SAT Encodings for Sudoku

Bug Catching in 2006 Fall

Sep. 26, 2006

Gi-Hwon Kwon
Various SAT Encoding

- C Program
- Optimal Path Planning
- Sudoku Puzzle
- Latin Square Problem
- Traveling Salesmen Problem

Encodings:
- Encoding 1
- Encoding 2
- Encoding 3
- Encoding n

CNF SAT Formula

SAT Solver
Agenda

• Introduction

• Background and Previous Encodings

• Optimized Encoding

• Experimental Results

• Conclusions
What is Sudoku?

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Sudoku Problem" /></td>
<td><img src="image2.png" alt="Sudoku Solution" /></td>
</tr>
</tbody>
</table>

Given a problem, the objective is to find a satisfying assignment w.r.t. Sudoku rules.

Sudoku rules

- There is a number in each **cell**.
- A number appears once in each **row**.
- A number appears once in each **column**.
- A number appears once in each **block**.
Sudoku as SAT Problem

Sudoku → Encoder → CNF → SAT Solver → SAT? → Decoder → Solution found

Symbol table

No solution found

Model
Previous Encodings

- **Minimal** encoding [Lynce & Ouaknine, 2006]
- **Extended** encoding [Lynce & Ouaknine, 2006]
- **Efficient** encoding [Weber, 2005]

Sudoku \[\rightarrow\] Encoder \[\rightarrow\] CNF \[\rightarrow\] SAT Solver \[\rightarrow\] SAT? \[\rightarrow\] Decoder

- symbol table
- model

\[\text{yes}\]
# Analysis of Previous Encodings

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Number of Variables</th>
<th>Number of Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal</td>
<td>$N^3$</td>
<td>$N^3 + \left( N^3 \cdot \left( \frac{N^3(N-1)}{2} \right) \right) \cdot 3 + k$</td>
</tr>
<tr>
<td>Efficient</td>
<td>$N^3$</td>
<td>$N^3 + \left( N^3 \cdot \left( \frac{N^3(N-1)}{2} \right) \right) \cdot 4 + k$</td>
</tr>
<tr>
<td>Extended</td>
<td>$N^3$</td>
<td>$\left( N^3 + N^3 \cdot \left( \frac{N^3(N-1)}{2} \right) \right) \cdot 4 + k$</td>
</tr>
</tbody>
</table>
Exponential Growth in Clauses

<table>
<thead>
<tr>
<th>size</th>
<th>minimal</th>
<th>efficient</th>
<th>extended</th>
</tr>
</thead>
<tbody>
<tr>
<td>9x9</td>
<td>8829</td>
<td>11745</td>
<td>11988</td>
</tr>
<tr>
<td>16x16</td>
<td>92416</td>
<td>123136</td>
<td>123904</td>
</tr>
<tr>
<td>25x25</td>
<td>563125</td>
<td>750625</td>
<td>752500</td>
</tr>
<tr>
<td>36x36</td>
<td>2450736</td>
<td>3267216</td>
<td>3271104</td>
</tr>
<tr>
<td>49x49</td>
<td>8473129</td>
<td>11296705</td>
<td>11303908</td>
</tr>
<tr>
<td>64x64</td>
<td>24776704</td>
<td>33034240</td>
<td>33046528</td>
</tr>
<tr>
<td>81x81</td>
<td>63779481</td>
<td>85037121</td>
<td>85056804</td>
</tr>
</tbody>
</table>
# Experimental Results

<table>
<thead>
<tr>
<th>size</th>
<th>level</th>
<th>minimal encoding</th>
<th>efficient encoding</th>
<th>extended encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>vars</td>
<td>clauses</td>
<td>time</td>
</tr>
<tr>
<td>9x9</td>
<td>easy</td>
<td>729</td>
<td>8854</td>
<td>0.00</td>
</tr>
<tr>
<td>9x9</td>
<td>hard</td>
<td>729</td>
<td>8859</td>
<td>0.00</td>
</tr>
<tr>
<td>16x16</td>
<td>easy</td>
<td>4096</td>
<td>92520</td>
<td>0.10</td>
</tr>
<tr>
<td>16x16</td>
<td>hard</td>
<td>4096</td>
<td>92514</td>
<td>0.46</td>
</tr>
<tr>
<td>25x25</td>
<td>easy</td>
<td>15625</td>
<td>563417</td>
<td>9.07</td>
</tr>
<tr>
<td>25x25</td>
<td>hard</td>
<td>15625</td>
<td>563403</td>
<td>time</td>
</tr>
<tr>
<td>36x36</td>
<td>easy</td>
<td>46656</td>
<td>2451380</td>
<td>time</td>
</tr>
<tr>
<td>36x36</td>
<td>hard</td>
<td>46656</td>
<td>2451400</td>
<td>time</td>
</tr>
<tr>
<td>49x49</td>
<td>easy</td>
<td>117649</td>
<td>8474410</td>
<td>time</td>
</tr>
<tr>
<td>64x64</td>
<td>easy</td>
<td>262144</td>
<td>24779088</td>
<td>stack</td>
</tr>
<tr>
<td>81x81</td>
<td>easy</td>
<td>531441</td>
<td>63783464</td>
<td>stack</td>
</tr>
</tbody>
</table>
## Experimental Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>minimal encoding</th>
<th></th>
<th>efficient encoding</th>
<th></th>
<th>extended encoding</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size</td>
<td>vars</td>
<td>clauses</td>
<td>time</td>
<td>vars</td>
<td>clauses</td>
<td>time</td>
</tr>
<tr>
<td></td>
<td>level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9x9</td>
<td>easy</td>
<td>729</td>
<td>8854</td>
<td>0.00</td>
<td>729</td>
<td>11770</td>
<td>0.00</td>
</tr>
<tr>
<td>9x9</td>
<td>hard</td>
<td>729</td>
<td>8859</td>
<td>0.00</td>
<td>729</td>
<td>11775</td>
<td>0.00</td>
</tr>
<tr>
<td>16x16</td>
<td>easy</td>
<td>4096</td>
<td>92520</td>
<td>0.10</td>
<td>4096</td>
<td>123240</td>
<td>0.09</td>
</tr>
<tr>
<td>16x16</td>
<td>hard</td>
<td>4096</td>
<td>92514</td>
<td>0.46</td>
<td>4096</td>
<td>123234</td>
<td>0.21</td>
</tr>
<tr>
<td>25x25</td>
<td>easy</td>
<td>15625</td>
<td>563417</td>
<td>9.07</td>
<td>15625</td>
<td>750917</td>
<td>17.48</td>
</tr>
<tr>
<td>25x25</td>
<td>hard</td>
<td>15625</td>
<td>563403</td>
<td>time</td>
<td>15625</td>
<td>750903</td>
<td>time</td>
</tr>
<tr>
<td>36x36</td>
<td>easy</td>
<td>46656</td>
<td>2451380</td>
<td>time</td>
<td>46656</td>
<td>3267860</td>
<td>time</td>
</tr>
<tr>
<td>36x36</td>
<td>hard</td>
<td>46656</td>
<td>2451400</td>
<td></td>
<td>46656</td>
<td>3267880</td>
<td></td>
</tr>
<tr>
<td>49x49</td>
<td>easy</td>
<td>117649</td>
<td>8474410</td>
<td></td>
<td>7649</td>
<td>11297986</td>
<td></td>
</tr>
<tr>
<td>64x64</td>
<td>easy</td>
<td>262144</td>
<td>63783404</td>
<td>stack</td>
<td>16624</td>
<td>33048912</td>
<td>stack</td>
</tr>
<tr>
<td>81x81</td>
<td>easy</td>
<td>531441</td>
<td>85041104</td>
<td>stack</td>
<td>531441</td>
<td>85060787</td>
<td>stack</td>
</tr>
</tbody>
</table>

Solution found

No solution found
Motivations

• Sudoku was regarded as SAT problem
  ➔ Extended encoding shows the best performance in our experiments

• Problems in previous works
  ▪ Too many clauses are generated (e.g. 85,056,804 clauses)
  ▪ Thus, large size puzzles are not solved
  ➔ The extended encoding must be optimized for large size puzzles
Agenda

• Introduction

• **Background and Previous Encodings**

• Optimized Encoding

• Experimental Results

• Conclusions
Encoding

- Knowledge compilation into a **target language**

  problem knowledge $\rightarrow$ CNF

- Knowledge about Sudoku

  - A number appears once in each **cell**
  - A number appears once in each **row**
  - A number appears once in each **col**
  - A number appears once in each **block**
  - A pre-assigned number
Variables

• Each cell has one number from 1..N
  ▪ \([1,1]=1\) or \([1,1]=2\) or …… or \([1,1]=N\)
  ▪ Each cell needs N boolean variables to consider all cases

• Total number of variables
  ▪ \(N^3\)

• Boolean variable name as a triple
  ▪ \((r,c,v)\) (i.e., \(x_{rcv}\)) iff \([r,c]=v\)
  ▪ \(\neg(r,c,v)\) (i.e., \(\neg x_{rcv}\)) iff \([r,c]\neq v\)
Cell Rule $\rightarrow$ CNF

A number appears once in each cell

There is **at least** one number in each cell (definedness)

$$Cell_d = \bigwedge_{r=1}^{N} \bigwedge_{c=1}^{N} \bigvee_{v=1}^{N} (r, c, v)$$

There is **at most** one number in each cell (uniqueness)

$$Cell_u = \bigwedge_{r=1}^{N} \bigwedge_{c=1}^{N} \bigwedge_{v_i=1}^{N-1} \bigwedge_{v_j=v_i+1}^{N} \neg((r, c, v_i) \land (r, c, v_j))$$
Row Rule $\rightarrow$ CNF

Each number appears at least once in each row (definedness)
\[ \text{Row}_d = \bigwedge_{r=1}^{N} \bigwedge_{v=1}^{N} \bigvee_{c=1}^{N} (r, c, v) \]

Each number appears at most once in each row (uniqueness)
\[ \text{Row}_u = \bigwedge_{r=1}^{N} \bigwedge_{v=1}^{N} \bigwedge_{c_i=1}^{N-1} \bigwedge_{c_j=c_i+1}^{N} \neg((r, c_i, v) \land (r, c_j, v)) \]
Column Rule $\rightarrow$ CNF

A number appears once in each column

Each number appears at least once in each column (definedness)

$$Col_d = \bigwedge_{c=1}^N \bigwedge_{v=1}^N \bigvee_{r=1}^N (r, c, v)$$

Each number appears at most once in each column (uniqueness)

$$Col_u = \bigwedge_{c=1}^N \bigwedge_{v=1}^N \bigwedge_{r_i=1}^{N-1} \bigwedge_{r_j=r_i+1}^N \neg((r_i, c, v) \land (r_j, c, v))$$
Block Rule $\rightarrow$ CNF

Each number appears **at least** once in each block \[(\text{definedness})\]

\[
Block_d = \bigwedge_{r_{offs}=1}^{subN} \bigwedge_{c_{offs}=1}^{subN} \bigwedge_{v=1}^{N} \bigvee_{r=1}^{N} \bigvee_{c=1}^{N} (r_{offs} \ast subN + r, c_{offs} \ast subN + c, v)
\]

Each number appears **at most** once in each block \[(\text{uniqueness})\]

\[
Block_u = \bigwedge_{r_{offs}=1}^{subN} \bigwedge_{c_{offs}=1}^{subN} \bigwedge_{v=1}^{N} \bigwedge_{r=1}^{N} \bigwedge_{c=r+1}^{N} \\
\neg((r_{offs} \ast subN + (r \mod subN), c_{offs} \ast subN + (r \mod subN), v) \\
\land (r_{offs} \ast subN + (c \mod subN), c_{offs} \ast subN + (c \mod subN), v))
\]
Pre-Assigned Fact $\rightarrow$ CNF

As a constant; the number is never changed

It can be represented as a {\textit{unit clause}}

\[
\text{Assigned} = \bigwedge_{i=1}^{k} \{(r, c, a) \mid \exists_{1 \leq a \leq N} \bullet [r, c] = a\}
\]

where $k$ is a number of pre-assigned numbers
Previous Encodings

**Minimal encoding**  [Lynce & Ouaknine, 2006]

\[
\phi = Cell_d \cup Row_u \cup Col_u \cup Block_u \cup Assigned
\]

sufficient to characterize the puzzle

**Extended encoding**  [Lynce & Ouaknine, 2006]

\[
\phi = Cell_d \cup Cell_u \cup Row_d \cup Row_u \cup Col_d \cup Col_u \\
\cup Block_d \cup Block_u \cup Assigned
\]

minimal encoding with redundant clauses

**Efficient encoding**  [Weber, 2005]

\[
\phi = Cell_d \cup Cell_u \cup Row_u \cup Col_u \cup Block_u \cup Assigned
\]

between minimal encoding and extended encoding
# Analysis (Recap)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Number of Variables</th>
<th>Number of Clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal</td>
<td>$N^3$</td>
<td>$N \times N + \left( N \times N \times \left( \frac{N \times (N - 1)}{2} \right) \right) \times 3 + k$</td>
</tr>
<tr>
<td>Efficient</td>
<td>$N^3$</td>
<td>$N \times N + \left( N \times N \times \left( \frac{N \times (N - 1)}{2} \right) \right) \times 4 + k$</td>
</tr>
<tr>
<td>Extended</td>
<td>$N^3$</td>
<td>$\left( N \times N + N \times N \times \left( \frac{N \times (N - 1)}{2} \right) \right) \times 4 + k$</td>
</tr>
</tbody>
</table>
Agenda

• Introduction

• Background and Previous Encodings

• Optimized Encoding

• Experimental Results

• Conclusions
Example

For example, consider the cell \([1,1]\)
- Four cases are considered; thus, four variables are needed
  \((1,1,1), (1,1,2), (1,1,3), (1,1,4)\)
Variables

• A pre-assigned cell reduces the cases to be considered
  ▪ Because the cell has a fixed number
  ▪ The pre-assigned cell does not need a variable at all
  ▪ It affects other cells located at the same row, or column, or block.

• For example, consider the cell \([1,1]\)
  ▪ The case \([1,1]=1\) is not allowed since \([4,1]=1\) are already assigned
  ▪ The case \([1,1]=3\) is not allowed since \([1,4]=3\) are already assigned
  ▪ The case \([1,1]=4\) is not allowed since \([1,3]=4\) are already assigned
  ▪ Thus, the only case to be considered is \([1,1]=2\)

\((1,1,2)\)
Variables

• Let $V$ be a set of variables

$$V = \bigcup_{r=1}^{N} \bigcup_{c=1}^{N} \bigcup_{v=1}^{N} \{(r, c, v) \mid [r, c] = \text{empty} \land \neg \text{affected}(r, c, v)\}$$

$$\text{affected}(r, c, v) = \text{sameRow}(r, c, v) \lor \text{sameCol}(r, c, v) \lor \text{sameBlock}(r, c, v)$$

$$\text{sameRow}(r, c, v) = \exists_{i:1..N} \cdot i \neq c \Rightarrow [r, i] = v$$

$$\text{sameCol}(r, c, v) = \exists_{i:1..N} \cdot i \neq r \Rightarrow [i, c] = v$$

$$\text{sameBlock}(r, c, v) = \exists_{i: \text{originRow..subN}} \cdot \exists_{i: \text{originCol..subN}} \cdot (i \neq r \land j \neq c) \Rightarrow [i, j] = v$$
Example

\[
V = \left\{\begin{array}{|c|c|}
(1,1,2) & (1,2,1) \\
(1,2,2) & \\
(2,1,2) & (2,2,1) \\
(2,1,3) & (2,2,2) \\
(2,1,4) & (2,2,4) \\
(2,3,1) & (2,3,2) \\
(2,4,1) & (2,4,2) \\
(3,1,2) & (3,2,1) \\
(3,1,4) & (3,2,2) \\
(3,1,4) & (3,2,4) \\
(3,3,1) & (3,3,2) \\
(3,3,3) & (3,4,1) \\
(3,3,3) & (3,4,2) \\
(3,4,4) & \\
1 & 3 \\
4 & 3 \\
\end{array}\right\}
\]

these parts are excluded
Cell Rule $\rightarrow$ CNF

A number appears once in each cell

There is **at least** one number in each cell  (definedness)

$$Cell_d' = \bigcup_{r=1}^{N} \bigcup_{c=1}^{N} \{ \bigvee_{v=1}^{N} (r, c, v) \mid (r, c, v) \in V \}$$

There is **at most** one number in each cell  (uniqueness)

$$Cell_u' = \bigcup_{r=1}^{N} \bigcup_{c=1}^{N} \bigcup_{v_i=1}^{N-1} \bigcup_{v_j=v_i+1}^{N} \{ \neg (r, c, v_i) \lor \neg (r, c, v_j) \mid (r, c, v_i) \in V \land r, c, v_j \in V \}$$
**Example**

<table>
<thead>
<tr>
<th>(1,1,2)</th>
<th>(1,2,1)</th>
<th>(1,2,2)</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,1,2)</td>
<td>(2,2,1)</td>
<td>(2,3,1)</td>
<td>(2,4,1)</td>
<td></td>
</tr>
<tr>
<td>(2,1,3)</td>
<td>(2,2,2)</td>
<td>(2,3,2)</td>
<td>(2,4,2)</td>
<td></td>
</tr>
<tr>
<td>(2,1,4)</td>
<td>(2,2,4)</td>
<td>(3,1,2)</td>
<td>(3,1,4)</td>
<td></td>
</tr>
<tr>
<td>(3,2,2)</td>
<td>(3,3,1)</td>
<td>(3,3,2)</td>
<td>(3,4,1)</td>
<td></td>
</tr>
<tr>
<td>(3,2,4)</td>
<td>(3,3,3)</td>
<td>(3,4,2)</td>
<td>(3,4,4)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>(4,3,2)</td>
<td>(4,4,2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4,3,2)</td>
<td>(4,4,4)</td>
<td></td>
</tr>
</tbody>
</table>

**Cell\textsubscript{d}' =**

\[
\{(1,1,2)\} \\
\{(1,2,1) \lor (1,2,2)\} \\
\{(2,1,2) \lor (2,1,3) \lor (2,1,4)\} \\
\{(2,2,1) \lor (2,2,2) \lor (2,2,4)\} \\
\ldots \\
\{(4,3,2)\} \\
\{(4,4,2) \lor (4,4,4)\}
\]

**Cell\textsubscript{u}' =**

\[
\{¬(1,2,1) \lor ¬(1,2,2)\} \\
\{¬(2,1,2) \lor ¬(2,1,3)\} \\
\{¬(2,1,2) \lor ¬(2,1,4)\} \\
\{¬(2,1,3) \lor ¬(2,1,4)\} \\
\ldots \\
\{¬(4,4,2) \lor ¬(4,4,4)\}
\]
Each number appears **at least** in each row (definedness)

\[ \text{Row}_d' = \bigcup_{r=1}^{N} \bigcup_{v=1}^{N} \{ \bigvee_{c=1}^{N} (r, c, v) \mid (r, c, v) \in V \} \]

Each number appears **at most** in each row (uniqueness)

\[ \text{Row}_u' = \bigcup_{r=1}^{N} \bigcup_{v=1}^{N} \bigcup_{c_i=1}^{N-1} \bigcup_{c_j=c_i+1}^{N} \{ \neg (r, c_i, v) \vee \neg (r, c_j, v) \}
\]

\[ \mid (r, c_i, v) \in V \wedge (r, c_j, v) \in V \} \]
### Example

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1,2)</td>
<td>(1,2,1)</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>(2,1,2)</td>
<td>(2,2,1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,1,3)</td>
<td>(2,2,2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,1,4)</td>
<td>(2,2,4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3,1,2)</td>
<td>(3,2,1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3,1,3)</td>
<td>(3,2,2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3,1,4)</td>
<td>(3,2,4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{Row}_d' = \{(1,2,1)\} \]
\[ \quad \{(1,1,2) \lor (1,2,2)\} \]
\[ \quad \{(2,2,1) \lor (2,3,1) \lor (2,4,1)\} \]
\[ \quad \{(2,1,2) \lor (2,2,2) \lor (2,3,2) \lor (2,4,2)\} \]
\[ \quad \ldots \ldots \]
\[ \quad \{(4,3,2) \lor (4,4,2)\} \]
\[ \quad \{(4,4,4)\} \]

\[ \text{Row}_u' = \{- (1,1,2) \land - (1,2,2)\} \]
\[ \quad \{- (2,2,1) \land - (2,3,1)\} \]
\[ \quad \{- (2,2,1) \land - (2,4,1)\} \]
\[ \quad \{- (2,3,1) \land - (2,4,1)\} \]
\[ \quad \ldots \ldots \]
\[ \quad \{- (4,3,2) \land - (4,4,2)\} \]
Column Rule $\rightarrow$ CNF

A number appears once in each column

Each number appears **at least** in each column (definedness)

$$Col_{d}' = \bigcup_{c=1}^{N} \bigcup_{v=1}^{N} \{ \bigvee_{r=1}^{N} (r, c, v) \mid (r, c, v) \in V \}$$

Each number appears **at most** in each column (uniqueness)

$$Col_{u}' = \bigcup_{c=1}^{N} \bigcup_{v=1}^{N} \bigcup_{r_i=1}^{N-1} \bigcup_{r_j=r_i+1}^{N} \{ \neg (r_i, c, v) \lor \neg (r_j, c, v) \mid (r_i, c, v) \in V \land (r_j, c, v) \in V \}$$
Example

\[
\begin{array}{cccc}
(1,1,2) & (1,2,1) & 4 & 3 \\
(2,1,2) & (2,2,1) & (2,3,1) & (2,4,1) \\
(2,1,3) & (2,2,2) & (2,3,2) & (2,4,2) \\
(2,1,4) & (2,2,4) & (2,3,4) & (2,4,4) \\
(3,1,2) & (3,2,1) & (3,3,1) & (3,4,1) \\
(3,1,3) & (3,2,2) & (3,3,2) & (3,4,2) \\
(3,1,4) & (3,2,4) & (3,3,4) & (3,4,4) \\
1 & 3 & (4,3,2) & (4,4,2) \\
& & (4,3,2) & (4,4,4) \\
\end{array}
\]

\begin{align*}
Col_d^\prime &= \{(1,1,2) \lor (2,1,2) \lor (3,1,2)\} \\
& \quad \{(2,1,3)\} \\
& \quad \{(2,1,4) \lor (3,1,4)\} \\
& \quad \ldots \\
& \quad \ldots \\
& \quad \ldots \\
& \quad \{(2,4,2) \lor (3,4,2) \lor (4,4,2)\} \\
& \quad \{(3,4,4) \lor (4,4,4)\} \\
\end{align*}

\begin{align*}
Col_u^\prime &= \{\neg(1,1,2) \lor \neg(2,1,2)\} \\
& \quad \{\neg(1,1,2) \lor \neg(3,1,2)\} \\
& \quad \{\neg(2,1,2) \lor \neg(3,1,2)\} \\
& \quad \{\neg(2,1,4) \lor \neg(3,1,4)\} \\
& \quad \ldots \\
& \quad \ldots \\
& \quad \ldots \\
& \quad \{\neg(3,4,4) \lor \neg(4,4,4)\} \\
\end{align*}
Each number appears **at least** in each block (definedness)

$$Block_d' = \bigcup_{r_{offs}=1}^{subN} \bigcup_{c_{offs}=1}^{subN} \bigcup_{v=1}^{N} \big\{ \bigvee_{r=1}^{subN} \bigwedge_{c=1}^{subN} (r_{offs} * subN + r, c_{offs} * subN + c, v) \big\} \setminus \big\{ (r_{offs} * subN + r, c_{offs} * subN + c, v) \in V \big\}$$

Each number appears **at most** in each block (uniqueness)

$$Block_u' = \bigcup_{r_{offs}=1}^{subN} \bigcup_{c_{offs}=1}^{subN} \bigcup_{v=1}^{N} \bigcup_{r=1}^{N} \bigcup_{c=r+1}^{N} \big\{ \neg (r_{offs} * subN + (r \mod subN), c_{offs} * subN + (r \mod subN), v) \big\} \bigvee \big\{ \neg (r_{offs} * subN + (c \mod subN), c_{offs} * subN + (c \mod subN), v) \big\} \big\setminus \big\{ (r_{offs} * subN + (r \mod subN), c_{offs} * subN + (r \mod subN), v) \in V \big\} \bigvee \big\{ (r_{offs} * subN + (c \mod subN), c_{offs} * subN + (c \mod subN), v) \in V \big\}$$
Example

<table>
<thead>
<tr>
<th></th>
<th>(1,2,1)</th>
<th>(1,2,2)</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1,2)</td>
<td>(1,2,1)</td>
<td>(1,2,2)</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>(2,1,2)</td>
<td>(2,1,3)</td>
<td>(2,2,1)</td>
<td>(2,3,1)</td>
<td>(2,4,1)</td>
</tr>
<tr>
<td>(2,1,3)</td>
<td>(2,2,2)</td>
<td>(2,2,4)</td>
<td>(2,3,2)</td>
<td>(2,4,2)</td>
</tr>
<tr>
<td>(2,1,4)</td>
<td>(2,2,4)</td>
<td>(2,3,2)</td>
<td>(2,4,1)</td>
<td>(2,4,2)</td>
</tr>
<tr>
<td>(3,1,2)</td>
<td>(3,2,2)</td>
<td>(3,3,1)</td>
<td>(3,4,1)</td>
<td>(3,4,2)</td>
</tr>
<tr>
<td>(3,1,4)</td>
<td>(3,2,4)</td>
<td>(3,3,2)</td>
<td>(3,4,1)</td>
<td>(3,4,2)</td>
</tr>
<tr>
<td></td>
<td>(3,2,4)</td>
<td>(3,3,3)</td>
<td>(3,4,3)</td>
<td>(3,4,4)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>(4,3,2)</td>
<td>(4,4,2)</td>
<td>(4,4,4)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(4,4,2)</td>
<td>(4,4,4)</td>
</tr>
</tbody>
</table>

\[
\text{Block}_d' = \begin{cases} 
(1,2,1) \lor (2,2,1) \\
(1,1,2) \lor (1,2,2) \lor (2,1,2) \lor (2,2,2) \\
(2,1,3) \\
(3,3,3) \\
(3,4,4) \lor (4,4,4)
\end{cases}
\]

\[
\text{Block}_u' = \begin{cases} 
\neg(1,2,1) \lor \neg(2,2,1) \\
\neg(1,1,2) \lor \neg(1,2,2) \\
\neg(1,1,2) \lor \neg(2,1,2) \\
\neg(1,1,2) \lor \neg(2,2,2) \\
\neg(3,4,4) \lor \neg(4,4,4)
\end{cases}
\]
Optimized Encoding

The resulting CNF formula

\[ \phi = \text{Cell}_d \cup \text{Cell}_u \cup \text{Row}_d \cup \text{Row}_u \cup \text{Col}_d \cup \text{Col}_u \cup \text{Block}_d \cup \text{Block}_u \]

\( \phi \) is **satisfiable** iff Sudoku has a **solution**

**Smaller** variables and clauses than previous encodings

Number of variables are reduced 12 times on average in our experiments

Number of clauses are reduced 79 times on average in our experiments
Agenda

• Introduction

• Background and Previous Encodings

• Optimized Encoding

• Experimental Results

• Conclusions
# Experimental Results

<table>
<thead>
<tr>
<th>size</th>
<th>level</th>
<th>vars</th>
<th>clauses</th>
<th>time</th>
<th>size</th>
<th>level</th>
<th>vars</th>
<th>clauses</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>9x9</td>
<td>easy</td>
<td>729</td>
<td>12013</td>
<td>0.00</td>
<td>9x9</td>
<td>hard</td>
<td>729</td>
<td>12018</td>
<td>0.00</td>
</tr>
<tr>
<td>9x9</td>
<td>hard</td>
<td>729</td>
<td>12018</td>
<td>0.00</td>
<td>9x9</td>
<td>easy</td>
<td>729</td>
<td>12013</td>
<td>0.00</td>
</tr>
<tr>
<td>16x16</td>
<td>easy</td>
<td>4096</td>
<td>124008</td>
<td>0.01</td>
<td>16x16</td>
<td>hard</td>
<td>4096</td>
<td>124002</td>
<td>0.01</td>
</tr>
<tr>
<td>16x16</td>
<td>hard</td>
<td>4096</td>
<td>124002</td>
<td>0.01</td>
<td>16x16</td>
<td>easy</td>
<td>4096</td>
<td>124008</td>
<td>0.01</td>
</tr>
<tr>
<td>25x25</td>
<td>easy</td>
<td>15625</td>
<td>752792</td>
<td>0.07</td>
<td>25x25</td>
<td>hard</td>
<td>15625</td>
<td>752778</td>
<td>0.21</td>
</tr>
<tr>
<td>25x25</td>
<td>hard</td>
<td>15625</td>
<td>752778</td>
<td>0.21</td>
<td>25x25</td>
<td>easy</td>
<td>15625</td>
<td>752792</td>
<td>0.07</td>
</tr>
<tr>
<td>36x36</td>
<td>easy</td>
<td>46656</td>
<td>3271748</td>
<td>0.50</td>
<td>36x36</td>
<td>hard</td>
<td>46656</td>
<td>3271768</td>
<td>0.67</td>
</tr>
<tr>
<td>36x36</td>
<td>hard</td>
<td>46656</td>
<td>3271768</td>
<td>0.67</td>
<td>36x36</td>
<td>easy</td>
<td>46656</td>
<td>3271748</td>
<td>0.50</td>
</tr>
<tr>
<td>49x49</td>
<td>easy</td>
<td>117649</td>
<td>11305189</td>
<td>1.47</td>
<td>49x49</td>
<td>hard</td>
<td>117649</td>
<td>11305189</td>
<td>1.47</td>
</tr>
<tr>
<td>64x64</td>
<td>easy</td>
<td>262144</td>
<td>33048912</td>
<td>stack</td>
<td>64x64</td>
<td>hard</td>
<td>262144</td>
<td>33048912</td>
<td>stack</td>
</tr>
<tr>
<td>81x81</td>
<td>easy</td>
<td>531441</td>
<td>85060787</td>
<td>stack</td>
<td>81x81</td>
<td>hard</td>
<td>531441</td>
<td>85060787</td>
<td>stack</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>extended encoding</th>
<th>proposed encoding</th>
<th>analysis of pre-assigned cells</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>k</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>104</td>
</tr>
<tr>
<td></td>
<td></td>
<td>98</td>
</tr>
<tr>
<td></td>
<td></td>
<td>292</td>
</tr>
<tr>
<td></td>
<td></td>
<td>278</td>
</tr>
<tr>
<td></td>
<td></td>
<td>644</td>
</tr>
<tr>
<td></td>
<td></td>
<td>664</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1282</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2384</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3983</td>
</tr>
</tbody>
</table>
81x81 Puzzle

Variables are reduced 30 times

Clauses are reduced 320 times
Variable Reduction

The chart shows the comparison of variable reduction for different sizes (9x9, 16x16, 25x25, 36x36, 49x49, 64x64, 81x81) between an extended method and a proposed method. The extended method reduces a significantly larger number of variables compared to the proposed method, especially at larger sizes.
Clause Reduction

![Graph showing clause reduction comparison between extended and proposed methods across different sizes.](image)
Time Reduction

The graph shows the time reduction for different sizes. The x-axis represents the size of the input, and the y-axis represents the time in seconds. The bars are divided into two categories: extended and proposed. The extended method shows a significant increase in time for larger sizes, whereas the proposed method maintains a more consistent performance across all sizes.
Variable Reduction Ratio

The graph shows the relationship between variable reduction and the percentage k (%). The variable reduction increases significantly as k (%) increases from 32% to 61%.
Clause Reduction Ratio

![Graph showing the clause reduction ratio as a function of k (%)](image)
Agenda

• Introduction

• Background and Previous Encodings

• Optimized Encoding

• Experimental Results

• Conclusions
Conclusions

Previous encodings

J. Ouaknine, Sudoku as a SAT Problem, 2006
T. Weber, A SAT-based Sudoku Solver, 2005

Props and cons

+ Ideal encoding techniques
+ Well used for small puzzles
  – Too many clauses
  – Hard to handle large size puzzles such as 81x81
Conclusions

Proposed techniques

- Optimized encoding used to reduce a formula

Results from 11 different size puzzles

- All given puzzles are successfully solved
- Number of variables is greatly reduced
- Number of clauses is greatly reduced
- Execution time is greatly reduced
- Finally, encoding time is greatly reduced

Thank You!!