Domain Splitting, Local Search

CPSC 322 – CSP 4

Textbook §4.6, §4.8

February 4, 2011
Discussion of feedback

• Pace
  – 2 “fine”, 1 “could go faster”
  – 2: recap too long, 3: “sometimes rushed later (as a consequence)"

• Coloured card questions
  – Some more explanation would be good
  – More consistent: get everyone to vote”

• Which parts are most important?
  – Definitions + algorithms. Examples are for illustration

• Hard concepts:
  – Arc consistency: today + work in Alspace + practice exercise
  – Alternative formulation of CSP as graph search: after class
Discussion of feedback

• Midterm: review & sample questions?
  – Midterm date confirmed: Mon, Feb 28, 3pm (1 to 1.5 hours, TBD)
  – Sample midterm has been on WebCT for ~2 weeks
    • Topics: everything up to (including all of) CSP
    • The topic of its question 2 (Planning based on CSP) won’t be on the midterm
  – Should we do a midterm review session?

• More explanation of practice exercises?
  – I’ll show where they are in WebCT
  – If you have trouble with them, please come to office hours

• How will what we learn eventually be applied in making an intelligent agent?
  – Game AI: lots of search
  – Reasoning under constraints is core to making intelligent decisions
    • With CSPs, we’re right in the middle of it!
We’ll now focus on CSP.
Lecture Overview

Arc consistency
- Recap
- Complexity analysis
- Domain Splitting

• Intro to Local Search
Arc Consistency

Definition:
An arc \(<x, r(x,y)\> is arc consistent if for each value \(x\) in \(\text{dom}(X)\) there is some value \(y\) in \(\text{dom}(Y)\) such that \(r(x,y)\) is satisfied.

A network is arc consistent if all its arcs are arc consistent.

Not arc consistent: No value in domain of B that satisfies A<B if A=3
Arc consistent: Both B=2 and B=3 have ok values for A (e.g. A=1)
Arc Consistency

Arc consistent: For each value in \( \text{dom}(C) \), there is one in \( \text{dom}(A) \) that satisfies \( A > C \) (namely \( A = 3 \)).

Not arc consistent: No value in domain of \( B \) that satisfies \( A < B \) if \( A = 3 \).
Arc Consistency

Not arc consistent anymore: For C=2, there is no value in \( \text{dom}(A) \) that satisfies \( A > C \)
Which arcs need to reconsider?

- When we reduce the domain of a variable $X$ to make an arc $\langle X, c \rangle$ arc consistent, which arcs do we need to reconsider?

  - You do not need to reconsider other arcs
    - If an arc $\langle X, c' \rangle$ was arc consistent before, it will still be arc consistent
    - Nothing changes for arcs of constraints not involving $X$

\[ Z_1 \text{ } \overset{c_1}{\longrightarrow} \text{ } Z_2 \text{ } \overset{c_2}{\longrightarrow} \text{ } Z_3 \text{ } \overset{c_3}{\longrightarrow} \text{ } X \]

\[ \text{EVERY ARC } \langle Z, c' \rangle \text{ WHERE } c' \neq c \text{ INVOlVES } Z \text{ AND } X: \]

\[ X \overset{c}{\longrightarrow} Y \text{ } \overset{c_4}{\longrightarrow} \text{ } A \]
Lecture Overview

• Arc consistency
  - Recap
  - Complexity analysis
  - Domain Splitting

• Intro to Local Search
Arc Consistency Algorithm: Complexity

- Worst-case complexity of arc consistency procedure on a problem with N variables
  - Let \( d \) be the max size of a variable domain
  - Let \( c \) be the number of constraints
- How often will we prune the domain of variable \( V \)? \( O(d) \) times
- How many arcs will be put on the ToDoArc list when pruning domain of variable \( V \)?
  - \( O(\text{degree of variable } V) \)
  - In total, across all variables: sum of degrees of all variables = …
    - \( 2\times\text{number of constraints}, \text{i.e. } 2\times c \)
- Together: we will only put \( O(dc) \) arcs on the ToDoArc list
- Checking consistency is \( O(d^2) \) for each of them
- Overall complexity: \( O(cd^3) \)
- Compare to \( O(d^N) \) of DFS!! Arc consistency is MUCH faster
Lecture Overview

• Arc consistency
  - Recap
  - Complexity analysis
    Domain Splitting

• Intro to Local Search
Can we have an arc consistent network with no solution?

**YES**

**NO**

- Example: vars A, B, C with domain {1, 2} and constraints $A \neq B$, $B \neq C$, $A \neq C$

- Or see Alspace CSP applet Simple Problem 2
Domain splitting (or case analysis)

• Arc consistency ends: Some domains have more than one value → may or may not have a solution
  A. Apply Depth-First Search with Pruning or
  B. Split the problem in a number of disjoint cases:

        CSP with dom(X) = \{x_1, x_2, x_3, x_4\} becomes

        CSP_1 with dom(X) = \{x_1, x_2\} and
        CSP_2 with dom(X) = \{x_3, x_4\}

• Solution to CSP is the union of solutions to CSP_i
Domain splitting

- Each smaller CSP is easier to solve
  - Arc consistency might already solve it
- For each subCSP, which arcs have to be on the ToDoArcs list when we get the subCSP by splitting the domain of X?

  arcs <Z, r(Z,X)>

  arcs <Z, r(Z,X)> and <X, r(Z,X)>

  All arcs
Domain splitting in action

- Trace it on “simple problem 2”
Searching by domain splitting

If domains with multiple values
Split on one

CSP₁, apply AC

If domains with multiple values
Split on one

CSPₙ, apply AC

If domains with multiple values.....Split on one

How many CSPs do we need to keep around at a time?
With depth m and b children at each split: $O(bm)$. It’s a DFS.
Learning Goals up to here

• Define/read/write/trace/debug the arc consistency algorithm. Compute its complexity and assess its possible outcomes

• Define/read/write/trace/debug domain splitting and its integration with arc consistency
Lecture Overview

- Arc consistency
  - Recap
  - Complexity analysis
  - Domain Splitting

Intro to Local Search
Local Search: Why

- Solving a CSP is NP hard
  - Search space for many CSPs is huge
  - Exponential in the number of variables
  - Even arc consistency with domain splitting isn’t enough

- Alternative: local search
  - Algorithms that often find a solution quickly
  - But that cannot prove that there is no solution

- Useful method in practice
  - Best available method for many constraint satisfaction and constraint optimization problems
  - Extremely general!
    - Works for problems other than CSPs
    - E.g. arc consistency only works for CSPs
Local Search

- Idea:
  - Consider the space of complete assignments of values to variables (all possible worlds)
  - Neighbours of a current node are similar variable assignments
  - Move from one node to another according to a function that scores how good each assignment is
Definition: A local search problem consists of a:

**CSP**: a set of variables, domains for these variables, and constraints on their joint values. A node in the search space will be a complete assignment to all of the variables.

**Neighbour relation**: an edge in the search space will exist when the neighbour relation holds between a pair of nodes.

**Scoring function**: this can be used to incorporate information about how many constraints are violated. It can also incorporate information about the cost of the solution in an optimization context.
Example: Sudoku as a local search problem

CSP: usual Sudoku CSP
- One variable per cell; domains \{1,\ldots,9\};
- Constraints:
  each number occurs once per row, per column, and per 3x3 box

Neighbour relation: value of a single cell differs

Scoring function: number of constraint violations
Search Space

Only the current node is kept in memory at each step.
Very different from the search approaches we have seen so far!
Local search does NOT backtrack!
Example: Hill climbing for Sudoku

Assign random numbers between 1 and 9 to blank fields

```
  9 3 6 2 8 1 4
  6       5
  3 1 9
  5 8 2 7
  4 7 6
  8       3
 1 7 5 9 3 4 2
```
Example: Hill climbing for Sudoku

Assign random numbers
between 1 and 9 to blank fields

Repeat
– For each cell & each number:
  Evaluate how many constraint violations the assignment would yield
– Choose the cell and number that leads to the fewest violated constraints; change it

Until solved
Example: Hill climbing for Sudoku

Example for one local search step:
Reduces #constraint violations by 3:
- Two 1s in the first column
- Two 1s in the first row
- Two 1s in the top-left box
General Local Search Algorithm

You can be smart about selecting a variable and a new value for it: variable and value selection heuristics based on local information. E.g., for each neighbour, you can evaluate how many constraints are unsatisfied.

Hill climbing: select Y and V to minimize #unsatisfied constraints at each step.
Learning Goals for local search (started)

• Implement local search for a CSP.
  – Implement different ways to generate neighbors
  – Implement scoring functions to solve a CSP by local search through either greedy descent or hill-climbing.

• Assignment 1 is due on Monday

• Local search practice exercise is on WebCT
• Programming assignment (part of assignment #2) is available on WebCT (due Wednesday, Feb 23rd)

• Coming up: more local search, Section 4.8