Enhanced Gaussian Elimination in DPLL-based SAT Solvers

Mate Soos

UPMC LIP6, PLANETE team INRIA, SALSA team INRIA

10th of July 2010
Table of Contents

1 Context

2 Gaussian elimination in SAT Solvers

3 Results

4 Conclusions
Outline

1. Context
   - Cryptographic problems
   - Gaussian elimination

2. Gaussian elimination in SAT Solvers
   - Datastructures, algorithms
   - Row and Column Elimination by XOR
   - Independent sub-matrixes
   - Skipping parts of matrix to treat

3. Results

4. Conclusions
DPLL-based SAT solvers

Solves a problem in CNF

CNF is an “and of or-s”

\[
\neg x_1 \lor \neg x_3 \quad \neg x_2 \lor x_3 \quad x_1 \lor x_2
\]

Uses DPLL(\(\varphi\)) algorithm

1. If formula \(\varphi\) is trivial, return SAT/UNSAT
2. Picks a variable \(v\) to branch on
3. \(v := \text{true}\)
4. Simplifies formula to \(\varphi'\) and calls DPLL(\(\varphi'\))
5. if SAT, output SAT
6. if UNSAT, \(v := \text{false}\)
7. Simplifies formula to \(\varphi''\) and calls DPLL(\(\varphi''\))
8. if SAT, output SAT
9. if UNSAT, output UNSAT
## Cryptographic problems

### Crypto problems are given in ANF

0 = \( ab \oplus b \oplus bc \)

0 = \( a \oplus d \oplus c \oplus bd \)

0 = \( bc \oplus cd \oplus bd \)

0 = \( d \oplus ab \oplus 1 \)

### Methods to solve ANF

1. Put into matrix, Gauss eliminate:

\[
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

2. Convert to CNF. Notice: it’s same as above, but \( ab = a \times b \) is included, and less info (rows) needed.

3. Other methods (e.g. F4/F5)
Gaussian elimination

Theory

- Solving a Gaussian elimination problem with DPLL-based SAT solvers is exponentially difficult
- Even though Gaussian elimination is poly-time
  - Theoretically, Gaussian elimination in SAT solvers is useful

Practise

- Designers of SAT solvers have grown accustomed to solving worst-case exponential problems really fast
- But Gaussian elimination is different:

<table>
<thead>
<tr>
<th>Matrix size: $n \times n$, MiniSat time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
</tr>
<tr>
<td>0.02</td>
</tr>
</tbody>
</table>

- Practical usefulness is still elusive
Gauss and Crypto

The two approaches

- Only-Gauss approach problem: too many rows needed, too large matrix
- Only-SAT approach problem: Can’t “see” the matrix, can’t find truths from it

A hybrid approach

Executing Gauss. elimination at every decision step in the SAT solver, we can mix the two approaches

SAT Solver

Gauss \leftrightarrow DPLL

At every decision, exchange of information
1. Context
   - Cryptographic problems
   - Gaussian elimination

2. Gaussian elimination in SAT Solvers
   - Datastructures, algorithms
   - Row and Column Elimination by XOR
   - Independent sub-matrixes
   - Skipping parts of matrix to treat

3. Results

4. Conclusions
# Implementation

<table>
<thead>
<tr>
<th>A-matrix</th>
<th>N-matrix</th>
</tr>
</thead>
</table>
|\[ \begin{bmatrix}
  v_{10} & v_8 & v_9 & v_{12} & \text{aug} \\
  1 & 1 & 1 & 1 & 0 \\
  0 & 0 & 1 & 1 & 1 \\
  0 & 1 & 0 & 1 & 1 \\
  0 & 1 & 0 & 0 & 1 \\
\end{bmatrix} \] | \[ \begin{bmatrix}
  v_{10} & v_8 & v_9 & v_{12} & \text{aug} \\
  1 & 1 & 1 & 1 & 0 \\
  0 & 0 & 1 & 1 & 1 \\
  0 & 1 & 0 & 1 & 1 \\
  0 & 1 & 0 & 0 & 1 \\
\end{bmatrix} \] |
Datastructures, algorithms

### Implementation

<table>
<thead>
<tr>
<th>A-matrix with ( v8 ) assigned to true</th>
<th>N-matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| \( \begin{array}{cccc|c}
 v10 & v8 & v9 & v12 & \text{aug} \\
 1 & - & 1 & 1 & 1 \\
 0 & - & 1 & 1 & 1 \\
 0 & - & 0 & 1 & 0 \\
 0 & - & 0 & 0 & 0 \\
\end{array} \) | \( \begin{array}{cccc|c}
 v10 & v8 & v9 & v12 & \text{aug} \\
 1 & 1 & 1 & 1 & 1 \\
 0 & 0 & 1 & 1 & 1 \\
 0 & 1 & 0 & 1 & 1 \\
 0 & 1 & 0 & 0 & 1 \\
\end{array} \) |

Mate Soos (UPMC LIP6, PLANETE team INRIA, SALSA team INRIA)

Gauss in SAT solvers

10th of July 2010 9 / 23
### Implementation

**A-matrix**

<table>
<thead>
<tr>
<th></th>
<th>v10</th>
<th>v8</th>
<th>v9</th>
<th>v12</th>
<th>aug</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**N-matrix**

<table>
<thead>
<tr>
<th></th>
<th>v10</th>
<th>v8</th>
<th>v9</th>
<th>v12</th>
<th>aug</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Resulting xor-clause:

\[ v_8 \oplus v_{12} \]
Implementation

A-matrix with $v_8$ assigned to true

<table>
<thead>
<tr>
<th>$v_{10}$</th>
<th>$v_8$</th>
<th>$v_9$</th>
<th>$v_{12}$</th>
<th>aug</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

N-matrix

<table>
<thead>
<tr>
<th>$v_{10}$</th>
<th>$v_8$</th>
<th>$v_9$</th>
<th>$v_{12}$</th>
<th>aug</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Resulting xor-clause:

$v_{12} = \text{false} \leftarrow v_8 \oplus v_{12}$
Example

- If variable $a$ is not present anywhere but in 2 XOR-s:

$$a \oplus b \oplus c \oplus d = \text{false}$$
$$a \oplus f \oplus g \oplus h = \text{false}$$

- Then we can remove $a$, the two XOR-s, and add the XOR:

$$f \oplus g \oplus h \oplus b \oplus c \oplus d = \text{false}$$

Theory

- This is variable elimination at the XOR-level
- It is equivalent to VE at CNF level
- But it doesn’t make sense to do this at CNF level:
  - → results in far more (and larger) clauses
- For us it helps: removes 1 column ($a$) and one row from the matrix
Independent sub-matrixes

Reasoning

- Gaussian elimination is approx. $O(nm^2)$ algorithm
- Making two smaller matrixes from one bigger one leads to speedup
- If matrix has non-connected components, cutting up is orthogonal to algorithm output

![Diagram of independent sub-matrixes]

Mate Soos (UPMC LIP6, PLANETE team INRIA, SALSA team INRIA)

Gauss in SAT solvers

10th of July 2010
Independent sub-matrixes

Algorithm
Let us build a graph from the XOR-s:
- Vertexes are the variables
- Edge runs between two vertexes if they appear in an XOR
- Independent graph components are extracted

Advantages
- In case of 2 roughly equal independent sub-matrixes:
  \[ cnm^2 \rightarrow 2c'(n/2)(m/2)^2 = c'nm^2/4 \]
- Better understanding of problem structure:
  - E.g. number of shift registers in a cipher
  - Number of S-boxes in cipher
  - Problem similarities

Mate Soos  (UPMC LIP6, PLANETE team)  Gauss in SAT solvers  10th of July 2010  12 / 23
Not treating parts of the matrix

Reasoning

- Let’s assume the leftmost column updated is the $c^{th}$
- Let’s assume the topmost “1” in this column was in row $r$
- Then, the rows above $r$ cannot have changed their leading 1

Example

```
<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Mate Soos (UPMC LIP6, PLANETE team)
Gauss in SAT solvers
Auto turn-off heuristics

Reasoning
- Gauss doesn’t work well for all restarts
- If it doesn’t bring enough benefits, switch it off
- Performance is measured by percentage of times confl/prop is generated

Quantitatively
- If \( 2\text{numGaussConfl} + \text{numGaussProp} < 0.05\text{numGaussCalled} \)
  → Then turn it off
- Conflict is preferred — we can return immediately
More efficient data structure

### Data structure

- Bits are packed — faster row xor/swap
- Augmented column is non-packed — faster checking
- Two matrixes are stored as an interlaced continuous array
- \[ A[0][0] \ldots A[0][n], \ N[0][0] \ldots N[0][n], \ldots A[m][0] \ldots N[m][n] \]

### Advantages

- When doing row-xor both matrixes’ rows are xor-ed
- When doing row-swap both matrixes’ rows are swapped
- We can now operate on *one* continuous data in both operations
1 Context
- Cryptographic problems
- Gaussian elimination

2 Gaussian elimination in SAT Solvers
- Datastructures, algorithms
- Row and Column Elimination by XOR
- Independent sub-matrixes
- Skipping parts of matrix to treat

3 Results

4 Conclusions
Before: “Extending SAT Solvers to Cryptographic Problems”

- Worked only on few instances
- Had to be tuned for each instance
- Gave approx. 5-10% speedup

Now: “Enhanced Gaussian Elimination in DPLL-based SAT Solvers”

- Matrix discovery is automatic
- Less tuning necessary – turn-off is automatic
- Works on more types of instances
- Gives up to 30%-45% speedup
# Results — RCX

Table: Avg. time (in sec.) to solve 100 random problems

<table>
<thead>
<tr>
<th>no. help bits</th>
<th>55</th>
<th>54</th>
<th>53</th>
<th>52</th>
<th>51</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>no RCX + no Gauss</td>
<td>0.69</td>
<td>1.26</td>
<td>1.38</td>
<td>2.19</td>
<td>6.25</td>
<td>10.40</td>
</tr>
<tr>
<td>RCX + no Gauss</td>
<td>0.65</td>
<td>0.89</td>
<td>1.30</td>
<td>2.36</td>
<td>5.76</td>
<td>8.87</td>
</tr>
<tr>
<td>no RCX + Gauss</td>
<td>0.55</td>
<td>0.91</td>
<td>1.06</td>
<td>1.89</td>
<td>3.87</td>
<td>7.76</td>
</tr>
<tr>
<td>RCX + Gauss</td>
<td>0.52</td>
<td>0.69</td>
<td>0.90</td>
<td>1.85</td>
<td>3.81</td>
<td>6.20</td>
</tr>
<tr>
<td>Vars removed on avg</td>
<td>36.27</td>
<td>36.42</td>
<td>37.30</td>
<td>37.07</td>
<td>38.32</td>
<td>37.94</td>
</tr>
</tbody>
</table>

Mate Soos (UPMC LIP6, PLANETE team) Gauss in SAT solvers 10th of July 2010
## Results — Gauss

**Table:** Avg. time (in sec.) to solve 100 random problems

<table>
<thead>
<tr>
<th></th>
<th>Bivium</th>
<th>Trivium</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>no. help bits</td>
<td>54</td>
</tr>
<tr>
<td>RCX</td>
<td>0.89</td>
<td>1.30</td>
</tr>
<tr>
<td>Gauss+RCX</td>
<td>0.69</td>
<td>0.90</td>
</tr>
<tr>
<td>RCX</td>
<td>66.57</td>
<td>86.42</td>
</tr>
<tr>
<td>Gauss+RCX</td>
<td>40.57</td>
<td>68.16</td>
</tr>
</tbody>
</table>
### Table: Avg. time (in sec.) to solve 100 random problems

<table>
<thead>
<tr>
<th>HiTag2</th>
<th>no. help bits</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCX</td>
<td></td>
<td>4.78</td>
<td>11.73</td>
<td>30.70</td>
<td>76.44</td>
<td>233.61</td>
<td>719.86</td>
<td>1666.99</td>
</tr>
<tr>
<td>Gauss+RCX</td>
<td></td>
<td>4.76</td>
<td>11.64</td>
<td>29.03</td>
<td>77.19</td>
<td>220.64</td>
<td>701.46</td>
<td>1636.77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grain</th>
<th>no. help bits</th>
<th>109</th>
<th>108</th>
<th>107</th>
<th>106</th>
</tr>
</thead>
<tbody>
<tr>
<td>RCX</td>
<td></td>
<td>168.51</td>
<td>291.29</td>
<td>540.14</td>
<td>1123.08</td>
</tr>
<tr>
<td>Gauss+RCX</td>
<td></td>
<td>193.09</td>
<td>359.58</td>
<td>608.47</td>
<td>1133.75</td>
</tr>
</tbody>
</table>
1 Context
   - Cryptographic problems
   - Gaussian elimination

2 Gaussian elimination in SAT Solvers
   - Datastructures, algorithms
   - Row and Column Elimination by XOR
   - Independent sub-matrixes
   - Skipping parts of matrix to treat

3 Results

4 Conclusions
Conclusions

- Gaussian elimination can bring benefits for specific applications
- Better understanding of the problem could be gained

Possible future work

- Automatic cut-off value finding
- Better heuristics to decide when to execute Gaussian elim.
- Add support for sparse matrix representation
Thank you for your time