Foundations of Artificial Intelligence

CS472/3
Lecture #6

Bart Selman

Today's Lecture

Problem Solving as Search, cont.
Constraint satisfaction problems

Readings: R&N, Chapter 3.

Constraint Satisfaction Problems (CSP)

A powerful representation for (discrete) search problems.
Led to “constraint programming”.

A **Constraint Satisfaction Problem (CSP)** is defined by:

X is a set of n variables $X_1, X_2, \ldots, X_n$,
each defined by its finite domain $D_1, D_2, \ldots, D_n$.

C is a set of constraints $C_1, C_2, \ldots, C_m$.

Slide CS472–3

Constraints

A **constraint** C_i restricts the set of possible values
that can be assigned to the variables in the constraint.
In other words, a constraint specifies which values are
compatible for the variables in the constraint.
A **solution** is an assignment of values to the variables
that satisfies all constraints.

Slide CS472–4

Example: Graph Coloring

Slide CS472–5

Any NP-complete problem can be efficiently formulated as a finite CSP problem.

Why? — ...

Slide CS472–6

Constraint Satisfaction Problems (CSP)

For a given CSP the problem is one of the following:

- find all solutions
- find one solution
  - just a feasible solution, or
  - a “reasonably good” feasible solution, or
  - the optimal solution given an objective
- determine if a solution exists

How to View a CSP as a Search Problem?

Initial State – state in which all the variables are unassigned.

Operators – assign a value to a variable from a set of possible values.

Goal test – check if all the variables are assigned and all the constraints are satisfied.

Solving CSPs

figure backtrack search

Branching Factor

Hypothesis 1 – any unassigned variable at a given state can be assigned a value by an operator: branching factor as high as sum of all domains.

Better approach – since order of variable assignment not relevant, consider as the successors of a node just the different values of a *single* unassigned variable: max branching factor = max domain.

Maximum Depth of Search Tree

n the number of variables; all the solutions are at depth n. What are the implications in terms of using DFS vs. BFS?

CSP – Goal Decomposed into Constraints

How to exploit it?

- **Backtracking** only insert successors if *consistent* with constraints.
- **Constraint propagation** "looking ahead" to remove inconsistencies.

- **Forward Checking** — each time variable is instantiated, remove other inconsistent values
- **Consistency** — state is arc-consistent, if every variable has some value that is consistent with each of its constraints (consider pairs of variables)
- **K-Consistency** generalizes arc-consistency. Consistency of groups of K variables.

Branching:

Most-constrained variable first; Most-constraining variable; Least-constraining value. (p. 105 R&N)

Example forward-checking / arc consistency.

Slide CS472–13

Send More Money as a CSP

Variables:

$S = \{0, \ldots, 9\}$; $E = \{0, \ldots, 9\}$;
$N = \{0, \ldots, 9\}$; $D = \{0, \ldots, 9\}$; $M = \{0, \ldots, 9\}$;
$O = \{0, \ldots, 9\}$; $R = \{0, \ldots, 9\}$; $Y = \{0, \ldots, 9\}$;

Constraints:

send $= 1000 \times S + 100 \times E + 10 \times N + D$;
more $= 1000 \times M + 100 \times O + 10 \times R + E$;
money $= 10000 \times M + 1000 \times O + 100 \times N + 10 \times E + Y$;
send + more = money;
each letter has a different digit (S $\neq$ E, S $\neq$ N, etc);

Slide CS472–14

Dramatic recent progress in **Constraint Satisfaction.**

For example, we can now handle problems with **10,000** to **100,000** variables, and up to **1,000,000** constraints.