Foundations of Artificial Intelligence

CS472/3
Lecture #7

Bart Selman

Slide CS472–1

Today’s Lecture

Local Search, Cont.
GSAT, Simulated Annealing,
Readings: R&N, Chapter 4.

Slide CS472–2
Local Search / Hillclimbing

function HILL-CLIMBING(problem) returns a solution state
inputs: problem, a problem
static: current, a node
next, a node

current ← MAKE-NODE(INITIAL-STATE(problem))
loop do
next ← a highest-valued successor of current
if VALUE(next) < VALUE(current) then return current
current ← next
end

Slide CS472–3

Notes
“successor” normally called “neighbor”
You search in the neighborhood of current state.
When does Hill-Climbing stop?
Current approaches: Often simply keep going!
Use: time limit.
Simple method: very fast, uses minimal memory, surprisingly effective!
10,000+ variables, 1,000,000+ constraints.

Slide CS472–4
Example

A wide variety of key CS problems can be translated into a propositional logical formalization

e.g., \((A \lor B \lor C) \land (\neg B \land C \lor D) \land (A \lor \neg C \lor D)\)

and solved by finding a truth assignment to the propositional variables \((A, B, C,\ldots)\) that makes it true, i.e., a model.

If a formula has a model, we say that it is “satisfiable”.

Special kind of CSP.

Applications:

planning and scheduling

circuit diagnosis and synthesis
deductive reasoning
software testing
etc.

Slide CS472–5

Slide CS472–6
Satisfiability Testing

Best-known method: Davis-Putnam Procedure (1960)

Backtrack search (DFS) through the space of truth assignments (with unit-propagation).

Slide CS472–7

\[
(A \lor C) \land (\neg A \lor C) \land (B \lor \neg C) \land (A \lor \neg B)
\]

\[
\begin{array}{c}
C \land (B \lor \neg C) \land \neg B \\
C \land (B \lor \neg C)
\end{array}
\]

Slide CS472–8
To date, Davis-Putnam still the *fastest* sound and *complete* method.

However, there are classes of formulas where the procedure scales badly.

Consider an *incomplete* local search procedure.

---

**Greedy Local Search — GSAT**

Begin with a random truth assignment (assume CNF).

Flip the value assigned to the variable that yields the *greatest number of satisfied clauses*.

(Note: Flip even if there is no improvement.)

Repeat until a model is found, or have performed a specified maximum number of flips.

If a model is still not found, repeat the entire process, starting from a different initial random assignment.
\begin{tabular}{|c c c| c c c| c|}
\hline
$A$ & $B$ & $C$ & $(A \lor C)$ & $\land$ & $(\neg A \lor C)$ & $\land$ & $(B \lor \neg C)$ & \text{Score} \\
\hline
F & F & F & $\times$ & $\checkmark$ & $\checkmark$ & & 2 \\
F & F & T & $\checkmark$ & $\checkmark$ & $\times$ & & 2 \\
F & T & T & $\checkmark$ & $\checkmark$ & $\checkmark$ & & 3 \\
\hline
\end{tabular}

(Selman, Levesque, and Mitchell 1992)

Slide CS472–11

How well does it work?

First intuition: It will get \textbf{stuck} in local minima, with a few unsatisfied clauses.

Note we are not interested in \textbf{almost} satisfying assignments
\textit{E.g.}, a plan with one “magic” step is useless.
Contrast with optimization problems.

Surprise: It often finds \textbf{global} minimum!
\textit{I.e.}, finds satisfying assignments.

Slide CS472–12
<table>
<thead>
<tr>
<th>form. vars</th>
<th>GSAT m.flips</th>
<th>retries</th>
<th>time</th>
<th>Davis-Putnam choices</th>
<th>depth</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>250</td>
<td>6</td>
<td>0.5 sec</td>
<td>77</td>
<td>11</td>
<td>1 sec</td>
</tr>
<tr>
<td>70</td>
<td>350</td>
<td>11</td>
<td>1 sec</td>
<td>42</td>
<td>15</td>
<td>15 sec</td>
</tr>
<tr>
<td>100</td>
<td>500</td>
<td>42</td>
<td>6 sec</td>
<td>$10^3$</td>
<td>19</td>
<td>3 min</td>
</tr>
<tr>
<td>120</td>
<td>600</td>
<td>82</td>
<td>14 sec</td>
<td>$10^5$</td>
<td>22</td>
<td>18 min</td>
</tr>
<tr>
<td>140</td>
<td>700</td>
<td>53</td>
<td>14 sec</td>
<td>$10^6$</td>
<td>27</td>
<td>5 hrs</td>
</tr>
<tr>
<td>150</td>
<td>1500</td>
<td>100</td>
<td>45 sec</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>200</td>
<td>2000</td>
<td>248</td>
<td>3 min</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>300</td>
<td>6000</td>
<td>232</td>
<td>12 min</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>500</td>
<td>10000</td>
<td>996</td>
<td>2 hrs</td>
<td>$10^{30}$</td>
<td>&gt; 100</td>
<td>$10^{19}$ yrs</td>
</tr>
</tbody>
</table>

Slide CS472–13

Search space

Slide CS472–14
**Improvements to Basic Local Search**

Issue: How to move more quickly to successively lower plateaus?

Avoid getting “stuck” / local minima.

Idea: introduce uphill moves (“noise”) to escape from long plateaus (or true local minima)

Noise strategies:

a) **Simulated Annealing**
   
   Kirkpatrick *et al.* 1982; Metropolis *et al.* 1953

b) **Mixed Random Walk**
   
   Selman and Kautz 1993

---

**Slide CS472–15**

---

**Simulated Annealing**

- Noise model based on statistical mechanics.
- Pick a random variable
  
  - If flip improves assignment: do it.
  - else flip with probability $p = e^{-\delta/T}$ (“upward”).
  - $\delta$ number of additional clauses becoming *unsatisfied*
  - $T$ = “temperature”
  
  Higher temperature = greater likelihood of upward moves.

Slowly decrease $T$ from high temperature to near zero.

What is $p$ for $T \to \inf$? For $T \to 0$? For $\delta = 0$?

---

**Slide CS472–16**
Sim. Annealing introduced as analogue to a **physical process**
Way to grow crystals.
Kirkpatrick *et al.* 1982; Metropolis *et al.* 1953
Physical analogy remains elusive.
Can prove that with exponential schedule will converge to
global optimum.
Difficult to be more precise about convergence rate.
See recent work on rapidly mixing Markov chains.
Key aspect: **upwards moves / sideways moves.**
Expensive, but if you have time can be best.
(hundreds of papers per year / many applications)

---

**Random Walk**

Random walk SAT algorithm:

1. *Pick random truth assignment.*
2. *Repeat until all clauses satisfied:*
   - *Flip variable from any unsatisfied clause.*

- Solves 2-SAT (2 variables per clause) in $O(n^2)$ flips.
  - (Papadimitriou 1992) Why limited value?
- Does not work at all for hard $k$-SAT ($k \geq 3$).
Mixing Random Walk with Greedy Local Search

• With probability $p$, walk,
  i.e., pick a variable in some
  unsatisfied clause and flip it;
  with probability $(1 - p)$ make a greedy flip,
  i.e., one that makes greatest decrease in number of
  unsatisfied clauses.

• Value for parameter $p$ determined empirically,
  by finding best setting for a problem class.

Experimental Results: Hard Random 3CNF

<table>
<thead>
<tr>
<th>vars</th>
<th>GSAT basic</th>
<th></th>
<th>GSAT walk</th>
<th></th>
<th>Simul. Ann.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>eff.</td>
<td></td>
<td>time</td>
<td>eff.</td>
</tr>
<tr>
<td>100</td>
<td>.4</td>
<td>.12</td>
<td></td>
<td>.2</td>
<td>1.0</td>
</tr>
<tr>
<td>200</td>
<td>22</td>
<td>.01</td>
<td></td>
<td>4</td>
<td>.97</td>
</tr>
<tr>
<td>400</td>
<td>122</td>
<td>.02</td>
<td></td>
<td>7</td>
<td>.95</td>
</tr>
<tr>
<td>600</td>
<td>1471</td>
<td>.01</td>
<td></td>
<td>35</td>
<td>1.0</td>
</tr>
<tr>
<td>800</td>
<td>*</td>
<td>*</td>
<td></td>
<td>286</td>
<td>.95</td>
</tr>
<tr>
<td>1000</td>
<td>*</td>
<td>*</td>
<td></td>
<td>1095</td>
<td>.85</td>
</tr>
<tr>
<td>2000</td>
<td>*</td>
<td>*</td>
<td></td>
<td>3255</td>
<td>.95</td>
</tr>
</tbody>
</table>

Slide CS472–19

Slide CS472–20
• Time in seconds (SGI Challenge).
• Effectiveness: prob. that random initial assignment leads to a solution.
• Complete methods, such as DP, up to 400 variables.
• *Mixed Walk better than*
  *Simul. Ann. better than*
    *Basic GSAT better than*
      *Backtracking (Davis-Putnam).*