Local Restarts in SAT

Vadim Ryvchin rvadim@tx.technion.ac.il
Information Systems Engineering, IE, Technion, Haifa, Israel, and
Design Technology Solutions Group, Intel Corporation, Haifa, Israel.
Ofer Strichman ofers@ie.technion.ac.il
Information Systems Engineering, IE, Technion, Haifa, Israel.

Editor: Youssef Hamadi, Eric Monfroy and Frederic Saubion

Abstract

Most or even all competitive DPLL-based SAT solvers have a "restart" policy, by which the solver is forced to backtrack to decision level 0 according to some criterion. Although not a sophisticated technique, there is mounting evidence that this technique has crucial impact on performance. The common explanation is that restarts help the solver avoid spending too much time in branches in which there is neither an easy-to-find satisfying assignment nor opportunities for fast learning of strong clauses. All existing techniques rely on a global criterion such as the number of conflicts learned as of the previous restart, and differ in the method of calculating the threshold after which the solver is forced to restart. This approach disregards, in some sense, the original motivation of avoiding 'bad' branches. It is possible that a restart is activated right after going into a good branch, or that it spends all of its time in a single bad branch. We suggest instead to *localize* restarts, i.e., apply restarts according to measures local to each branch. This adds a dimension to the restart policy, namely the decision level in which the solver is currently in. Our experiments with both Minisat 2007 and Eureka show that with certain parameters this improves the run time by 15% - 30% on average (when applied to the 100 test benchmarks of SAT-race'06), and reduces the number of time-outs. We begin the paper by considering various possible explanations for the effectiveness of restarts.

Keywords: SAT solving, Restarts.

1. Introduction: Why Do Restarts Work ?

Most or even all competitive DPLL SAT solvers have a "restart" policy, a strategy initially proposed by Gomes et al. (1998). Most modern DPLL solvers restart after a certain number of conflict clauses have been learned. The search changes from one restart to the next either due to randomness in the search algorithm, due to the change in the input formula as a result of learning new clauses, or both.

The argument given by Gomes et al. (1998) for the success of restarts was in fact only relevant for solvers with a certain level of randomness. They observed empirically that the distribution of running time, when solving the same formula many times with a solver that has a certain element of randomness (assuming each run is initiated with a different seed), is 'heavy-tailed'. One may interpret this phenomenon as follows. Every random seed defines a deterministic solver. The heavy tail phenomenon means that at any given time along the time axes, there are still some deterministic solvers that require exponential more time to complete the search.[1] Restarting the solver with a different seed, then, can be thought of as attempting to find a fast deterministic solver rather than getting stuck with a slow one. This clearly reduces the variance in solving time, or, in the words of Gomes et al. (1998), it makes the solver more robust.

Many of the modern solvers, however, do not support randomness, but still use restarts. Consecutive runs are different simply because, as mentioned above, learned clauses are retained, and hence the input problem changes after each restart. The effectiveness of restarts in this context can probably be explained in a similar manner, although we are not aware of an empirical study that supports this view. It is very likely that a similar heavy-tail phenomenon exists. In other words, starting from a different input formula – albeit a logically equivalent one – can have a drastic influence on the solving process (because the decision heuristic is influenced by the input formula), similar to the effect of randomizing the search.

An additional reason for the success of restarts emerges from statistics we collected regarding the size of learned clauses at each level. Our data shows that conflict clauses at the lower decision levels (typically up to decision level 20 or 30) are smaller on average.[2] The graphs in Fig. 1 show this phenomenon quite clearly. Each of these graphs is taken from a different benchmark family. It shows the average size of learned conflict clauses per decision level, when ran with the default Minisat 2007 (Een and Sorensson (2006)) configuration except that the restarts mechanism is turned off. A similar pattern can be seen in almost all industrial examples in the benchmark set of the SAT'06 competition.

One may speculate that one of the effects of restarts is lowering the average size of learned clauses, since it forces the solver to lower decision levels. Our statistics show that indeed with successful restarts strategies the average conflict clause size is smaller. Yet it is hard to establish causality in this case: was it the fact that the average decision level in which clauses are learned became smaller, or is there some other intermediate factor that causes a reduction in both the average size of learned clauses and the average decision level in which these clauses are learned? while we cannot establish causality, we can still check how restarts affect these two figures.

We recorded the number of clauses that are learned in the first 20 decision levels, and the first 100 decision levels (Minisat solves these four formulas with maximal decision level lower than 100), both with and without restarts (specifically, we ran it with the best restart strategy that is later reported in Fig. 3). The table in Fig. 2 summarizes these results. It shows that activating restarts more than doubles the ratio of the clauses that are learned in earlier decision levels. It is possible, then, that the decision level in which clauses are learned is a mediator between restarts and the size of the learned clauses. In other words, restarts lead to more decisions in lower decision levels, which are smaller on average, and these, in turn, lead to more effective learning and shorter run times.

We continue in the next section by surveying competitive restart strategies and presenting a technique that improves all of them. Most of these results already appeared in our

1. We use here the terminology used in the original paper by Gomes et al. (1998). One may argue that this is only an approximation to the real definition of heavy tail in statistics, because here for every reasonable search algorithm one may think of a bound on the worst running time.
2. As a side note let us also mention that they tend to grow on average as time passes, but this is not related to the argument here.

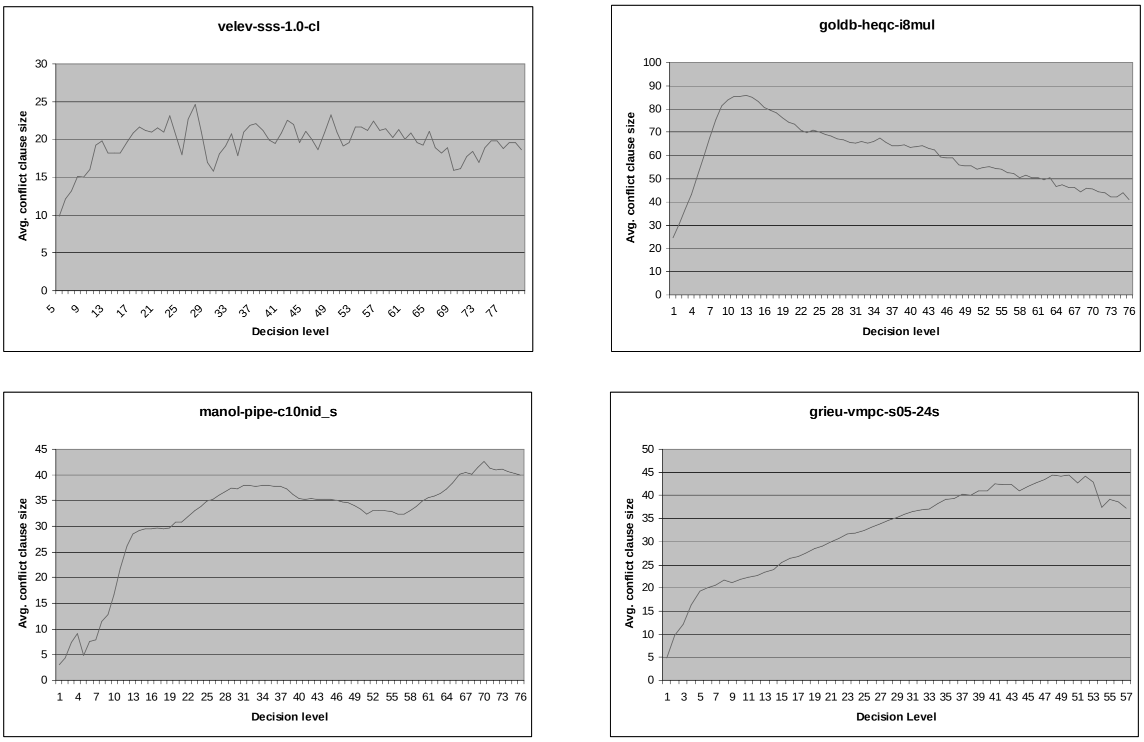

Figure 1: Each of the charts displays the average size of conflict clauses learned at each decision level. These figures were recorded while solving each of the four CNF formulas (taken from different benchmark families) with Minisat, without restarting. The charts show clearly that there is an increase in the average size of learned conflict clauses in the first few dozen decision levels.

	No restarts			Restarts		
Instance	DL 1–20	DL 1–100	Ratio	DL 1–20	DL 1–100	Ratio
velev-sss-1.0-cl	2047	6182	0.331	3123	4691	0.66
goldb-heqc-i8mul	456466	1022810	0.446	252952	425603	0.594
manol-pipe-c10nid_s	9208	1193490	0.0077	1693	26173	0.064
grieu-vmpc-s05-24s	16474	139518	0.118	125999	247307	0.509
Average:			0.225			0.477

Figure 2: Restarts tend to increase the ratio of clauses that are learned in low decision levels, as demonstrated by the statistics associated with the above four instances (it more than doubles the ratio of clauses learned in the first 20 decision levels). As demonstrated in Fig. 1, in these low decision levels the average size of learned clauses is typically smaller than the average.

SAT'08 short paper (Ryvchin and Strichman (2008)). In this extended version we added the above hypothesis regarding the reason for the success of restarts, and a more thorough evaluation of one of our suggested strategies (the 'dynamic restart' strategy, which is described in the next section).

2. Global vs. Local Restarts

Different restart policies are used by different solvers. A recent survey by Huang (2007) includes several types of restart policies. We briefly describe various types of popular restart techniques based on that survey and on some new developments.

1. *Arithmetic (or fixed) series.* Parameters: x, y. A policy in which there is a restart every x conflicts, which is increased by y every restart. Some sample values are: in zchaff 2004 $x = 700$, in Berkmin $x = 550$, in Siege $x = 16000$ and in Eureka $x = 2000$. In all of these solvers the series is in fact fixed (i.e., $y = 0$), owing to the observation that completeness is meaningless in the realm of timeouts.

2. *Geometric series.* Parameters: x, y. A policy in which, initially, there is a restart every x conflicts. The value of x is then multiplied by a factor of y in each restart, for some $y > 1$. This policy is used in Minisat 2007 with $x = 100$ and $y = 1.5$.

3. *Inner-Outer Geometric series.* Parameters: x, y, z. An idea suggested by Biere and implemented in PicoSAT (Biere (2008)), by which restarts follow what can be seen as a two dimensional pattern that increases geometrically in both dimensions. The inner loop multiplies a number initialized to x by z at each restart. When this number is larger than a threshold y, it is reset back to x and the threshold y is also multiplied by z (this is the outer loop). Hence, both the inner and outer loops follow a geometric series, and the whole series creates an oscillating pattern.

4. *Luby et al. series* (Luby et al. (1993)). Parameter: x. A policy by which restarts are performed according to the following series of numbers: 1, 1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, 1,... multiplied by the constant x (called the *unit-run*). Formally, let t_i denote the i-th number in this series. Then t_i is defined recursively:

$$t_i = \begin{cases} 2^{k-1} & \text{if } \exists k \in \mathbb{N}.\ i = 2^k - 1 \\ t_{i-2^{k-1}+1} & \text{if } \exists k \in \mathbb{N}.\ 2^{k-1} \leq i < 2^k - 1 \end{cases}$$

   This is a well-defined series, as the two conditions are mutually-exclusive. This policy has some nice theoretical characteristics in a class of randomized algorithms called Las Vegas algorithms[3], but the relevance of these results to DPLL has only been empirical so far – it is not clear what is the reason that it works well in practice. The experiments reported by Huang (2007) show that it outperforms the other restart strategies, and indeed this is now the restart method of choice of several state-of-the-art solvers, such as TinySAT (Huang (2007)) and RSAT (Pipatsrisawat and Darwiche (2007)).

3. Algorithms that use randomness, but the quality of the result is not affected by it. Typically randomness in such algorithms only affects run-times.

For completeness of this list, we should also mention that there is a family of techniques in which 'restart' does not entail backtracking to level 0, but rather to some decision level which is lower than what is computed as the backtracking level by a conflict analysis procedure. Such a procedure was proposed, for example, by Lynce et al. (2001). We did not experiment with these techniques, however.

All of the strategies listed above are based on a global counter of conflict clauses, and therefore they measure progress over many branches together. Assuming that the motivation for restarts is to prevent the solver from getting stuck in a bad branch (which can, informally, be defined as a branch which neither contains an easy-to-find satisfying assignment nor leads to efficient learning that directs the solver to a different search-space or to a proof of unsatisfiability), such a global policy may miss the point.

For example, it is possible that the solver spent a significant amount of time searching in a branch, eventually left it, and very soon after that it restarts (since the global threshold was reached), although there is no knowledge yet about the potential of the current branch. It is also possible that the restart is too late, for example if it spends all its time between restarts in a single bad branch.

A possibly better strategy is to localize the measure of difficulty of branches, and restart when the branch is more difficult than some threshold. Each of the global strategies mentioned above can be applied locally, because we can count the number of conflicts under each branch easily, as follows. For each decision level d we maintain a counter $c(d)$, which is initially (when a decision is made at that level) set to the global number of conflicts. When backtracking back to that level, we examine the difference between the current global number of conflicts, and $c(d)$. This difference reflects the number of conflicts that were encountered above level d, since the last time a decision was made at this level. If this difference is larger than some strategy-dependent threshold, we restart.

Dynamic strategies

Locality opens a new dimension, namely that of the decision level. In other words, the threshold can be a function of the level in which the solver is currently in. We call such strategies *dynamic*. It can be expected that the work done between two visits to a decision level (from decision to backtracking back to that level) will be smaller as the decision level increases. The data we introduced in the introduction also supports this direction: it suggests that decreasing the threshold for restarts when the decision level goes up can force the solver to learn stronger facts first.

Each of the strategies above can be made dynamic. Such dynamic strategies, then, are two-dimensional, as they compose a base (static) strategy with a strategy of adjusting it according to the decision level. The dynamic element of the composed strategy is defined by two parameters: d and min, where d is the number by which the threshold is decreased at each decision level (here we assume a linear adjustment), and min is the minimal threshold for performing restart. The restart threshold is given by

$$\max(x - i \cdot d, min) \ , \tag{1}$$

where x represents the threshold as computed by the base strategy and i is the decision level.

We focused on two base strategies — the first is the fixed series strategy and the second is the Inner-Outer strategy. Hence, we have:

5. Dynamic-fix. Parameters: parameters of the fixed series strategy + d, min. A policy by which at decision level i there is a restart every $\max(x - i \cdot d, min)$ conflicts, where x is calculated via the fixed series strategy.

6. Dynamic-IO. Parameters: parameters of the Inner Outer strategy + d, min. A policy by which at decision level i there is a restart every $\max(x - i \cdot d, min)$ conflicts, where x is calculated via the IO strategy.

3. Experimental Results

Making the strategy local instead of global requires re-tuning of the parameters – there is no reason to believe that parameters that optimize a global restart policy also optimize a local one. Hence a major empirical evaluation is needed in order to check the effect of locality on each of these strategies. This is the subject of the current section.

The table in Fig. 3 shows results with 40 different restart configurations, when implemented on top of Minisat 2007 (Een and Sorensson (2006)), and ran on the 100 industrial benchmarks that were used as preparation for SAT-race'06 (divided evenly to the two test-sets TS1 and TS2). A similar table for the latest version of Eureka (Nadel et al. (2006)), with 41 configurations, appears in Fig. 4. The set of configurations is not identical, but close, because we chose them dynamically: when a good strategy was found, we tried to change it incrementally. The tables are sorted according to the type of strategy, local/global, and parameters. The third column indicates whether this strategy is implemented globally or locally. Timeout was set to 30 minutes. Instances that timed-out are included and contribute 30 minutes (we added them to the SAT or UNSAT column according to our prior knowledge of the expected result). Instances that none of our configurations nor any SAT'06-race competitor can solve are not included. The overall number of timeouts and total run time are given in the last two columns, where time is measured in hours. All together the two tables represent over 40 days of CPU time.

The first column indicates the position of each solver when measured by the total run time, and the best three configurations according to this measure are preceded by '✓'. With both solvers, the best three configurations that we tried are local (also when measured by time-outs).

To the extent that the benchmark set is representative of industrial problems, and that Minisat 2007 and Eureka represent state-of-the-art solvers, it seems that locality can help with the four types of strategies that we tried. The following table shows, for the Luby and Inner-Outer strategies, the figures corresponding to the best local and best global configurations that we could find.

	Minisat				Eureka			
Strategy	Global		Local		Global		Local	
	TO	Time	TO	Time	TO	Time	TO	Time
Luby	11	8.98	9	7.89	9	8.90	8	8.40
IO	10	8.86	8	7.38	9	8.64	8	8.12

There seems to be such an advantage for the local geometric and local arithmetic strategies as well, but more global configurations of these strategies need to be tested in order to draw concrete conclusions. If we take the default parameters of Minisat and Eureka as best of their respective global strategies, then this can be said with some confidence.

What about the dynamic strategy? *dynamic-fix* does not seem to score well in general, at least not with the 4 parameters set that we tried, but it performs well with unsatisfiable instances. In the case of the first table (Minisat), the dynamic strategies with parameters 1000,0.1,20,10 and 1000,0.1,10,10 arrive at the second and third places, respectively, if we measure only unsatisfiable instances.

Fig. 5 shows our experiments with *dynamic-IO*, based on the best configuration of the Inner-Outer strategy as shown in Fig. 3 (i.e., with parameters 100,1000,1.1). The results show that none of our attempts to make this particular IO strategy dynamic was successful. Whether there exists parameters that make the dynamic approach worth while is a question left for further investigation.

Summary

Localizing the criterion of competitive restart strategies reduces the solving time of industrial benchmarks. The technique suggested here, of measuring the hardness of a branch by the number of conflicts learned under it is easy to compute and effective. As future research we suggest to look for additional easy-to-compute measures for the quality of the branch (e.g., it is possible that measures such as the size of the backtrack can be factored effectively in the restart policy). Additional observations regarding the effect of restarts, as was discussed in the introduction, can be helpful for optimizing this technique further.

References

Armin Biere. PicoSAT essentials. *JSAT*, 2008. (to be published).

Niklas Een and Niklas Sorensson. Minisat v2.0 (beta). In *Solvers description, SAT-race*. 2006. `http://fmv.jku.at/sat-race-2006/descriptions/27-minisat2.pdf`.

Carla P. Gomes, Bart Selman, and Henry A. Kautz. Boosting combinatorial search through randomization. In *AAAI/IAAI*, pages 431–437, 1998.

Jinbo Huang. The effect of restarts on the efficiency of clause learning. In *IJCAI*, pages 2318–2323, 2007.

Michael Luby, Alistair Sinclair, and David Zuckerman. Optimal speedup of Las Vegas algorithms. In *ISTCS*, pages 128–133, 1993.

Lynce, Luis Baptista, and Joao P. Marques Silva. Stochastic systematic search algorithms for satisfiability. In *LICS Workshop on Theory and Applications of Satisfiability Testing*, pages 190–204, 2001.

Alexander Nadel, Moan Gordon, Amit Palti, and Ziyad Hana. Eureka-2006 SAT solver. In *Solvers description, SAT-race*. 2006.

		G/		TS1				TS2				Overall	
Place	Strategy	L	Parameters	SAT	UNSAT	TO	Total	SAT	UNSAT	TO	Total	TO	Time
✓3	Arith	L	100,10	1.12	2.06	4	3.18	2.17	2.59	6	4.75	10	7.93
26	Arith	L	10,1	2.12	2.62	6	4.74	2.42	2.99	6	5.41	12	10.15
8	Arith	L	100,1	1.89	1.96	4	3.85	2.37	2.84	6	5.21	10	9.05
6	Arith	L	100,20	2.49	1.99	6	4.48	2.32	2.21	5	4.53	11	9.02
12	Arith	L	100,40	2.51	1.95	6	4.47	2.11	2.74	6	4.86	12	9.33
10	Arith	L	1000,0.1	2.3	2.05	4	4.35	1.89	2.85	6	4.74	10	9.09
9	Arith	L	1000,1	2.15	1.93	5	4.08	2.07	2.9	6	4.97	11	9.05
32	Arith	L	1000,10	2.76	2.13	7	4.89	2.72	2.99	8	5.71	15	10.6
34	Arith	L	1000,20	3.13	2.07	8	5.2	2.61	2.93	5	5.54	13	10.74
21	Arith	L	2500,1	2.11	2.38	6	4.49	2.37	3.03	7	5.39	13	9.89
24	Arith	L	3,1	2.47	1.87	3	4.34	2.88	2.81	9	5.69	12	10.03
29	Arith	L	3,10	2.69	1.92	6	4.61	2.95	2.92	9	5.87	15	10.48
14	Arith	L	5,0.2	2.41	1.62	6	4.04	2.59	2.85	8	5.43	14	9.47
15	Arith	L	5000,1	2.33	2.48	7	4.81	2.13	2.56	4	4.69	11	9.5
18	Arith	L	6,1	2.02	2.23	5	4.25	2.61	2.86	8	5.46	13	9.71
27	Geom.	L	10,1.1	2.53	2.03	6	4.56	2.5	3.18	8	5.68	14	10.24
37	Geom.	L	10,1.5	2.46	2.63	7	5.08	2.62	3.29	6	5.91	13	10.99
40	Geom.	L	10,2	2.89	2.77	9	5.65	3.03	3.39	9	6.42	18	12.07
16	Geom.	L	100,1.1	1.71	2.16	3	3.86	2.55	3.14	8	5.69	11	9.56
38	Geom.	L	100,1.5	3.33	2.71	9	6.03	2.94	2.77	6	5.71	15	11.75
36	Geom.	L	100,2	2.33	2.86	7	5.19	2.42	3.35	7	5.76	14	10.95
33	Geom. *	G	100,1.5	1.6	2.76	6	4.36	3.06	3.22	8	6.28	14	10.64
11	IO	G	100,1000,1.1	2.68	2.07	6	4.75	1.72	2.86	7	4.57	13	9.32
4	IO	G	100,1000,1.5	1.81	2.04	4	3.86	2.04	2.97	6	5	10	8.86
39	IO	G	100,1000,2	2.81	2.16	8	4.97	3.33	3.48	10	6.81	18	11.78
✓1	IO	L	100,1000,1.1	1.59	2	4	3.59	1.27	2.51	4	3.78	8	7.38
7	IO	L	100,1000,1.5	2.22	2.02	5	4.24	1.92	2.88	6	4.8	11	9.04
30	IO	L	100,1000,2	2.89	2.22	8	5.11	2.6	2.79	7	5.39	15	10.5
22	Luby	G	32	2.22	1.49	3	3.71	3.06	3.15	10	6.21	13	9.91
23	Luby	G	128	3.08	1.76	6	4.84	2.21	2.89	7	5.1	13	9.94
13	Luby	G	512	2.84	1.93	7	4.77	1.92	2.64	5	4.56	12	9.33
5	Luby	G	1024	2.26	1.97	5	4.22	2.02	2.74	6	4.76	11	8.98
✓2	Luby	L	32	1.6	1.15	3	2.75	2.22	2.92	6	5.14	9	7.89
25	Luby	L	128	2.75	2.01	7	4.76	2.29	3.02	7	5.32	14	10.08
17	Luby	L	512	2.18	2.08	5	4.26	2.33	3.1	6	5.43	11	9.69
19	Luby	L	1024	2.71	2.02	4	4.73	1.94	3.05	7	5	11	9.73
28	D-arith	L	1000,0.1,10,10	3.45	1.02	6	4.47	2.7	3.13	8	5.84	14	10.31
20	D-arith	L	1000,0.1,20,10	2.92	0.99	4	3.91	2.77	3.1	8	5.87	12	9.78
31	D-arith	L	1000,10,10,10	3.5	2	8	5.51	1.64	3.41	7	5.05	15	10.56
35	D-arith	L	1000,10,20,10	3.22	2.02	8	5.24	2.25	3.4	8	5.65	16	10.89

Figure 3: Results, in hours, based on Minisat 2007. The original configuration of Minisat 2007 is marked with *.

Place	Strategy	G/ L	Parameters	TS1 SAT	TS1 UNSAT	TS1 TO	TS1 Total	TS2 SAT	TS2 UNSAT	TS2 TO	TS2 Total	Overall TO	Overall Time
39	Arith	L	10,0.1	2.34	1.26	4	3.6	2.78	4.22	11	7	15	10.59
38	Arith	L	10,1	1.92	1.67	4	3.59	2.93	4.06	10	6.98	14	10.58
41	Arith	L	100,1	2.19	1.63	3	3.81	3.24	4.04	10	7.28	13	11.09
17	Arith	L	100,10	1.78	1.11	2	2.89	2.8	3.44	7	6.24	9	9.13
✓2	Arith	L	1000,1	1.6	1.04	2	2.64	2.74	2.72	6	5.46	8	8.09
5	Arith	L	1000,10	1.63	0.96	2	2.59	3.05	2.68	5	5.72	7	8.31
✓1	Arith	L	1000,20	1.83	0.92	2	2.75	2.57	2.67	5	5.24	7	7.98
40	Arith	L	20,0.1	2.47	1.35	4	3.82	2.65	4.23	11	6.87	15	10.69
31	Arith	L	20,1	2.4	1.32	3	3.72	2.63	3.69	9	6.32	12	10.04
14	Arith	L	2000,1	1.76	1.1	2	2.86	3.4	2.81	6	6.21	8	9.08
32	Arith	L	3,1	2.04	1.19	3	3.23	3.4	3.43	9	6.83	12	10.06
8	Arith	L	3,10	1.63	1	2	2.63	2.66	3.24	6	5.89	8	8.52
4	Arith	L	3,20	1.7	0.9	2	2.6	2.47	3.21	7	5.68	9	8.28
21	Arith	L	3,40	1.79	0.92	2	2.71	3.54	3.39	8	6.93	10	9.64
37	Arith	L	5,0.2	2.29	1.23	3	3.53	3.17	3.85	10	7.02	13	10.55
18	Arith	L	5000,1	1.71	1.08	2	2.79	3.01	3.44	7	6.45	9	9.24
19	Arith*	G	2000,0	2.15	1.07	3	3.22	3.17	3	6	6.17	9	9.39
29	Geom.	L	10,1.1	2.2	1.07	3	3.26	3.27	3.49	9	6.76	12	10.03
36	Geom.	L	10,1.5	1.89	1.1	2	2.99	3.17	4.23	10	7.4	12	10.39
25	Geom.	L	10,2	1.96	1.32	2	3.28	3.14	3.38	9	6.52	11	9.80
11	Geom.	L	100,1.1	1.98	0.9	2	2.88	2.8	3.1	7	5.9	9	8.78
28	Geom.	L	100,1.5	1.73	0.95	2	2.68	3.46	3.78	9	7.24	11	9.93
30	Geom.	L	100,2	2.11	1.01	2	3.12	3.16	3.75	7	6.91	9	10.04
10	IO	G	100,1000,1.1	1.54	0.93	2	2.47	3.05	3.12	7	6.17	9	8.64
15	IO	G	100,1000,1.5	1.59	0.9	1	2.49	3.01	3.57	8	6.58	9	9.08
26	IO	G	100,1000,2	2.12	0.87	3	2.99	3.34	3.48	8	6.83	11	9.82
✓3	IO	L	100,1000,1.1	1.72	0.88	2	2.6	2.82	2.7	6	5.52	8	8.12
22	IO	L	100,1000,1.5	2.19	0.86	3	3.05	3.14	3.55	8	6.68	11	9.73
34	IO	L	100,1000,2	2.34	1.1	3	3.44	3.13	3.76	8	6.88	11	10.32
16	Luby	G	32	1.83	1.03	3	2.86	2.97	3.29	7	6.26	10	9.12
12	Luby	G	128	2.17	0.87	2	3.05	2.92	2.94	7	5.86	9	8.90
13	Luby	G	512	1.59	1	2	2.59	3.18	3.27	7	6.46	9	9.05
23	Luby	G	1024	2.22	1.09	3	3.31	3.58	2.88	6	6.46	9	9.76
9	Luby	L	32	1.67	0.94	1	2.61	2.75	3.17	7	5.92	8	8.53
7	Luby	L	128	1.71	0.91	1	2.62	2.84	2.96	6	5.79	7	8.41
6	Luby	L	512	1.6	0.94	2	2.54	3.14	2.72	6	5.86	8	8.40
27	Luby	L	1024	2.33	1.1	3	3.43	3.6	2.87	7	6.47	10	9.90
24	D-arith	L	1000,0.1,10,10	1.91	1.34	3	3.25	3.26	3.27	8	6.53	11	9.77
35	D-arith	L	1000,0.1,20,10	1.86	1.71	4	3.57	3.15	3.66	9	6.81	13	10.38
20	D-arith	L	1000,10,10,10	1.88	1.2	2	3.08	3.25	3.28	5	6.53	7	9.61
33	D-arith	L	1000,10,20,10	1.82	1.31	2	3.13	3.25	3.74	8	6.98	10	10.11

Figure 4: Results, in hours, based on Eureka. The original configuration of Eureka is marked with *.

	TS1				TS2				Overall	
Parameters	SAT	UNSAT	TO	Total	SAT	UNSAT	TO	Total	TO	Time
-1,100	2.59	1.93	7	4.52	1.88	2.88	6	4.76	13	9.28
0.05,10	2.49	1.99	6	4.49	1.88	2.91	6	4.79	12	9.28
0.05,20	2.46	1.75	5	4.22	2.29	2.91	7	5.19	12	9.41
0.05,50	2.05	1.99	5	4.04	2.30	2.91	7	5.21	12	9.25
0.05,100	2.42	2.08	6	4.50	2.67	2.87	6	5.53	12	10.03
0.1,10	3.36	1.97	6	5.33	2.83	2.95	8	5.78	14	11.11
0.1,20	2.04	1.97	5	4.01	2.41	2.95	7	5.36	12	9.37
0.1,50	2.88	1.95	6	4.83	2.56	2.94	7	5.50	13	10.33
0.1,100	2.22	1.97	7	4.19	1.89	2.89	8	4.78	15	8.97
0.5,10	1.83	1.95	5	3.78	1.83	2.50	6	4.33	11	8.11
0.5,20	1.42	1.79	3	3.20	2.38	2.89	6	5.27	9	8.47
0.5,50	1.98	2.07	5	4.05	1.91	2.85	7	4.76	12	8.81
0.5,100	1.93	2.07	5	4.00	2.48	2.82	5	5.29	10	9.30
1,10	2.05	1.95	4	3.99	2.38	2.95	8	5.33	12	9.32
1,20	2.36	2.23	7	4.59	1.59	2.57	5	4.16	12	8.75
1,50	1.85	1.97	6	3.83	2.29	2.86	7	5.15	13	8.97
1,100	2.17	1.94	5	4.11	1.75	3.00	7	4.76	12	8.87
5,10	3.07	1.96	5	5.03	2.66	2.86	7	5.52	12	10.55
5,20	2.48	1.93	8	4.41	2.29	2.65	6	4.95	14	9.36
5,50	2.16	2.05	5	4.22	1.74	2.91	6	4.65	11	8.87
5,100	2.17	2.08	8	4.26	2.39	2.92	7	5.30	15	9.56
10,10	2.97	1.95	5	4.91	2.19	3.21	6	5.40	11	10.32
10,20	3.50	1.86	6	5.37	1.89	2.96	6	4.85	12	10.22
10,50	2.54	1.69	5	4.24	2.13	2.58	7	4.71	12	8.95
10,100	2.29	1.99	6	4.28	1.74	2.88	7	4.62	13	8.90
*	1.59	2.01	4	3.60	1.28	2.52	4	3.80	8	7.40

Figure 5: Results, in hours, of various dynamic strategies combined with the best local restart strategy of Minisat (Inner-Outer with parameters 100,1000,1.1). These results show that the dynamic strategy is not helpful with the 25 configurations that we tried. The two parameters are d and min. The last line is the base strategy, i.e., $d = 0$.

Knot Pipatsrisawat and Adnan Darwiche. Rsat 2.0: Sat solver description. SAT competition'07, 2007.

Vadim Ryvchin and Ofer Strichman. Local restarts. In *SAT*, pages 271–276, 2008.