Careful Ranking of Multiple Solvers with Timeouts and Ties

Allen Van Gelder
http://www.cse.ucsc.edu/∼avg

University of California, Santa Cruz, CA 95064

Abstract. In several fields, Satisfiability being one, there are regular competitions to compare multiple solvers in a common setting. Due to the fact some benchmarks of interest are too difficult for all solvers to complete within available time, time-outs occur and must be considered. Through some strange evolution, time-outs became the only factor that was considered in evaluation. Previous work in SAT 2010 observed that this evaluation method is unreliable and lacks a way to attach statistical significance to its conclusions. However, the proposed alternative was quite complicated and is unlikely to see general use.
This paper describes a simpler system, called *careful ranking*, that permits a measure of statistical significance, and still meets many of the practical requirements of an evaluation system. It incorporates one of the main ideas of the previous work: that outcomes had to be freed of assumptions about timing distributions, so that non-parametric methods were necessary. Unlike the previous work, it incorporates ties.
The *careful ranking* system has several important non-mathematical properties that are desired in an evaluation system: (1) the relative ranking of two solvers cannot be influenced by a third solver; (2) after the competition results are published, a researcher can run a new solver on the same benchmarks and determine where the new solver would have ranked; (3) small timing differences can be ignored; (4) the computations should be easy to understand and reproduce. Voting systems proposed in the literature lack some or all of these properties.
A property of *careful ranking* is that the pairwise ranking might contain cycles. Whether this is a bug or a feature is a matter of opinion. Whether it occurs among leaders in practice is a matter of experience.
The system is implemented and has been applied to the SAT 2009 Competition. No cycles occurred among the leaders, but there was a cycle among some low-ranking solvers. To measure robustness, the new and current systems were computed with a range of simulated time-outs, to see how often the top rankings changed. That is, times above the simulated time-out are reclassified as time-outs and the rankings are computed with this data. *Careful ranking* exhibited many fewer changes.

1 Introduction and Overview

Empirical comparison of computational performance is an important technique for advancing the state of the art in software. In Propositional Satisfiability and several related fields testing programs on benchmarks is complicated by the fact that time limits must be set, because some benchmarks of interest are too difficult for all programs to complete within available time. Programs can fail to complete a test due to exhausting time or some other resource, often memory. There is no clearly correct way to integrate the results of failed tests and completed tests, to compute a single "figure of merit."

To focus the discussion, let us assume that the property we wish to measure is speed of solution, and we are evaluating the results of a SAT competition. If every program could complete every test, we would simply add up the times for each program and rank them according to this total, with smallest being best. From this point of view, time-outs and other failures are defects in the experiment.

In actuality, there is not time for every program to complete every test, and failures do occur. This leads to what is called *censored data* in the literature: there "really" is a value for the time the program would have taken on a benchmark, we just did not find out what that value is. The question is, what is a good way to rank the programs, based on the data that *is* available. Logically, we would want this ranking method to produce the same results as the ideal experiment, to the extent possible.

The ranking method that has been used in recent SAT competitions, which we shall call ***solution-count ranking***,[1] is to set some time limit *ad hoc*, and simply count how many tests are successfully completed. Through some strange evolution, time-outs, which are the manifestations of defects in the experiment, became the only factor that was considered in evaluation. Total CPU time is used as a tie-breaker only if solution counts are equal.

Previous work by Nikolić observed that **solution-count ranking** is unreliable and lacks a way to attach statistical significance to its conclusions [Nik10]. However, the proposed alternative was quite complicated and had some practical drawbacks. The purpose of this paper is to describe and propose a simpler system that meets the practical requirements for ranking solvers in a SAT competition (endorsed by a survey of solver developers and users),[1] and also gives information about the statistical significance of the results, or lack thereof.

Definition 1 ***Practical requirements***:

1. The relative ranking of two solvers cannot be influenced by a third solver.
2. After the competition results are published, a researcher can run a new solver on the same benchmarks and determine where the new solver would have ranked.
3. Small timing differences can be ignored.
4. The computations should be easy to understand and reproduce.

One earlier method, called the *purse* method,[2] lacked properties (1) and (2) and fell into disfavor after a few trials. □

[1] See `http://www.satcompetition.org/2009/spec2009.html`, where it is called "Lexicographical NBSOLVED, sum ti."

[2] See `http://www.satcompetition.org/2007/rules07.html`.

The methodology we propose, called ***careful ranking***, incorporates one of the main ideas of Nikolić: that outcomes must be free of assumptions about timing distributions, because we have no information about these distributions. Non-parametric methods are necessary. Unlike the previous work, our proposal incorporates ties to account for timing differences that are considered inconsequential for ranking purposes.

The **careful ranking** system has the important non-mathematical properties given in Definition 1. The main ingredient of **careful ranking** is that all pairs of competitors are compared in isolation, leading to a pair of "scores" that sum to zero. A large positive score indicates a significantly faster solver. The null hypothesis is that both solvers are equally fast "overall," or "in the long run." The *expected value* of the score is zero, under this hypothesis. The difference between zero and the observed score may be converted into a standard measure of statistical significance.

For a k-way competition, there are $k(k-1)/2$ pairwise matches. The results are expressed with a dominance matrix, as described in Section 5. The final ranking is extracted from this matrix.

There is a meta-ranking question to be addressed. How can we compare various ranking methods, since we do not know the "true answers?" The method we propose, and use, is to measure sensitivity to changes in the time limit. We do not know what would have happened if we used a larger time limit. But what would have happened under all *shorter* time limits can be determined from the available data.

The **careful ranking** system is implemented[3] and has been applied to the SAT 2009 Competition. The implementation is `csh` scripts, `sed`, and `awk`, which should be portable. No cycles occurred among the leaders, but there was a cycle among some low-ranking solvers. To measure robustness, the new and current systems were computed with a range of simulated time-outs, to see how often the top rankings changed. That is, times above the simulated time-out are reclassified as time-outs and the rankings are computed with this data.

2 Related Work

There is a large body of work on various aspects of experimental comparisons. We restrict ourselves to immediately related work on ranking solvers. Non-mathematical considerations for a scoring method are discussed in general terms by Le Berre and Simon [LBS04], and influenced several aspects of the method proposed here. One such aspect is our provision for many timing differences to be treated as a tie, because it appears that many people consider calling one solver the winner in these cases is a distortion. The reaction to this perceived distortion has been to reduce the importance of speed to nearly nothing, as long as the solver stays within the time limit. We hope that treating "minor" timing differences as ties will make the technique more acceptable than prior techniques that used time as the major consideration.

[3] Code is at `http://www.cse.ucsc.edu/∼avg/CarefulRanking/`.

Brglez and co-authors [BLS05,BO07] replicate instances into classes to gather statistics. Their goals are quite different from ranking a competition. Nikolić [Nik10] extends these ideas to compare more than two programs. The non-mathematical, practical issues mentioned in Definition 1 are not considered in these papers.

Pulina conducted an extensive empirical evaluation of several scoring methods [Pul06]. One criterion he used is similar to the one we use, decreasing the time limit and measuring stability. Our proposed method is significantly different from those he analyzed. Most or all of the comparison methods he studied lacked the independence from a third solver. Thus a later researcher could not see where new work fit into a previous competition.

Pulina introduced the idea of viewing the ranking problem as a *voting situation*: each benchmark "votes" for the solvers (the "candidates") by a preference ballot that ranks them by solution time. This is a very attractive idea, but unfortunately, none of the well-known proposed voting methods satisfy the criteria of Definition 1 and elaborated further in the URL given there. There is a vast literature on this subject, as surveyed by Levin and Nalebuff [LN95], and more recently treated by Pomerol and Barba-Romero [PBR00] and Tideman [Tid06].

A detailed comparison with all proposals would take us far afield, so we restrict attention to the Schulze method, which has enjoyed recent popularity [Sch03]. That popularity is not surprising, because the Schulze method, unlike many other proposals (such as Borda), permits voters to vote equal preferences among subsets of the candidates (e.g., D=1, (A, C)=2, B=7 is a valid ballot in a field of 10).

Suppose a competition is being run with six solvers and 63 benchmarks with Schulze ranking (the example may use many combinations of numbers), and the following events transpire. After 60 benchmarks have been run on all solvers, the Schulze ranking is computed and solver A is uniquely winning. On each of the last three benchmarks, solver A has the best performance of any solver and solver D has the *worst* performance of any solver. (For example, solver D might time out on the last three benchmarks). However, when the Schulze ranking is computed using all 63 benchmarks, D wins. No, this is not a typo. See Appendix B of [Sch03] for complete details.[4]

It is impossible to imagine that any organizers of a competition would adopt the Schulze method, if they know about this possibility. Moreover, this is not a quirk in the Schulze method. It is known to be present in a large class of methods that satisfy the *Condorcet principle* [LN95,Tid06]. The phenomenon is known as the *no-show paradox* [Mou88], because solver A would have been better off without the "support" of the last three benchmarks, on all of which A was the clear winner.

The above example is possible under Schulze ranking and many other voting systems because it does not satisfy criterion (1) in Definition 1, that other solvers should not be able to affect the *relative ranking* of solvers A and D.

[4] The example cited has some pairwise ties among candidates, for simplicity of presentation, but these ties can be removed by "fuzzing" without changing the outcomes.

3 What is a Tie?

Say we are comparing solvers (R, S) on a set of benchmarks, $\{B_i \mid i = 1, \ldots, n\}$, with time limit τ. The data is two lists of numbers, $t_i(R)$ and $t_i(S)$, the solution times of the two solvers on B_i. Numbers are floating point and include Inf to denote a failure of any kind. (We choose not to distinguish among failure reasons, except that a *wrong answer* means the solver is disqualified and its matches are not scored.) All data other than Inf is between 0 and τ, and we call these *finite* times.

We interpret the lists $t_i(R)$ and $t_i(S)$ as a series of n mini-matches, each with a stake of one point. A tie awards 0 to each solver. A win for R gives R a score of 1 and gives S a score of -1, and the reverse if S wins.

Clearly if $t_i(R) = t_i(S)$, the result is a tie. The question is what other outcomes should be considered a tie. The current method treats any pair of finite times as a tie; the only win is a finite time vs. Inf. The opposite extreme is that any $t_i(R) < t_i(S)$ is a win for R. Nikolić performed a theoretical analysis that depended on a complete absence of ties, so he "discarded" benchmarks where all times were under 5 seconds, which got rid of all the exact equalities with finite times, and then treated any finite time difference as a win [Nik10]. This is essentially the same as saying any pair of times under 5 seconds is a tie.

Our thesis is that some time differences should be considered "inconsequential" in the sense that someone trying to select the better solver between R and S for use in an application would not be influenced these time differences. We hypothesize that on longer runs, larger time differences would be considered inconsequential, so we want to define a **tie zone** whose width grows as run times get longer. We also believe that most people agree that below some threshold, *all* time differences are inconsequential. The user decides where to set this threshold, which we call **noise**, and which is the only user-specified parameter needed to specify the tie zone.

The growth rate we choose is founded in recurrent-event theory. We model the solver's computation as a long series of search events with independent outcomes. The probability that a search event has a successful outcome is very small, and the solver terminates upon the first successful search event. This is the well-known Poisson process. The *standard deviation* of the time to termination is proportional to the *square root* of the *average* time to termination. If two solvers have the same (theoretical) average time to termination on benchmark B_i, then their time difference is a random variable with mean zero and standard deviation proportional to the square root of their common average. We propose that observed time differences less than some number of standard deviations should be considered as ties, because they do not provide compelling evidence that one solver's average is really shorter than the other's. This is purely an heuristic model, of course.

Once we accept the idea that it is sensible for the tie zone to grow proportionally to $\sqrt{(t_i(R) + t_i(S))/2}$, the square root of the average of the two observed solving times, all that remains is to choose a constant of proportionality. The user makes this choice indirectly by specifying a scalar parameter called **noise**.

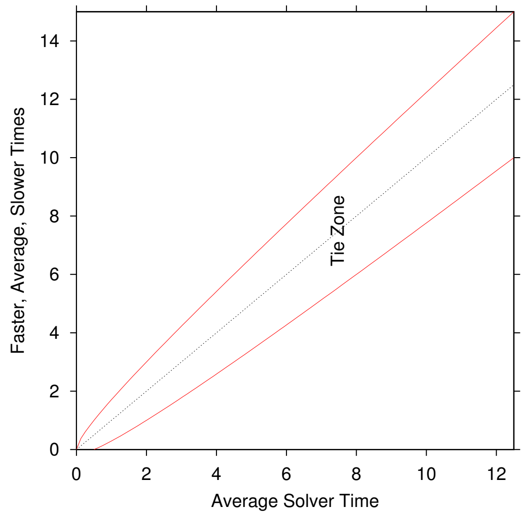

Fig. 1. Tie Zone for **noise** = 1 minute; times in minutes. Lower curve: faster solver time; upper curve: slower solver time; middle line: average of upper and lower curves.

As stated above, the intuition is that all solution times at or below the **noise** level should be treated as indistinguishable. Figure 1 shows the tie zone for **noise** = one minute. Any pair of times, both under one minute, fall into the tie zone.

To generate the desired tie zone, we define:

$$\begin{aligned} \alpha &= \sqrt{\textbf{noise} \,/\, 2} \\ \varDelta &= \alpha \,\sqrt{(t_i(R) + t_i(S))/2}. \end{aligned}$$

Then the "tie zone" extends from $(t_i(R)+t_i(S))/2 - \varDelta$ to $(t_i(R)+t_i(S))/2 + \varDelta$. For R to win it is necessary that

$$t_i(R) < (t_i(R) + t_i(S))/2 \; - \; \varDelta$$

Since $t_i(R) \geq 0$, S is assured of (at least) a tie whenever $t_i(S) \leq$ **noise**, as was desired by the user.

4 Pairwise Matches

Say we are comparing solvers (R, S) on a set of benchmarks, $\{B_i \mid i = 1, \ldots, n\}$, with time limit τ. The data is two lists of numbers, $t_i(R)$ and $t_i(S)$. As described in Section 3, we interpret the lists $t_i(R)$ and $t_i(S)$ as a series of n mini-matches, with each outcome for R being -1, 0, or 1. The (algebraic) total is the raw score for the match, denoted $\mathit{raw}(R, S)$. For simplicity, all pairs are processed, so S is compared with R at some point to get $\mathit{raw}(S, R)$, which of course equals $-\mathit{raw}(R, S)$.

The value of $\mathit{raw}(R, S)$ can be used to test the null hypothesis, which is that R and S have an equal probability of winning a random mini-match. We also need the number of decisive (non-tie) mini-matches, denoted $\mathit{decisive}(R, S)$. Then the Student t parameter is given by

$$\text{Student } t = \frac{\mathit{raw}(R, S)}{\sqrt{\mathit{decisive}(R, S)}}, \tag{1}$$

which expresses the raw score in standard deviations. Statistically, the match is modeled as $\mathit{decisive}(R, S)$ fair coin flips. If $\mathit{decisive}(R, S)$ is large, the distribution is close to Gaussian. We are certainly justified in rejecting the null hypothesis when $|t| \geq 2$, without figuring out the exact value of p, which is the probability of observing a t-value this large or larger. In this case $p < 0.03$.

To summarize, if Student $t \geq 2$ in (1), we may conclude that R is faster than S with high confidence, on a space of benchmarks for which the actual benchmarks used are representative. If Student $t = 1$ we may have "medium" confidence, because a value this large or larger would occur with probability about 0.16 if the solvers were really equally fast on average.

5 Competition Ranking

We propose to create a k-way competition ranking of solvers S_1, …, S_k by forming a $k \times k$ matrix M in which

$$M_{i,j} = \begin{cases} 1 & \text{if } \mathit{raw}(S_i, S_j) > 0 \\ 0.5 & \text{if } \mathit{raw}(S_i, S_j) = 0 \\ 0 & \text{if } \mathit{raw}(S_i, S_j) < 0 \end{cases} \tag{2}$$

This matrix can be interpreted as specifying a directed graph (also called M), where solvers are vertices and an edge from S_i to S_j exists wherever $M_{i,j} \neq 0$. If $M_{i,j} = M_{j,i} = 0.5$, there are edges in both directions. If this graph is acyclic, it defines a total order among the solvers, which we call the *dominance* order. In practice, we usually are not concerned about establishing a total order among *all* participants; it is sufficient if there is a total order among the leaders, perhaps the 5–6 top ranks.

First, let us focus on the case that the leaders do not have any tied matches, not even with a non-leader. Things are slightly more complicated otherwise. In the case of no leader ties, adding up the rows of the leaders provides a "definitive" ranking. That is, if M restricted to the leaders defines an acyclic graph, each row sum is unique, and ranking the leaders by row sums is unambiguous.

However, it is possible, even in the case of no leader ties, that the graph has a cycle [Nik10]. This possibility is present because ties are not transitive in mini-matches. In other words, on a specific benchmark B_i, it is possible that S_1 ties with S_2 and S_2 ties with S_3, but S_1 wins or loses against S_3. We can easily create a set of timings on three benchmarks so that $\mathit{raw}(S_1, S_2) = 1$, $\mathit{raw}(S_2, S_3) = 1$, and $\mathit{raw}(S_3, S_1) = 1$. Longer and more complex cyclic structures can be constructed, as well. If a cyclic structure is present, then at least two row sums must be equal (still in the case of no leader ties in matches).

The conclusion from the preceding discussion is that row sums can provide quick hints, are easily interpreted, but may be inconclusive. If used carelessly in the presence of ties, they can be misleading. On the positive side, we expect them to be adequate for most situations. But "most" is not good enough, so we need a procedure that always gives an unambiguous result.

Example 2 This small example shows some complications that can arise, involving pairwise ties and cycles. Let us assume that the time-out is 15 and that a difference of 3 is a winning margin in the range of times shown below, while a difference of 2 is a tie. The left side shows times for three solvers on three benchmarks. The middle shows the raw scores. The right side shows the dominance matrix.

	S_1	S_2	S_3
B_1	10	13	14
B_1	14	12	10
B_3	12	11	14

	S_1	S_2	S_3
S_1	0	1	0
S_2	-1	0	1
S_3	0	-1	0

	S_1	S_2	S_3
S_1	0	1	0.5
S_2	0	0	1
S_3	0.5	0	0

Although S_1 beats S_2 and S_2 beats S_3, still S_3 ties S_1, so all three are cyclically related. However, no row-sums are equal. □

Treating M simply as a connected, directed graph, its vertices (the solvers) can be partitioned into strongly connected components. (For small graphs, the famous linear-time procedure is unnecessary; matrix multiplications and additions suffice.) The component graph, obtained by collapsing every strongly connected component to a single node, defines a total order.

We propose that all solvers living in the same ***strongly connected component* (SCC)** of the graph M described in (2) shall be equally ranked; otherwise the relative ranking is determined by the component graph. This policy provides an unambiguous specification for all situations.

If a tie-break is necessary (e.g., an indivisible trophy is awarded), we recommend that all solvers in a single SCC shall be ranked among themselves by the sums of their raw scores within the SCC. That is, if $S_1, \ldots, S_k$ comprise an SCC, then

$$\mathit{TieBreak}(S_i) = \sum_{j=1}^{k} \mathit{raw}(S_i,\, S_j)$$

This amounts to treating each mini-match among $S_1, \ldots, S_k$ as a single-point contest between two solvers in a round-robin event similar to teams in a league playing a season, so we call this the ***round-robin tie-break*** method. It is also known as Copeland's method in the voting-system literature [PBR00]. The advantage of this method is that it is easily understood and familiar. Its disadvantage is that the comparative ranking of S_1 and S_2 depends on mini-matches involving other solvers in the SCC.

Table 1. The dominance matrix for 16 solvers in the final phase, based on **careful ranking**.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	CircUs	0	0	0	0	0	0	0	1	1	1	1	1	0	0	1	0
2	LySAT_i	1	0	1	1	1	0	1	1	1	1	1	1	0	1	1	0
3	MXC	1	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0
4	ManySAT_1.1	1	0	1	0	0	0	1	1	1	1	1	1	0	1	1	0
5	MiniSAT_09z	1	0	1	1	0	0	1	1	1	1	1	1	0	1	1	0
6	MiniSat_2.1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	0
7	Rsat	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	0
8	SAT07_Rsat	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0
9	SAT07_picosat	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	SATzilla	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1	0
11	SApperloT	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
12	clasp	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
13	glucose	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	0
14	kw	1	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
15	minisat_cumr	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0	0
16	precosat	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

In practice, we expect SCCs to be about 2–4 solvers. Outcomes perceived as being “unfair” seem unlikely, because all the solvers involved are peers. In Example 2, the round-robin tie-break makes $S_1 > S_2 > S_3$. Notice that tweaking the times by ± 0.1 does not change the result, using this method. However, with the solution-count method, S_1 and S_2 are tied with the times as shown, but tweaking can make either one the winner.

6 Results on the SAT 2009 Competition

The final round of the Application section in the SAT 2009 Competition[5] was conducted with a time limit of 10000 seconds, used 292 benchmarks, and involved 16 solvers. The organizers were Daniel Le Berre, Laurent Simon, and Olivier Roussel. The discussion uses abbreviated solver names; please see the web page for complete names. The solvers were ranked for the competition using the **solution-count ranking** method described in Section 1. We computed the rankings that would have resulted using **careful ranking**. The dominance matrix discussed in Section 5 is shown in Table 1.

Examination of this matrix shows that solvers 1, 10, 14, and 15 are in one strongly connected component, so they share ranks 9–12, according to Section 5. All other solvers are not in any cycles, so have unique ranks.

[5] See `http://www.satcompetition.org/2009/`.

Table 2. Numbers of changes in top three ranks for two ranking methods and various time limits (seconds).

time range	solution-count	careful rank
1600–2000	8	4
2000–4000	10	0
4000–6000	4	0
6000–8000	0	0
8000–10000	1	0

6.1 Robustness of Ranking

We analyzed the robustness of the ranking methods by counting how many times there was some change in the top three ranks as the time limit was varied continuously. We note that `precosat` stayed in first place for all time limits 4000 seconds and above, in both ranking methods. Table 2 summarizes the numbers of changes in various ranges. (Returning to an earlier permutation is considered a change, too.) It seems clear that **careful ranking** is more robust by this criterion.

We offer this intuitive explanation for why **careful ranking** gives less variations as the time limit changes. With **solution-count ranking**, a mini-match victory is only temporary, as the time limit increases: S_1 wins the mini-match against S_2 only if S_1 succeeds and S_2 times out. But for a high enough time limit S_2 also succeeds (in theory), and the victory is wiped out. However, with **careful ranking**, once the time limit is sufficiently above the solving time of S_1 and S_2 still has not succeeded, the victory is permanent for this mini-match.

6.2 Differences in Ranking

The two ranking methods, **careful ranking** and **solution-count ranking**, disagreed on the third place solver with the final time limit of 10,000 seconds. `MiniSat_2.1` held third place behind `glucose` for all time limits above 2000 under **careful ranking**.

Under the **solution-count ranking**, `MiniSat_2.1` and `LySAT_i` exchanged places two and three several times, with `LySAT_i` finally taking the lead after about 8100. By the 10,000 mark `MiniSat_2.1` was in sixth place.

Under the **solution-count ranking**, `precosat` and `glucose` were apparently "neck and neck," as they each solved 204 instances. The tie-break was on cumulative CPU time, and `precosat` won. Other solvers were in the 190's well separated from the two leaders.

Quite a different picture emerges under **careful ranking**. We show three matches with their statistics. "Std. Devs." refers to the Student t from (1).

Winner	Loser	Raw Score	Std. Devs.	Prob. Faster
`precosat`	`glucose`	16	1.65	0.97
`glucose`	`MiniSat_2.1`	8	0.83	0.79
`MiniSat_2.1`	`LySAT_i`	8	0.86	0.80

In this ranking, `precosat` has a more convincing win than any of the others.

6.3 Tie-Break Illustration

Recall that Table 1 shows that solvers 1, 10, 14, and 15 are in one strongly connected component, sharing ranks 9–12, For purposes of illustration, we apply the round-robin tie-break procedure described in Section 5 to these four solvers, although in practice it is probably not important to break this tie.

The left side just below shows the raw scores for solvers involved. The right side shows the dominance subgraph.

	S_1	S_{10}	S_{14}	S_{15}
S_1	0	13	−9	1
S_{10}	−13	0	−8	3
S_{14}	9	8	0	−1
S_{15}	−1	−3	1	0

$$\begin{tikzcd}
S_1 \arrow[r] \arrow[dr] & S_{10} \arrow[d] \\
S_{14} \arrow[u] \arrow[ur] & S_{15} \arrow[l]
\end{tikzcd}$$

The round-robin ranking, based on the row-sums, gives $S_{14} > S_{10} > S_{15} > S_1$.

7 Conclusion

This paper described a new ranking system that provides a measure of statistical significance, allows for small timing differences to be treated as ties, and ensures that a pairwise comparison between two solvers is not influenced by a third solver. The latter property also allows later researchers to replicate the competition conditions and find out where their solver would have ranked. Another application of this technique is to evaluate whether software changes from one version to another caused a performance difference that is statistically significant, or whether the difference is in a range that might well just be random.

Acknowledgment We thank Daniel Le Berre, Laurent Simon, and Olivier Roussel for making the 2009 competition data available in text form.

References

[BLS05] F. Brglez, X. Y. Li, and M. F. Stallmann. On SAT instance classes and a method for reliable performance experiments with sat solvers. *Annals of Mathematics and Artificial Intelligence*, 43:1–34, 2005.

[BO07] F. Brglez and J. A. Osborne. Performance testing of combinatorial solvers with isomorph class instances. In *Workshop on Experimental Computer Science*, San Diego, 2007. ACM. (co-located with FCRC 2007).

[LBS04] D. Le Berre and L. Simon. The essentials of the sat 2003 competition. In *Proc. SAT*, 2004.

[LN95] J. Levin and B. Nalebuff. An introduction to vote-counting schemes. *The Journal of Economic Perspectives*, 9:3–26, 1995.

[Mou88] H. Moulin. Condorcet's principle implies the no-show paradox. *The Journal of Economic Theory*, 45:53–64, 1988.

[Nik10] M. Nikolić. Statistical methodology for comparison of SAT solvers. In *Proc. SAT*, pages 158–171, 2010.

[PBR00] J.-C. Pomerol and S. Barba-Romero. *Multicriterion Decision in Management: Principles and Practice*. Springer, 2000.

[Pul06] L. Pulina. Empirical evaluation of scoring methods. In *Third European Starting AI Researcher Symposium*, 2006.

[Sch03] M. Schulze. A new monotonic and clone-independent single-winner election method. In N. Tideman, editor, *Voting Matters, Issue 17*, pages 9–19. Oct. 2003. URL: `http://www.votingmatters.org.uk`.

[Tid06] N. Tideman. *Collective Decisions and Voting: the Potential for Public Choice*. Ashgate, 2006.