Constraint Programming

Lecture 8: Heuristics
Last Lecture: Exploiting Constraint Properties

- Reducing constraint checks by using properties of constraints:
  - Functional constraints.
  - Monotonic constraints.
- Avoiding constraint checks by using inference:
  - Bi-directionality.
  - Commutativity.
  - Irreflexivity.
  - Anti-functional.
  - Monotonicity.
This Lecture

• Heuristics.
• (we’ll return to consistency next lecture).
Heuristics
I’m sorry, Dave, I’m afraid I can’t do that

- Heuristically programmed ALgorithmic computer – HAL 9000
- Heuristics were part of the mid-1960’s optimism about AI – Human-like AI in 20 years!
- A heuristic in AI is an “educated guess” or “rule of thumb” that may help to solve some problem
Heuristics for Solving CSPs/COPs

• “Rules of thumb” that, if we follow them, are expected to lead to an improvement in general.
• Operations we can order heuristically:
  1. Variable instantiation order.
  2. Value assignment order.
  3. Constraint checking order.
• We look in detail at the first 2.
• If we had an oracle that always got no. 2 right we would march straight to a solution!
Static Heuristics

- Order in which variables/values selected/assigned chosen before search.
  - Order is set before search starts.
  - Order is fixed for entire search.
- E.g.
  - Assign variables in order $x_1, x_2, x_3, \ldots$
  - Assign values in ascending order.
Dynamic Heuristics

• Examine the state of the problem during search.
• Decide on “best” variable/value to assign next.
• E.g.
  • “Smallest-domain” variable ordering.
    • Affected by FC/MAC propagation, which differs based on the assignment made.
  • “Min-conflicts” value ordering.
Variable Instantiation Order

(or just Variable Order for short)
Variable Instantiation Order

- Can have a substantial effect on size of search tree:

\[ x_1, D_1 = \{2, 10, 16\}, x_2, D_2 = \{9, 12, 21\}, x_3, D_3 = \{9, 10, 11\}, x_4, D_4 = \{2, 5, 10, 11\} \]

- Using just backtracking, let’s try the static instantiation order: \( x_1, x_2, x_4, x_3 \).
Variable Instantiation Order: Example

\[ x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \]

\[ x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \]

- 2-way branching, backtracking
Variable Instantiation Order: Example

\[ x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \]

\[ x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \]
Variable Instantiation Order: Example

\[ x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \]

\[ x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \]
Variable Instantiation Order:

Example

\[ x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \]
\[ x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \]
Variable Instantiation Order:
Example

- Solution:
  13 nodes in tree.

\[ x_1, D_1 = \{2, 10, 16\} \]
\[ x_2, D_2 = \{9, 12, 21\} \]
\[ x_3, D_3 = \{9, 10, 11\} \]
\[ x_4, D_4 = \{2, 5, 10, 11\} \]
Variable Instantiation Order: Example

- Consider now the static instantiation order: $x_3, x_4, x_1, x_2$.

$x_1, D_1 = \{2, 10, 16\}$

$x_2, D_2 = \{9, 12, 21\}$

$x_3, D_3 = \{9, 10, 11\}$

$x_4, D_4 = \{2, 5, 10, 11\}$
Variable Instantiation Order: Example

\[ x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \]

\[ x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \]

No solutions

26 nodes
Variable Instantiation Order: Example

\[ x_1, D_1 = \{2, 10, 16\} \]
\[ x_2, D_2 = \{9, 12, 21\} \]
\[ x_3, D_3 = \{9, 10, 11\} \]
\[ x_4, D_4 = \{2, 5, 10, 11\} \]

No solutions
26 nodes

No solutions
26 nodes
Variable Instantiation Order: Example

\( x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \)

\( x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \)

\( x_3 = 9 \quad x_3 \neq 9 \)

No solutions
26 nodes

\( x_4 = 2 \)

No solutions
26 nodes
Variable Instantiation Order: Example

\( x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \)

\( x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \)
Variable Instantiation Order: Example

\( x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \)

\( x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \)
Variable Instantiation Order: Example

\[x_1, D_1 = \{2, 10, 16\}\]
\[x_2, D_2 = \{9, 12, 21\}\]
\[x_3, D_3 = \{9, 10, 11\}\]
\[x_4, D_4 = \{2, 5, 10, 11\}\]
Variable Instantiation Order:

Example

$x_1, D_1 = \{2, 10, 16\}$  
$x_2, D_2 = \{9, 12, 21\}$  
$x_3, D_3 = \{9, 10, 11\}$  
$x_4, D_4 = \{2, 5, 10, 11\}$

No solutions
26 nodes

$x_3 = 9$
$x_3 \neq 9$

$x_3 = 10$
$x_3 \neq 10$

No solutions
26 nodes

$x_3 = 11$

$x_4 = 2$
$x_4 \neq 2$

$x_4 = 5$

$x_1 = 2$
$x_1 \neq 2$

$x_1 = 10$
$x_1 \neq 10$

$x_2 = 9$
$x_2 \neq 9$

$x_2 = 12$

$x_1 = 16$
$x_1 \neq 16$
Variable Instantiation Order: Example

$x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\}$

$x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\}$

No solutions
26 nodes
$x_3 \neq 9$

No solutions
26 nodes
$x_3 \neq 10$

$x_4 \neq 2$

$x_4 = 2$

$x_1 = 2$

$x_1 = 10$

$x_1 \neq 10$

$x_2 = 9$

$x_2 \neq 9$

$x_2 = 12$

$x_2 \neq 12$

$x_2 = 21$
Variable Instantiation Order: Example

\[ x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \]

\[ x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \]
Variable Instantiation Order: Example

\[x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\}\]

\[x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\}\]
Variable Instantiation Order: Example

\[ x_1, D_1 = \{2, 10, 16\} \quad x_2, D_2 = \{9, 12, 21\} \]

\[ x_3, D_3 = \{9, 10, 11\} \quad x_4, D_4 = \{2, 5, 10, 11\} \]

- Solution: 83 nodes explored in total.
What’s Going on Here?

• In the example, it’s simple:
  - **First ordering**: for each variable the first value we assign that is compatible with past assignments is part of the solution.
  - **Second ordering**: because of the variable instantiation order, we make assignments that “look” good, but turn out to have no solutions beneath them – backtracking required.
What’s Going on Here? Thrashing

- Repeated failure for the same reason:
  
  \[ x_1 = a \]

  \[ x_{1000} D_{1000} = \{ a \} \]

  \[ x_1 \neq x_{1000} \]

- If \( x_1, x_{1000} \) are adjacent in the variable order, there is very little impact.
Do Variable Order Heuristics Still Matter when We Use FC?

• Yes:
  
  $x_1 = a$

  ...Thrashing

  $x_{999}, D_{999} = \{a, b\}$

  $x_{1000}, D_{1000} = \{a, b\}$

• Again, if $x_1, x_{999}$ are adjacent in the variable order, there is very little effect.
Do Variable Order Heuristics Still Matter when We Use MAC?

- Yes:
  - $x_1 = a$
  - $x_{998} D_{998} = \{a, b, c\}$
  - $x_{999} D_{999} = \{a, b, c\}$
  - $x_{1000} D_{1000} = \{a, b, c\}$

- Again, if $x_1, x_{998}$ are adjacent in the variable order, there is very little effect.
Rationale

- Make the **most difficult** choice first (**variables**).
  - Have the most far-reaching consequences for the rest of the problem.
  - Narrows down easy choices.
  - If we make the easy choices first, we can paint ourselves into a corner, as in the previous example.
- For the most difficult choice choose the option **most likely to succeed** (**values**).
Static Variable Heuristics
Static Variable Order Heuristics: Maximum Degree

• Orders variables in decreasing order of their degrees in the constraint graph.
• Break ties arbitrarily.
Static Variable Order Heuristics: Maximum Degree

- Orders variables in decreasing order of their degrees in the constraint graph.
- Break ties arbitrarily.
Static Variable Order Heuristics: Maximum Degree

• Remember the Crystal Maze?
• Solved easily by labelling the maximum degree variables first
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Maximum Cardinality

- Select first variable arbitrarily.
- Repeat until no more variables:
  - Append to order variable connected to largest group among those already selected.
Static Variable Order Heuristics: Minimum Width

• Ordered constraint graph:
  • Arrangement of constraint graph nodes into a fixed linear order.

• The **width** at an ordered node:
  • Number of arcs linking back to a previous node in the order.

• Width of an ordering:
  • Maximum width at any node.

• Width of a graph:
  • Minimum width of all its orderings.
Primal Graph: Reminder

- \( c(x_1, x_2, x_3), c(x_2, x_3, x_4), c(x_3, x_4, x_5). \)
Ordered Constraint Graph

- Arrangement of constraint graph nodes into a fixed linear order.
Width at an Ordered Node

• Number of edges linking to a previous variable in the ordering:

Width: 0
Width: 1
Width: 1
Width: 2
Width: 3

First
Second…
Width of an Ordering

- **Maximum** width at any ordered node:

  Width: 0
  Width: 1
  Width: 1
  Width: 2
  Width: 3

First
Second…
Width of a Constraint Graph

- **Minimum** width of all of its orderings.

  - For each ordering, write down:
    1. The width of each node.
    2. The width of the ordering.

- What is the width of the constraint graph?
Width of a Constraint Graph

- **Minimum** width of all of its orderings.

For each ordering, write down:
1. The width of each node.
2. The width of the ordering.

- What is the width of the constraint graph?
Static Variable Order Heuristics: Minimum Width

• There are efficient ways to find:
  • The width of the graph.
  • An ordering corresponding to this width.

• Why is minimum width a good idea?
  • Here $x_3$ is most difficult choice. Don’t want to leave until last:
Static Variable Order Heuristics: Relative Merits

- In a study in 1989:

- Max-card was seen to be the best, especially for finding all solutions.

- But this was a study on random binary problems only.
Dynamic Variable Order Heuristics
Dynamic Variable Order Heuristics: Smallest Domain

- Select the variable with the smallest number of values compatible with past assignments.
  - Dynamic because selection made based on state during search.
  - Different branches can have different variable orders.

\[
D_1 = \{1, 2\} \quad D_2 = \{1, 2\} \quad D_3 = \{2, 3\}
\]

\[
x_1 = 1 \quad x_1 \neq 1
\]

\[
x_2 = 1 \quad x_3 = 3
\]

\[
x_1 = 2
\]
Dynamic Variable Order Heuristics: Smallest Domain

• Very popular heuristic, known to perform very well.
• Caution: it has many names in the literature!
  • Search re-arrangement backtracking
  • Fail-first!
  • Minimum remaining values.
Dynamic Variable Order

Heuristics: Smallest Domain

• Backtrack vs FC, MAC:
  • Backtrack requires extra constraint checks to implement this heuristic. FC, MAC do not.
  • Why? Because all remaining domain elements in FC, MAC are compatible with past assignments.
  • In fact, if augment BT with a smallest domain heuristic, resulting algorithm has to do as much work as FC.
Dynamic Variable Order Heuristics: Brelaz

• Select the variable with the smallest number of values compatible with past assignments.
• Break ties by selecting the variable with the maximum degree in the constraint sub-graph of the future variables.
Dynamic Variable Order
Heuristics: dom/deg

• Select the variable that minimises the ratio domain size/degree in future constraint graph.

• Motivation:
  • When constraint graph is sparse, minimum domain less useful than an ordering based on degree.
  • When constraint graph is dense, minimum domain much better than degree alone.
  • Rather than break ties with one or the other, this heuristic combines them.
Static vs Dynamic Variable Orders: Performance

- In studies on **random** problems, dynamic orderings often seen to do better.
- On **structured** problems static orderings can often perform very well.
- Part of the art of constraint programming is choosing an appropriate heuristic for your problem.
Lecture Summary

• Variable instantiation order
  – Why it makes a difference

• Static variable ordering heuristics
  – Maximum degree
  – Maximum cardinality
  – Minimum width

• Dynamic variable ordering heuristics
  – Smallest domain (dom)
  – Brelaz
  – Dom/deg