The second QBF solvers comparative evaluation*

Daniel Le Berre[1], Massimo Narizzano[3],
Laurent Simon[2], and Armando Tacchella[3]

[1] CRIL, Université d'Artois,
Rue Jean Souvraz SP 18 – F 62307 Lens Cedex, France
leberre@cril.univ-artois.fr
[2] LRI, Université Paris-Sud
Bâtiment 490, U.M.R. CNRS 8623 – 91405 Orsay Cedex, France
simon@lri.fr
[3] DIST, Università di Genova,
Viale Causa, 13 – 16145 Genova, Italy
{mox,tac}@.dist.unige.it

Abstract. This paper reports about the 2004 comparative evaluation of solvers for quantified Boolean formulas (QBFs), the second in a series of non-competitive events established with the aim of assessing the advancements in the field of QBF reasoning and related research. We evaluated sixteen solvers on a test set of about one thousand benchmarks, selected from instances submitted to the evaluation and from those available at www.qbflib.org. In the paper we present the evaluation infrastructure, from the criteria used to select the benchmarks to the hardware set up, and we show different views about the results obtained, highlighting the strength of different solvers and the relative hardness of the benchmarks included in the test set.

1 Introduction

The 2004 comparative evaluation of solvers for quantified Boolean formulas (QBFs) is the second in a series of non-competitive events established with the aim of assessing the advancements in the field of QBF reasoning and related research. The non-competitive nature of the evaluation is meant to encourage the developers of QBF reasoning tools and the users of QBF technology to submit their work. Indeed, our evaluation does not award one particular solver, but instead draws a picture of the current state of the art in QBF solvers and benchmarks. Running the evaluation enables us to collect data regarding the strength of different solvers and methods, the relative hardness of the benchmarks, and to shed some light on the open issues for the researchers in the QBF community.

With respect to last year evaluation [1] we have witnessed to an almost 50% increase in the number of submitted solvers (from eleven to sixteen). While most of the participants are still complete solvers extending the well-known Davis,

* The authors would like to thank all the participants to the QBF evaluation for submitting benchmarks and solvers.

Putnam, Logemann, Loveland procedure (DPLL) [2, 3] for propositional satisfiability (SAT), the evaluation also hosted one incomplete solver (WalkQSAT [4]), a solver based on Q-resolution and expansion of quantifiers (quantor [5]), and a solver using ZBDDs (Zero-suppressed Binary Decision Diagrams) to obtain a symbolic implementation of the original DP algorithm extended to QBFs (QMRes [6]). The number of the participants and the variety of the technologies deployed confirm the vitality of the research on QBF reasoning tools. Regarding applications of QBF reasoning, three families of benchmarks obtained by encoding formal verification problems have been submitted, for a total of 88 instances. To these we must add 22 families and 814 instances that have been independently submitted to `www.qbflib.org` and that have been evaluated this year for the first time. Finally, the submission of a generator for model A and model B random instances [7] allowed us to run the participating solvers on a wide selection of instances generated according to these probabilistic models.

The evaluation consisted of two steps: (*i*) running the solvers on a selection of benchmarks, and (*ii*) analyzing the results. The first step is subject to the stringent requirements of getting meaningful results and completing the evaluation in a reasonable time. In order to mate these two requirements, we have extracted the non-random part of the evaluation test set by sampling the pool of available benchmarks – more than 5000 benchmarks and 68 families – to extract a much smaller, yet representative, test set of about five hundred instances. To these, we added representatives from model A and model B families of random benchmarks to obtain the final test set. Using the selected test set, we have completed the first step by running the solvers on a farm of PCs, each solver being restricted to the same amount of time and memory. The second step consisted of two stages. In the first stage we considered all the solvers and the benchmarks of the test set to give a rough, but complete, picture of the state of the art in QBF. By analyzing the results for problems and discrepancies among the solvers results, we were able to isolate solvers and instances that turned out to be problematic, and we removed them from the subsequent analysis. The second stage is an in-depth account of the results, where we tried to extract a narrow, but crisp, picture of the current state of the art.

The paper is structured as follows. In Section 2 we briefly describe all the QBF solvers that participated in the evaluation and all the benchmarks that we used to construct the test set. In Section 3 we present the first step of the evaluation, i.e., the choice of the test set and a description of the computing infrastructure where the evaluation ran. In Section 4 we present the results of the evaluation first stage, including a discussion about discrepancies in the results of the solvers and the problems detected in the benchmarks. In Section 5 we restrict our attention to the solvers and the benchmarks that passed the first stage and we present the result of the evaluation arranged solver-wise in Sub. 5.1, and benchmark-wise in Sub. 5.2. We conclude the paper in Section 6 with a balance about the evaluation, including a discussion of some open problems and challenges for the QBF community at large.

2 Solvers and Benchmarks

Sixteen solvers from eleven different authors participated to the evaluation this year. The requirements for the solvers where to be able to read the input instance in a common format (the Q-DIMACS format [8]), and to provide the answer (sat/unsat) in a given output format. Noticeably, all the solvers complied on the input requirements, which was not the case on the previous evaluation [1], while a few solvers required wrapper scripts to adapt the output format (SSOLVE, QSAT, and SEMPROP) or to load additional applications required to run the solver (OPENQBF, requiring the JVM). A short description of the solvers submitted to the evaluation follows:

- CLEARN by Andrew G.D. Rowley, is a search-based solver written in C++, featuring lazy data structures and conflict learning; the heuristic is a simple and efficient lexicographic ordering based on prefix level (outermost to innermost) and variable identifier (smallest first).
- GRL by Andrew G.D. Rowley, is a sibling of CLEARN, but with a different learning method described in [9].
- OPENQBF by Gilles Audemard, Daniel Le Berre and Olivier Roussel, is a search-based solver written in Java, featuring basic unit propagation and pure literal lookahead, plus a conflict backjumping lookback engine; the heuristic is derived from Böhm and Speckenmeyer's heuristic for SAT.
- ORSAT by Olivier Roussel, is search-based solver written in C++, featuring an algorithm based on relaxations to SAT, plus special purpose techniques to deal with universal quantifiers; the solver is currently in its early stage of development, so most of the features found in other, more mature, solvers are missing.
- QBFL by Florian Letombe, is a search-based solver written in C, packed with a number of features: trivial-truth, trivial-falsity, Horn and reverse-Horn formulas detection; QBFLis implemented on top of Limmat (version 1.3), and comes in two flavors: QBFL-JW uses an extension of the Jeroslow-Wang heuristic for SAT, while QBFL-BS uses an extension of Böhm and Speckenmeyer's heuristic for SAT.
- QSAT by Jussi Rintanen, is a search-based solver written in C, featuring a lookahead heuristic with failed literal rule, sampling, partial unfolding and quantifier inversion.
- QMRES by Guoqiang Pan and Moshe Y. Vardi, written in C and based on a symbolic implementation of the original DP algorithm, achieved using ZBDDs. The algorithm features multi-resolution, a simple form of unit propagation, and heuristics to choose the variables to eliminate.
- QUANTOR by Armin Biere, is a solver written in C based on Q-resolution (to eliminate existential variables) and Shannon expansion (to eliminate universal variables), plus a number of features, such as equivalence reasoning, subsumption checking, pure literal detection, unit propagation, and also a scheduler for the elimination step.
- QUBE by Enrico Giunchiglia, Massimo Narizzano and Armando Tacchella, is a search-based solver written in C++ featuring lazy data structures for unit and pure literal propagation; QUBE comes in two flavors: QUBE-BJ, featuring conflict- and solution-directed backjumping, and QUBE-LRN, featuring conflict and solution learning; the heuristic is an extension to QBF of ZCHAFF heuristic for SAT.
- SSOLVE by Rainer Feldmann and Stefan Schamberger, is a search-based algorithm written in C, featuring trivial truth and a modified version of Rintanen's method of

inverting quantifiers. The data structures used are extensions of the data structures of Max Böhm's SAT-solver.

SEMPROP by Reinhold Letz, is a search-based solver written in Bigloo (a dialect of Scheme), featuring dependency directed backtracking and lemma/model caching for false/true subproblems.

WALKQSAT by Ian Gent, Holger Hoos, Andrew G. D. Rowley, and Kevin Smyth, is the first incomplete QBF solver based on stochastic search methods. It is a sibling of CSBJwith WalkSAT as a SAT oracle and guidance heuristic.

YQUAFFLE by Yinlei Yu and Sharad Malik, is a search-based solver written in C++, featuring multiple conflict driven learning, solution based backtracking, and inversion of quantifiers.

Most of the solvers mentioned above are described in a booklet [10] prepared for the evaluation with the contributions of the solvers authors, with the exception of QSAT, SSOLVE, and SEMPROP, which are described, respectively, in [11], [12], and [13].

The evaluation received 88 benchmarks divided in 4 different families and a random generator, all from four different authors:

Biere (1 family, 65 instances) QBF encodings of the following problem: given an $\langle n \rangle$-bit-counter with optional reset (r) and enable (e) signals, check whether it is possible to reach the state where all $\langle n \rangle$ bits are set to 1 starting from the initial state where all bits are set to 0.

Katz (2 families, 20 instances) QBF encodings of symbolic reachability problems in hardware circuits.

Lahiri/Seshia (1 family, 3 instances) QBF encodings of convergence testing instances generated in term-level model checking.

Tacchella A generator for model A and model B random QBF instances, implemented according to the guidelines described in [7].

In order to obtain the evaluation test set, we have also considered 5558 benchmarks in 64 families from `www.qbflib.org`:

Ayari (5 families, 72 benchmarks) A family of problems related to the formal equivalence checking of partial implementations of circuits (see [14]).

Castellini (3 families, 169 benchmarks) Various QBF-based encodings of the bomb-in-the-toilet planning problem (see [15]).

Gent/Rowley (8 families, 612 benchmarks) Various encodings of the famous "Connect4" game into QBF [16].

Letz (1 family, 14 benchmarks) Formulas proposed in [13] generated according to the pattern $\forall x_1 x_3 \ldots x_{n-1} \exists x_2 x_4 \ldots x_n (c_1 \wedge c_n)$ where $c_1 = x_1 \wedge x_2$, $c_2 = \neg x_1 \wedge \neg x_2$, $c_3 = x_3 \wedge x_4$, $c_4 = \neg x_3 \wedge \neg x_4$, and so on. The instances consists of simple variable-independent subproblems but they should be hard for standard (i.e., without non-chronological backtracking) QBF solvers.

Mneimneh/Sakallah (12 families, 202 benchmarks) QBF encodings of vertex eccentricity calculation in hardware circuits [17].

Narizzano (4 families, 4000 benchmarks) QBF-based encoding of the robot navigation problems presented in [15].

Pan (18 families, 378 benchmarks) Encodings of modal K formulas satisfiability into QBF (see [18]). The original benchmarks have been proposed during the TANCS'98 comparison of theorem provers for modal logics [19].

Rintanen (5 families, 47 benchmarks) Planning, hand-made and random problems, some of which have been presented in [20].

Scholl/Becker (8 families, 64 benchmarks) encode equivalence checking for partial implementations problems (see [21]).

3 Evaluation: test set and infrastructure

As we outlined in Section 1, deciding the test set for the evaluation is complicated by two competing requirements: (*i*) obtaining meaningful data and (*ii*) completing the evaluation in reasonable time. To fulfill requirement (*i*) in the case of non-random benchmarks, we decided to extract a suitable subset from the pool of all the available benchmarks. In particular, we designed the selection process in such a way that the resulting test set is representative of the initial pool, yet it is not biased toward specific instances. This cannot be achieved by simply extracting a fixed proportion of benchmarks from all the available families, because some of them dominate others in terms of absolute numbers, e.g., the four Robots families account for more than 70% of the instances available on QBFLIB. In order to remove the bias, we have extracted a *fixed* number of instances from each available family. In this way, the extracted test set accounts for the same number of families as the initial pool (variety is preserved), but each family contains at most N representatives (bias is removed) We used the following algorithm:

- if the family in the original pool consists of $M < N$ benchmarks, then extract all M of them, while
- if the family consists of $M > N$ benchmarks, then extract only N instances by sampling the original ones uniformly at random.

Considering all the non-random benchmarks described in Section 2, we have extracted an evaluation test set of 522 instances divided into 68 families. As for random instances, the issue is further complicated by the fact that we have several parameters to choose when generating the benchmarks, namely the number of variables, the number of clauses, the number of alternations in the prefix, the number of literals per clause, the number of existential literals per clause, and the generation model (either model A or model B). We based our selection of random benchmarks on the experimental work presented in [22], and we generated formulas with $v = \{50, 100, 150\}$ variables, $a = \{2, 3, 4, 5\}$ alternations in the prefix, and a fixed number of 5 literals per clause. For each fixed value of a and v, we generated formulas ranging over clauses-to-variables ratio of 2 to 18 with step 2. We used the above parameters both to generate model A and model B instances, and a threshold 2 for the number of existential literals in the clause, which means at least 2 existential literals per clause in model A instances, and exactly 2 existential literals per clause in model B instances. Notice that while for model A instances we were able to choose parameters based on the previous experience of [7, 22], for model B instances, the only experimental account available is that of [7], which covers only part of the space explored in the evaluation.

Solver	Total		Sat		Unsat		Unique	
	#	Time	#	Time	#	Time	#	Time
SEMPROP	288	10303.40	133	2985.806	155	7317.55	5	814.56
QUANTOR	284	3997.10	126	2137.25	158	1859.85	10	2624.36
YQUAFFLE	256	6733.02	110	3152.37	146	3580.65	–	–
CLEARN	255	11565.30	116	3894.06	139	7671.27	–	–
SSOLVE	245	8736.64	114	3350.96	131	5385.68	–	–
GRL	240	11895.90	107	4577.22	133	7318.70	–	–
QUBE-BJ	239	9426.09	110	4538.27	129	4887.82	–	–
QUBE-LRN	237	8270.98	113	3365.83	124	4905.15	1	433.15
QMRES	224	6337.39	122	3315.42	102	3021.97	28	901.54
QSAT	218	8375.62	93	3307.13	125	5068.49	7	197.63
QBFL-JW	205	5573.55	83	2849.65	122	2723.90	–	–
CSBJ	205	6528.84	98	3407.16	107	3121.68	–	–
QBFL-BS	191	3076.62	75	1466.10	116	1610.52	–	–
OPENQBF	185	6598.94	78	3219.56	107	3379.38	–	–
WALKQSAT	163	7262.51	83	4113.37	80	3149.14	–	–
ORSAT	73	1243.83	37	1134.74	36	109.09	–	–

Table 1. Results of the evaluation first stage (non-random benchmarks).

Overall, the evaluation test set was completed by 432 random formulas divided into 24 families of 18 samples each, bringing the total number of benchmarks to 954.

As for the computing infrastructure, the evaluation ran on a farm of 10 identical rack-mount PCs, equipped with 3.2Ghz PIV processors, 1GB of RAM and running Debian/GNU Linux (distribution `sarge`). Considering that we had 954 benchmarks to run, we split the evaluation job evenly across (9) machines, using `perl` scripts to run subsets of 106 benchmarks on all the 16 solvers on each machine. This methodology has a two points in its favor. First, testing scripts are extremely lean and simple: one server script, plus as many client scripts as there are machines running, accounting for less than 100 lines of `perl` code. This makes the whole evaluation infrastructure lightweight and easy to debug. Second, by running clusters of benchmarks on the same machine, we are guaranteed that small differences that could exist even in identical hardwares, are compensated by the fact that a given benchmark is evaluated by all the participants on the very same machine. While noise in the order of one second does not matter much when comparing benchmarks to decide their hardness, it can make a big difference when the total runtime on the benchmark is in in the order of one second or less and we are comparing solvers. Finally, all the solvers where limited to 900 seconds of CPU time and 900MB of memory: in the following, when we say that an instance has been solved, we mean that this happened without exceeding the resource bounds above.

4 Evaluation: first stage results

In Table 1 and Table 2, we present the raw results of the evaluation concerning, respectively, non-random (522 benchmarks) and random (432 benchmarks) instances. Each table consists of nine columns that for each solver report its name (column "Solver"), the total number of instances solved and the cumulative time to solve them (columns "#" and "Time", group "Total"), the number

Solver	Total		Sat		Unsat		Unique	
	#	Time	#	Time	#	Time	#	Time
QuBE-lrn	426	3452.67	86	93.12	340	3359.55	–	–
QuBE-bj	418	4343.98	76	87.12	342	4256.86	–	–
ssolve	403	1028.30	86	346.87	317	681.43	1	0.45
semprop	384	3069.28	80	1614.06	304	1455.22	–	–
CLearn	338	5267.99	76	1939.28	262	3328.71	–	–
GRL	335	5975.14	73	1213.30	262	4761.84	–	–
qsat	321	3491.18	60	450.01	261	3041.17	1	208.12
CSBJ	320	5956.06	74	2038.39	246	3917.67	–	–
WalkQSAT	316	5838.05	75	2262.17	241	3575.88	–	–
openQBF	277	5525.99	64	334.51	213	5191.48	–	–
qbfl-jw	263	6380.26	94	58.35	169	6321.91	–	–
qbfl-bs	218	3265.93	92	1.06	126	3264.87	–	–
yQuaffle	197	3166.62	34	184.69	163	2981.93	–	–
QMRes	142	4091.29	53	1594.07	89	2497.22	–	–
quantor	120	263.75	52	0.78	68	262.97	–	–
orsat	60	1.24	–	–	60	1.24	–	–

Table 2. Results of the evaluation first stage (random benchmarks).

of instances found satisfiable and the time to solve them (columns "#" and "Time", group "Sat"), the number of instances found unsatisfiable and the time to solve them (columns "#" and "Time", group "Unsat"), and, finally, the number of instances uniquely solved and the time to solve them (columns "#" and "Time", group "Unique"); a "–" (dash) in the last two columns means that the solver did not conquer any instance uniquely. Both tables are sorted in descending order, according to the number of instances solved, and, in case of a tie, in ascending order according to the cumulative time taken to solve them.

Looking at the results on non-random instances in Table 1, we can see that all the solvers, with the only exception of orsat, were able to conquer at least 25% of the instances in this category. On the other hand, only two solvers, namely semprop and quantor, were able to conquer more than 50% of the instances. Overall, this indicates that given the current state of the art in QBF reasoning, the performance demand of the application domains is still exceeding the capabilities of most solvers. The performance of the solvers is also pretty similar: excluding orsat, there are only 125 instances (less than 25% of the total) separating the strongest solver (semprop), from the weakest solver (WalkQSAT), and the number of instances solved by the strongest five participants are in the range [288-245] spanning only 43 instances (less than 10% of the total). Some difference arises when considering the number of instances uniquely solved by a given solver: QMRes, quantor, semprop, qsat and QuBE-lrn are the only solvers able to conquer, respectively, 28, 10, 7, 5 and 1 instance. Noticeably, the strongest solvers in this respect, QMRes and quantor, are not extensions of the DPLL algorithm as all the other participants, indicating that the technologies on which they are based provide an interesting alternative to the classic search-based paradigm.

Looking at the results on random instances in Table 2, we can see that all the solvers, again with the only exception of orsat, were able to conquer at least 25% of the instances in this category, and six solvers were able to conquer more than 75% of the instances. Overall, this indicates that the choice of parameters

for the generation of random instances resulted in a performance demand well within the capabilities of most solvers. The performance of the solvers is however rather different: even excluding ORSAT, there are 306 instances (about 70% of the total) separating the strongest QUBE-LRN, from the weakest QUANTOR, and the number of instances solved by the strongest five participants are spread over 88 instances (about 20% of the total). There is no relevant change in the picture above when considering the number of instances uniquely solved by a given solver, since only SSOLVE and QSAT are able to uniquely conquer one instance each. Noticeably, some of the strongest solvers on random instances, namely SEMPROP, SSOLVE and CLEARN, are also among the strongest solvers on non-random benchmarks, indicating that these search-based engines feature relatively robust algorithms.

As we have anticipated in Section 1, a few discrepancies in the results of the solvers were detected during the analysis of the first stage results. A total of 32 discrepancies were detected, of which 9 regarding non-random instances, and the remaining regarding random instances. For each of the discrepancies we reran the solvers reporting a result different from the majority of the other solvers and/or the expected result of the benchmark. We also inspected the instances, looking for weird syntax and other pitfalls that may lead a correct solver to report an incorrect result. At the end of this analysis we excluded from the second stage the following solvers:

- QUBE-BJ and QUBE-LRN, responsible for all the discrepancies detected on random instances; although the satisfiability status of the random benchmarks is not known in advance, the two solvers do not agree with each other in 10/23 cases, and in 7/23 cases they do not agree with the majority of solvers.
- QSAT, reporting as unsatisfiable the benchmark `k_ph_n-21` of the k_ph_n family in the Pan series: these benchmarks ought to be satisfiable by construction (in modal K), and the correctness of the translations is not taunted by any other evidence in our data.
- CLEARNand GRL, reporting as unsatisfiable the benchmark `s27_d2_s` of the s_27 family in the Mneimneh/Sakallah series; the benchmark is both declared satisfiable by its authors and by all the other solvers.

We have also excluded the following benchmarks:

- the Connect2 family (Gent/Rowley), since on some of its instances QBFL-BS, QBFL-JW, QMRESand WALKQSAT, reported apparently incorrect results, if compared to the majority of the other solvers: examining the instances, it turns out that they contain existentially quantified sets declared as separate but adjacent lists in the Q-DIMACS prefix. Although the Q-DIMACS standard does not disallow this syntax, we believe that this might be the cause of the problems,for some solvers are not prepared to handle this kind of input correctly.
- the Logn family (Rintanen), since two of its instances are pure SAT with unbound variables containing an empty input clause: their correct satisfiability

Category	Solver	Total		Sat		Unsat		Unique	
		#	Time	#	Time	#	Time	#	Time
Formal Verification	QUANTOR	74	2854.87	34	1424.66	40	1430.21	10	2624.36
	SEMPROP	71	2064.38	29	294.58	42	1769.8	2	545.51
	YQUAFFLE	68	1239.56	28	319.56	40	920	–	–
	QBFL-JW	65	967.25	26	311.44	39	655.81	–	–
	CSBJ	59	1247.82	27	371.96	32	875.86	–	–
	SSOLVE	59	1814.76	26	231.47	33	1583.29	–	–
	QBFL-BS	56	262.61	25	244.81	31	17.8	–	–
	QMRES	51	1838.81	28	1166.75	23	672.06	10	171.24
	OPENQBF	49	1459.92	20	170.26	29	1289.66	–	–
	WALKQSAT	40	1599.04	21	771.14	19	827.9	–	–
	ORSAT	27	774.68	20	718.77	7	55.91	–	–
Planning	YQUAFFLE	100	3261.00	37	1763.32	63	1497.68	4	1843.74
	SSOLVE	93	2786.63	35	353.63	58	2433.00	–	–
	QBFL-JW	92	2243.19	30	706.42	62	1536.77	–	–
	SEMPROP	88	3935.57	27	348.03	61	3587.54	2	563.87
	QUANTOR	85	479.50	28	311.18	57	168.32	1	37.35
	QBFL-BS	84	569.73	21	0.79	63	568.94	–	–
	CSBJ	84	1052.90	30	872.49	54	180.41	–	–
	OPENQBF	80	3404.27	27	1867.38	53	1536.89	–	–
	WALKQSAT	55	143.52	16	0.69	39	142.83	–	–
	QMRES	37	2513.36	18	490.71	19	2022.65	–	–
	ORSAT	38	65.04	11	13.47	27	51.57	–	–
Miscellaneous	QMRES	132	1980.92	74	1657.88	58	323.04	20	749.35
	QUANTOR	117	635.82	63	401.31	54	234.51	–	–
	SEMPROP	117	4293.44	70	2339.32	47	1954.12	6	281.64
	SSOLVE	81	4115.79	48	2762.40	33	1353.39	–	–
	YQUAFFLE	76	2212.21	38	1051.83	38	1160.38	1	179.42
	WALKQSAT	62	5517.68	44	3340.22	18	2177.46	–	–
	CSBJ	54	4222.49	36	2158.17	18	2064.32	–	–
	OPENQBF	46	1710.80	26	1160.66	20	550.14	–	–
	QBFL-BS	41	2243.40	29	1220.50	12	1022.90	–	–
	QBFL-JW	38	2362.21	27	1831.79	11	530.42	–	–
	ORSAT	7	403.59	6	402.50	1	1.09	–	–

Table 3. Results of the evaluation second stage (non-random benchmarks).

status is thus "false", but some of the solvers (namely QMRES, SEMPROPand YQUAFFLE) report them as satisfiable.

The data obtained by disregarding the above solvers and benchmarks is free of any discrepancy. Clearly, for instances that were conquered by just one solver, and for which we do not know the satisfiability status in advance, the possibility that the solver is wrong still exists, but we consider this as unavoidable given the current state of the art.[1]

5 Evaluation: second stage results

5.1 Solver-centric view

In Table 3 we report second stage results about non-random benchmarks (510 benchmarks, 11 solvers), divided into three categories:

Formal Verification 29 families and 220 benchmarks, including Ayari, Biere, Katz, Mneimneh/Sakallah, and Scholl/Becker instances.

[1] Notice that the same problem exists in the SAT competition when a solver is the only one to report about an instance and the answer is "unsatisfiable".

Category	Solver	Total		Sat		Unsat		Unique	
		#	Time	#	Time	#	Time	#	Time
Model A	SSOLVE	187	1025.92	64	346.64	123	679.28	16	498.75
	SEMPROP	168	3055.93	58	1602.71	110	1453.22	–	–
	CSBJ	104	5629.68	52	1722.02	52	3907.66	–	–
	WALKQSAT	100	5461.75	53	1896.45	47	3565.3	–	–
	QBFL-JW	94	1926.84	72	0.64	22	1926.2	–	–
	QBFL-BS	91	378.23	72	0.74	19	377.49	–	–
	OPENQBF	69	3092.8	42	183.21	27	2909.59	–	–
	YQUAFFLE	53	1375.92	27	139.79	26	1236.13	–	–
	QMRES	31	1591.19	31	1591.19	0	0	–	–
	QUANTOR	30	0.61	30	0.61	0	0	–	–
	ORSAT	0	0	0	0	0	0	–	–
Model B	SSOLVE	216	2.38	22	0.23	194	2.15	–	–
	SEMPROP	216	13.35	22	11.35	194	2	–	–
	CSBJ	216	326.38	22	316.37	194	10.01	–	–
	WALKQSAT	216	376.3	22	365.72	194	10.58	–	–
	OPENQBF	208	2433.19	22	151.3	186	2281.89	–	–
	QBFL-JW	169	4453.42	22	57.71	147	4395.71	–	–
	YQUAFFLE	144	1790.7	7	44.9	137	1745.8	–	–
	QBFL-BS	127	2887.7	20	0.32	107	2887.38	–	–
	QMRES	111	2500.1	22	2.88	89	2497.22	–	–
	QUANTOR	90	263.14	22	0.17	68	262.97	–	–
	ORSAT	60	1.24	0	0	60	1.24	–	–

Table 4. Results of the evaluation second stage (random benchmarks).

Planning 16 families and 122 benchmarks, including Castellini, Gent/Rowley, Narizzano, and part of Rintanen instances.

Miscellaneous 21 families and 168 benchmarks, including Letz, Pan and the remaining Rintanen instances.

Table 3 is arranged analogously to Tables 1 and 2, except an additional column that indicates the category. The solvers are classified independently for each category, and in descending order according to the number of instances solved: in case of ties, the solvers are prioritized according the time taken to solve the benchmarks, shortest time first.

Looking at Table 3, the first observation is that the solvers performed slightly better on the planning category: the strongest one in the category (YQUAFFLE) solves 82% of the instances, the weakest one (ORSAT) solves about 30% of the instances, and 7 out of the 11 solvers admitted to the second stage are able to solve more than 50% of the category. On the other hand, the strongest solver in the miscellaneous category (QMRES) solves 78% of the instances, but most of the solvers (8 out of 11) do not go beyond the 50% threshold; in the formal verification arena, the strongest solver in the category (QUANTOR) does not get beyond a mere 33% of the total instances. Also significant is the fact that in the planning category the strongest solver is DPLL-based (YQUAFFLE), while both in the formal verification category and in the miscellaneous category the strongest solvers (respectively QUANTOR and QMRES) express alternative paradigms.

Focusing on formal verification category, we can see that all the solvers are pretty much in the same capability ballpark. Considering the three strongest solvers, namely QUANTOR, SEMPROP and YQUAFFLE, we can see that both QUANTOR and SEMPROP are able to uniquely conquer 10 and 2 instances, respectively, while YQUAFFLE is subsumed by the portfolio constituted by all the

other solvers. At the same time, QUANTOR, with 38.58s average solution time, and SEMPROP, with 29.07s average solution time, seem to be slightly less optimized than YQUAFFLE, which fares a respectable 18.22s average solution time. Among the other solvers, it is worth noting that QBFL-JW and QBFL-BS perform quite nicely in terms of average solution time (14.88s and 4.68s, respectively), and QMRES stands out for its ability to conquer 10 instances that defied all the other participants.

As for the planning category, we can see that given the relative easiness of the benchmarks selected for the evaluation, the differences between the solvers are substantially smoothed. This is also witnessed by the fact that only 4, 2 and 1 instances where uniquely conquered by, respectively, YQUAFFLE (which is also the strongest in this category), SEMPROP and QUANTOR. One possible explanation of these results is that most of the benchmarks in this category, with the only exception of Gent and Rowley's connect[3-9] families, have been around for quite some time, so developers had access to them for tuning their solvers before the evaluation.

Finally, considering the miscellaneous category, the first thing to be observed is that most of these benchmarks come from the Pan families. Since such benchmarks are derived from translations of structured modal K instances [19], and the translation algorithm applied is the same for all the benchmarks, it is reasonable to assume that the original structure, although obfuscated by the translation, carries over to the QBF instances. In conclusion, the best solvers in this category are probably those that can discover and take advantage of such a hidden structure. Looking at the results it seems that QMRES, both the strongest solver, and the only one able to conquer 20 instances (more than 10% of the total), is clearly the strongest reasoner in this category. Also QUANTOR and SEMPROP perform quite nicely by conquering 117 instances: SEMPROP, although slightly slower than QUANTOR on average, is also able to uniquely conquer 6 instances. Noticeably, the fourth strongest solver (SSOLVE) trails the path of the strongest three at a consistent distance (36 instances, about 20% of the total instances).

In Table 4 we report second stage results about random benchmarks (432 benchmarks, 11 solvers), divided into two categories:

Model A 12 families and 216 benchmarks, generated according to the guidelines presented in Section 3 to cover the space $a = \{2, 3, 4, 5\}$, $v = \{50, 100, 150\}$, where a is the number of alternations in the prefix and v is the number of variables: each of the 12 families corresponds to a given setting of a, v and contains formulas with a ratio r clauses/variables in the range 2 to 18 (step 2), and 2 instances per each value of a, v, and r.

Model B 12 families and 216 benchmarks, generated according to the same parameters as model A families.

Table 4 is arranged analogously to Table 3.

Looking at Table 4, the first observation is that the solvers performed very well on model B instances: the strongest one in the category (SSOLVE) solves 100% of the instances, the weakest one (ORSAT) solves about 27%, and 9 out of the 11 solvers admitted to the second stage are able to solve more than 50% of

the instances in this category. Model A instances turned out to be slightly more difficult to solve since the strongest solver (again SSOLVE) conquered about 86% of the instances, and with the only exception of SSOLVE and SEMPROP, all the other solvers do not go beyond the 50% threshold. Noticeably, most solvers that do very well on non-random instances have troubles with the random ones: this is the case of QMRES, QUANTOR and YQUAFFLE, and the phenomenon is particularly evident on model A instances. This seems to validate analogous results in SAT, where solvers that are extremely good on non-random instances, fail to be effective on random ones. On the other hand SSOLVE and SEMPROP partially contradict this result, in that they are the most effective on random instances, and still reasonably effective on non-random instances as we have seen before. While QMRES and QUANTOR abandon the top positions on the random benchmarks, WALKQSAT performs much better on this kind of instances, possibly indicating that its incomplete algorithm is much more suited to random rather than structured instances. On the other hand, QUANTOR and QMRES results, are a clear indication that their non-DPLL based algorithms have been tuned heavily on structured instances, and are possibly less adequate for randomly generated ones.

5.2 Benchmark-centric view

In Table 5 we show the classification of the non-random benchmarks included in the evaluation test set according to the solvers admitted to the second stage. Table 5 consists of nine columns where for each family of instances we report the name of the family (column Family), the number of instances included in the family, the number of instances solved, the number of such instances found SAT and the number found UNSAT (group "Overall", columns "N", "#", "S", "U", respectively), the time taken to solve the instances (column "Time"), the number of easy, medium and medium-hard instances (group "Hardness", columns "EA", "ME", "MH"). The number of instances solved and the cumulative time taken for each family is computed considering the "SOTA solver", i.e., all the second stage solvers running in parallel. A benchmark is thus solved if at least one of the solvers conquers it, and the time taken is the best among the times of the solvers that conquered the instance. The benchmarks are classified according to their hardness with the following criteria: easy benchmarks are those solved by all the solvers, medium benchmarks are those solved by at least two solvers, medium-hard benchmarks are those solved by one reasoner only, and hard benchmarks are those that remained unsolved. Finally, Table 5 is divided into three sections grouping respectively, families of formal verification, planning and miscellaneous benchmarks.

According to the data summarized in Table 5, the non-random part of the evaluation second stage consisted of 510 instances, of which 383 have been solved, 172 declared satisfiable and 211 declared unsatisfiable, resulting in 38 easy, 289 medium, 56 medium-hard, and 127 hard instances (respectively, the 7%, 56%, 10%, and 24% of the test set). These results indicate that the selected non-random benchmarks are not trivial for current state-of-the-art QBF solvers,

Family	Overall				Time	Hardness		
	N	#	S	U		EA	ME	MH
Adder	8	6	2	4	132.54	0	2	4
C432	8	8	3	5	1787.99	0	5	3
C499	8	5	3	2	19.97	0	4	1
C5315	8	4	2	2	437.76	0	3	1
C6288	8	1	1	0	3.93	0	0	1
C880	8	2	2	0	1.66	0	2	0
comp	8	8	4	4	0.62	0	8	0
counter	8	4	4	0	0.05	2	2	0
DFlipFlop	8	8	0	8	3.16	1	7	0
jmc_quant1	8	3	2	1	18.68	0	1	2
jmc_quant2	8	3	2	1	11.45	0	0	3
MutexP	7	7	7	0	7.30	2	5	0
s1196	6	0	0	0	–	0	0	0
s1269	8	0	0	0	–	0	0	0
s27	4	4	1	3	4.43	1	3	0
s298	8	1	1	0	452.96	0	0	1
s3271	8	8	0	8	1.90	0	8	0
s3330	8	1	1	0	154.87	0	0	1
s386	8	0	0	0	–	0	0	0
s499	8	3	3	0	70.41	0	1	2
s510	8	0	0	0	–	0	0	0
s641	8	1	1	0	350.81	0	0	1
s713	8	1	1	0	287.14	0	0	1
s820	8	0	0	0	–	0	0	0
SzymanskiP	8	8	0	8	211.20	0	8	0
term1	8	8	4	4	164.71	0	7	1
uclid	3	0	0	0	–	0	0	0
VonNeumann	8	8	0	8	16.40	0	8	0
z4ml	8	8	4	4	0.03	5	3	0
Blocks	8	8	2	6	215.02	0	7	1
Connect3	8	8	0	8	6.92	0	8	0
Connect4	8	8	0	8	3.31	3	5	0
Connect5	8	8	0	8	6.72	1	7	0

Family	Overall				Time	Hardness		
	N	#	S	U		EA	ME	MH
Connect6	8	8	0	8	2.99	1	7	0
Connect7	8	8	0	8	5.22	0	8	0
Connect8	8	8	0	8	6.29	0	8	0
Connect9	3	3	0	3	2.81	0	3	0
RobotsD2	8	8	6	2	256.12	0	8	0
RobotsD3	8	7	6	1	2411.13	0	4	3
RobotsD4	8	7	5	2	580.83	0	5	2
RobotsD5	8	8	4	4	871.52	0	7	1
Toilet	8	8	5	3	7.75	2	6	0
ToiletA	8	8	1	7	6.99	3	5	0
ToiletC	8	8	3	5	0.95	4	4	0
ToiletG	7	7	7	0	0.07	6	1	0
Chain	8	8	8	0	9.25	0	8	0
Impl	8	8	8	0	0.06	4	4	0
k_branch_n	8	4	4	0	269.07	0	1	3
k_branch_p	8	3	0	3	5.86	0	1	2
k_d4_n	8	8	8	0	32.64	0	1	7
k_d4_p	8	8	0	8	12.44	0	7	1
k_dum_n	8	8	8	0	5.34	0	8	0
k_dum_p	8	8	0	8	6.88	0	8	0
k_grz_n	8	8	8	0	1069.34	0	7	1
k_grz_p	8	8	0	8	9.19	0	7	1
k_lin_n	8	8	8	0	21.54	1	6	1
k_lin_p	8	8	0	8	4.80	0	8	0
k_path_n	8	8	8	0	3.55	0	8	0
k_path_p	8	8	0	8	11.84	0	8	0
k_ph_n	8	7	7	0	184.35	1	5	1
k_ph_p	8	3	0	3	1.46	0	3	0
k_poly_n	8	8	8	0	5.59	0	8	0
k_poly_p	8	8	0	8	0.15	0	8	0
k_t4p_n	8	8	8	0	94.10	0	2	6
k_t4p_p	8	8	0	8	22.92	0	4	4
Tree	8	8	2	6	4.82	1	7	0

Table 5. Classification of non-random benchmarks (second stage data)

since there is a little number of easy instances, and a substantial percentage of medium-to-hard instances. At the same time, the test set is not overwhelming, since most of the non-easy instances could be considered of medium difficulty, i.e., they are solved by at least two solvers.

The cumulative results about Table 5 are not balanced across each single category: formal verification families contributed 110 hard and 22 medium-hard benchmarks, planning families contributed only 2 hard and 7 medium-hard benchmarks, and the miscellaneous families (essentially the Pan families) contributed 15 hard and 27 medium-hard benchmarks. The families submitted for the evaluation resulted pretty hard for the solvers: only 4 out of 8 instances in the counter (Biere) benchmarks, 3 out of 8 in the jmc (Katz) benchmarks, and none of the uclid (Lahiri/Seshia) benchmarks were solved. Quite interesting are also the results for the Mneimneh/Sakallah's and Gent/Rowley's families, which have never been extensively tested before: the benchmarks in the former resulted quite hard in accordance to what reported in [17] (only 20% of the instances solved), while the latter resulted well within the capabilities of the current state-of-the-art solvers, but not trivial (100% of the instances solved, only 5 easy instances). Among the "older" benchmarks, the Adder family (Ayari), and the C499, C5315, C6288, and C880 families (Scholl/Becker) are still quite challenging as they resulted in the last year evaluation [1].

Family	Overall				Time	Hardness		
	N	#	S	U		EA	ME	MH
2qbf-5cnf-50var	18	18	3	15	17.63	0	15	3
2qbf-5cnf-100var	18	17	2	15	453.77	0	13	4
2qbf-5cnf-150var	18	14	2	12	46.41	0	11	3
3qbf-5cnf-50var	18	18	9	9	17.83	0	18	0
3qbf-5cnf-100var	18	17	8	9	6.7	0	15	2
3qbf-5cnf-150var	18	16	8	8	23.16	0	16	0
4qbf-5cnf-50var	18	18	6	12	10.18	0	17	1
4qbf-5cnf-100var	18	16	4	12	2.49	0	15	1
4qbf-5cnf-150var	18	17	5	12	6.38	0	15	2
5qbf-5cnf-50var	18	17	10	7	9.07	0	17	0
5qbf-5cnf-100var	18	15	9	6	21.7	0	15	0
5qbf-5cnf-150var	18	16	10	6	48.98	0	16	0

Family	Overall				Time	Hardness		
	N	#	S	U		EA	ME	MH
2qbf-5cnf-50var	18	18	1	17	0.45	0	18	0
2qbf-5cnf-100var	18	18	1	17	7.49	0	18	0
2qbf-5cnf-150var	18	18	0	18	0.58	0	18	0
3qbf-5cnf-50var	18	18	2	16	0.01	11	7	0
3qbf-5cnf-100var	18	18	2	16	0.1	2	16	0
3qbf-5cnf-150var	18	18	2	16	0.15	0	18	0
4qbf-5cnf-50var	18	18	2	16	0.15	0	18	0
4qbf-5cnf-100var	18	18	2	16	0.19	0	18	0
4qbf-5cnf-150var	18	18	2	16	0.28	0	18	0
5qbf-5cnf-50var	18	18	4	14	0.1	1	17	0
5qbf-5cnf-100var	18	18	2	16	0.22	0	18	0
5qbf-5cnf-150var	18	18	2	16	0.53	0	18	0

Table 6. Classification of random benchmarks (second stage data)

In Table 6 we show the classification of the random benchmarks included in the evaluation test set according to the solvers admitted to the second stage. Table 6 is arranged similarly to Table 5: Table 6 (left) is about model A instances, Table 6 (right) is about model B instances. According to the data summarized in Table 6, the random part of the evaluation second stage consisted of 432 instances, of which 415 have been solved, 98 declared satisfiable and 317 declared unsatisfiable, resulting in 14 easy, 385 medium, 16 medium-hard, and 17 hard instances. These results indicate that the selected non-random benchmarks are not trivial for current state-of-the-art QBF solvers, although less challenging than the non-random ones. All other things being equal, model A instances provide a much more challenging test set than model B instances. Using model A instances, an increasing number of variables determines an increase in the number of hard instances in the case of 2- and 3-qbfs, but this is not confirmed in the case of 4- and 5-qbfs. The number of alternations seems not correlated with the hardness: considering 150 variables benchmarks, there are 4 hard 2-qbfs, 2 hard 3-qbfs, 1 hard 4-qbf, and 2 hard 5-qbfs. Although the number of samples is too small for each single point to draw definitive conclusions, the variety of solvers used for the evaluation supports the conclusions drawn in [22], and restated also in [1], that model A instances seem to show a counterintuitive relationship between hardness and the number of alternations in their prefix.

6 Conclusions

The final balance of the second QBF comparative evaluation can be summarized as follows:

- 16 solvers participated, 15 complete and 1 incomplete: 13 search-based algorithms, 1 (QUANTOR) based on Q-resolution and expansion, and 1 (QMRES) based on a symbolic implementation.
- 88 formal verification benchmarks, plus a random generator were submitted.
- State-of-the-art solvers, both for random and non-random benchmarks, have been identified; also, a total of 144 challenging benchmarks that cannot be solved by none of the participants have been identified to set the reference point for future developments in the field.

– Some of the challenges outlined last year in [1] have been tackled by the participants this year: Challenge 1, about the 2003 hard benchmarks, was attempted by most solvers with noticeable progress made; Challenge 8, about alternative paradigms to search-based QBF solvers, was undertaken quite successfully by QUANTOR and QMRES; finally, all the other challenges have been at least surfaced by most of the participants, a good indicator of the stimulus that the QBF evaluation is providing to the researchers.

The evaluation also evidenced some critical points:

– The QBF evaluation is still a niche event if compared to the SAT competition: 55 SAT solvers from 27 authors and 999 benchmarks were submitted this year to the SAT competition.
– QBF encoding of real-world applications (e.g., Ayari's hardware verification problems, Sakallah's vertex eccentricity problems, etc.) contributed a lot to the pool of 144 challenging benchmarks. This shows that QBF developers must improve the performance of their solvers before these can be practical for industrial-sized benchmarks.
– The question of how to check the answer of the QBF solvers in an effective way is still unanswered. Specifically, the questions of what is a good certificate of satisfiability/unsatisfiability for QBF, and, if this proves too huge to be practical, what is a good approximation of such certificate, remain open.

The last point is not only an issue for the QBF evaluation, but also for the implementation of QBF solvers: indeed, while only two versions of one solver were found incorrect in the SAT competition, we had problems with 4 solvers in the QBF evaluation. Overall, the evaluation showed the vitality of QBF as a research area. Despite some technological limitations and some maturity issues, it is our opinion that the development of effective QBF solvers and the use of QBF-based automation techniques can be regarded as promising research directions.

References

1. D. Le Berre, L. Simon, and A. Tacchella. Challenges in the QBF arena: the SAT'03 evaluation of QBF solvers. In *Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*. Springer Verlag, 2003.
2. M. Davis and H. Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.
3. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.
4. A. G. D. Rowley I. P. Gent, H. H. Hoos and K. Smyth. Using stochastic local search to solve quantified boolean formulae. In *9th Conference on Principles and Practice of Constraint Programming (CP 2003)*, volume 2833 of *Lecture Notes in Computer Science*. Springer Verlag, 2003.
5. A. Biere. Resolve and Expand. In *Seventh Intl. Conference on Theory and Applications of Satisfiability Testing*, 2004. Extended Abstract.

6. Guoqiang Pan and Moshe Y. Vardi. Symbolic decision procedures for qbf. In *10th Conference on Principles and Practice of Constraint Programming (CP 2004)*, 2004.
7. Ian Gent and Toby Walsh. Beyond NP: the QSAT phase transition. In *Proc. of AAAI*, pages 648–653, 1999.
8. E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2001. `www.qbflib.org`.
9. Andrew G D Rowley Ian P Gent. Solution learning and solution directed backjumping revisited. Technical Report APES-80-2004, APES Research Group, February 2004.
10. D. Le Berre, M. Narizzano, L. Simon, and A. Tacchella, editors. *Second QBF solvers evaluation*. Pacific Institute of Mathematics, 2004. Available on-line at `www.qbflib.org`.
11. 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*, number 2250 in LNAI, pages 362–376. Springer, 2001.
12. R. Feldmann, B. Monien, and S. Schamberger. A distributed algorithm to evaluate quantified boolean formula. In *Proceedings of the Seventeenth National Conference in Artificial Intelligence (AAAI'00)*, pages 285–290, 2000.
13. R. Letz. Lemma and model caching in decision procedures for quantified boolean formulas. In *Proceedings of Tableaux 2002*, LNAI 2381, pages 160–175. Springer, 2002.
14. Abdelwaheb Ayari and David Basin. Bounded model construction for monadic second-order logics. In *12th International Conference on Computer-Aided Verification (CAV'00)*, number 1855 in LNCS, pages 99–113. Springer-Verlag, 2000.
15. C. Castellini, E. Giunchiglia, and A. Tacchella. Sat-based planning in complex domains: Concurrency, constraints and nondeterminism. *Artificial Intelligence*, 147(1):85–117, 2003.
16. Andrew G D Rowley Ian P Gent. Encoding connect 4 using quantified boolean formulae. Technical Report APES-68-2003, APES Research Group, July 2003.
17. M. Mneimneh and K. Sakallah. Computing Vertex Eccentricity in Exponentially Large Graphs: QBF Formulation and Solution. In *Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT 2003)*, volume 2919 of *Lecture Notes in Computer Science*. Springer Verlag, 2003.
18. Guoqiang Pan and Moshe Y. Vardi. Optimizing a BDD-based modal solver. In *Proceedings of the 19th International Conference on Automated Deduction*, 2003.
19. P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics k, kt, s4. *Journal of Automated Reasoning*, 24(3):297–317, 2000.
20. Jussi Rintanen. Improvements to the evaluation of quantified boolean formulae. In *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence (IJCAI'99)*, pages 1192–1197, Stockholm, Sweden, July 31-August 6 1999. Morgan Kaufmann.
21. C. Scholl and B. Becker. Checking equivalence for partial implementations. In *38th Design Automation Conference (DAC'01)*, 2001.
22. E. Giunchiglia, M. Narizzano, and A. Tacchella. An Analysis of Backjumping and Trivial Truth in Quantified Boolean Formulas Satisfiability. In *Seventh Congress of the Italian Association for Artificial Intelligence (AI*IA 2001)*, volume 2175 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2001.