# An Empirical Perspective on Ten Years of QBF Solving

Paolo Marin[2], Massimo Narizzano[1], Luca Pulina[3], Armando Tacchella[1], and Enrico Giunchiglia[1]

[1] DIBRIS, Università di Genova, Via Opera Pia, 13 – 16145 Genova – Italy
{giunchiglia,narizzano,tacchella}@unige.it
[2] Lehrstuhl für Rechnerarchitektur, Georges-Köhler-Allee 051 – 79110 Freiburg i.B. – Germany marin@informatik.uni-freiburg.de
[3] POLCOMING, Università di Sassari, Viale Mancini n. 5 – 07100 Sassari – Italy
lpulina@uniss.it

**Abstract.** Twelve years have elapsed since the first QBF evaluation was held as an event linked to SAT conferences. During this period, researchers have strived to propose new algorithms and tools to solve challenging problems, with evaluations periodically trying to assess the current state of the art. In this paper, we present an experimental account of solvers and benchmarks with the aim to understand the progress, if any, in the QBF arena. Unlike typical evaluations, the analysis is not confined to the snapshot of submitted solvers and problems, but rather we consider several tools that were proposed over the last decade, and we run them on different problem sets. The main contribution of our analysis, which is also the message we would like to pass along to the research community is that some faded-to-oblivion techniques turn out to be still quite effective.

## 1 Introduction

The first non-competitive QBF solvers evaluation (QBFEVAL'03) [3] was held as an associated event of the SAT 2003 conference. If the purpose of QBFEVAL'03 was to assess the state of the art in the relatively young – in the time – QBF reasoning field, the ensuing QBFEVAL series was established with the purpose of measuring the progress in QBF reasoning techniques – see, e.g., [18]. Since the last evaluation, what has been the progress (if any) in the QBF arena? After more than a decade of new solvers being developed and new challenge problems being proposed, we believe that QBFEVAL and, more recently, QBF Gallery [6] events offer a series of snapshots about QBF solving and related aspects, but somehow fail to provide a long-term picture about what has been achieved.

Covering the whole time span of QBFEVAL and QBF Gallery events, our experiments enable us to assess the progress in the QBF field, and put the current state of the art in a historical perspective. In order to achieve this goal, the experimental setup is not confined to a snapshot in time offered by recently proposed systems. In particular, as far as systems are concerned, we consider

some *legacy solvers*, i.e., tools that were proposed in the literature, but are not considered in more recents comparative events, e.g., because they are no longer maintained or updated. We call *new solvers* all the other tools that we consider and which are not legacy. In particular, out of 9 solvers considered, the legacy ones are AIGSolve [19], aqme [21], quantor [4], QuBE [8], sKizzo [1], and StruQS [22]. These tools are chosen among winners of at least one category in the past QBFEVAL events, conditioned to their maintenance ending before 2010. The set of new solvers is assembled by including the winners of the last QBF Gallery 2014, namely depqbf [15], ghostq [13] and rareqs [10]. As for problems, we consider two different pools, namely QBF Gallery 2014 Track 1 and QBF Gallery 2014 Track 2. Overall, the problem set is purposefully biased towards more recently submitted instances, in order to (try to) assess legacy solvers on problems that are probably "unseen" to them, i.e., for which their developers did not have a chance to optimize the solver.

The main conclusion that we draw by analyzing the results of our comparison is that the techniques implemented in legacy solvers are far from being outdated. Just to get an idea of what we observe – more details can be found in Section 4 – consider that, if we rank the tools using the number of problems solved, then it turns out that at least two legacy solvers rank among the first three solvers, for all the pools considered. Further evidence in this direction can be obtained considering the "state-of-the-art" (SOTA) solver abstraction, i.e., the ideal system that always fares the best time among the systems in a solver portfolio. If we build "legacy-SOTA" and "new-SOTA" solvers based on the corresponding portfolios of legacy and new solvers, then we observe that legacy-SOTA outperforms new-SOTA – and this remains true even looking at specific subcategories in most cases. While it is difficult to single out the contribution of specific algorithmic techniques by looking at the performances of implemented systems – most of which are closed source – the results we observe strongly suggest that, while new solvers are better engineered than legacy ones, the latter have some combination of techniques that are probably worth taking into account for further developments.

The rest of the paper is structured as follows. In Section 2 we review QBF syntax and semantics. In Section 3 we briefly describe the solvers and the problems used in our experiments. Section 4 presents the results, while in Section 5 we conclude the paper with some final remarks.

## 2 Preliminaries

In this section we consider the definition of QBFs and their satisfiability as given in the literature of QBF decision procedures (see, e.g., [9, 2, 4]), and we define features describing the structure of QBFs.

*Syntax and Semantics* A *variable* is an element of a set $P$ of propositional letters and a *literal* is a variable or the negation thereof. We denote with $|l|$ the variable occurring in the literal $l$, and with $\bar{l}$ the *complement* of $l$, i.e., $\neg l$ if $l$ is a variable

and $|l|$ otherwise. A literal is *positive* if $|l| = l$ and *negative* otherwise. A *clause* $C$ is an $n$-ary ($n \geq 0$) disjunction of literals such that, for any two distinct disjuncts $l, l'$ in $C$, it is not the case that $|l| = |l'|$. A *propositional formula* is a $k$-ary ($k \geq 0$) conjunction of clauses. A *quantified Boolean formula* is an expression of the form

$$Q_1 z_1 \dots Q_n z_n \Phi \tag{1}$$

where, for each $1 \leq i \leq n$, $z_i$ is a variable, $Q_i$ is either an existential quantifier $Q_i = \exists$ or a universal one $Q_i = \forall$, and $\Phi$ is a propositional formula in the variables $\{z_1, \dots, z_n\}$. The expression $Q_1 z_1 \dots Q_n z_n$ is the *prefix* and $\Phi$ is the *matrix* of (1). A literal $l$ is *existential* if $|l| = z_i$ for some $1 \leq i \leq n$ and $\exists z_i$ belongs to the prefix of (1), and it is *universal* otherwise.

The semantics of a QBF $\varphi$ can be defined recursively as follows. A QBF clause is *contradictory* exactly when it does not contain existential literals. If the matrix of $\varphi$ contains a contradictory clause then $\varphi$ is false. If the matrix of $\varphi$ has no conjuncts then $\varphi$ is true. If $\varphi = Qz\psi$ is a QBF and $l$ is a literal, we define $\varphi_l$ as the QBF obtained from $\psi$ by removing all the conjuncts in which $l$ occurs and removing $\bar{l}$ from the others. Then we have two cases. If $\varphi$ is $\exists z\psi$, then $\varphi$ is true exactly when $\varphi_z$ or $\varphi_{\neg z}$ are true. If $\varphi$ is $\forall z\psi$, then $\varphi$ is true exactly when $\varphi_z$ and $\varphi_{\neg z}$ are true. The QBF satisfiability problem (QSAT) is to decide whether a given formula is true or false. It is easy to see that if $\varphi$ is a QBF without universal quantifiers, solving QSAT is the same as solving propositional satisfiability (SAT).

*Representing QBFs* To correlate the structure of QBFs with the performances of solvers, we extract representative features from QBFs — see, e.g., [21]. A first class is given by syntactic features:

- $c$, total number of clauses; $c_1$, $c_2$, $c_3$ total number of clauses with 1, 2 and more than two existential literals, respectively; $c_h$, $c_{dh}$ total number of Horn and dual-Horn clauses, respectively;
- $v$, total number of variables; $v_\exists$, $v_\forall$, total number of existential and universal variables, respectively; $l_{tot}$, total number of literals; $vs$, $vs_\exists$, $vs_\forall$, distribution of the number of variables per quantifier set, considering all the variables, and focusing on existential and universal variables, respectively; $s$, $s_\exists$, $s_\forall$, number of total, existential and universal, quantifier sets;
- $l$, distribution of the number of literals in each clause; $l_+$, $l_-$, $l_\exists$, $l_{\exists+}$, $l_{\exists-}$, $l_\forall$, $l_{\forall+}$, $l_{\forall-}$, distribution of the number of positive, negative, existential, positive existential, negative existential, universal, positive universal, negative universal number of literals in each clauses, respectively.
- $r$, distribution of the number of variable occurrences $r_+$, $r_-$, $r_\exists$, $r_{\exists+}$, $r_{\exists-}$, $r_\forall$, $r_{\forall+}$, $r_{\forall-}$, distribution of the number of positive, negative, existential, positive existential, negative existential, universal, positive universal, negative universal variable occurrences, respectively.

We also take into account the following combined features:

- $\frac{c}{v}$, the classic clauses-to-variables ratio, and for each $x \in \{l, r\}$ the following ratios (on mean values):
  - $\frac{x_+}{x}$, $\frac{x_-}{x}$, $\frac{x_+}{x_-}$, balance ratios;
  - $\frac{x_\exists}{x}$, $\frac{x_{\exists+}}{x}$, $\frac{x_{\exists-}}{x}$, $\frac{x_{\exists+}}{x_\exists}$, $\frac{x_{\exists-}}{x_\exists}$, $\frac{x_{\exists+}}{x_{\exists-}}$, $\frac{x_{\exists+}}{x_+}$, $\frac{x_{\exists-}}{x_-}$, balance ratios (existential part);
  - $\frac{x_\forall}{x}$, $\frac{x_{\forall+}}{x}$, $\frac{x_{\forall-}}{x}$, $\frac{x_{\forall+}}{x_+}$, $\frac{x_{\forall-}}{x_-}$, $\frac{x_{\forall+}}{x_\forall}$, $\frac{x_{\forall-}}{x_\forall}$, $\frac{x_{\forall+}}{x_{\forall-}}$, balance ratios (universal part);
- $\frac{c_1}{c}$, $\frac{c_2}{c}$, $\frac{c_3}{c}$, $\frac{c_h}{c}$, $\frac{c_{dh}}{c}$, $\frac{c_h}{c_{dh}}$, i.e., balance ratios between different kinds of clauses.

A second class of features is computed on graph models of QBFs. From previous related work on SAT, see, e.g. [17], we borrow variable graphs (VG) and the clause graphs (CG). The former has a node for each variable and an edge between variables that occur together in at least one clause, while the latter has nodes representing clauses and an edge between two clauses whenever they share a negated literal. For each graph, we consider the average value on their node degree. Finally, we also consider a treewidth measure $tw_p$ which accounts for the treewidth of the VG adjusted to keep into account that only elimination orders compatible to the prefix $p$ are viable — see [20, 23] for details, and also for extensive empirical evidence about the correlation of $tw_p$ with hardness of QBFs.

## 3 Setup

In this section we present solvers and problems that we selected for our analysis. As for solvers, we consider systems participating to QBF Gallery 2014[1] as well as solvers participating to past QBFEVAL editions. Considering the former, we choose the winners of Track 1 and Track 2 [12] which are shortly described in the following.

DEPQBF (v. 3.0.4) [15] is a search-based solver performing non-chronological backtracking from conflicts and solutions; DEPQBF can select branching variables without following the prefix order by leveraging a compact representation of the dependencies among variables.

GHOSTQ (v. qdimacs-gal-2014) [13] is a non-prenex DPLL-based solver which makes use of auxiliary variables to force necessary assignments, i.e., to force a value to an existential (resp. universal) variable if the opposite value directly makes the formula evaluate to false (resp. true). Additionally, it features a CEGAR-based learning to further prune the search space when the last decision literal is existential (resp. universal) and a conflict (resp. solution) is detected.

RAREQS (v. 1.1) [10] is a counterexample guided abstraction refinement (CEGAR) based solver which performs a kind of resolution and expansion procedure but in a depth-first way, i.e., by expanding first only one value of a variable, and learns abstractions of the local partial solutions to refine the global solution.

[1] `http://qbf.satisfiability.org/gallery`

We did not consider the system HIQQER [12] because we could not find a version available for download. In the remainder of the paper, we refer to this pool of solvers as S-NEW.

Solvers participating to past editions of QBFEVAL – to which we refer as S-LEGACY from now on – are described in the following.

- AIGSOLVE [19] uses And-Inverter Graphs (AIGs) as the main data structure, and AIG-based operations to reason about the input formula. The solver includes preliminary phases devoted to simplification, structure extraction and early quantification of the input formula.
- AQME [21] is a multi-engine solver, i.e., a tool using Machine Learning techniques to select among its reasoning engines the one which is more likely to yield optimal results. The reasoning engines of AQME are a subset of those submitted to QBFEVAL'06, namely 2CLSQ, QUANTOR, QUBE, SKIZZO, and SSOLVE. Engine selection is performed according to the adaptive strategy described in [21].
- QUANTOR [4] is 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 [8] is a solver that first applies, among other simplification techniques, deep equivalence reasoning and removes variables by Q-Resolution. Then, it uses a search-based decision procedure that performs monotone and "don't care" literal propagation, non-chronological backtracking from conflicts and solutions, in which it produces and removes less clauses/terms made tautological by blocking universal/existential literals than its predecessor.
- SKIZZO [1] is a reasoning engine for QBF featuring several techniques, including search, resolution and skolemization.
- STRUQS [22] main feature is a dynamic combination of search – with solution- and conflict-backjumping – and variable-elimination. The key point in this approach is to implicitly leverage graph abstractions of QBFs to yield structural features which support an effective decision between search and variable elimination.

We included AIGSOLVE because it is the only system employing AIG-based operations to reason on input QBF. We involved AQME for its multi-engine architecture; as a by-product, it can return an approximated picture of state-of-the-art QBF solvers back in 2006, so it can be used as "yardstick" to assess improvements. QUANTOR, QUBE, and SKIZZO implement key QBF solution techniques, namely resolution and expansion, DPLL-search, and Skolemization, respectively. Finally, we included STRUQS because it represents the first — and, to the best of our knowledge, the only — attempt to combine dynamically very different solution techniques. Almost all the S-LEGACY solvers also collected accolades in past QBF evaluations. AIGSOLVE was the winner of the QBFEVAL'10 small hard track, while AQME was the system able to solve the highest number of formulas in QBFEVAL'07, '08, and in the main track of QBFEVAL'10. QUANTOR was the

winner of QBFEVAL'04, while QuBE won the 2QBF track of QBFEVAL'10. Finally, sKizzo has been the winner of QBFEVAL'05 and '07.

We evaluate the above mentioned systems on different pools of problem instances. The syntax of the instances is prenex-CNF using the `qdimacs` 1.1 format. The problem pools we consider are briefly outlined in the following.

- The formulas included in QBF Gallery 2014 Track 1. These are 276 instances collectively denoted as qbfg-t1.
- The formulas included in QBF Gallery 2014 Track 2. These are collectively denoted as qbfg-t2.

The pool qbfg-t2 includes formulas coming from six different families, namely:

`bomb` and `dungeon` [14] are encodings of conformant planning problems with optimal length and uncertainty of the initial state.

`complexity` [11] result from a QBF encoding of automatic reduction between decision problems. The original problem is undecidable in general, but it can be reduced to $\Sigma_2^p$ if the dimension of the reduction is fixed and given, and the size of the inputs is bounded.

`hardness` [16] Black-Box bounded model checking instances for an incomplete parametrized arbiter of a bus system.

`planning` [5] This instance set include different planning problems encoded into QBF using two different strategies: the first one is based on the iterative squaring formulation, and the second one relies on a more compact tree-like encoding.

`testing` [24] The solutions to these problems are test patterns for sequential circuits coming from ISCAS 89 and ITC 99 benchmarks having a maximum amount of inputs set to don't care.

## 4 Experimental Analysis

In this section we report and analyze the results of our empirical evaluation. All the experiments ran on a cluster of Intel Xeon E3-1245 PCs at 3.30 GHz equipped with 64 bit Ubuntu 12.04. All solvers were limited to 600 seconds of CPU time and to 4GB of memory.

### 4.1 QBF Gallery 2014 formulas – Track 1

The aim of our first experiment is to evaluate the selected solvers in the qbfg-t1 pool of instances. In Table 1 we report the raw results of such evaluation. Looking at the results, we can see that only 6 solvers out of 9 were able to solve at least 25% of the test set. If we rank solvers according to the number of problems solved within the time limit, then the best system is AIGSolve, which can solve about 42% of the test set, followed by QuBE and aqme. To find the best solver in s-new, namely ghostq, we must go down to the fourth position. ghostq performs only slightly worse than aqme, and it tops at 33% of the

| Solver | Total | | True | | False | | Unique | |
|---|---|---|---|---|---|---|---|---|
| | # | Time | # | Time | # | Time | # | Time |
| AIGSolve | 116 | 5333.01 | 56 | 2177.45 | 60 | 3155.56 | 22 | 1458.26 |
| QuBE | 106 | 8764.73 | 53 | 3997.78 | 53 | 4766.95 | 8 | 1195.58 |
| aqme | 97 | 3287.20 | 39 | 1098.00 | 58 | 2189.20 | – | – |
| ghostq | 91 | 4814.73 | 48 | 2912.38 | 43 | 1902.17 | 4 | 158.97 |
| depqbf | 88 | 2388.32 | 39 | 1163.15 | 49 | 1225.17 | 5 | 454.77 |
| rareqs | 79 | 2588.64 | 32 | 1593.25 | 47 | 995.39 | 6 | 787.33 |
| sKizzo | 51 | 948.81 | 18 | 556.76 | 33 | 392.06 | – | – |
| quantor | 50 | 1498.37 | 28 | 911.72 | 22 | 586.65 | 2 | 161.67 |
| StruQS | 43 | 6092.64 | 31 | 4052.98 | 12 | 2039.66 | 1 | 16.53 |

**Table 1.** Runtime of solvers on qbfg-t1. For each solver, the table reports its name (column "Solver"), the total number of instances solved and the cumulative time to solve them (columns "#" and "Time", group "Total"), the number of instances found satisfiable and the time to solve them (columns "#" and "Time", group "True"), the number of instances found unsatisfiable and the time to solve them (columns "#" and "Time", group "False"), and, finally, the number of instances uniquely solved and the time to solve them (columns "#" and "Time", group "Unique"); a "–" (dash) means that the solver did not solve any instance. The table is 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.

test set. This result is relevant for our case in point, particularly if we consider that both AIGSolve and QuBE are systems dating back to 2010, while aqme combines solvers dating back to QBFEVAL 2006. Finally, despite quantor and StruQS were not able to solve more than 20% of qbfg-t1, still they were the only ones able to solve some instances — 2 and 1, respectively.

If we consider the structure of the instances comprised in qbfg-t1, then we can observe several structural differences between those solved by at least one solver and those that remained unsolved. For instance, if we focus on the formula size in terms of variables $v$ and clauses $c$, then we can see that unsolved instances feature, on average, higher values of both parameters, i.e., they are somewhat larger. Looking at the median values $\hat{v}$ and $\hat{c}$ of the parameters $v$ and $c$, we can see that $\hat{v} = 3412$ if the population is restricted to solved instances, whereas $\hat{v} = 10188$ on the population of unsolved ones. A similar picture holds for $c$, with $\hat{c} = 14818$ and $\hat{c} = 57130$ for solved and unsolved instances, respectively. As expected, $tw_p$ is also indicative of this spread, since $\hat{tw}_p = 486$ for solved instances, whereas $\hat{tw}_p = 1102$ for unsolved ones.

Another perspective about the results of Table 1 can be obtained by resorting to the *state-of-the-art solver* abstraction (sota in the following), i.e., the ideal solver that always fares the best time among all the solvers in a portfolio. In this case, sota was able to cope with about 73% of qbfg-t1 (202 formulas). What is more relevant is that all the systems contributed to its composition. In particular, the main contributors — in percentage — were AIGSolve, depqbf,

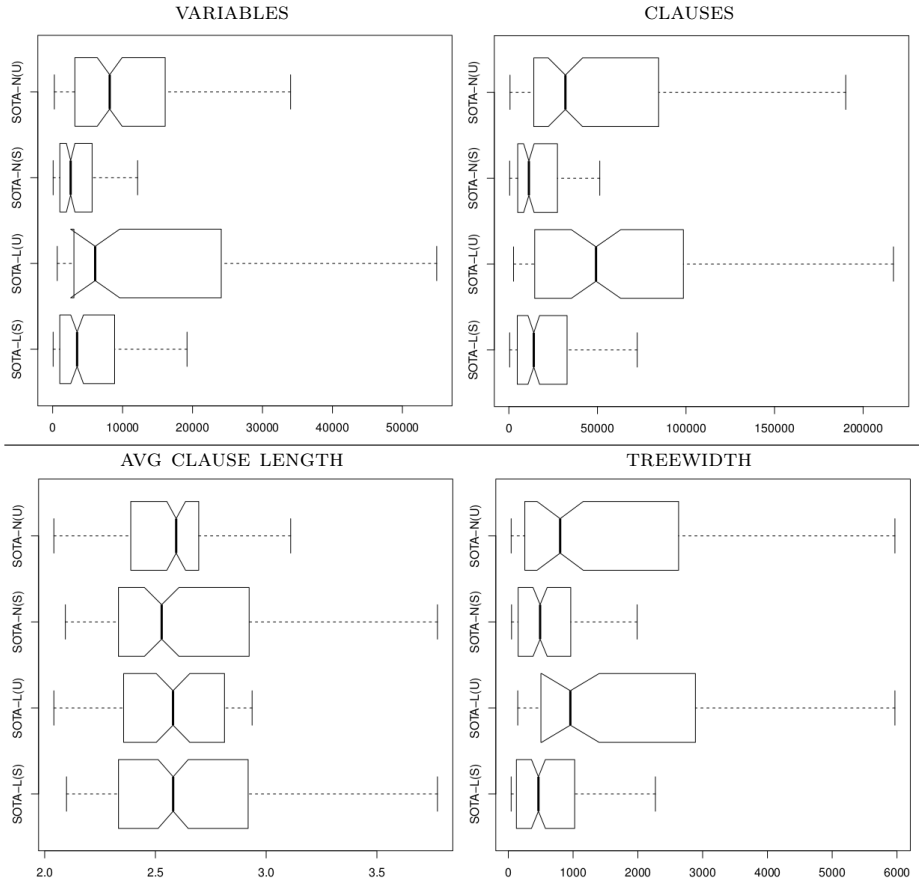


**Fig. 1.** Box-plots of different features distributions related to QBFs comprised in QBFGT1. Features are $v$ and $c$ (top-left and top-right, respectively), $l$ (bottom left), and $tw_p$ (bottom right). Each distribution in the plots is labeled as follows: "SOTA-L" and "SOTA-N" are placeholders for SOTA-LEGACY and SOTA-NEW, respectively, while "S" and "U" – in parentheses – stand for "solved" and "unsolved". For each plot, we show a box-and-whiskers diagram representing the median (bold line), the first and third quartile (bottom and top edges of the box), the minimum and maximum (whiskers at the top and the bottom) of a distribution. An approximated 95% confidence interval for the difference in the two medians is represented by the notches cut in the boxes: if the notches of two plots do not overlap, this is strong evidence that the two medians differ. Finally, in the $y$-axes of each plot are reported the values of the related features.

and RAREQS with 22%, 20%, and 17%, respectively. Notice that 2 out of 3 of the main contributors are indeed in S-NEW.

With the aid of the SOTA abstraction we can also compare the overall performances of solvers in S-LEGACY vs. those in S-NEW. In order to do that, we compute two abstractions, namely SOTA-LEGACY — considering only legacy systems

— and SOTA-NEW— considering only new solvers. The rationale of this analysis is twofold: on one hand, we want to evaluate the advancement of the state of the art with respect to legacy systems (and related solving techniques); on the other, we want to look for patterns, expressed by means of features, enabling us to spot differences in the type of QBFs solved by old and new systems. As far as advancing the state of the art is concerned, we report that SOTA-LEGACY solves 185 formulas — about 92% of those solved by SOTA — while SOTA-NEW tops at 70% (142 instances). In view of these results, and considering that most of the formulas in QBFG-T1 where not available at the time in which the solvers in S-LEGACY were developed, there does not seem to be a stark advancement in solvers' abilities from S-LEGACY to S-NEW.

As for the nature of the instances solved by legacy vs. new solvers, we can try to observe differences in the structure of QBFs solved by solvers in SOTA-LEGACY and solvers in SOTA-NEW. In Figure 1 we present the distributions of four features across four different populations obtained by combining SOTA-LEGACY, SOTA-NEW with solved and unsolved formulas. In the figure, we can see that the parameter $l$ (average clause length) is not significantly different among the various classes of problems — all the notches overlap. If we consider $v$ (number of variables) then we see that for SOTA-NEW the value of $\hat{v}$ is significantly different between solved and unsolved instances, while the same is not true for SOTA-LEGACY. Therefore, it seems that the sheer number of variables matters most for solvers in SOTA-NEW. However, also notice that there is no significant difference between SOTA-LEGACY and SOTA-NEW when considering (un)solved formulas. As for $c$ (number of clauses) both SOTA-LEGACY and SOTA-NEW are sensitive to this parameter: higher values of $c$ imply harder formulas. Also in this case, no significant difference can be spotted when considering SOTA-LEGACY and SOTA-NEW on (un)solved formulas. Finally, looking at the distributions of $tw_p$, we can see that its median value is not a significant hardness predictor for solvers in SOTA-NEW, whereas it is a hardness predictor for solvers in SOTA-LEGACY, but there are no differences when considering (un)solved formulas. Overall, we can conclude that no clear pattern emerges that could help to differentiate (un)solved formulas between solvers in SOTA-NEW and SOTA-LEGACY, at least looking at the parameters shown in Figure 1.

### 4.2 QBF Gallery 2014 formulas – Track 2

Our next experiment aims at assessing solvers on the pool QBFG-T2. Before delving into the analysis, we wish to point out that there are structural differences between the formulas in QBFG-T2 and those in QBFG-T1. On average, they are characterized by a smaller number of median variables $\hat{v}$ (3374 vs. 4708), but a considerably larger number of median clauses $\hat{c}$ (29492 vs. 17397). Formulas in QBFG-T2 are also characterized by a relatively small value of universal variables since $\frac{\hat{v}_\forall}{\hat{v}} = 0.006$ in the case of QBFG-T2, while the same ratio is 0.02 in the case of QBFG-T1. Finally, we report that QBFG-T1 formulas usually have a higher value of average clause length since $\hat{l} = 2.58$, whereas the same value is 2.37 for QBFG-T2.

| Family | Solver | Total | | True | | False | | Unique | |
|---|---|---|---|---|---|---|---|---|---|
| | | # | Time | # | Time | # | Time | # | Time |
| bomb (132) | AIGSolve | 83 | 1003.23 | 40 | 165.20 | 43 | 838.03 | – | – |
| | RAREQS | 83 | 1420.61 | 34 | 165.41 | 49 | 1255.19 | 6 | 1094.71 |
| | QUANTOR | 82 | 923.25 | 53 | 217.92 | 29 | 705.33 | – | – |
| | AQME | 80 | 674.38 | 53 | 345.02 | 27 | 329.36 | – | – |
| | DEPQBF | 67 | 2410.16 | 40 | 1693.36 | 27 | 716.79 | – | – |
| | sKizzo | 57 | 609.41 | 31 | 2.39 | 26 | 607.03 | – | – |
| | GHOSTQ | 56 | 532.47 | 29 | 42.66 | 27 | 489.81 | – | – |
| | QuBE | 47 | 1168.86 | 23 | 470.47 | 24 | 698.39 | – | – |
| | StruQS | 36 | 1051.46 | 19 | 813.58 | 17 | 237.88 | – | – |
| complexity (104) | RAREQS | 75 | 1559.65 | 29 | 466.77 | 46 | 1092.88 | 15 | 1148.51 |
| | DEPQBF | 49 | 1553.73 | 22 | 1086.35 | 27 | 467.38 | – | – |
| | GHOSTQ | 42 | 1791.86 | 11 | 499.21 | 27 | 467.38 | – | – |
| | QuBE | 39 | 1273.95 | 19 | 277.87 | 20 | 996.09 | – | – |
| | AQME | 33 | 528.28 | 15 | 188.76 | 18 | 339.52 | – | – |
| | QUANTOR | 26 | 170.44 | 11 | 11.29 | 15 | 159.14 | – | – |
| | StruQS | 21 | 1855.53 | 13 | 1677.81 | 8 | 177.72 | – | – |
| | AIGSolve | 15 | 70.26 | 7 | 12.24 | 8 | 58.02 | – | – |
| | sKizzo | 9 | 316.60 | 4 | 315.82 | 5 | 0.78 | – | – |
| dungeon (107) | QUANTOR | 104 | 525.30 | 18 | 54.81 | 86 | 470.48 | – | – |
| | AQME | 104 | 1121.43 | 18 | 86.11 | 86 | 1035.32 | – | – |
| | AIGSolve | 87 | 1220.22 | 17 | 417.12 | 70 | 803.10 | – | – |
| | RAREQS | 57 | 1870.73 | 18 | 54.89 | 39 | 1815.85 | – | – |
| | DEPQBF | 44 | 535.22 | 18 | 300.44 | 26 | 234.77 | – | – |
| | QuBE | 34 | 1429.60 | 7 | 212.89 | 27 | 1216.71 | – | – |
| | GHOSTQ | 7 | 385.11 | 4 | 4.62 | 3 | 380.49 | – | – |
| | sKizzo | 2 | 0.99 | – | – | 2 | 0.99 | – | – |
| | StruQS | 1 | 21.96 | 1 | 21.96 | – | – | – | – |
| hardness (114) | StruQS | 88 | 7826.42 | 1 | 372.74 | 87 | 7453.68 | 12 | 3033.19 |
| | QuBE | 76 | 1346.11 | – | – | 76 | 1346.11 | 2 | 328.75 |
| | GHOSTQ | 51 | 2649.30 | 2 | 239.56 | 49 | 2409.74 | 1 | 224.22 |
| | AQME | 50 | 265.14 | – | – | 50 | 265.14 | – | – |
| | RAREQS | 14 | 1431.05 | – | – | 14 | 1431.05 | – | – |
| | AIGSolve | 12 | 2038.84 | – | – | 12 | 2038.84 | – | – |
| | DEPQBF | 8 | 617.99 | – | – | 8 | 617.99 | – | – |
| | QUANTOR | – | – | – | – | – | – | – | – |
| | sKizzo | – | – | – | – | – | – | – | – |
| planning (147) | AIGSolve | 147 | 2371.36 | 38 | 114.02 | 109 | 2257.34 | 10 | 861.70 |
| | RAREQS | 137 | 1093.01 | 38 | 125.66 | 99 | 967.35 | – | – |
| | QUANTOR | 131 | 6750.13 | 37 | 122.68 | 94 | 6627.44 | – | – |
| | AQME | 123 | 9263.25 | 37 | 464.97 | 86 | 8798.28 | – | – |
| | sKizzo | 74 | 71.57 | 34 | 24.02 | 40 | 47.55 | – | – |
| | DEPQBF | 57 | 5134.24 | 29 | 1876.90 | 28 | 3257.34 | – | – |
| | QuBE | 14 | 1270.35 | 12 | 743.61 | 2 | 526.74 | – | – |
| | GHOSTQ | 11 | 2155.26 | 8 | 1420.71 | 3 | 734.55 | – | – |
| | StruQS | 4 | 1229.67 | 4 | 1229.67 | – | – | – | – |
| testing (131) | AQME | 71 | 2675.64 | 64 | 2339.68 | 7 | 335.95 | 1 | 3.00 |
| | StruQS | 65 | 1770.09 | 63 | 1488.09 | 2 | 282.00 | 4 | 236.18 |
| | DEPQBF | 57 | 692.38 | 46 | 672.96 | 11 | 19.42 | 2 | 359.15 |
| | AIGSolve | 51 | 4194.44 | 46 | 4163.65 | 5 | 30.79 | 2 | 11.88 |
| | QuBE | 41 | 765.08 | 31 | 734.85 | 10 | 30.23 | 1 | 1.24 |
| | RAREQS | 34 | 428.00 | 22 | 317.04 | 12 | 110.95 | 1 | 0.53 |
| | GHOSTQ | 32 | 269.13 | 29 | 66.10 | 3 | 203.03 | – | – |
| | QUANTOR | 26 | 121.15 | 25 | 110.52 | 1 | 10.63 | – | – |
| | sKizzo | 1 | 0.02 | 1 | 0.02 | – | – | – | – |

**Table 2.** Performances of QBF solvers on QBFG-T2: The table is split in six horizontal parts, one for each family. The first column contains families names, as well as its total amount of instances. The rest of the table is organized as Table 1.

In Table 2, we show the results of our experiments on QBFG-T2. The formulas in `bomb`, when compared to the whole QBFG-T2 formulas, are characterized by higher median values of $\frac{l_-}{l}$ (0.96 vs 0.84), $\frac{c}{v}$ (13.61 vs 8.74), and $tw_p$ (914 vs 758). On this subcategory, the best systems are AIGSOLVE, RAREQS, and QUANTOR, which are the only ones able to solve more than 60% of the total. Also in this case, two of the top three performers are solvers in S-LEGACY. However, we should also point out that RAREQS is the only system able to solve instances uniquely. Overall, it seems that the solvers which are not purely search-based are also the most effective ones in this subcategory. This difference cuts across the separation between S-LEGACY and S-NEW, and it could be due to the fact that these formulas are relatively easy to expand into SAT instances, so solvers featuring this technique, e.g., QUANTOR and RAREQS, handle them more effectively. Indeed, if we consider the SOTA abstraction, its major contributors are QUANTOR and RAREQS, with 41 and 38 formulas, respectively. Overall, SOTA is able to solve 77% of the total (102 instances out of 132). In spite of the very good performances of RAREQS, still SOTA-LEGACY solved 96 instances, while SOTA-NEW 89, thus confirming the picture that we observed in QBFG-T1.

Regarding the results on `complexity`, looking at Table 2 we can see that the best solvers are all comprised in S-NEW. Noticeably, this is the only subcategory of QBFG-T2 and the only case throughout our experimental analysis in which this is true. RAREQS, DEPQBF, and GHOSTQ are able to solve 75, 49, and 42 instances, respectively. In particular, RAREQS solves 15 of them uniquely. Looking at the structure of QBFs, we can see that `complexity` instances are smaller than `bomb` ones: the median values of $c$, $v$, and $l$ are 1101, 2533, and 6601, respectively. Moreover, with respect to `bomb`, we also report a smaller values of $\frac{\hat{v}_\forall}{\hat{v}}$, and of median clause-to-variable ratio $\frac{c}{v}$. On the other hand, the parameter $\hat{l}$ on these formulas is 2.66, higher than QBFG-T2 (2.37) and `bomb` (2.07). Since DEPQBF and GHOSTQ do not perform very well on `bomb` and also on other subcategories in QBFG-T2, we conjecture that (*i*) relatively small instances with (*ii*) relatively small number of universally quantified variables even with (*iii*) relatively long clauses, could correlate with positive performances of DEPQBF and GHOSTQ. Considering the SOTA abstraction, we report that it solves the same number of formulas solved by the best solver (RAREQS). Unsurprisingly, in this case SOTA-NEW outperforms SOTA-LEGACY — 75 and 44 solved instances, respectively.

Considering `dungeon`, we can see that the three best solvers are comprised in S-LEGACY. QUANTOR and AQME solved 97% of `dungeon`, while AIGSOLVE topping at about 81%. The structure of `dungeon` is characterized by large values of $\hat{v}$, $\hat{c}$, and $\hat{l}$ (27781, 128155, and 265184, respectively). On the other hand, we report small values of $\hat{l}$ (1.99) and $\hat{v_\forall}$ (5). Moreover, it is worth noticing that `dungeon` formulas have a large amount of $c_1$ and $c_h$ (number of unary and Horn clauses, respectively) with respect to the whole QBFs in QBFG-T2. The value of $\hat{c_1}$ related to `dungeon` is 20299, while the one reported for QBFG-T2 is 3. Considering $\hat{c_h}$, the values in `dungeon` and QBFG-T2 are 125611 and 23277, respectively. Given, e.g., the large number of unary and Horn clauses, these formulas should not be particularly challenging in general. Despite that, looking

at the result we can see that otherwise effective solvers such as GHOSTQ solved only 6% of the total. This fact makes us conjecture that for this family sheer size becomes an issue for some solvers. Finally, we report that SOTA can solve all but one formula (106 solved out of 107) and, in this case, SOTA-LEGACY outperforms SOTA-NEW (106 and 57 solved instances, respectively).

Considering `hardness`, looking at Table 2 we can see that the best system is STRUQS with 88 solved formulas, followed by QUBE and GHOSTQ with 76 and 51 solved instances, respectively. This result is quite surprising because, considering the results described so far, STRUQS always ranks among the worst three solvers. To investigate this phenomenon, we analyzed the structure of the instances comprised in `hardness`. First, we report that, on one hand, both $\hat{v}$ and $\hat{c}$ are relatively small (2191 and 7793, respectively); on the other, we can report for `hardness` the highest value of several features, such as $\hat{l}$ (9.80), $\frac{\hat{v_\forall}}{\hat{v}}$ (in percentage, 5%), and the number of quantified sets $\hat{s}$ (26, against a value of 3 reported for QBFG-T2). This can partially explain the performance of STRUQS because its hybrid resolution-search algorithm works best with small formulas having many quantifier alternations. Finally, we report that SOTA was able to solve 91 instances, and its best contributors – in percentage – are QUBE, STRUQS, and GHOSTQ, with 65%, 16%, and 13% of the total, respectively. Also in this case, SOTA-LEGACY outperformed SOTA-NEW (90 and 51 solved instances, respectively).

Concerning the results on `planning`, we can see from Table 2 the best solver is AIGSOLVE, able to deal with all the instances in the family. It is followed by RAREQS and QUANTOR, that solved 137 and 131 instances, respectively. The picture seems to be very similar to `bomb` and, indeed we can report that this family is characterized by a low value of $\hat{v}$ (1947 vs. 3374 of QBFG-T2), but large values of $\hat{l_{tot}}$) (326955 vs 96532) and $\hat{c}$ (112826 vs 29492). These data also implies that `planning` has the largest value of the median clause to variable ratio. Finally, we report that variables in `planning` are highly connected: the median value of the VG node degree is 170.05, while the same value in QBFG-T2 is 21.61. As a final comment, we report that the performances of SOTA-LEGACY and SOTA-NEW are quite close in this case, with 147 and 137 instances solved, respectively.

To conclude, looking at the results on `testing`, we can see that AQME is the best solver, dealing with about 54% of the instances. It is worth noticing that AQME solved 13% of the instances running SSOLVE [7], a system dating back to year 2000. Second and third are STRUQS and DEPQBF, solving about 50% and 43%, respectively. Values of dimensional features of these formulas are quite similar to `bomb`, with the noticeable exception of the total amount of universal variables (more than 2% of the total), that makes the family more similar to `hardness`, which may also can explain the good performances of STRUQS. As a final consideration, we report that SOTA solved 69% of the total, and its major contributors is DEPQBF (53%). Notice that also in this case SOTA-LEGACY outperforms SOTA-NEW (85 and 65 solved instances, respectively).

## 5 Conclusions

In the paper we have shown the results of a massive evaluation of QBF solvers and benchmarks. The picture that we have obtained is significant both because it is the first historical perspective on QBF solving technologies, and because of the results that emerged clearly from the analysis. In particular, we have shown that recently proposed solvers might benefit from some techniques implemented in legacy ones which defy aging. Indeed, new solvers seems to be fairly well engineered – the majority of the overall SOTA solver is made by new systems – and they made a relevant contribution to the QBF field, as witnessed by the fact that they are most often among the ones solving a formula uniquely. However, by comparing the SOTA-LEGACY and SOTA-NEW abstractions we have also shown that legacy systems still outperform the new ones in many problem categories. Therefore, we believe that it would be interesting to blend new techniques, e.g., CEGAR or dependency schemas, with legacy ones – modulo the inevitable engineering challenges that might arise – in order to really push forward the state of the art in the QBF arena. A contribution to the development and optimization of such blended solvers might come, e.g., from the significant number of problems that emerged as challenging throughout our evaluation.

## References


1. M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Eleventh International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2004)*, volume 3452 of *Lecture Notes in Computer Science*. Springer Verlag, 2004.
2. M. Benedetti. sKizzo: a Suite to Evaluate and Certify QBFs. In *20th Int.l. Conference on Automated Deduction*, volume 3632 of *LNCS*, pages 369–376. Springer Verlag, 2005.
3. D. L. 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*, pages 468–485. Springer Verlag, 2003.
4. A. Biere. Resolve and Expand. In *Seventh Intl. Conference on Theory and Applications of Satisfiability Testing (SAT'04)*, volume 3542 of *LNCS*, pages 59–70. Springer Verlag, 2005.
5. M. Cashmore, M. Fox, and E. Giunchiglia. Partially grounded planning as quantified boolean formula. In D. Borrajo, S. Kambhampati, A. Oddi, and S. Fratini, editors, *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy, June 10-14, 2013*. AAAI, 2013.
6. A. V. G. F. Lonsing, M. Seidl. QBF gallery 2013, 2013. `http://www.kr.tuwien.ac.at/events/qbfgallery2013/`.
7. R. Feldmann, B. Monien, and S. Schamberger. A distributed algorithm to evaluate quantified boolean formulae. In *Proceedings of the Seventeenth National Conference in Artificial Intelligence (AAAI'00)*, pages 285–290. AAAI Press / The MIT Press, 2000.

8. E. Giunchiglia, P. Marin, and M. Narizzano. Qube7.0. *JSAT*, 7(2-3):83–88, 2010.
9. E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause-Term Resolution and Learning in Quantified Boolean Logic Satisfiability. *Artificial Intelligence Research*, 26:371–416, 2006.
10. M. Janota, W. Klieber, J. Marques-Silva, and E. Clarke. Solving QBF with counterexample guided refinement. In *Theory and Applications of Satisfiability Testing–SAT 2012*, pages 114–128. Springer Berlin Heidelberg, 2012.
11. C. Jordan and L. Kaiser. Experiments with reduction finding. In M. Jarvisalo and A. Van Gelder, editors, *Theory and Applications of Satisfiability Testing SAT 2013*, volume 7962 of *Lecture Notes in Computer Science*, pages 192–207. Springer Berlin Heidelberg, 2013.
12. C. Jordan and M. Seidl. The QBF Gallery 2014, 2014.
13. W. Klieber, S. Sapra, S. Gao, and E. Clarke. A non-prenex, non-clausal QBF solver with game-state learning. In *Theory and Applications of Satisfiability Testing–SAT 2010*, pages 128–142. Springer, 2010.
14. M. Kronegger, A. Pfandler, and R. Pichler. Conformant planning as a benchmark for QBF solvers. In *Intl. Workshop on Quantified Boolean Formulas (QBF 2013)*, page 15, 2013.
15. F. Lonsing and A. Biere. Depqbf: A dependency-aware QBF solver. *JSAT*, 7(2-3):71–76, 2010.
16. C. Miller, C. Scholl, and B. Becker. Proving QBF-hardness in Bounded Model Checking for Incomplete Designs. In *14th International Workshop on Microprocessor Test and Verification, MTV 2013, Austin, TX, USA, December 11-13, 2013*, pages 23–28. IEEE Computer Society, 2013.
17. E. Nudelman, K. Leyton-Brown, A. Devkar, Y. Shoham, and H. Hoos. SATzilla: An Algorithm Portfolio for SAT. In *In Seventh International Conference on Theory and Applications of Satisfiability Testing, SAT 2004 Competition: Solver Descriptions*, pages 13–14, 2004.
18. C. Peschiera, L. Pulina, A. Tacchella, U. Bubeck, O. Kullmann, and I. Lynce. The seventh QBF solvers evaluation (QBFEVAL'10). In *Theory and Applications of Satisfiability Testing–SAT 2010*, pages 237–250. Springer Berlin Heidelberg, 2010.
19. F. Pigorsch and C. Scholl. An AIG-based QBF-solver using SAT for preprocessing. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 170–175. IEEE, 2010.
20. L. Pulina and A. Tacchella. Treewidth: a useful marker of empirical hardness in quantified Boolean logic encodings. In *15th Int.l Conf. on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 5330 of *LNCS*, pages 528–542. Springer, 2008.
21. L. Pulina and A. Tacchella. A self-adaptive multi-engine solver for quantified Boolean formulas. *Constraints*, 14(1):80–116, 2009.
22. L. Pulina and A. Tacchella. A structural approach to reasoning with quantified Boolean formulas. In *21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pages 596–602, 2009.
23. L. Pulina and A. Tacchella. An empirical study of QBF encodings: from treewidth estimation to useful preprocessing. *Fundamenta Informaticae*, 102(3):391–427, 2010.
24. M. Sauer, S. Reimer, I. Polian, T. Schubert, and B. Becker. Provably optimal test cube generation using quantified boolean formula solving. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 533–539, Jan 2013.