Application: Focused Search
Constraint Based Systems

Reading: Chapter 6
Chess article (see links)
My topic is Computer Science. My collection might look like:

- http://liinwww.ira.uka.de/bibliography/
- http://www.lcs.mit.edu/
- http://www.cs.cmu.edu/
- http://www.acm.org
Goal test = classifier

- Given a new web page, is it relevant to my topic?
  - www.ncstrl.org → yes
  - http://www.metmuseum.org → no

- How does it work?
  - Give it an example set of documents: training set
  - Using machine learning techniques, learn from the set of training examples, rules that can be applied to unseen data
The Crawl

- Start at a random page in the topic collection
  - [http://www.acm.org](http://www.acm.org)

- What is the successor-fn?

- What AI search algorithm should we use?
What heuristics should we use?

- Can the link itself tell us?
- One step away and test
- Hubs
  - A page that has many links to other sites
- Authorities
  - A page that has many incoming links
  - Especially incoming links from other authoritative sites
Identifying hubs and authorities

- Each node has two scores, iteratively determined
  - $a(v)$: number of incoming edges from relevant nodes
  - $h(v)$: number of outgoing edges to relevant nodes
- Weight these scores by the relevance scores of the pages they point to (a probability between 0 and 1)
Constraint Satisfaction Problems

Standard representation allows a general-purpose approach to search

- Set of **variables**, $X_1, X_2 \ldots X_n$
  - Each variable has domain $D_i$ of possible values
- Set of **constraints**, $C_1, C_2, \ldots C_n$
- **State** = assignment of values to some or all variables
  - $X_i=v_i$ , $X_j=v_j$
- **Solution** = complete assignment that satisfies all constraints
- Some solutions maximize an **objective function**
Cryptograms

QFL HCVPS
THE BRAIN

PX V ANSWLCEZK NCJVS; PQ XQVCQX
IS A WONDERFUL ORGAN; IT STARTS

QFL BPSZQL RNZ JLQ ZT PS QFL BNCSPSJ
THE MINUTE YOU GET UP IN THE MORNING

VSW WNLX SNQ XQNT ZSQPK RNZ JLQ QN
AND DOES NOT STOP UNTIL YOU GET TO

QFL NEEPGL
THE OFFICE

ROBERT FROST
**CSP algorithm**

*Depth-first search often used*

- **Initial state**: the empty assignment `{}`; all variables are unassigned
- **Successor fn**: assign a value to any variable, provided no conflicts w/ constraints
  - All CSP search algorithms generate successors by considering possible assignments for only a single variable at each node in the search tree
- **Goal test**: the current assignment is complete
- **Path cost**: a constant cost for every step
Local search

- Complete-state formulation
  - Every state is a complete assignment that might or might not satisfy the constraints

- Hill-climbing methods are appropriate
Dimensions of CSPs

- Discrete/finite domains
  - Map coloring
  - 8 queens

- Nonlinear constraints
  - No general solution

- Unary constraints

- Absolute constraints

- Infinite domains
  - Scheduling calendars (discrete)
  - Schedule experiments for Hubble

- Linear constraints

- Binary/N-ary constraints

- Preferences
  - Prof. McKeown prefers after 1pm
Backtracking search

function BACKTRACKING-SEARCH(csp) returns solution/failure
  return RECURSIVE-BACKTRACKING([], csp)

function RECURSIVE-BACKTRACKING(assigned, csp) returns solution/failure
  if assigned is complete then return assigned
  var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assigned, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assigned, csp) do
    if value is consistent with assigned according to CONSTRAINTS[csp] then
      result ← RECURSIVE-BACKTRACKING([var = value | assigned], csp)
      if result ≠ failure then return result
  end
  return failure
M GWD’X URNMRPR MD QD
QCXLINMACR, QNXFWZOF M QH
ULMDOMDO Q KFQDOR WC
ZDGRLARQL.

AWWGS QNNRD
M GWD’X URNMRPR MD QD
A BCD’E A AD D
QCXRLNMCR, QNXFWZOF M QH
E A E
ULMDOMDO Q KFQDOR WC
AD AD D C
ZDGLARQL.

DB

AWWGS QNNRD
B D
M GWD’X URNMRPR MD QD
A BCD’I A AD D
QCXRLNMCR, QNXFWZOF M QH
I A I
ULMDOMDO Q KFQDOR WC
AD AD D C
ZDGLARQL.
DB
AWWGS QNNRD
B D
M GWD’X URNMRPR MD QD
A BCD’O A AD D
QCXRLNMCR, QNXFWZOF M QH
O A O
ULMDOMDO Q KFQDOR WC
AD AD D C
ZDGRLARQL.
DB

AWWGS QNNRD
B D
M GWD’X URNMRPR MD QD
A BCD’U A AD D
QCXRLNMCR, QNXFWZOF M QH
U A U
ULMDOMDO Q KFQDOR WC
AD AD D C
ZDGRLARQL.
DB
AWWGS QNNRD
B D
What additional knowledge about language would be helpful?
General purpose methods for efficient implementation

- Which variable should be assigned next?
- in what order should its values be tried?
- Can we detect inevitable failure early?
- Can we take advantage of problem structure?
Order

- Choose the most constrained variable first
  - The variable with the fewest remaining values
  - Minimum Remaining Values (MRV) heuristic

- What if there are >1?
  - Tie breaker: Most constraining variable
  - Choose the variable with the most constraints on remaining variables
Order on value choice

- Given a variable, chose the least constraining value
  - The value that rules out the fewest values in the remaining variables
M GWD’X URNMRPR MD QD
I N I IN AN
QCXRLNMCR, QNXFWZOF M QH
A I A I A
ULMDOMDO Q KFQDOR WC
IN IN A AN
ZDGRLARQL.
N A

AWWGS QNNRD
A N
Forward Checking

- Keep track of remaining legal values for unassigned variables
- Terminate search when any variable has no legal values
M GWD’X URNMRPR MD QD
I I I A
QCXRLNMCR, QNXFWZOF M QH
A I A I A
ULMDOMDO Q KFQDOR WC
I I A A
ZDGRLARQL.
A

When M I selected D (F N S T)
When QA selected D (N S T)
I DON’T BELIEVE IN ANNUAL QCXRLNMCR, QNXFWZOF MD QH
AFTERLIFE ALTHOUGH I AM ULMDOMDO Q KFQDOR WC
RINGING A HANGOVER OF ZDGLARQL.
NDERWEAR

AWWGS QNNRD
WOODY ALLEN
I DON’T BELIEVE IN AN UNDERWEAR.

Select UC -> the domain of P is empty. Requires backtracking even before taking the next branch. Avoids selection Z, K next before discovering the error.
I don’t believe in an QCXRLNMCR, QNXFWZOOF M QH A FTERL I FE ALTHO UGH I AM ULMDOMDO Q KFQDOR WC BR IN G I NG A CHANGE OF ZDGRLARQL. UNDERWEAR AWWGS QNNRD WOODY ALLEN
## Speed-ups

<table>
<thead>
<tr>
<th>Problem</th>
<th>Backtracking</th>
<th>BT+MRV</th>
<th>Forward Checking</th>
<th>FV+MRV</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA</td>
<td>(&gt;1000K)</td>
<td>(&gt;1000K)</td>
<td>2K</td>
<td>60</td>
</tr>
<tr>
<td>N-Queens</td>
<td>(&gt;40000K)</td>
<td>13,500K</td>
<td>(&gt;40000K)</td>
<td>817K</td>
</tr>
<tr>
<td>Zebra</td>
<td>3,859K</td>
<td>1K</td>
<td>35K</td>
<td>.5K</td>
</tr>
<tr>
<td>Random1</td>
<td>415K</td>
<td>3K</td>
<td>26K</td>
<td>2K</td>
</tr>
<tr>
<td>Random2</td>
<td>942K</td>
<td>27K</td>
<td>77K</td>
<td>15K</td>
</tr>
</tbody>
</table>
**Other types of improvements**

- **Constraint propagation**
  - Propagate implications of a constraint on one variable onto other variables. Not only values, but constraints between other variables.

- **Intelligent backtracking**
  - Conflict-directed backtracking: Save possible conflicts for a variable value on assignment.
  - Back jump to assignment of values.
Crossword Puzzles


\begin{align*}
\text{slot 1A} & : v & p & q & q^{(\infty)} \\
\text{AS} & : .5 & .250 & .190 \\
\text{IN} & : .3 & .617 & .645 \\
\text{IS} & : .2 & .133 & .165 \\
\text{slot 3A} & : v & p & q & q^{(\infty)} \\
\text{FUN} & : .7 & .350 & .314 \\
\text{TAD} & : .3 & .650 & .686 \\
\end{align*}

\begin{align*}
\text{slot 1D} & : v & p & q & q^{(\infty)} \\
\text{IT} & : .4 & .400 & .496 \\
\text{IF} & : .3 & .350 & .314 \\
\text{AT} & : .3 & .250 & .190 \\
\text{slot 2D} & : v & p & q & q^{(\infty)} \\
\text{NAG} & : .4 & .267 & .331 \\
\text{SAG} & : .3 & .383 & .355 \\
\text{NUT} & : .3 & .350 & .314 \\
\end{align*}

\begin{align*}
\text{slot 5A} & : \\
\text{slot 4D} & : \\
\end{align*}