Metodi di Calcolo dei Punteggi per la Valutazione di Sistemi di Ragionamento Automatico

Scoring Methods for the Evaluation of Automated Reasoning Systems

Massimo Narizzano Luca Pulina Armando Tacchella

SOMMARIO/*ABSTRACT*

La comunità di ricerca sul Ragionamento Automatico ha visto consolidarsi la prassi relativa ad eventi competitivi in cui un insieme di sistemi viene eseguito su un insieme di istanze di problemi col proposito di stilare una classifica basata sulle prestazioni dei sistemi. Alla base di tale classifica si ha il metodo utilizzato per calcolare il punteggio di ogni sistema, ossia la procedura usata per derivare una quantità numerica che dovrebbe riassumere le prestazioni di un sistema rispetto agli altri e rispetto all'insieme di istanze. In questo articolo analizziamo diversi metodi per il calcolo dei punteggi, includendo sia metodi utilizzati in competizioni per sistemi di ragionamento automatico, sia metodi classici basati su elementi di teoria del voto, ed un nuovo metodo da noi introdotto. La nostra ricerca ha lo scopo di stabilire quale metodo massimizza le misure di efficacia da noi sviluppate per quantificare le proprietà desiderabili delle procedure per il calcolo dei punteggi. Il nostro è un metodo empirico, dato che compariamo i metodi di calcolo dei punteggi ottenendo le relative misure di efficacia dai dati della valutazione sperimentale di sistemi per la soluzione di formule in logica proposizionale con quantificatori tenutasi nel 2005. I risultati dei nostri esperimenti offrono utili indicazioni sui punti di forza e di debolezza dei metodi di calcolo dei punteggi da noi analizzati e permettono di formulare alcune conclusioni indipendenti dallo specifico metodo adottato.

The automated reasoning research community has grown accustomed to competitive events where a pool of systems is run on a pool of problem instances with the purpose of ranking the systems according to their performances. At the heart of such ranking lies the method used to score the systems, i.e., the procedure used to compute a numerical quantity that should summarize the performances of a system with respect to the other systems and to the pool of problem instances. In this paper we evaluate several scoring methods, including methods used in automated reasoning contests, as well as methods based on voting theory, and a new method that we introduce. Our research aims to establish which of the above methods maximizes the effectiveness measures that we devised to quantify desirable properties of the scoring procedures. Our method is empirical, in that we compare the scoring methods by computing the effectiveness measures using the data from the 2005 comparative evaluation of solvers for quantified Boolean formulas. The results of our experiments give useful indications about the relative strengths and weaknesses of the scoring methods, and allow us to infer also some conclusions that are independent of the specific method adopted.

1 Introduction

The automated reasoning research community has grown accustomed to competitive events where a pool of systems is run on a pool of problem instances with the purpose of ranking the systems according to their performances. A non exhaustive list of such contests includes the CADE ATP System Competition (CASC) [1], the SAT Competition [2], the QBF Evaluation [3] for quantified Boolean formulas (QBFs) solvers, and the International Planning Competition (see, e.g., [4]). At the heart of the ranking that determines the winner of such events, lies the method used to score the systems, i.e., the procedure used to compute a numerical quantity that should summarize the performances of a system. Usually such quantities cannot be interpreted as absolute measures of merit, but they should represent the relative strength of a system with respect to the other competitors based on the difficulty of the problem instances used in the contest. There is a general agreement that, although the results of automated reasoning systems competitions may provide less insight than controlled experiments in the spirit of [5], yet they play a fundamental role in the advancement of the state of the art.

In this paper we evaluate two different sets of scoring methods. The first set is comprised of some methods used in automated reasoning systems contests, namely CASC [1], the SAT competition [2], and the QBF evaluation [3], and a new method called YASM ("Yet Another Scoring Method") that we are evaluating as a candidate scoring method for the 2006 competitive evaluation of QBF solvers. The second set of scoring methods is comprised of procedures based on voting systems, namely

Borda count [6], range voting [7] and sum of victories (based on ideas from [8]). Using these methods amounts to considering the solvers as candidates and the problem instances as voters. Each voter ranks the candidates, i.e., the solver with the best performance on the instance is the preferred candidate, and all the other solvers are ranked accordingly. Finally, the votes are pooled to elect the winner of the contest.

Our research aims to establish which of the above methods maximizes the measures that we devised to quantify desirable properties of the scoring procedures. In particular, our measures should account for:

- the degree of (dis)agreement between the different scoring methods;
- the degree of stability of each scoring method with respect to perturbations (i) in the size of the test set, (ii) in the amount of resources available (CPU time), and (iii) in the quality of the test-set;
- the representativeness of each scoring method with respect to the state of the art expressed by the competitors.

In order to evaluate the relative quality of the scoring methods under test, we compute the above measures using part of the results from the 2005 comparative evaluation of QBF solvers (QBFEVAL 2005) [9]. Our analysis is thus empirical, but we believe that the scenario of QBFEVAL 2005 is representative enough in order to allow some generalizations of our results, under the hypotheses presented in Subsection 2.2.

The paper is structured as follows. In Section 2 we introduce our case study, the 2005 comparative evaluation of QBF solvers [9], and we outline the working hypotheses underlying our analysis. In Section 3 we present the scoring methods, and the effectiveness measures used to compare and evaluate the methods. The results of such comparison are the subject of Section 4, where we analyze the data in order to pin down the methods that enjoy the best performances overall. We conclude the paper in Section 5 with a discussion about the impact of our results on the evaluations of automated reasoning systems.

2 Preliminaries

2.1 QBFEVAL 2005

QBFEVAL 2005 [9] is the third in a series of non-competitive events established with the aim of assessing the advancements in the field of QBF reasoning and related research. QBFEVAL 2005 accounted for 13 competitors, 553 QBFs and three QBF generators submitted. The test set was assembled using a selection of 3191 QBFs obtained considering the submissions and the instances archived in QBFLIB [10]. The results of QBFEVAL 2005 can be listed in a table, that we call RUNS in the following. RUNS is comprised of four attributes (column names): SOLVER, INSTANCE, RESULT, and CPUTIME. The attributes SOLVER and INSTANCE report which solver is run on which instance. RESULT is a four-valued attribute: SAT, i.e., the instance was found satisfiable by the solver, UNSAT, i.e., the instance was found unsatisfiable by the solver, TIME, i.e., the solver exceeded a given time limit without solving the instance (900 seconds in QBFEVAL 2005)[1], and FAIL, i.e., the solver aborted for some reason (e.g., a crash, or insufficient memory to complete the task). Finally, CPUTIME reports the CPU time spent by the solver on the given instance, in seconds.[2]

2.2 Working hypotheses

The scoring methods that we evaluate, the measures that we compute and the results that we obtain, are based on the assumption that a table identical to RUNS as described above is the only input required by a scoring method. As a consequence, the scoring methods (and thus our analysis) do not take into account (i) memory consumption, (ii) correctness of the solution, and (iii) "quality" of the solution. As for (i), it turns out that measuring and comparing memory consumption is a tough exercise during a contest, mainly for two reasons. First, there are several definitions of memory consumption, e.g., peak memory usage as opposed to the total number of bytes in memory read/write operations, and it is not clear which one should be preferred. Second, it is complicated to measure memory consumption for systems as black boxes, which is usually the only view that the organizers of the contest have. Regarding (ii), checking the correctness of the solution is desirable for most automated reasoning systems. Unfortunately, the size of the certificate can be prohibitive in practice, and this is precisely the case of certificates for QBF (un)satisfiability which is a PSPACE-complete problem. In the case of search-based solvers one could exploit the ideas of [11] to produce a clause-term resolution proof that certifies the result of the solver. Notice that such a proof is not guaranteed to be small. However, its size is proportional to the amount of space searched by the solver, and we can expect it to be reasonable once the solution time, and hence the amount of space explored, is reasonably small. In the general case, producing reasonably sized certificates is still a research issue (see, e.g., [12, 13]), and thus we do not have any certificates associated with QBFE-

[1] The solvers are not given the time limit as input, thus they have no ways of tuning their internal heuristics in order to take the time resource bound into account.

[2] With the only exception of WalkQSAT, all the solvers submitted to QBFEVAL'05 are deterministic, and thus only one run for each solver and benchmark (modulo noise) is enough to assess its performances. The case of WalkQSAT is in principle more complex, as it would require multiple runs to be averaged in order to obtain its performance on a single benchmark. Here we assume that the only time we sampled is indeed the average: such assumption is clearly not valid in general, but since we are comparing scoring methods (not solvers) it seems perfectly reasonable in this context.

VAL 2005 RUNS table. Finally, (iii) matters only if there is indeed some solution on which quality indicators can be computed for the sake of comparison. For instance, should a resolution proof be available for all the solvers, one could compare the quality of the solution by comparing, e.g., the number of deduction steps in each proof. No such indicators can be computed for simple SAT/UNSAT results, and, as we said before, obtaining solutions from QBF solvers is not an easy task.

Given the setting described above, one more working hypothesis concerns CPU times, that we will assume to be unaffected by noise. It turns out that such assumption makes for a rather idealistic model, at least on the current QBFEVAL 2005 platform[3]. Even under light-load conditions, the noise affecting the CPU time measured by the operating system can be substantial, with standard errors in the order of 1% to 10% of the average CPU time over several runs of the same program with the same inputs. Besides, according to some preliminary experiments that we performed, the distribution of such errors does not appear to be normal, unless CPU times in the order of 1000 seconds are considered, and the variance of the observed times grows proportionally with their mean, i.e., a higher run time implies higher noise in the measure.

3 Methods and measures

3.1 State of the art scoring methods

In the following we describe in some details the state of the art scoring methods used in our analysis. For each method we describe only those features that are relevant for our purposes. Further details can be found in the references provided.

CASC [1] The CASC scoring method applied to our setting yields the following guidelines. Solvers are ranked according to the number of problems solved, i.e., the number of times RESULT is either SAT or UNSAT. In case of a tie, the solver faring the lowest average on CPUTIME fields over the problems solved is preferred.

QBF evaluation [3] QBFEVAL scoring method is the same as CASC, except that ties are broken using the sum of CPUTIME fields over the problems solved.

SAT competition [2] The last SAT competition uses a *purse-based method*, i.e., the score of a solver on a given instance, is obtained by adding up three purses:

- the solution purse, which is divided equally among all solvers that solve the problem;
- the speed purse, which is divided unequally among all the competitors that solve the problem, first by computing the speed factor $F_{s,i}$ of a solver s on a problem instance i:
$$F_{s,i} = \frac{k}{1 + T_{s,i}} \quad (1)$$
where k is an arbitrary scaling factor (we set $k = 10^4$ according to [14]), and $T_{s,i}$ is the time spent by s to solve i; then by computing the speed award $A_{s,i}$, i.e., the portion of speed purse awarded to the solver s on the instance i:
$$A_{s,i} = \frac{P_i \cdot F_{s,i}}{\sum_r F_{r,i}} \quad (2)$$
where r ranges over the solvers, and P_i is the total amount of the speed purse for the instance i.
- the series purse, which is divided equally among all solvers that solve at least one problem in a given series (a series is a family of instances that are somehow related, e.g., different QBF encodings for some problem in a given domain).

The overall score of a solver is just the sum of its scores on all the instances of the test set, and the winner of the contest is the solver with the highest sum.

Borda count [6] Suppose that n solvers are participating to the contest. Each voter (instance) ranks the candidates (solvers) in ascending order considering the value of the CPUTIME field. Let $p_{s,i}$ be the position of a solver s in the ranking associated with instance i ($1 \leq p_{s,i} \leq n$). The score of s computed according to Borda count is just $S_{s,i} = n - p_{s,i}$. In case of time limit attainments and failures, we default $S_{s,i}$ to 0. The total score S_s of a solver s is the sum of all the scores, i.e., $S_s = \sum_i S_{s,i}$, and the winner is the solver with the highest score.

Range voting [7] The ranking position $p_{s,i}$ of each solver s on each instance i is computed as in Borda count. Then an arbitrary scale is used to associate a weight w_p with each of the n positions and the score $S_{s,i}$ is computed as $S_{s,i} = w_p \cdot p_{s,i}$ (default to 0 in case of time limit attainment or failure) and $S_s = \sum_i S_{s,i}$, as in Borda count. To compute w_p in our experiments we use a geometric progression with a common ratio $r = 2$ and a scale factor $a = 1$, i.e., $w_p = ar^{n-p}$ with $1 \leq p \leq n$.

Sum of victories A non-Condorcet method based on ideas from [8] which works as follows. A $n \times n$ square matrix M is computed, where the entries $M_{s,t}$ with $s \neq t$ account for the number of times that solver s is faster than t on some instance, while $M_{s,t} = 0$ if $s = t$. No points are awarded in case of ties, time limit attainment and failure. The score S_s according to which a solver s is ranked is computed as $S_s = \sum_t M_{s,t}$.

[3] A farm of identical 3GHz PIV PCs with 1GB of main memory, running `Debian GNU/Linux` (`sarge` distribution with a 2.4.x kernel)

3.2 YASM: Yet Another Scoring Method

While the scoring methods used in CASCs and QBF evaluations are straightforward, they do not take into account some aspects that are indeed considered by the purse-based method used in the last SAT competition. On the other hand, the purse-based method requires some oracle to assign purses to the problem instances, so the results can be influenced heavily by the oracle. YASM is a first attempt to combine the two approaches: a rich method like the purse-based one, but using the data obtained from the runs only. As such, YASM requires a preliminary classification whereby a hardness degree H_i is assigned to each problem instance i using the same equation as in CASC [1]:

$$H_i = 1 - \frac{S_i}{S_t} \tag{3}$$

where S_i is the number of solvers that solved i, and S_t is the total number of participants to the contest. Considering equation (3), we notice that $0 \leq H_i \leq 1$, where $H_i = 0$ means that i is relatively easy, while $H_i = 1$ means that i is relatively hard. We can then compute the score $SB_{s,i}$ of a solver s on a given instance i:

$$SB_{s,i} = k \cdot H_i \cdot \frac{L - T_{s,i}}{L - M_i} \tag{4}$$

where k is a constant (we fix $k = 100$ to normalize H_i in the range $[0; 100]$), L is the time limit, $T_{s,i}$ is the CPU time used up by s to solve i ($T_{s,i} \leq L$), and $M_i = min_s\{T_{s,i}\}$. Notice that $SB_{s,i} = 0$ whenever RESULT is TIME and we force $SB_{s,i} = 0$ also when RESULT is FAIL. M_i is the time spent on the instance i by the *SOTA solver* defined in [9] to be the ideal solver that always fares the best time among all the participants. The score of a solver SB_s is just the sum of the scores obtained on the instances, i.e., $SB_s = \sum_i SB_{s,i}$.

The score SB_s introduced so far is just a combined speed/solution bonus, and it does not take into account explicitly the number of times that RESULT is either TIME or FAIL. To complete the YASM method and compute the total score S_s of a solver s, we extend the basic scoring mechanism of equations (3) and (4) by awarding bonuses to solvers in such a way that the amount of bonuses received is inversely proportional to the number of times that the solvers reach the time limit or fail. Let Γ be the set of instances used for the contest and $\Gamma_{l,s}$ (resp. $\Gamma_{f,s}$) be the set of instances on which the solver s reaches the time limit (resp. fails). We compute the time limit bonus LB_s of a solver s as:

$$LB_s = k_l \cdot \frac{|\Gamma| - |\Gamma_{l,s}|}{|\Gamma|} + k_l \cdot C_{l,s} \tag{5}$$

and the fail bonus FB_s as:

$$FB_s = k_f \cdot \frac{|\Gamma| - |\Gamma_{f,s}|}{|\Gamma|} + k_f \cdot C_{f,s} \tag{6}$$

CASC	QBF	SAT	YASM	Borda	r.v.	s.v.
–	1	0.71	0.86	0.86	0.71	0.86
	–	0.71	0.86	0.86	0.71	0.86
		–	0.86	0.71	0.71	0.71
			–	0.71	0.71	0.71
				–	0.86	1
					–	0.86
						–

Table 1: Homogeneity between scoring methods.

where k_l and k_f are constant parameters (we set $k_l = k_f = 1/2 \cdot max_s\{S_s\}$); $C_{l,s}$ and $C_{f,s}$ are the *time limit coefficient* and the *failure coefficient* respectively. $C_{x,s}$ with $x \in \{f, l\}$ is given by:

$$C_{x,s} = \begin{cases} \frac{\sum_{i \in \Gamma_{x,s}} H_i}{|\Gamma_{x,s}|} & \text{if } |\Gamma_{x,s}| > 0 \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

Equations (5) and (6) convey the intuition that a good solver is one that reaches the time limit or fails in a small number of cases (first addendum), and when it does, this happens mainly on hard instances (second addendum). Notice that LB_s (resp.FB_s) has the highest value when $|\Gamma_{l,s}| = 0$ (resp. $|\Gamma_{f,s}| = 0$), i.e., there are no instances on which s reaches the time limit (resp. fails). In this case, considering our choice of k_l and k_f detailed above, $LB_s = FB_s = max_s\{S_s\}$.

The total score S_s of a solver according to YASM is thus computed with a weighted sum of three elements:

$$S_s = \alpha \cdot SB_s + \beta \cdot LB_s + \gamma \cdot FB_s \tag{8}$$

The parameters α, β and γ can be tuned to improve the quality of the scoring method, or to take into account specific characteristics of the solver, e.g., in a competition of incomplete solvers it may be reasonable to set $\gamma = 0$. In the analysis hereafter presented we set $\alpha = 0.5$, $\beta = 0.25$ and $\gamma = 0.25$.

3.3 Effectiveness measures

This Subsection describes the measures that we introduce to assess quantitatively the effectiveness of the scoring methods.

Homogeneity The rationale behind this measure is to verify that, on a given test set, the scoring methods considered (i) do not produce exactly the same solver rankings, but, at the same time, (ii) do not yield radically different solver rankings. If (i) was the case, evaluating different methods would not make a lot of sense, since any of them would produce the same results. If (ii) was the case, since we have no clue about the absolute value of the competitors, it would be impossible to decide which method is the right choice. Measuring homogeneity is thus fundamental

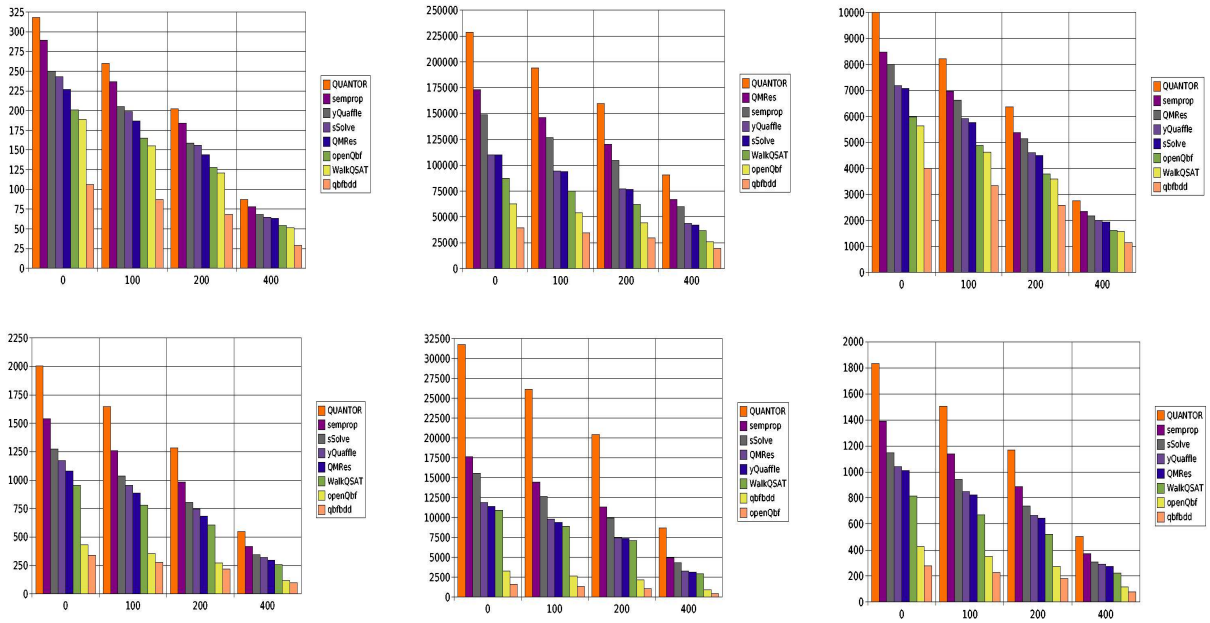

Figure 1: RDT-stability plots.

to ensure that we are considering alternative scoring methods, yet we are performing an apple-to-apple comparison.

To measure homogeneity, we consider the Kendall rank correlation coefficient τ [15]. τ is computed between any two rankings and is such that $-1 \leq \tau \leq 1$, where $\tau = -1$ means perfect disagreement, i.e., one ranking is the opposite of the other, $\tau = 0$ means independence, i.e., the rankings are not comparable, and $\tau = 1$ means perfect agreement, i.e., the two rankings are the same. For $0 \leq \tau \leq 1$, increasing values of τ imply increasing agreement between the rankings. Table 1 shows the values of τ computed for the scoring methods considered, arranged in a symmetric matrix where we omit the row names, and the elements below the diagonal (r.v. is a shorthand for range voting, while s.v. is a shorthand for sum of victories). Values of τ close to, but not exactly equal to 1 are desirable. Table 1 shows that this is indeed the case for the scoring methods considered using QBFEVAL 2005 data. Only two couples of methods (QBF-CASC and sum of victories-Borda) show perfect agreement, while all the other couples agree to some extent, but still produce different rankings.

RDT-stability Stability on a randomized decreasing test set (RDT-stability for short) aims to measure how much a scoring method is sensitive to perturbations that diminish the size of the original test set. We evaluate RDT-stability using several test sets computed by removing instances uniformly at random from the original test set. On the reduced test sets we compute the scores, and then we take the median over the reduced test sets in order to establish a new ranking. If such ranking is the same as the one obtained on the original test set, then we conclude that the scoring method is RDT-stable up to the number of instances discarded from the original test set.

More precisely, assume that n solvers are participating to the contest, let Γ be the set of instances used for the contest, and R_q be the ranking produced by a given scoring method q. We consider a set of k test sets $\{\Gamma_{m,1}, \ldots, \Gamma_{m,k}\}$, where each $\Gamma_{m,i}$ is obtained by discarding, uniformly at random without repetitions, m instances from Γ. For each $\Gamma_{m,i}$, we then apply q considering only the entries in RUNS whose INSTANCE values appear in $\Gamma_{m,i}$. Thus, for each $\Gamma_{m,i}$, we obtain the set of n scores $\{S_1|\Gamma_{m,i}, \ldots, S_n|\Gamma_{m,i}\}$. Let $\overline{S}_j|\Gamma_m = \mathit{median}_i(S_j|\Gamma_{m,i})$. The set $\{\overline{S}_1|\Gamma_m, \ldots, \overline{S}_n|\Gamma_m\}$ is used to produce a ranking $R_{q,m}$. If $R_{q,m}$ is the same as R_q, then we say that q is RDT-stable up to m. In the experiments presented in Section 4 the value of k is always 100.

DTL-stability Stability on a decreasing time limit (DTL-stability for short) aims to measure how much a scoring method is sensitive to perturbations that diminish the maximum amount of CPU time granted to the solvers. Let L be initial value of such quantity. To compute DTL-stability of a scoring method q, we apply it considering a new time limit L' such that $L - L' = t$ and $t > 0$. If the ranking R'_q obtained using L' instead of L is the same as the original ranking R_q, then we conclude that the scoring method is DTL-stable up to t, i.e., the amount of time subtracted from the original time limit.

SBT-stability Stability on a solver biased test set (SBT-stability for short) aims to measure how much a scoring method is sensitive to a test set that is biased in favor of a given solver. Let Γ be the original test set, and Γ_s be the subset of Γ such that the solver s is able to solve exactly the instances in Γ_s. Let $R_{q,s}$ be the ranking obtained by applying the scoring method q on Γ_s. If $R_{q,s}$ is the same as the original ranking R_q, then the scoring method q is SBT-stable with respect to the solver s. Notice that a good SBT-stability is necessary for a scoring method to detect the absolute quality of the participants, even against flaws in the design of the test set.

SOTA-relevance This measure aims to understand the relationship between the ranking obtained with a scoring method and the strength of a solver, as witnessed by its contribution to the SOTA solver. As mentioned in Subsection 3.2, the SOTA solver is the ideal solver that always fares the best time among all the participants. Indeed, a participant contributes to the SOTA solver whenever it is the fastest solver on some instance. The count of such events for a given solver is the quantitative measure of the SOTA contribution. We measure the SOTA-relevance using Kendall coefficient to compare the ranking induced by the SOTA contribution with the rankings obtained using each of the scoring methods considered.

4 Experimental results

In this section we present the experimental evaluation of the scoring methods presented in the Subsections 3.1 and 3.2, using the measures introduced in Subsection 3.3.

We start from RDT-stability and the plots in Figure 1. The histograms are arranged in two rows and three columns: the first row shows, from left to right, the plots regarding QBF/CASC, SAT and YASM scoring methods, while the second row shows, again from left to right, the plots regarding Borda count, range voting and sum of victories. Each histogram in Figure 1 reports, on the x-axis the number of problems m discarded from the original test set (0, 100, 200 and 400 out of 551) and on the y-axis the score. For each value of the x-axis, eight bars are displayed, corresponding to the scores of the solvers admitted to the second stage of QBFEVAL 2005. In all the plots of Figure 1 the legend is sorted according to the ranking computed by the specific scoring method, and the bars are also displayed accordingly. This makes easier to identify perturbations of the original ranking, i.e., the leftmost group of bars corresponding to $m = 0$.

Considering Figure 1, we can immediately conclude that all the scoring methods considered are RDT-stable up to 400. This means that a random sample of 151 instances is sufficient for all the scoring methods to reach the same conclusions that each one reaches on the much heftier test set of 551 instances used in QBFEVAL 2005. A general consideration that may be extracted from these results is that a substantial number of problem instances is not needed in order to perform a proper evaluation, at least with the scoring methods that we consider. On the other hand, RDT-stability does not allow us to discriminate among the different scoring methods, since all of them perform equally well under this point of view.

We continue our analysis by considering DTL-stability. Figure 2 reports six histograms arranged in the same way as Figure 1, except that the x-axis now reports the amount of CPU time seconds used as a time limit when evaluating the scores of the solvers. The leftmost value is $L = 900$, i.e., the original time limit that produces the ranking according to which the legend and the bars are sorted, and then we consider the values $L' = \{700, 500, 300, 100, 50, 10, 1\}$ corresponding to $t = \{200, 400, 600, 800, 850, 890, 899\}$.

Considering Figure 2 we see that CASC/QBF scoring methods (Figure 2, top-left) are DTL-stable up to $t = 400$. For $L' = 300$, there is a noticeable perturbation, i.e., an exchange of position in the ranking between WALKQSAT and OPENQBF. The ranking then stabilizes until $L' = 100$ and then, as one would expect, it is perturbed more heavily in the rightmost part of the plot. Also the SAT competition scoring method (Figure 2, top-center) is DTL-stable up to $t = 400$, while perturbations occur for higher values of t. YASM (Figure 2, top-right) DTL-stable up to $t = 850$. For $t = 890$ and $t = 899$ there is a relatively high instability, but, as for all the other scoring methods analyzed so far, the relative position of the best solver and the worst solver does not change. Borda count (Figure 2, bottom-left) is remarkably DTL-stable up to $t = 890$, and it is the best scoring method as far DTL-stability is concerned. Finally, range voting (Figure 2, bottom-center) and sum of victories (Figure 2, bottom-right) are both DTL-stable up to $t = 850$. In conclusion, while Borda count shows excellent DTL-stability, YASM turns out to be the best among some of the methods used in automated reasoning contests. Overall, we notice that decreasing the time limit substantially, even up to one order of magnitude, is not influencing the stability of the scoring methods considered, except for some minor perturbations for QBF/CASC and SAT methods. Moreover, independently from the scoring method used and the amount of CPU time granted, the best solver is always the same. A general consideration that may be extracted from these results is that increasing the time limit of a competition is useless, unless the increase is substantial, i.e., orders of magnitude.

Figure 3 shows the plots with the results of the SBT-stability measure for each scoring method (the layout is the same as Figures 1 and 2). The x-axis reports the name of the solver s used to compute the solver-biased test set Γ_s and the y-axis reports the score value. For each of the Γ_s's, we report eight bars showing the scoring obtained by the solvers using only the instances in Γ_s. The order of the bars (and of the legend) corresponds to the ranking obtained with the given scoring method considering the original test

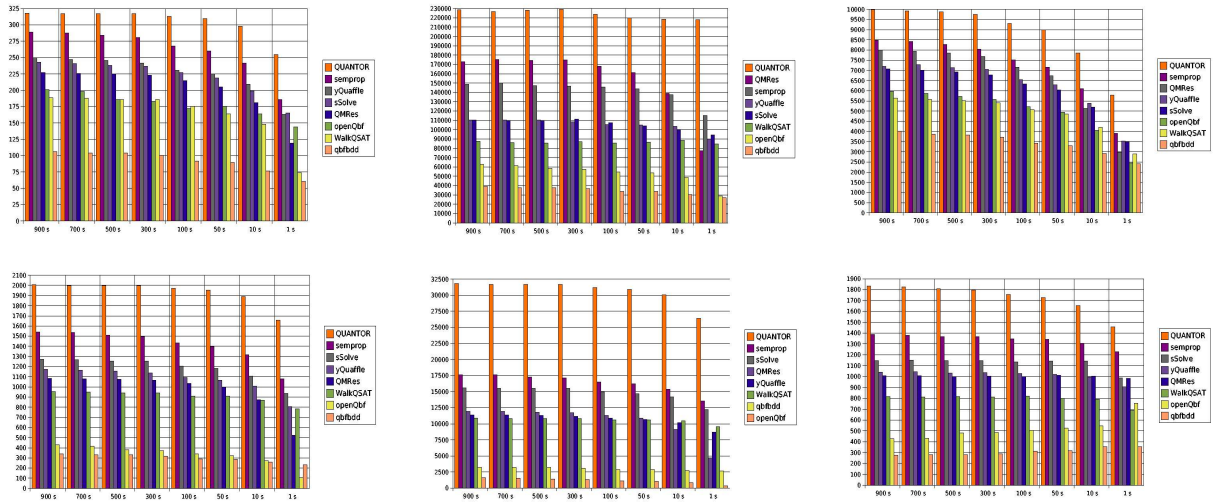

Figure 2: DTL-stability plots.

	CASC/QBF	SAT	YASM	Borda	r.v.	s.v.
OPENQBF	0.43	0.57	0.36	0.79	0.79	0.71
QBFBDD	0.43	0.43	0.36	0.79	0.86	0.79
QMRES	0.64	0.86	0.76	0.71	0.86	0.71
QUANTOR	1	0.86	0.86	0.93	0.86	1
SEMPROP	0.93	0.71	0.71	0.93	0.86	0.93
SSOLVE	0.71	0.57	0.57	0.86	0.79	0.86
WALKQSAT	0.57	0.57	0.43	0.64	0.79	0.71
YQUAFFLE	0.71	0.64	0.57	0.86	0.86	0.86
Mean	0.68	0.65	0.58	0.81	0.83	0.82

Table 2: Comparing rankings on solver-biased test sets vs. the original test set

	SOTA ranking
CASC	0.57
QBF	0.57
SAT	0.64
YASM	0.57
Borda	0.71
range voting	0.86
sum of victories	0.71

Table 3: Comparing scoring methods and SOTA contributes.

set Γ.

As we can see from Figure 3 (top-left), CASC/QBF scoring methods are SBT-stable with respect to their alleged winner QUANTOR, while they are not SBT-stable with respect to all the other solvers: for each of the Γ_s where $s \neq$ QUANTOR, the original ranking is perturbed and the winner becomes s. The SAT competition scoring method (Figure 3, top-center) and YASM (Figure 3, top-right) are not SBT-stable with respect to any solver, not even with respect to their alleged winner. Borda count (Figure 3, bottom-left) is not SBT-stable with respect to any solver, but the alleged winner (QUANTOR) is always the winner on the biased test sets. Moreover, the rankings obtained on the test sets biased on QUANTOR and SEMPROP are not far from the ranking obtained on the original test set. Also range voting (Figure 3, bottom-center), is not SBT-stable with respect to any solver, but the solvers ranking first and last do not change over the biased test sets. Finally, sum of victories (Figure 3, bottom-right) is SBT-stable only with respect to its alleged winner QUANTOR.

Looking at the results presented above, we can only conclude that all the scoring methods are sensitive to a bias in the test set. However, some methods seem more robust than others, e.g., the winner according to Borda count and range voting seems less bias-prone than the winner according to other methods such as SAT or YASM. SBT-stability is an on-off criteria which does not allow us to quantify these subtle differences. In order to do so, in Table 2, for each scoring method we compute the Kendall coefficient between the ranking obtained on the original test set Γ and each of the rankings obtained on the Γ_s test sets. In Table 2 each column, but the first, is relative to a scoring method, and each row, but the first and the last, is relative to a specific Γ_s. The last row reports the mean value of the coefficients for each scoring method. Overall, YASM turns out to be the method more sensitive to a bias in the test set, immediately followed by CASC/QBF and SAT. On the other hand, the methods based on voting theory are all more robust than automated reasoning contests scoring methods. A general consideration that we can extract is that, whatever the choice of the scoring method, it is very important to assemble a test set in such a way to minimize bias, e.g., by extracting random samples from a large instance base.

We conclude our analysis with Table 3, showing the relationship between the ranking computed by each scoring method, and the ranking induced by the contribution of

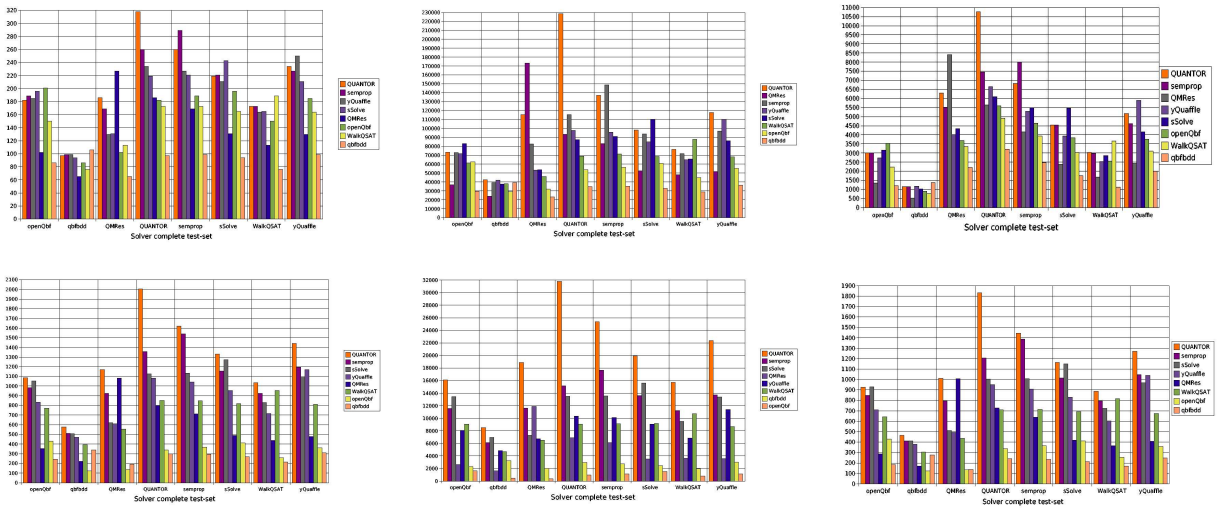

Figure 3: SBT-stability plots.

each solver to the SOTA solver. Column 2 of Table 3 shows the values of the Kendall coefficient for each scoring method. Notice that the results of this table are very close to those summarized in the last row of Table 2. This seems to imply that robustness and SOTA relevance are somehow related qualities of a scoring method. Under this point of view, the methods based on voting theory seem to have an edge over automated reasoning contests scoring methods.

5 Conclusions

Summing up, our analysis allowed us to put forth some considerations that are independent of the specific scoring method is used. First, a *large* test set is not necessarily a *better* test set: the results about DRT-stability show that there is no visible difference in rankings even when slashing the original test by 70%. Second, increasing the time limit does not seem to change the overall piucture, unless you are prepared to increase it substantially, e.g., by orders of magnitude: DTL-stability shows that there is no visible difference in rankings even unless the initial time limit is reduced substantially. Third, the composition of the evaluation test set may heavily influence the final ranking: the results on SBT-stability tell us that no scoring method considered is immune from bias in the original test set. As for the specific properties of the scoring methods, those based on voting theory seem to have an edge over automated reasoning contests scoring methods, particularly for SBT-stability and SOTA-relevance.

REFERENCES

[1] G. Sutcliffe and C. Suttner. The CADE ATP System Competition. `http://www.cs.miami.edu/~tptp/CASC`.

[2] D. Le Berre and L. Simon. The SAT Competition. `http://www.satcompetition.org`.

[3] M. Narizzano, L. Pulina, and A. Taccchella. QBF solvers competitive evaluation (QBFEVAL). `http://www.qbflib.org/qbfeval`.

[4] D. Long and M. Fox. The 3rd International Planning Competition: Results and Analysis. *Artificial Intelligence Research*, 20:1–59, 2003.

[5] J. N. Hooker. Testing Heuristics: We Have It All Wrong. *Journal of Heuristics*, 1:33–42, 1996.

[6] D. G. Saari. *Chaotic Elections! A Mathematician Looks at Voting*. American Mathematical Society, 2001.

[7] RangeVoting.org. `http://math.temple.edu/~wds/crv/`.

[8] The Condorcet Method. Reference available on line from `http://en.wikipedia.org/wiki/Condorcet_method` . Visited in April 2006.

[9] M. Narizzano, L. Pulina, and A. Tacchella. The third QBF solvers comparative evaluation. *Journal on Satisfiability, Boolean Modeling and Computation*, 2:145–164, 2006. Available on-line at `http://jsat.ewi.tudelft.nl/`.

[10] E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formulas satisfiability library (QBFLIB), 2001. `www.qbflib.org`.

[11] E. Giunchiglia, M. Narizzano, and A. Tacchella. Clause-Term Resolution and Learning in Quantified Boolean Logic Satisfiability. *Artificial Intelligence Research*, 2006. Accepted for publication.

[12] Y. Yu and S. Malik. Verifying the Correctness of Quantified Boolean Formula(QBF) Solvers: Theory and Practice. In *ASP-DAC*, 2005.

[13] M. Benedetti. Extracting Certificates from Quantified Boolean Formulas. In *IJCAI 2005*, 2005.

[14] A. Van Gelder, D. Le Berre, A. Biere, O. Kullmann, and L. Simon. Purse-Based Scoring for Comparison of Exponential-Time Programs, 2006. Unpublished draft.

[15] M. Kendall. *Rank Correlation Methods*. Charles Griffin & Co. Ltd., 1948. Reference available on line from `http://en.wikipedia.org/wiki/Rank_correlation`.