included in DPLL-like complete solvers.

Acknowledgments

References

[1] Mézard M, Parisi G and Zecchina R, *Analytic and algorithmic solution of random satisfiability problems*, 2002 *Science* **297** 812
[2] Weigt M, White R A, Szurmant H, Hoch J A and Hwa T, *Identification of direct residue contacts in protein-protein interaction by message passing*, 2009 *Proc. Nat. Acad. Sci.* **106** 67
[3] Richardson T J and Urbanke R L, *The capacity of low-density parity-check codes under message-passing decoding*, 2000 *IEEE Trans. Inf. Theory* **47** 599
[4] Donoho D L, Maliki A and Montanari A, *Message-passing algorithms for compressed sensing*, 2009 *Proc. Nat. Acad. Sci.* **106** 18914
[5] Hsu E I, Muise C J, Beck J C and McIlraith S A, *From backbone to bias: probabilistic inference and heuristic search performance*, 2008 www.cs.mu.oz.au/cp2008/post-proceedings/106long.pdf
[6] Dequen G and Dubois O, *An efficient approach to solving random k-sat problems*, 2006 *J. Autom. Reasoning* **37** 261
[7] Heule M J H and van Maaren H, *March dl: Adding adaptive heuristics and a new branching strategy*, 2006 *J. Satisfiability, Boolean Modeling Comput.* **2** 47
[8] Giunchiglia E, Narizzano M and Tacchella A, *QuBE: A system for deciding quantified boolean formulas satisfiability*, 2001 *Proc. Int. Joint Conf. on Automated Reasoning (IJCAR'2001)* p 364
[9] Li C M and Anbulagan, *Heuristics based on unit propagation for satisfiability problems*, 1997 *Proc. 15th Int. Joint Conf. on Artifical Intelligence (IJCAI'97)* p 366
[10] Dubois O and Dequen G, *A backbone-search heuristic for efficient solving of hard 3-SAT formulae*, 2001 *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI'01)* p 248
[11] Yin M, Zhou J, Sun J and Gu W, *Heuristic survey propagation algorithm for solving QBF Problem*, 2011 *J. Softw.* **22** 1538
[12] Biere A, Heule M, van Maaren H and Walsh T, 2009 *Handbook of Satisfiability* (Amsterdam: IOS Press)
[13] Interian Y, Corvera G, Selman B and Williams R, *Finding small unsatisfiable cores to prove unsatisfiability of qbfs*, 2006 *9th Int. Symp. on AI and Mathematics (Fort Lauderdale, Florida)* www.cs.cornell.edu/gec27/report/wmq.asp
[14] Gent I and Walsh T, *Beyond NP: the QSAT phase transition*, 1999 *Proc. 16th National Conf. Artificial Intelligence (AAAI/IAAI)* p 648
[15] Chen H and Interian Y, *A model for generating random quantified boolean formulas*, 2005 *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)* p 66
[16] Yedidia J S, Freeman W T and Weiss Y, *Understanding belief propagation and its generalizations*, 2003 *Exploring Artificial Intelligence in the New Millennium* ed G Lakemeyer and B Nebel (San Francisco: Morgan Kaufmann) ch 8, p 239
[17] Braunstein A, Mézard M and Zecchina R, *Survey propagation: an algorithm for satisfiability*, 2005 *Random Struct. Alg.* **27** 201
[18] Zhang L and Malik S, *Towards a symmetric treatment of satisfaction and conflicts in quantified Boolean formula evaluation*, 2002 *Proc. 8th Int. Conf. on Principles and Practice of Constraint Programming (CP'2002)* p 200
[19] Moskewicz M, Madigan C, Zhao Y, Zhang L and Malik S, *Chaff: engineering an efficient SAT solver*, 2001 *Proc. 39th Design Automation Conf. (DAC'01)* p 530
[20] Zhang L, Madigan C F, Moskewicz M H and Malik S, *Efficient conflict driven learning in a Boolean satisfiability solver*, 2001 *Proc. Int. Conf. on Computer-Aided Design (ICCAD'01)* p 279
[21] Lonsing F and Biere A, *DepQBF: a dependency-aware QBF Solver*, 2010 *J. Satisfiability Boolean Modeling Comput.* **7** 71
[22] Giunchiglia E, Narizzano M and Tacchella A, 2001 *QBFLIB: Quantified Boolean Formulas Satisfiability Library* www.qbflib.org
[23] Krzakala F, Montanari A, Ricci-Tersenghi F, Semerjian G and Zdeborová L, *Gibbs states and the set of solutions of random constraint satisfaction problems*, 2007 *Proc. Nat. Acad. Sci.* **104** 10318
[24] Mertens S, Mézard M and Zecchina R, *Threshold values of random K-SAT from the cavity method*, 2006 *Random Struct. Alg.* **28** 340
[25] http://panzhang.net/qbf