Search with Constraints

COMP8620, S2 2008
Advanced Topics in A.I.
Jason Li
The Australian National University
Overview

- What you will get from this lecture:
  - What is Constraint Programming
  - What it’s good for
  - What is [Arc | Generalized-Arc | Path | Bound ] Consistency, and how they’re useful
  - What drives efficient searches with constraints
  - How to play Sudoku!
Introduction

- **Constraints**
  - Specifies what you can/can’t do!
  - Find a solution that conform to all the rules

- **Search with constraints**
  - Yes, you still have to search!
  - Once a variable assignment is made, your other choices are limited.
  - You can limit your search space.
Search with Constraints

- Most typical example:

```
SEND
+ MORE
-----
MONEY
```
Holy Grail

- Dream of A.I. and declarative programming
  - Freuder, Walsh
  - User describes a problem
  - Computer comes up with a solution
Constraints and Culture

- Prevalent in many East-Asian cultures
  - Think via constraints
  - You can’t do this, can’t do that...
  - Heavy punishments for social constraint violation
- Compare versions of hell:
Real-world applications

- They are everywhere!
  - Delivery before 10am
  - Within 4km radius of Civic
  - Achieve a grade of at least 80

- Car-assembly lines
  - No more than 2 cars in every 5 are red
Constraint Satisfaction

- In Constraint Satisfaction Problem (CSP), we have:
  - A set of variables (V)
  - Each variable has a set of values (D)
    - Usually finite
    - \{true,false\}, \{red,green,blue\}, \{0,1,2,3,…\}
  - Set of constraints (C)
    - Describe the relationship between the variables
- Goal:
  - Find assignment of each variable to a value such that all constraints are satisfied
CSP Examples

- Sudoku
  - Variables
    - Each entry in the table $X_{row,col}$
  - Domain
    - Each variable between (1..9)
  - Constraints
    - Row: AllDifferent($X_{x,1}, \ldots, X_{x,ncol}$)
    - Column: AllDifferent($X_{1,y}, \ldots, X_{nrow,y}$)
    - Square: AllDifferent($X_{1,1}, \ldots, X_{3,3}$), etc.
CSP Examples

- Olympic games scheduling
  - Variables for each event
    - 50mFreeStyleMen, 100mFreeStyleWomen, 10mDivingMen, etc…
  - Domain is the time for the event
    - Monday9am, Tuesday3pm, etc…
  - Constraints:
    - 50mFreeStyleMen != Monday9am
    - Venue: AllDifferent([50mFreeStyleMen, 100mFreeStyleWomen, … ])
    - Capacity: AtMost(3, [50mFreeStyleMen, 100mFreeStyleWomen, …], Tuesday12pm)
**Constraints**

- A constraint consists of
  - A list of \( n \) variables
  - A relation with \( n \) columns
- Example: \( a \times b = 6 \)
  - \((a, b)\)
  - Relations: see table:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>
Binary/Non-Binary Constraints

- **Binary**
  - Scope over 2 variables
    - Not Equal: $a \neq b$
    - Ordering: $a < b$
    - Topological: $a$ is disconnected to $b$

- **Non-Binary**
  - More than 2 variables
    - AllDifferent($x_1, x_2, x_3, \ldots$)
    - $x^2 + 2y^2 - z^2 = 0$
Non-Binary Constraints

- Most non-binary constraints can be reduced to binary constraints
  - AllDifferent(a, b, c) ⇔ a ≠ b, b ≠ c, a ≠ c
- Advantages of non-binary constraints:
  - Polynomial algorithms – efficient solving
  - More on this later on!
Arc-Consistency

- A binary constraint \( \text{rel}(X1,X2) \) is arc-consistent (AC) iff
  - For every value \( v \) for \( X1 \), there is a consistent value \( v' \) for \( X2 \), and vice versa
  - In this case, \( v' \) is called the support of \( v \)

- Example:
  - Both \( x \), \( y \) are prime numbers, \( x \) is less than 10
  - For \( \text{GreaterThan}(x, y) \):
    - 2,3,5 are all supports of \( x = 7 \)
    - But 7 is NOT!
Enforcing Arc-Consistency

- We enforce arc-consistency by deleting domain values that cannot have support
  - As they cannot participate in the solution
- It may remove support for other values
- Complexity: $O(nD^2)$
  - $n$: number of constraints
  - $D$: domain size
Enforcing Arc-Consistency

- Example:
  - X: \{ 1,2,3,4,5,6,7,8,9,10 \}
  - Y: \{ 3,4,5,6,7 \}
  - Z: \{ 6 \}
  - Constraints:
    - X < Y
    - Y >= Z
Enforcing Arc-Consistency

- Example: (Enforcing Y >= Z)
  - X: \{ 1,2,3,4,5,6,7,8,9,10 \}
  - Y: \{ 3,4,5,6,7 \}
  - Z: \{ 6 \}
  - Constraints:
    - X < Y
    - Y >= Z
Enforcing Arc-Consistency

- Example: (Enforcing X < Y)
  - X: \{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}
  - Y: \{ 3, 4, 5, 6, 7 \}
  - Z: \{ 6 \}
  - Constraints:
    - X < Y
    - Y >= Z
Generalized Arc-Consistency

- For CSP with non-binary constraints, it is Generalised Arc-Consistent (GAC) iff:
  - For every variable \( x \) in \( V \)
  - For every constraint \( C(x, y_1, \ldots, y_n) \)
  - For every value \( d \) in \( D(x) \)
    - There are values \( d_1, \ldots, d_n \) in \( D(y_1), \ldots, D(y_n) \)
    - Such that \( C(d, d_1, \ldots, d_n) \) is true.

- \( \text{GAC} = \text{AC} \) for binary constraints
GAC in Action… Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(col_1)
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{8\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{4\}
GAC in Action… Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(row_1)
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{8\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{4\}
GAC in Action... Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(row_2)
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{8\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{4\}
GAC in Action… Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(sq_1)
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{8\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{4\}
GAC in Action… Sudoku!

- Look at Col 1:
  - Enforce: AllDifferent(row_4)
    - \{1,2,3,5,6,7,9\}
    - \{1,2,3,5,6,7,9\}
    - \{8\}
    - \{1,2,3,5,6,7,9\}
    - \{1,2,3,5,6,7,9\}
    - \{1,2,3,5,6,7,9\}
    - \{1,2,3,5,6,7,9\}
    - \{4\}
Look at Col 1:
- Enforce: AllDifferent(row_5)
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{8\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{4\}
GAC in Action… Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(row_6)
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{8\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{4\}
GAC in Action… Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(sq_4)
  - $\{1,2,3,5,6,7,9\}$
  - $\{1,2,3,5,6,7,9\}$
  - $\{8\}$
  - $\{1,2,3,5,6,7,9\}$
  - $\{1,2,3,5,6,7,9\}$
  - $\{1,2,3,5,6,7,9\}$
  - $\{1,2,3,5,6,7,9\}$
  - $\{1,2,3,5,6,7,9\}$
  - $\{4\}$
GAC in Action… Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(row_7)
  - {1,2,3,5,6,7,9}
  - {1,2,3,5,6,7,9}
  - {8}
  - {1,2,3,5,6,7,9}
  - {1,2,3,5,6,7,9}
  - {1,2,3,5,6,7,9}
  - {1,2,3,5,6,7,9}
  - {1,2,3,5,6,7,9}
  - {1,2,3,5,6,7,9}
  - {4}
GAC in Action… Sudoku!

- Look at Col 1:
- Enforce: AllDifferent(row_8)
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{8\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{1,2,3,5,6,7,9\}
  - \{4\}
Path-Consistency

- Problem: sometimes arc-consistency is not sufficient
  - Usually involves transitivity of relations
- Example (Arc-consistent problem):
  - A $\neq$ B, B $\neq$ C, C $\neq$ A
  - A: \{1,2\}, B: \{2,3\}, C: \{2,3\}
  - For A=2, there’s no solution, but domains are arc consistent
  - Hence, we need path-consistency
    - Also known as (2,1)-consistency
Path-Consistency

- A binary CSP is Path-Consistent iff:
  - For every pair of variable x, y in V
    - with constraint C(x,y)
  - And every other variable z in V
    - With constraint C(x,z), C(y,z)
  - For every pair d1 in D(x), d2 in D(y)
    - Such that C(d1,d2) is true
  - There is a value d in D(z)
    - Such that C(d1,d) C(d2, d) is true.
Path-Consistency in Action!

- Consider the problem again:
  - A \neq B, B \neq C, C \neq A
  - A: \{1,2\}, B: \{2,3\}, C: \{2,3\}
- Arc-Consistency:
  - A=2 gets support: B=3, C=3
  - However, this violates path-consistency!
Path-Consistency

- Path-consistency enforces every constraint work with every other constraint
  - May work with binary constraint network with infinite domains (spatial-temporal reasoning)
- It may still be insufficient, where sometimes 3 constraints must be checked together
  - i,j-consistency
Path Consistency in Action!

- A different perspective:
Path-Consistency in Action!

- A different perspective:
Infinite Domains

- Path-consistency can also work with constraint networks with infinite domains (Montanari 78, Van Beek, 92)
  - Reason about relations between the entities
  - Complexity is $O(n^3)$, if path-consistency implies consistency
    - Any path-consistent CSP has a realization
Path-Consistency

- A confusion of relationships:

Diagram showing relationships between mother-in-law, father-in-law, mother, wife, and daughter.
Path-Consistency

- Applying path-consistency:
Path-Consistency

- Applying path-consistency one more time:
Bound-Consistency

- Sometimes, there are a lot of possibilities in a large number of ordered values
  - It’s inefficient to check all cases
  - Only the bounds are interesting
- When to use bound consistency?
  - Domain is ordered
  - Only necessary to enforce Arc-Consistency on the max/min elements
Bound-Consistency

- A CSP is Bound-Consistent if
  - For every $x$ in $V$ and constraint $C(x,y_1,\ldots,y_n)$
  - For both $d = \min(D(x))$ and $d = \max(D(x))$
  - There are values $d_1,\ldots,d_n$ in $D(y_1),\ldots,D(y_n)$, such that $C(d,d_1,\ldots,d_n)$ is true.
Bound-Consistency in Action!

- **Linear Inequalities:**
  - Constraint: $2X + 3Y \leq 10$
  - $X: \{1,2,3,4,5,6,7,8,9,10\}$
  - $Y: \{1,2,3,4,5,6,7,8,9,10\}$

- **Bound-consistent constraints:**
  - $\text{UpperBound}(X) \leq \left\lfloor \frac{10 - 3\times \text{LowerBound}(Y)}{2} \right\rfloor$
  - $\text{UpperBound}(Y) \leq \left\lfloor \frac{10 - 2\times \text{LowerBound}(X)}{3} \right\rfloor$
Bound-Consistency in Action!

- Linear Inequalities:
  - Constraint: $2X + 3Y \leq 10$

- Enforcing
  - $\text{UpperBound}(X) \leq \lfloor (10 - 3*\text{LowerBound}(Y))/2 \rfloor$
  - $X: \{1,2,3,4,5,6,7,8,9,10\}$
  - $Y: \{1,2,3,4,5,6,7,8,9,10\}$
Bound-Consistency in Action!

- Linear Inequalities:
  - Constraint: \(2X + 3Y \leq 10\)
- Enforcing
  - \(\text{UpperBound}(Y) \leq \lfloor (10 - 2*\text{LowerBound}(X))/3 \rfloor\)
  - \(X: \{1,2,3,4,5,6,7,8,9,10\}\)
  - \(Y: \{1,2,3,4,5,6,7,8,9,10\}\)
- Note only 2 bounds needs checking
Maintaining local-consistency

- **Tree Search**
  - Assign variable to value
  - Enforce consistency
    - Remove future values / add constraints
  - If no possible values can be assigned, backtrack

- **Local search**
  - Generate a complete assignment
  - Make changes in violated constraints
What makes constraints fast

- How the problem is modelled:
- Heuristic-guided search
- Efficient propagation of constraints
  - When enforcing constraints, prune as much as possible, but not at too greater costs
Trade offs

- Too strong consistency:
  - Too much overhead at each node of the search tree
- Too weak consistency:
  - Not pruning enough of the search space
- Unfortunately, this can only be determined empirically
  - Carry out experiments, see which one does better
Trade offs

- Propagation vs. Search
  - In general, we don’t want to spend more time enforcing consistency than doing search
  - Problem dependent
In Summary

- Inference with constraints prunes potential search-space
- A CSP $<V,D,C>$ is consisted of
  - Variables
  - Domains
  - Constraints
In Summary

- Arc-Consistency removes impossible values for each binary constraint
- Generalized-Arc-Consistency removes impossible values for non-binary constraints
- Path-Consistency removes impossible values between constraints
- Bound-Consistency checks AC for upper and lower bounds in ordered domains
Search with Constraints

Part 2
Overview

- What are Global Constraints?
  - Why we use them
  - AllDifferent constraint
  - The Marriage Theorem and Hall Intervals
  - Puget’s Algorithm

- What is Symmetry?
  - How to break them
Global Constraints

- A constraint involving a arbitrary number of variables
  - AllDifferent
  - LexOrder
- Can be modelled with binary constraints
  - E.g. AllDifferent(X1,X2,X3)
    $\Leftrightarrow X_1 \neq X_2, X_2 \neq X_3, X_1 \neq X_3$
- In practice, they can be more efficiently solved without this decomposition
Golomb Ruler

- Marking ticks on a ruler
- Unique distance between any two ticks
- Applications
  - X-Ray Crystallography
  - Radio Astronomy
- Problem 006 in CSPLib
Golomb Ruler

- Naive solution: exponentially long ruler
  - Ticks at 0, 1, 3, 7, 15, 31, 63, 127, 255, etc…
- Key is to find a ruler of minimal length
  - Known for up to 23 ticks
  - Distributed internet project for larger lengths
Golomb Ruler as CSP

- Explicit Representation
- Variable: $X_i$ for each tick
  - Auxillary Variables: $D_{ij} = |X_i - X_j|$
- Constraints:
  - $X_i < X_j$ for all $i < j$
  - AllDifferent($D_{11}$, $D_{12}$, $D_{13}$, ... )
  - Minimize($X_n$)
AllDifferent Constraint

- One of the oldest global constraints
  - ALICE: [Lauriere 78]
- They are everywhere!
  - Golomb Ruler: \texttt{AllDifferent}(D_{11}, D_{12}, D_{13}, \ldots)
  - Standard constraint
  - Incorporated by all constraint solvers
AllDifferent Constraint

- Can be modelled with binary constraints
  - AllDifferent(X1,X2,X3)
    \[\Leftrightarrow X1\neq X2, X2\neq X3, X1\neq X3\]

- However, this may be done more efficiently
  - X1: \{1,2\}, X2: \{1,2\}, X3: \{1,2,3\}
  - X3 can never be 1 or 2
  - How can we exploit this?

- Efficient algorithms:
  - [Puget AAAI’98] Bound Consistency Algorithm runs in \(O(n \log n)\)
Bound Consistency with AllDifferent

- Uses Hall’s Theorem
  - Also termed “Marriage Theorem”
  - Given $k$ sets
  - There is an unique and distinct element in each set iff
    - For $0 < j \leq k$
    - Any union of $j$ of the sets has at least $j$ elements

- Example:
  - $X1: \{1,2\}, X2: \{1,2\}$ is okay
  - $X1: \{1,2\}, X2: \{1,2\}, X3: \{1,2\}$ will not work.
Marriage Theorem

- There are $n$ men and women in a town…
  - Each man is happy to be married to any woman
  - Each woman has some preferred men to marry (subset of all men)
- Given $j$ women…
  - The number of men they wish to marry must be $j$ or more!
Following Hall’s (Marriage) Theorem

- Hall Interval
  - Interval of domain values that has just as many variables as domain values
  - E.g. $X1:\{1,2\}, \ X2:\{1,2\}$
    - Two variables in the interval $\{1,2\}$

- AllDifferent is Bound-Consistent iff:
  - Each interval in the domain do not cover more variables than its length (Hall Interval)
  - A variable with possible domain value outside a Hall Interval do not have value within it
Bound-Consistency in Action!

- Consider our old example:
  - X1: \{1,2\}, X2: \{1,2\}, X3: \{1,2,3\}
  - AllDifferent(X1,X2,X3)
- Obviously:
  - [1…2] is a Hall Interval covering X1, X2
  - X3 has a value outside a Hall Interval, therefore we prune it
- Result is Bound-Consistent:
  - X1:{1,2}, X2: \{1,2\}, X3: \{3\}
Puget’s Algorithm

- Naive implementation consider $O(n^2)$ intervals
- Puget order the intervals in $O(n \log n)$ time
  - Then go through them in order
- Best BC-Algorithm for AllDifferent
  - Still, in some problems GAC can do better
    - Problems closer to combinatorial problems
Further problem in Golomb Ruler

- Problem: there are trivial repetitions of the same solution in the search space.
  - Ruler 1: Ticks at: 0, 1, 4, 6
  - Ruler 2: Ticks at: 0, 2, 5, 6
- Is there any fundamental difference between the above two rulers?
Symmetry

- Symmetries occur frequently in Constraint Programming and Search
  - Any permutations of rows or columns of a table
  - Real-world scheduling problems
- It’s a very active area of CP research
Breaking Symmetry

- In Golomb Ruler, we ensure the ruler cannot be reversed.
- Easiest way to break symmetry is to add additional constraints:
  - $D_{12} < D_{n-1,n}$
- Another symmetry: preventing permutations in the ticks:
  - $X_1 < X_2 < \ldots X_n$
Another Example of Symmetry

- Consider two bins:
  - Example of row-symmetry (Walsh)

a)  
\[
\begin{array}{c|c}
\hline
A & B \\
\hline
5 & 6 \\
3 & 4 \\
1 & 2 \\
\hline
\end{array}
\]

\[
A \quad \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

b)  
\[
\begin{array}{c|c}
\hline
A & B \\
\hline
6 & 5 \\
4 & 3 \\
2 & 1 \\
\hline
\end{array}
\]

\[
A \quad \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 \\
0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 \\
\end{array}
\]

Another Example of Symmetry

- Such symmetry can be broken by lexicographical constraints
  - $\text{Row}(A) \leq \text{LEX} \text{ Row}(B)$

<table>
<thead>
<tr>
<th></th>
<th>a)</th>
<th>b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Breaking Symmetry

- Symmetries hide in all kind of problems
- They are expensive!
  - Therefore breaking them is essential in making CP/Search efficient
- General methods may add exponential number of constraints! [Crawford, Ginsberg and Luks]
- Other way of breaking symmetry
  - Ignore them in search strategy
Breaking Symmetry

- Two major strategies in modifying search strategy:
  - During Search:
    - After exploring a branch, implicitly add constraints preventing exploring other symmetric branches
  - Dominance Detection:
    - Before exploring a branch, check if it is dominated by a previously visited branch
Summary

- Global Constraints are efficient with clever algorithms
  - AllDifferent and Marriage Theorem
- Symmetries can surface in many problems
  - They can be broken by adding constraints
    - Most times lexicographical constraints
  - They can also be broken by modifying search strategies
Get your hands dirty!

- Most constraint solvers have two layers
  - High level: non-expert user input
  - Low level: for computer/expert user
- They come with easy examples!
- Try some yourself!
  - MiniZinc/FlatZinc:
    - http://www.g12.csse.unimelb.edu.au/minizinc/
  - Tailor/Minion:
    - http://www.dcs.st-and.ac.uk/~andrea/tailor/index.html
  - GeCode:
    - http://www.gecode.org/