QingTing: A Fast SAT Solver Using Local Search and Efficient Unit Propagation

Xiao Yu Li   Matthias F. Stallmann   Franc Brglez
Dept. of Computer Science, NC State University
Raleigh, North Carolina, USA

http://pluto.cbl.ncsu/EDS/QingTing

Outline

- Motivation
- QingTing1
  - Improving UnitWalk
  - Faster Unit Propagation
- QingTing2
  - Switching strategy
  - Combine QingTing1 with WalkSAT
- Conclusions
Thanks!

Edward Hirsch and Arist Kojevnikov for distribution of

UnitWalk 0.944 (UW1)

and

UnitWalk 0.981 (UW2)

Motivation: UnitWalk Algorithm

Incomplete local search algorithm

Uses unit propagation intensively

Competitive with other solvers such as chaff, sato (complete solvers) and WalkSAT, GSAT (incomplete solvers)
This Talk: A Look at Two Benchmarks

**Structured benchmark:** sched6.cnf  
Variables = 808  
Clauses = 12024  
PC class size = 32

**Random 3-SAT benchmark:** uf250-1065-087  
Variables = 250  
Clauses = 1065  
PC class size = 128

sched6: UW1 vs Chaff  (1)

Comparison between UnitWalk1 and chaff  
on a structured instance.

Exponential Distribution  
Mean = 18.0  Std deviation = 20.1

exp. d.  
18.0/20.1
sched6: UW1 vs Chaff  (2)

UW1 solves the instance with less mean runtime and less variation than chaff.

uf250-1065-87: UW1 vs Chaff  (1)

Comparison between UnitWalk1 and chaff on a randomly generated instance.
UW1 outperforms chaff again.

Our Solver: QingTing1

How to improve UnitWalk?

- UnitWalk uses counter-based adjacency list as underlying data structure
- Improve its unit propagation

QingTing1 (QT1) uses two well-known unit propagation techniques used in complete solvers:

- sato’s unit propagation algorithm
- chaff’s watched literal lazy data structure
sched06: QT1 vs UW1   (1)

How much better can we do with better unit propagation techniques on a structured instance?

sched06: QT1 vs UW1   (2)

With all other parameters, e.g. flips, approximately the same, QT1 runs five times faster than UW1.
uf250-1065-87: QT1 vs UW1  (1)

How much better can we do on a randomly generated instance?

uf250-1065-87: QT1 vs UW1  (2)

Not much! The 5%-level t-test shows no difference. Lack of improvement due to cost of maintaining dynamic data structure.

**t-test result:**
\[ t = 1.88 < 1.97 \]
Switching Strategy

Another Local search algorithm: WalkSAT

Observations:
1) QT1 works well on structured instances, but not on random instances
2) WalkSAT-Novelty works well on random SAT instances

Questions:
1) How do we know which one to use (switching strategy)?
2) How do we decide with low cost?

Switching Strategy: Method A (1)

Random Assignment

Formula

Formula Empty?

Yes
Terminate

No

Formula empty when each clause is empty or satisfied.
Switching Strategy: Method A (2)

Method A measures the number of random assignments and normalize with respect to the number of variables:

\[
\text{random assignment} \% = \frac{\text{# of random assignments}}{\text{# of variables}}
\]

Repeat the process 128 times and calculate the average.

Method A: An Example

Random Assignment: \((x, y, z)\), \((-x, y)\), \((-x, y, z)\), \((-y, -z)\), \((-x, z)\)

\[
\begin{align*}
x &= 1 & \quad (y) & \quad (y, z) & \quad (-y, -z) & \quad (z) \\
z &= 0 & \quad (y) & \quad (y) & \quad ( ) \\
y &= 0 & \quad ( ) & \quad ( ) & \quad ( ) 
\end{align*}
\]

formula empty!
Method A: Experiments

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Random Assignment % Mean/Std</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured</td>
<td>0.98/0.05</td>
<td>near-exp</td>
</tr>
<tr>
<td>Instances</td>
<td>0.98/0.06</td>
<td>near-exp</td>
</tr>
<tr>
<td>Random</td>
<td>0.98/0.06</td>
<td>near-exp</td>
</tr>
<tr>
<td>Instances</td>
<td>0.96/0.07</td>
<td>near-exp</td>
</tr>
</tbody>
</table>

Method A is no good. There is no difference!

Switching Strategy: Method B

Unit Propagation

Random Assignment → Formula

Formula Empty? → Yes, Terminate

No

Repeat the procedure 128 times and calculate average Random assignments %
Method B: On the Same Example

Random Assignment

\[(x \ y \ z) \ (-x \ y) \ (-x \ y \ z) \ (-y \ -z) \ (-x \ z)\]

\[x = 1\]

\[(y) \ (y \ z) \ (-y \ -z) \ (z)\]

Unit propagate with \(y = 1\)

\[(-z) \ (z)\]

Unit propagate with \(z = 0\)

\[() \ ()\]

formula empty!

Method B: Experiments (1)

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Random Assignment % Mean/Std</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bw_large_a</td>
<td>0.010/0.006</td>
<td>near-exp</td>
</tr>
<tr>
<td>Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>logistics_b</td>
<td>0.015/0.002</td>
<td>normal</td>
</tr>
<tr>
<td>Random</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uf250-1065-27</td>
<td>0.150/0.025</td>
<td>normal</td>
</tr>
<tr>
<td>Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uf250-1065-87</td>
<td>0.156/0.026</td>
<td>normal</td>
</tr>
</tbody>
</table>

Random assignment % has large gap between the two groups, but is it because of 3-SAT?
### Method B: Experiments (2)

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Random Assignment %</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structured</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bw_large_a</td>
<td>0.010/0.006</td>
<td>near-exp</td>
</tr>
<tr>
<td>instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bw_large_a_3sat</td>
<td>0.046/0.024</td>
<td>near-exp</td>
</tr>
<tr>
<td>logistics_b</td>
<td>0.015/0.002</td>
<td>normal</td>
</tr>
<tr>
<td>logistics_b_3sat</td>
<td>0.049/0.066</td>
<td>near-exp</td>
</tr>
<tr>
<td><strong>Random</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uf250-1065-27</td>
<td>0.150/0.025</td>
<td>normal</td>
</tr>
<tr>
<td>Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uf250-1065-87</td>
<td>0.156/0.026</td>
<td>normal</td>
</tr>
</tbody>
</table>

Random assignment % increases but the gap still exists!

What about other randomly generated 3-SAT?

### Method B: Experiments (3)

<table>
<thead>
<tr>
<th>Benchmark Name</th>
<th>Random Assignment %</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Structured</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bw_large_a</td>
<td>0.010/0.006</td>
<td>near-exp</td>
</tr>
<tr>
<td>instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bw_large_a_3sat</td>
<td>0.046/0.024</td>
<td>near-exp</td>
</tr>
<tr>
<td>logistics_b</td>
<td>0.015/0.002</td>
<td>normal</td>
</tr>
<tr>
<td>logistics_b_3sat</td>
<td>0.049/0.066</td>
<td>near-exp</td>
</tr>
<tr>
<td><strong>Random</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uf250-1065-27</td>
<td>0.150/0.025</td>
<td>normal</td>
</tr>
<tr>
<td>Instances</td>
<td></td>
<td></td>
</tr>
<tr>
<td>uf250-1065-87</td>
<td>0.156/0.026</td>
<td>normal</td>
</tr>
<tr>
<td>hgen2-v250-s2</td>
<td>0.159/0.022</td>
<td>normal</td>
</tr>
<tr>
<td>hgen2-v250-s500</td>
<td>0.161/0.019</td>
<td>near-normal</td>
</tr>
</tbody>
</table>
QingTing2 and UnitWalk2

QingTing2
- Switching strategy (low overhead)
- On randomly generated instances, QT2 behaves like WalkSAT-Novelt
- On structured instances, QT2 behaves like QT1

UnitWalk2
- UnitWalk2 is the latest version of UnitWalk and it also does some structure detection

sched6: QT2/QT1 vs UW2/UW1 (1)

What will the improvements be on a structured instance?

![Graph showing solvability vs runtime](image)
sched6: QT2/QT1 vs UW2/UW1  (2)

QT1 runs five times faster than UW1

sched6: QT2/QT1 vs UW2/UW1  (3)

QT1 runs five times faster than UW1

UW2 and UW1 are the same (t-test: t = 1.44 <1.97 )
sched6: QT2/QT1 vs UW2/UW1 (4)

QT1 runs five times faster than UW1

UW2 and UW1 are the same (t-test: $t = 1.44 < 1.97$)

QT1 and QT2 are the same (t-test: $t = 1.18 < 1.97$)

uf250..87: QT2/QT1 vs UW2/UW1 (1)

What about random 3-SAT instances?
uf250..87: QT2/QT1 vs UW2/UW1 (2)

UW1 performs the same as QT1 (t-test: $t = 1.88 > 1.97$)

![Graph showing runtime vs solvability for uf250..87-
uw1 and uf250..87-qt1.](image1)

exp. d. 16.7/17.2

exp. d. 12.3/12.5

uf250..87: QT2/QT1 vs UW2/UW1 (3)

UW1 performs the same as QT1 (t-test: $t = 1.88 > 1.97$)

UW2 outperforms UW1 by a factor of 31 ...

![Graph showing runtime vs solvability for uf250..87-
uw1, uf250..87-qt1, uf250..87_uw2.](image2)

exp. d. 0.39/0.29

exp. d. 16.7/17.2

exp. d. 12.3/12.5
uf250..87: QT2/QT1 vs UW2/UW1 (4)

UW1 performs the same as QT1 (t-test: $t = 1.88 > 1.97$)

UW2 outperforms UW1 by a factor of 31 ...

QT2 outperforms UW2 slightly (t-test: $t = 2.24 > 1.97$)

Summary and Conclusions (1)

- QingTing1
  - Improving UnitWalk
  - Faster Unit Propagation

- QingTing2
  - Switching strategy
  - Combine QingTing1 with WalkSAT
Summary and Conclusions (2)

Future research directions

1. Use switching strategy in a complete solver

2. Use learning techniques in local search algorithms

3. Combine the theory and the experimental methodology