Foundations of AI

8. Satisfiability and Model Construction

Davis-Putnam, Phase Transitions, GSAT and GWSAT

Wolfram Burgard & Bernhard Nebel
The Davis-Putnam Procedure

DP Function

Given a set of clauses \( \Delta \) defined over a set of variables \( \Sigma \), return “satisfiable” if \( \Delta \) is satisfiable. Otherwise return “unsatisfiable”.

1. If \( \Delta = \emptyset \) return “satisfiable”
2. If \( \square \in \Delta \) return “unsatisfiable”
3. **Unit-propagation Rule**: If \( \Delta \) contains a unit-clause \( C \), assign a truth-value to the variable in \( C \) that satisfies \( C \), simplify \( \Delta \) to \( \Delta' \) and return \( \text{DP}(\Delta') \).
4. **Splitting Rule**: Select from \( \Sigma \) a variable \( \nu \) which has not been assigned a truth-value. Assign one truth value \( t \) to it, simplify \( \Delta \) to \( \Delta' \) and call \( \text{DP}(\Delta') \)
   a. If the call returns “satisfiable”, then return “satisfiable”
   b. Otherwise assign the other truth-value to \( \nu \) in \( \Delta \), simplify to \( \Delta'' \) and return \( \text{DP}(\Delta'') \).
Example (1)

$$\Delta = \{\{a, b, \neg c\}, \{\neg a, \neg b\}, \{c\}, \{a, \neg b\}\}$$
Example (2)

\[ \Delta = \{ \{a, \neg b, \neg c, \neg d\}, \{b, \neg d\}, \{c, \neg d\}, \{d\} \} \]
Properties of DP

- DP is complete, correct, and guaranteed to terminate.
- DP constructs a model, if one exists.
- In general, DP requires exponential time (splitting rule!)
- DP is polynomial on horn clauses, i.e. clauses with at most one positive literal.
  \((\neg A_1 \lor \ldots \lor \neg A_n \lor B \equiv \land_i A_i \Rightarrow B)\)
  
  → **Heuristics** are needed to determine which variable should be instantiated next and which value should be used

  → In all SAT competitions so far, DP-based procedures have shown the best performance.
Local Search Methods for Solving Logical Problems

In many cases, we are interested in finding a satisfying assignment of variables (example CSP), and we can sacrifice completeness if we can “solve” much large instances this way.

Standard process for optimization problems: Local Search

- Based on a (random) configuration
- Through local modifications, we hope to produce better configurations

→ Main problem: local maxima
Dealing with Local Maxima

As a measure of the value of a configuration in a logical problem, we could use the number of satisfied constraints/clauses.

But local search seems inappropriate, considering we want to find a global maximum (all constraints/clauses satisfied).

By restarting and/or injecting noise, we can often escape local maxima.

Actually: Local search performs very well for finding satisfying assignments of CNF formulae (even without injecting noise).
GSAT

Procedure GSAT

INPUT: a set of clauses \( \alpha \), MAX-FLIPS, and MAX-TRYES

OUTPUT: a satisfying truth assignment of \( \alpha \), if found

begin
  for \( i := 1 \) to MAX-TRYES
    \( T := \) a randomly-generated truth assignment
    for \( j := 1 \) to MAX-FLIPS
      if \( T \) satisfies \( \alpha \) then return \( T \)
      \( \nu := \) a propositional variable such that a change in its truth assignment gives the largest increase in the number of clauses of \( \alpha \) that are satisfied by \( T \).
      \( T := T \) with the truth assignment of \( \nu \) reversed
    end for
  end for
return “no satisfying assignment found”
end
The Search Behavior of GSAT

- In contrast to normal local search methods, we must also allow sideways movements!
- Most time is spent searching on plateaus.
### Application of GSAT

<table>
<thead>
<tr>
<th>formulas</th>
<th>GSAT</th>
<th>DP</th>
</tr>
</thead>
<tbody>
<tr>
<td>vars</td>
<td>clauses</td>
<td>M-FLIPS</td>
</tr>
<tr>
<td>50</td>
<td>215</td>
<td>250</td>
</tr>
<tr>
<td>100</td>
<td>430</td>
<td>500</td>
</tr>
<tr>
<td>140</td>
<td>602</td>
<td>700</td>
</tr>
<tr>
<td>150</td>
<td>645</td>
<td>1500</td>
</tr>
<tr>
<td>300</td>
<td>1275</td>
<td>6000</td>
</tr>
<tr>
<td>500</td>
<td>2150</td>
<td>10000</td>
</tr>
</tbody>
</table>
GSAT + Noise

In order to escape from plateaus, we can inject noise.

1. **Simulated annealing (cooling)**

2. **Random walk:**
   With probability $p$, choose a variable in a clause that is still unsatisfied, and change its assignment. With probability $1 - p$, use the GSAT strategy.

3. **Random noise:**
   Like random walk, but choose any clause (can be already satisfied).
Results on Random-3CNF Formulae

- Time in CPU-seconds on an SGI Challenge 100 MHz
- All values are given for the best parameter assignment (re starts, \( p \)), where \textbf{walk} and \textbf{noise} require no restarts.
- **“*”** means more than 1000 restarts or more than 20 CPU-hours.
Results on Circuit Design Problems

Circuit design problems for random circuits to test integer programming approaches.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ids</td>
<td>clauses</td>
<td>time</td>
<td>time</td>
<td>flips</td>
</tr>
<tr>
<td>f16a1</td>
<td>1650</td>
<td>2039</td>
<td>114</td>
<td>709895</td>
</tr>
<tr>
<td>f16b1</td>
<td>1728</td>
<td>78</td>
<td>452</td>
<td>2870019</td>
</tr>
<tr>
<td>f16c1</td>
<td>1580</td>
<td>758</td>
<td>3.5</td>
<td>12178</td>
</tr>
<tr>
<td>f16d1</td>
<td>1230</td>
<td>1547</td>
<td>174</td>
<td>872219</td>
</tr>
<tr>
<td>f16e1</td>
<td>1245</td>
<td>2156</td>
<td>1.7</td>
<td>2090</td>
</tr>
</tbody>
</table>

Similar results for other circuit design and diagnosis problems.
State of the Art

• SAT competitions since beginning of the ´90
• Current SAT competitions ([http://www.satlive.org/](http://www.satlive.org/)):
  – 2002:
    • Largest solved instances:
      – 100,000 variables
      – 1,000,000 clauses
    • Smallest unsolved instances:
      – 100 variables
      – 1,000 clause
Concluding Remarks

- Main problem: **Determination of the parameters** MAX-TRIES, \( p \) … but seems connected to problem class.
- Second problem: Cannot find an assignment (that exists) under certain circumstances → incompleteness regarding satisfiability.
  → Area of application: Only for use with very large problem instances for which it is worth determining parameters.
- GWSAT seems to do better than GSAT.
- Tabu-search probably does better than GWSAT – some authors say.
  → At the limit of what can be handled, a bit of “black magic” is required.
  → Local search seems to model our “intuition”.