Constraint Satisfaction

CSE 473
Oren Etzioni
University of Washington
Question from last time.

- What’s the relationship between Uniform Cost search and Dijkstra’s algorithm?
- Answer: essentially the same.
- Paradox: Dijkstra’s algorithm is “fast” $O(e \cdot \log n)$ and search is “slow” $O(b^d)$, but they are the same...how can that be?
Today: Constraint Satisfaction Problems

Definition
Factoring state spaces

• Backtracking policies
• Variable-ordering heuristics
• Preprocessing algorithms
Constraint Satisfaction

• Kind of search in which
  States are factored into sets of variables
  Search = assigning values to these variables
  Structure of space is encoded with constraints

• Backtracking-style algorithms work

• But other techniques add speed
  Propagation
  Variable ordering
  Preprocessing
Chinese Food as Search?

• States?
  • Partially specified meals

• Operators?
  • Add, remove, change dishes

• Start state?
  • Null meal

• Goal states?
  • Meal meeting certain conditions (rating?)
Factoring States

• Rather than state = meal
• Model state’s (independent) parts, e.g.
  
  Suppose every meal for n people
  Has n dishes plus soup

  Soup =
  Meal 1 =
  Meal 2 =

... Meal n =

• Or... physical state =
  X coordinate =
  Y coordinate =
Chinese Constraint Network

Soup
  - Must be Hot&Sour

Appetizer

Chicken Dish
  - No Peanuts

Pork Dish

Vegetable
  - No Peanuts

Total Cost < $30

Seafood

Rice
  - Not Both Spicy

Not Both Spicy

Not Chow Mein
CSPs in the Real World

- Scheduling space shuttle repair
- Airport gate assignments
- Transportation Planning
- Supply-chain management
- Computer configuration
- Diagnosis
- UI optimization
- Etc...
Binary Constraint Network

- Any FD constraints can be reduced to ‘binary’
- Set of n variables: \( x_1 \ldots x_n \)
- Value domains for each variable: \( D_1 \ldots D_n \)
- Set of binary constraints (also “relations”)
  \[
  R_{ij} \subseteq D_i \times D_j
  \]
  Specifies which values pair \((x_i, x_j)\) are consistent

- \( V \) for each country
- Each domain = 4 colors
- \( R_{ij} \) enforces \( \neq \)
Binary Constraint Network

Partial assignment of values = tuple of pairs

\{...(x, a)...\} means variable x gets value a...

Tuple = consistent if all constraints satisfied

Tuple = full solution if consistent + has all vars

Tuple \{((x_i, a_i) ... (x_j, a_j)) = consistent w/ a set of vars \{x_m ... x_n\}

iff \exists a_m ... a_n such that

\{((x_i, a_i)...(x_j, a_j), (x_m, a_m)...(x_n, a_n)) \} = consistent
Difficulty of CSP

• How hard is it to solve a Boolean CSP? (so we why do we care about algorithms?)

• CSPs can have infinite domains (e.g., integers)

• Linear programming = integer domain + linear constraints.
Cryptarithmetic

- State Space
  - Set of states
  - Operators [and costs]
  - Start state
  - Goal states

- Variables?
- Domains (variable values)?
- Constraints? (alldiff(..) can be represented as binary inequalities!)
Classroom Scheduling

- Variables?
- Domains (possible values for variables)?
- Constraints?
N Queens

• *As a CSP?
N Queens

- Variables = board columns
- Domain values = rows

\{(x_1, 2), (x_2, 4), (x_3, 1)\} consistent with \(x_4\)
- Shorthand: “\{2, 4, 1\} consistent with \(x_4\)"
**CSP as a search problem?**

- **What are states?**  
  
  *(N variables)*

- **What are the operators?**  
  
  *(assign D possible values)*

- **What is the branching factor?**  
  
  \(O(N \times D)\)

- **Commutative** → **D**

- **Initial state?**

- **Goal test?**

```
 Q
 Q
 Q
```
Chronological Backtracking (BT)
(e.g., depth first search)

Consistency check performed in the order in which vars were instantiated. If c-check fails, try next value of current var. If no more values, backtrack to most recent var.
Backjumping (BJ)

- Similar to BT, but more efficient when no consistent instantiation can be found for the current var
- Instead of backtracking to most recent var...
  BJ reverts to deepest var which was c-checked against the current var

BJ Discovers
(2, 5, 3, 6) inconsistent with $x_6$
No sense trying other values of $x_5$
Conflict-Directed Backjumping (CBJ)

- More sophisticated backjumping behavior
- Each variable has conflict set CS
  Set of vars that failed c-checks w/ current val
  Update this set on every failed c-check
- When no more values to try for \( x_i \)
  Backtrack to deepest var, \( x_d \), in CS(\( x_i \))

\textbf{And} update CS(\( x_d \)):=CS(\( x_d \))∪CS(\( x_i \))-{\( x_d \)}

CBJ Discovers (2, 5, 3) inconsistent with \{\( x_5 \), \( x_6 \)\}

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( x_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Q</td>
<td></td>
<td></td>
<td>1,2,3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CS(\( x_5 \)) = 1,2,3
CS(\( x_6 \)) = 1,2,3,5
Fig. 2. A fragment of the BT backtrack tree for the 6-queens problem.
Forward Checking (FC)

- Perform Consistency Check *Forward*
- Whenever a var is assigned a value
  Prune inconsistent values from
  As-yet unvisited variables
  Backtrack if domain of any var ever collapses

FC only visits consistent nodes
  but not *all* such nodes
  skips (2, 5, 3, 4) which CBJ visits
But FC can’t detect that
  (2, 5, 3) inconsistent with \(\{x_5, x_6\}\)