SATbed: An Environment for Reliable Performance Experiments with SAT Instance Classes and Algorithms

Franc Brglez   Matthias F. Stallmann   Xiao Yu Li

Dept. of Computer Science, NC State University
Raleigh, North Carolina, USA
http://www.cbl.ncsu.edu/OpenExperiments/SAT/

Outline

• SATbed Context and Motivation
• SATbed Architecture and User Inputs
• SATbed Output Data Examples

See the paper for more details on generation of sat/unsat scheduling problems, the sched family, that raise some unexpected and yet-to-be explained challenges for the current generation of SAT solvers.
Thanks!

Chaff team, for distribution of chaff

Satire team, for distribution of satire

Sato team, for distribution of sato and satoL

SatLIB team, for distribution of satlib

Reviewers for constructive suggestions that helped shape this research.

SATbed Context and Motivation --

• linking SAT, reliability and statistics
• survival function and lifetime of hardware components (evaluated on a class of replicated components)
• solvability function and runtime of SAT solvers (evaluated on equivalence classes of SAT instances)
• why heavy-tail distribution in SAT
• how many samples are enough (what size the equivalence class)

SAT’2002: The Role of a Skeptic Agent in Testing and Benchmarking of SAT Algorithms

Revision with AMAI: On SAT Instance Classes and a Method for Reliable Performance Experiments with SAT Solvers
A Quote from SAT2003

“We can develop SAT solvers only so far …”
Toby Walsh, Challenges in SAT (and QBF)

The premise of this talk:

Far is a relative term. By improving the reliability of experimental procedures in use today, we can and should be able to develop SAT solvers not only to go further, but also at a rate is accelerating …

Lead-In: Last Slide from SAT2002

(1) how much solver improvement is due to design
   how much solver improvement is due to chance?

(2) ask not what solver can do for a single instance,
   ask what solver can do for a class of instances!

In SATbed (SAT2003), a class is NOT
   • a collection of unrelated cnf instances
   • a collection of same-size (random) cnf instances

In SATbed, a class is derived from a single reference:
   • preserving the syntactic structure of the reference
   • preserving the solution landscape of the reference
Four ways to re-write a cnf formula

<table>
<thead>
<tr>
<th>Class</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ref</td>
<td>{ 1 2 3} { 4 5 6} {-1 2 -3 4}</td>
</tr>
<tr>
<td>I-class</td>
<td>{ 3 2 1} { 2 -1 4 -3} { 6 4 5}</td>
</tr>
<tr>
<td>P-class</td>
<td>{ 4 6 2} { 5 1 3} {-4 6 -2 5}</td>
</tr>
<tr>
<td>C-class</td>
<td>{-1 2 3} { 4 -5 6} { 1 2 -3 4}</td>
</tr>
<tr>
<td>PC-class</td>
<td>{-4 6 2} { 5 -1 3} { 4 6 -2 5}</td>
</tr>
</tbody>
</table>

On SAT, Reliability, and Statistics

SAT solver runtime distribution may be exponential...
... but may not be observed reliably unless

1. experiments are repeated on the same instance (with a different starting point)
2. experiments are repeated on the class of instances with the same “solvability” (e.g., PC-class)

Component lifetime distribution (e.g., a fuse) is exponential;...
MTTF is determined by an experiment based on

A: testing N replicated components from a single supplier
and NOT B: testing N components from N suppliers

Would YOU fly an airplane whose components are tested using method B??
Reliability of Boeing 720 A/C Units (1)

Field study, 213 A/C units (Prochan, Technometrics, 1963)

Observations recorded:
  time-to-failure for each of 213 (identical!) units

MTTF = 93.3 hours
Std = 107 hours

... close fit to exponential distribution!

Reliability of Boeing 720 A/C Units (2)

Field study, 213 A/C units (Prochan, Technometrics, 1963)

Observations recorded:
  time-to-failure for each of 213 (identical!) units

MTTF = 93.3 hours
Std = 107 hours

... close fit to exponential distribution!
Reliability of Other Hardware Components

Projector lightbulbs: MTTF 500 hours
Mission-critical fuses: MTTF 500,000 hours
Computer hard disks: MTTF n/a

Again, ... close fit to exponential distribution!

Unsolvability of SAT Solvers (normalized)

sato on queen16_PC (32 instances): MTTS 0.07 seconds
unitwalk on sched07s (128 instances): MTTS 0.40 seconds
chaff on sched07s (128 instances): MTTS 13.4 seconds

Again, ... close fit to exponential distribution!

... same solvers, other problem classes, different distributions
Solvability of SAT Solvers (denormalized)

unitwalk on sched07s (128 instances): MTTS 0.40 seconds
chaff on sched07s (128 instances): MTTS 13.4 seconds

... same solvers, other problem classes, different distrib.

Again, ... chi2-test confirms a fit to exponential distribution!

Why a Heavy-Tail Distribution in SAT?

These 100 instances are "too different"!
(uf250-1065_R)

Choose:
- four instances => four PC classes of size 128 each

1) each of four PC classes has exponential d.
2) means differ significantly
How Many Samples Are Enough (1)?

howManySamplesAreEnough.png

solverID = unitwalk,
classID = uf250-1065_027_PC (128 instances)

How stable are consecutive means (and confidence int.)

How Many Samples Are Enough (2)?

howManySamplesAreEnough2.png

solverID = unitwalk,
classID = uf250-1065_027_PC (128 instances)

How stable are cumulative means (and confidence int.)
SATbed Architecture and User Inputs --

Six stages of SATbed architecture to encapsulate, chain, and execute a flow of:
- sat/unsat instance generators
- equivalence class generators
- sat solvers
- sat postprocessors (including verification, censored data, and statistics report generation)
- special-purpose statistics report generators
- special-purpose viewing tools and utilities

Two example configuration files (few lines) to invoke hundreds of experiments and organize benchmark and results data sets into ready-to-traverse hierarchies.

The SATbed Architecture

Six stages execute stand-alone or in a full/partial chain:

REFGEN → CLASSGEN → SOLVER → POST_PROCESS → STATISTICS → VIEWER

Each stage encapsulates any number of standard and user-supplied programs.
A single configuration file defines each experiment.

Each stage interfaces to:
- standard benchmark and class instance files
- user-supplied instance files
- internally generated, solver-specific data files
The SATbed Architecture (stages 1-4)

REFGEN ➔ CLASSGEN ➔ SOLVER ➔ POST_PROCESS

REFGEN encapsulates cnf instance generators
CLASSGEN encapsulates class generators (given instance)
SOLVER encapsulates SAT solvers applied to class instances
POST_PROCESS encapsulates
  solver-specific rawData-to-standardTable parsers
  generic solver verifier (given instance and solution)
  generic report generator of web-browsable statistics

The SATbed Architecture (stages 5-6)

... following the POST_PROCESS stage:

 ➔ STATISTICS ➔ VIEWER

STATISTICS aggregates results data across solvers and
encapsulates specialized statistics packages
  • Smirnov-Kolmogorov test for 2 distributions
  • t-test for 2 distributions
  • etc.

VIEWER encapsulates specialized programs that generate
  • additional ready-to-view tables
    for plotting packages
  • statistics summary tables (in LaTex)
  • etc.
demo1: REFGEN to POST_PROCESS

START = REFGEN
STOP = POST_PROCESS
BENCHMARK_DIR = benchmarks/demos
RESULTS_DIR = results/demos
REFGEN = sched_small_s
sched_small_s = sched_classic.tcl 0 3 yes
CLASSGEN = I,PC
I = rotate_formula I
PC = rotate_formula PC
CLASS_SIZE = 32
SEED = 33,1546,1968
SOLVER = sato
sato = sato_encap.tcl
TIME_OUT = 3600
SOLVE_CLASSES = sched_small_s
POST_PROCESS = sato_pp
sato_pp = sato_postProcess.tcl

demo1: SATbed Invocation

Starting sched_classic.tcl
..size = 0: 12 vars, 26 clauses
..size = 1: 31 vars, 98 clauses
Executing stage CLASSGEN
with I PC

../bin/@SATbed demo-1.cfg

... starting sato_encap.tcl
completing sato_encap.tcl
...
**demo1: REFGEN to CLASSGEN**

- **BENCHMARK_DIR** = benchmarks/demos
- **REFGEN** = sched_small_s
  - sched_small_s = sched_classic.tcl 0 3 yes
- **CLASSGEN** = I,PC
  - I = rotate_formula I
  - PC = rotate_formula PC
  - **CLASS_SIZE** = 32
  - **SEED** = 33,1546,1968

**demo1: 1 SOLVER to POST_PROCESS**

- **SOLVER** = sato
  - sato = sato_encap.tcl
  - **TIME_OUT** = 3600
  - **SOLVE_CLASSES** = sched_small_s
- **POST_PROCESS** = sato_pp
  - sato_pp = sato_postProcess.tcl
demo2: 2 SOLVERs to POST_PROCESS

SOLVER = dp0_nat,unitwalk
dp0_nat =
  dp0_nat_encap.tcl
unitwalk =
  unitwalk_encap.tcl
TIME_OUT = 3600
SOLVE_CLASSES =
  sched03s_v00095_PC

POST_PROCESS =
  dp_nat_pp,unitwalk
dp_nat_pp =
  dp_nat_postProcess.tcl
unitwalk =
  unitwalk_postProcess.tcl

SATbed Output Data Examples --

Defaults for typical cases of equivalence-class instances
(32--128 instances per class):

• censoring data from experiment(s)
• statistics tables (*.stat)
• solvability functions (*.solv)
• independent group sample variability (*.box1)
• cummulative group sample variability (*.box2)

User-selected outputs:

• comparative stats reports (LaTex table)
Ex: Censoring Data from Experiment(s)

(1) count ‘unsolved’ and ‘verFail-solved’ class instances

<table>
<thead>
<tr>
<th>0_instance</th>
<th>B_resolvedAs</th>
<th>C_verify</th>
<th>D_timeout</th>
<th>E_runtime</th>
<th>L_implications</th>
<th>J_decisions</th>
<th>K_backtracks</th>
</tr>
</thead>
<tbody>
<tr>
<td>i0000.cnf</td>
<td>--sat--</td>
<td>verPass</td>
<td>60</td>
<td>1.382622</td>
<td>413</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>i0001.cnf</td>
<td>--sat--</td>
<td>verPass</td>
<td>60</td>
<td>29.772735</td>
<td>7547</td>
<td>203</td>
<td>196</td>
</tr>
<tr>
<td>i0002.cnf</td>
<td>--unsolved--</td>
<td>verPass</td>
<td>60</td>
<td>0.464843</td>
<td>86</td>
<td>165</td>
<td>9</td>
</tr>
<tr>
<td>i0028.cnf</td>
<td>--sat--</td>
<td>verPass</td>
<td>60</td>
<td>7.661937</td>
<td>1803</td>
<td>69</td>
<td>57</td>
</tr>
<tr>
<td>i0029.cnf</td>
<td>--unsolved--</td>
<td>verSkip</td>
<td>60</td>
<td>62.141199</td>
<td>21356</td>
<td>887</td>
<td>886</td>
</tr>
<tr>
<td>i0030.cnf</td>
<td>--sat--</td>
<td>verPass</td>
<td>60</td>
<td>4.86656</td>
<td>1699</td>
<td>82</td>
<td>73</td>
</tr>
<tr>
<td>i0031.cnf</td>
<td>--sat--</td>
<td>verPass</td>
<td>60</td>
<td>18.597445</td>
<td>11134</td>
<td>865</td>
<td>846</td>
</tr>
<tr>
<td>i0032.cnf</td>
<td>--sat--</td>
<td>verPass</td>
<td>60</td>
<td>0.688729</td>
<td>197</td>
<td>22</td>
<td>8</td>
</tr>
</tbody>
</table>

(2) sampleSize/classSize < 0.95 ==> ‘incomplete distrib.’

<table>
<thead>
<tr>
<th>0_LABLES</th>
<th>timeOut</th>
<th>nUnSolved</th>
<th>nVerFail</th>
<th>nVerSkip</th>
<th>A_classSize</th>
<th>B_sampleSize</th>
<th>C_distribution</th>
<th>D_initV</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_runtime</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>1.38</td>
</tr>
<tr>
<td>L_implications</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>4.13</td>
</tr>
<tr>
<td>J_decisions</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>23.0</td>
</tr>
<tr>
<td>K_backtracks</td>
<td>60</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>19.0</td>
</tr>
</tbody>
</table>

Ex: Three stat Files From the Web

**solverID = dp0_nat**  **classID = sched03s_v00095_PC**

<table>
<thead>
<tr>
<th>0_LABLES</th>
<th>A_classSize</th>
<th>B_sampleSize</th>
<th>C_distribution</th>
<th>D_initV</th>
<th>E_medV</th>
<th>F_meanV</th>
<th>G_stdV</th>
<th>H_cVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_runtime</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>1.38</td>
<td>2.05</td>
<td>6.60</td>
<td>9.59</td>
<td>1.45</td>
</tr>
<tr>
<td>L_implications</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>4.13</td>
<td>692</td>
<td>2291</td>
<td>3383</td>
<td>1.48</td>
</tr>
<tr>
<td>J_decisions</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>23.0</td>
<td>41.5</td>
<td>112</td>
<td>180</td>
<td>1.61</td>
</tr>
<tr>
<td>K_max_dec_lvl</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>19.0</td>
<td>28.0</td>
<td>97.8</td>
<td>176</td>
<td>1.80</td>
</tr>
</tbody>
</table>

**solverID = chaff**  **classID = sched03s_v00095_PC**

<table>
<thead>
<tr>
<th>0_LABLES</th>
<th>A_classSize</th>
<th>B_sampleSize</th>
<th>C_distribution</th>
<th>D_initV</th>
<th>E_medV</th>
<th>F_meanV</th>
<th>G_stdV</th>
<th>H_cVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_runtime</td>
<td>32</td>
<td>33</td>
<td>near-exponential</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>0.35</td>
</tr>
<tr>
<td>L_implications</td>
<td>32</td>
<td>33</td>
<td>normal</td>
<td>1206</td>
<td>1614</td>
<td>1687</td>
<td>363</td>
<td>0.21</td>
</tr>
<tr>
<td>J_decisions</td>
<td>32</td>
<td>33</td>
<td>normal</td>
<td>51.0</td>
<td>64.0</td>
<td>64.7</td>
<td>11.5</td>
<td>0.18</td>
</tr>
<tr>
<td>K_max_dec_lvl</td>
<td>32</td>
<td>33</td>
<td>near-normal</td>
<td>8.00</td>
<td>9.00</td>
<td>8.91</td>
<td>0.63</td>
<td>0.07</td>
</tr>
</tbody>
</table>

**solverID = sato**  **classID = sched03s_v00095_PC**

<table>
<thead>
<tr>
<th>0_LABLES</th>
<th>A_classSize</th>
<th>B_sampleSize</th>
<th>C_distribution</th>
<th>D_initV</th>
<th>E_medV</th>
<th>F_meanV</th>
<th>G_stdV</th>
<th>H_cVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_runtime</td>
<td>32</td>
<td>33</td>
<td>heavy-tailed</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>2.73</td>
</tr>
<tr>
<td>L_implications</td>
<td>32</td>
<td>33</td>
<td>near-normal</td>
<td>93.0</td>
<td>77.0</td>
<td>74.9</td>
<td>9.31</td>
<td>0.12</td>
</tr>
<tr>
<td>J_decisions</td>
<td>32</td>
<td>33</td>
<td>near-exponential</td>
<td>1.00</td>
<td>1.00</td>
<td>1.33</td>
<td>0.65</td>
<td>0.48</td>
</tr>
<tr>
<td>K_backtracks</td>
<td>32</td>
<td>33</td>
<td>impulse</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Ex: from *.solv files on the Web (1)

Analyzing the solvability of four solvers on the class of sched03s_v00095_PC (32 instances)

dp0_nat (vanilla DPLL) may benefit from “some heuristics”

Ex: from *.solv files on the Web (2)

Analyzing the solvability of four solvers on the class of sched03s_v00095_PC (32 instances)

chaff has unusual difficulty with this “sched” class
Ex: from *.solv files on the Web (3)

Analyzing the solvability of four solvers on the class of sched03s_v00095_PC (32 instances)

sato has significant edge over chaff for this “sched” class

Ex: from *.solv files on the Web (4)

Analyzing the solvability of four solvers on the class of sched03s_v00095_PC (32 instances)

unitwalk outperforms sato for this “sched” class
### Ex: Two stat Files From the Web

<table>
<thead>
<tr>
<th>solverID</th>
<th>classID</th>
<th>A_classSize</th>
<th>B_sampleSize</th>
<th>C_distribution</th>
<th>D_initV</th>
<th>E_medV</th>
<th>F_meanV</th>
<th>G_stdV</th>
<th>H_cVar</th>
</tr>
</thead>
<tbody>
<tr>
<td>dp0_nat</td>
<td>sched03s_v00095_PC</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>1.38</td>
<td>2.05</td>
<td>6.60</td>
<td>9.59</td>
<td>1.45</td>
</tr>
<tr>
<td>chaff</td>
<td>sched03s_v00095_PC</td>
<td>32</td>
<td>32</td>
<td>exponential</td>
<td>413</td>
<td>692</td>
<td>2291</td>
<td>3383</td>
<td>1.48</td>
</tr>
</tbody>
</table>

Should we compare **medians** (692 vs 1614 implications) or **means** (2291 vs 1687 implications) of an exponential versus a normal distribution?

### Ex: from a *.box1 file on the Web

**solverID = unitwalk,**
**classID = uf250-1065_027_PC (128 instances)**

How stable are consecutive means (and confidence int.)

![95% confidence interval of the mean](chart.png)

consecutive sample groups of size 16 (for $i > 0$)
Ex: from a *.box2 file on the Web

solverID = unitwalk,
classID = uf250-1065_027_PC (128 instances)

How stable are cumulative means (and confidence int.)

---

**Runtime (seconds)**

<table>
<thead>
<tr>
<th>cumulative sample groups in increments of 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

---

Ex: comparative report generation

**Inputs:**

- **RESULTS_DIR** = sat2003-results
- **solverID** = chaff, sato, satoL, unitwalk
- **classID** = uf250-1065_027_PC
- **costID** = runtime
- **stats** = initV, minV, meanV, maxV, stdev, distribution

**Output:**

<table>
<thead>
<tr>
<th>solverID</th>
<th>initV</th>
<th>minV</th>
<th>meanV</th>
<th>maxV</th>
<th>max/min</th>
<th>stdev</th>
<th>distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>unitwalk</td>
<td>0.72</td>
<td>0.08</td>
<td>6.84</td>
<td>32.1</td>
<td>402</td>
<td>6.46</td>
<td>exponential</td>
</tr>
<tr>
<td>satoL</td>
<td>46.8</td>
<td>0.22</td>
<td>26.5</td>
<td>56.0</td>
<td>255</td>
<td>19.6</td>
<td>near-exponential</td>
</tr>
<tr>
<td>chaff</td>
<td>115</td>
<td>8.66</td>
<td>69.4</td>
<td>356</td>
<td>41.1</td>
<td>62.0</td>
<td>exponential</td>
</tr>
<tr>
<td>sato</td>
<td>23.5</td>
<td>0.13</td>
<td>93.3</td>
<td>682</td>
<td>5243</td>
<td>106</td>
<td>exponential</td>
</tr>
</tbody>
</table>
Summary and Conclusions

- A real-time demo of the SATbed features presented in this talk is available on the Mac-OSX during this conference.

- A SATbed1.0 release, user guide, and tutorial is planned for end of May 2003 -- email will be sent to all attendees.

- In a talk on Thursday, we demonstrate merits of using SATbed routinely to analyze the performance improvements to a new experimental local search solver.

Home page for pubs, data sets and results and SATbed: http://www.cbl.ncsu.edu/OpenExperiments/SAT/