CS 520: Introduction to Artificial Intelligence

Prof. Louis Steinberg

Lecture 4:

Constraint satisfaction

Reading:

- **For next lecture: Russell & Norvig Chapter 5**
- **For next week: Graham**
  - **Read Ch 1-3 and 6**
  - **Skim Ch 4-5 and 7-9**

Lectures on Search

- Formulation of search problems.
  - State Spaces
- Uninformed (blind) search algorithms.
- Informed (heuristic) search algorithms.
- Constraint Satisfaction Problems.
- Game Playing Problems.

Informed Search

- Incorporate problem-specific knowledge into the search strategy.
- Only useful if extra knowledge exists.
- One common form:

  f(n) = estimate of distance to goal

Review - Informed Search

- **Functions used in ordering the open nodes**
  - **h(n): heuristic estimate of distance from n to goal**
  - **g(n): actual distance from start state to n**
  - **f(n) = g(n)+h(n):**
    estimate of distance from start to goal via

Review - Informed Search

- **Algorithms**
  - **Constant Cost: sort open using g**
  - **Greedy / Best First: sort open using h**
  - **A*: sort open using f**
- **Monotonic f**
  - **Never decreases on path from start to goal**
- **Admissible h**
  - **Never over estimates distance to goal**

Review - Informed Search

- **Optimality of A***
  - **Shortest path**
  - **No complete, shortest path algorithm can be guaranteed to expand fewer nodes**

Comparison of performance for 8-puzzle

	Search Cost			Effective Branching Factor		
d	IDS	A*(h_1)	A*(h_2)	IDS	A*(h_1)	A*(h_2)
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	364404	227	73	2.78	1.42	1.24
14	3473941	539	113	2.83	1.44	1.23
16	–	1301	211	–	1.45	1.25
18	–	3056	363	–	1.46	1.26
20	–	7276	676	–	1.47	1.27
22	–	18094	1219	–	1.48	1.28
24	–	39135	1641	–	1.48	1.26

Where do Heuristics Come From?

- Convert the original problem into a simpler one.
  - E.g., A decomposable problem.
- Solve the simple problem using a specialized method.

  E.g., By decomposition/recomposition.
- Use the solution of the simple problem as a guide to solving the original problem.
  - E.g., Let heuristic evaluation function h(s) compute the length of the solution to the simple problem.

Constraint Satisfaction Problem (CSP)

- Given:
  - A set of variables: x1 … xn
  - A finite domain of values for each variable: D1 …Dn
  - A set of constraints: C1 … Cm
- Find: An assignment of values to variables that satisfies the constraints.

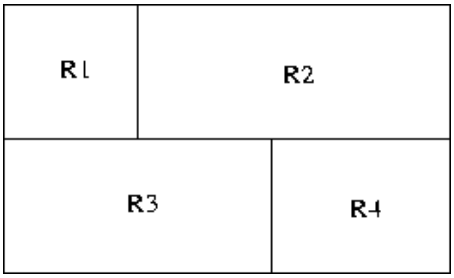
Definition of Constraint

- A constraint Ci references a set of variables:
  Xi1 … Xik
- Ci can be viewed as a function
  Di1 × Di2 × … Dik => Boolean
- Ci is satisfied by xi1 … xik if
  Ci(xi1, … , xik) = True
- A set of constraints C is satisfied if and only if for all Ci in C, Ci is satisfied

4-Queens Problem as a CSP

- Formulation of problem: for each column, choose a row in which to put a queen.
  - Variables: The columns A, B, C, D
  - Values: The rows: 1, 2, 3, 4
  - Domains: The domain of each variable is the set 1, 2, 3, 4
  - Constraints: For each pair of columns, the queens in those columns do not attack each other

House Floorplanning Problem

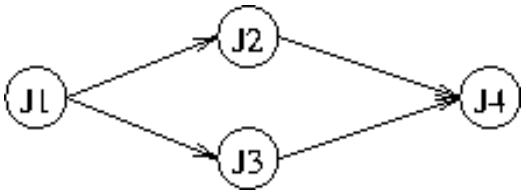


- **Design the floor plan of a house by choosing sizes and locations offour rooms so that:**
  - **Each room is at least 10 feet wide and 10 feet long.**
  - **Each room is at most 20 feet wide and 20 feet long.**
  - **Each room lies entirely inside the house.**
  - **No room overlaps another room.**
  - **The rooms entirely fill the house.**

Job Scheduling Problem

- **Assign a starting time to each of four jobs so that:**
  - **No two jobs are running at one time.**
  - **No job starts before its predecessors finish.**
  - **Each job finishes by its own deadline.**

Job	Duration	Deadline
J1	4	5
J2	12	20
J3	6	25
J4	9	35

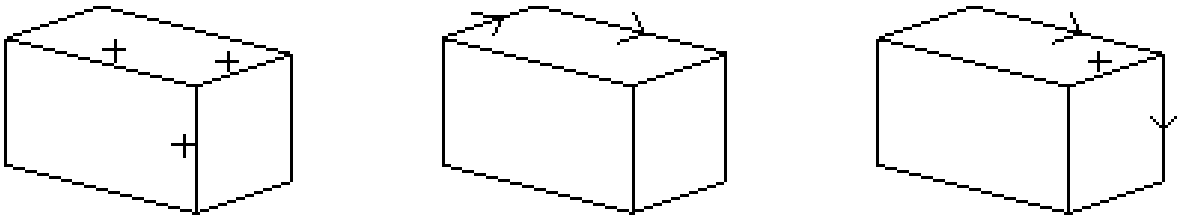


Cryptarithmetic Problem

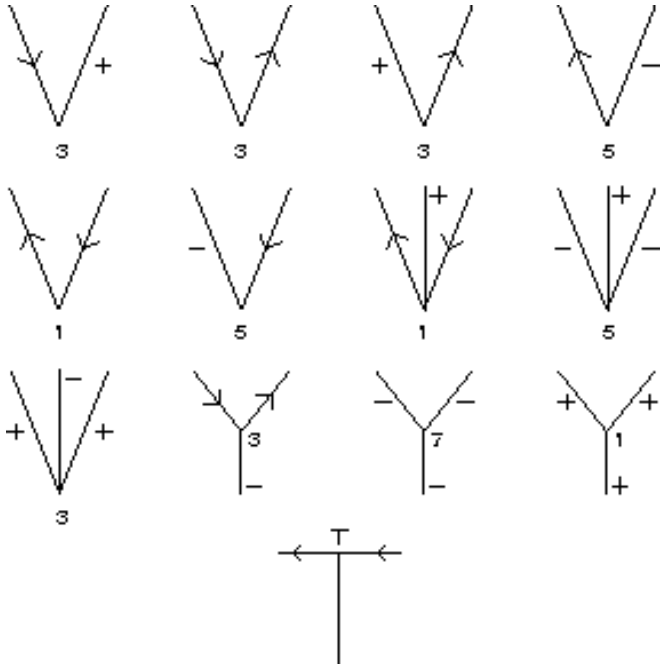
```
  SEND
+MORE
MONEY
```

- **Assign a digit 0 .. 9 to each letter so that:**
  - **All appearances of one letter get the same digit.**
  - **No two different letters get the same digit.**
  - **The sum is correct.**

Trihedral World



Trihedral World



Examples of Constraint Satisfaction Problems

- **N-Queens.**
- **House Floorplanning.**
- **Job scheduling.**
- **Crypt-Arithmetic.**
- **Scene Labeling.**
- **VLSI Layout and routing.**
- **Graph Coloring.**
- **Bin Packing.**

Arity of Constraints

- **Unary Constraint: References one variable.**
- **Binary Constraint: References two variables.**
- **N-Ary Constraint: References N variables.**
- **Global Constraint: References all variables.**

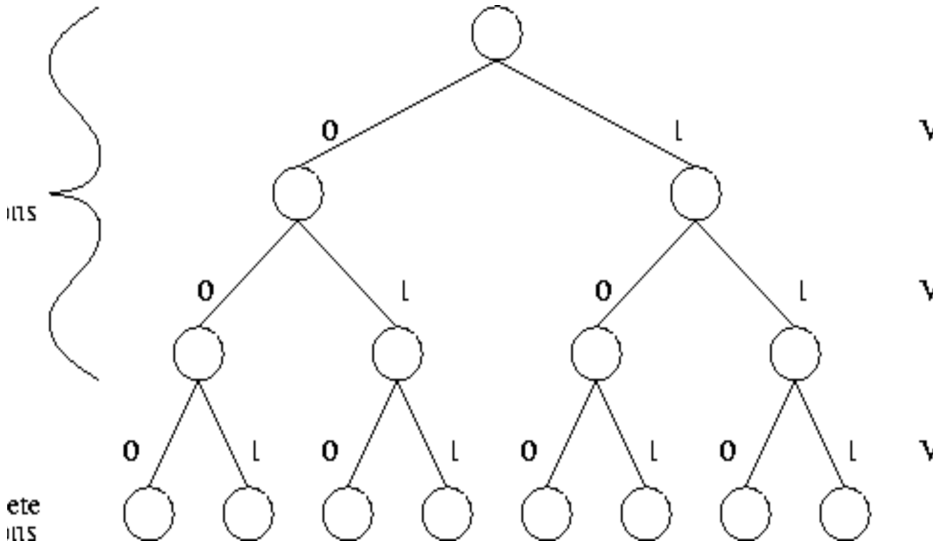
Binary CSPs

- **All constraints are unary or binary.**
- **Can be represented as a graph:**
  - **Vertices represent variables.**
  - **Edges represent constraints.**
- **The most widely studied type of CSP.**

CSP Tree v. State Space

- **Nature of the solution:**
  - **CSP: Solution is a (leaf) node.**
  - **State Space: Solution is a path.**
- **Location of solution:**
  - **CSP: Solutions are at a known depth.**
  - **State Space: Solutions may be at any depth.**
- **Source of complexity:**
  - **CSP: Complexity of constraints.**
  - **State Space: Complexity of space.**

CSP Search Tree



Properties of CSP Algorithms

- **A CSP algorithm is sound if every tuple it returns is actually a solution to the CSP.**
- **A CSP algorithm is complete if it returns some solution to the CSP whenever a solution exists.**

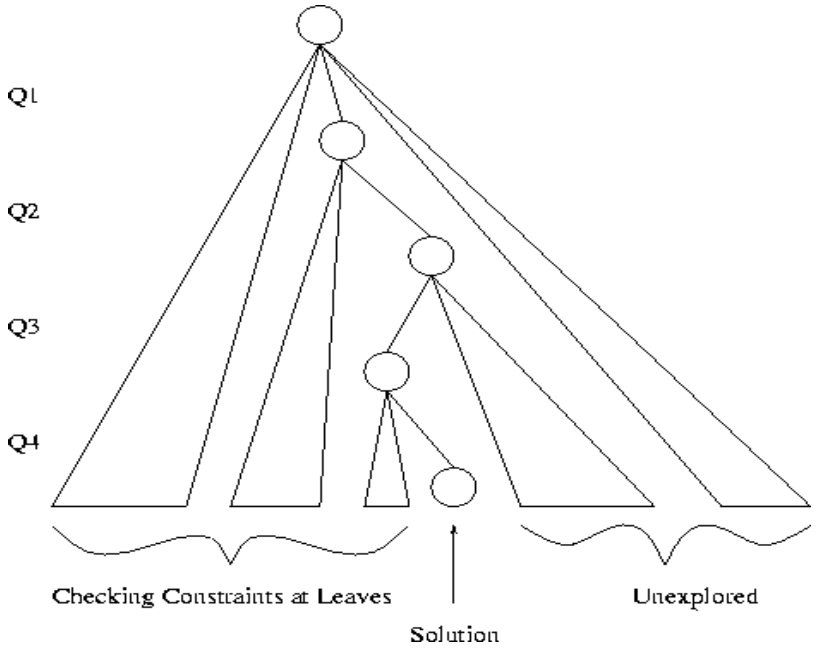
Simple Backtracking: ``Generate and Test''

```
Search(State, Variables, Domains, Constraints):
;;State = Assignments of values to some variables.
;;Variables = Set of unbound variables.
;;Domains = Sets of possible values for each variable.
;;Constraints = Tests on legal combinations of values.
If (Variables is empty)
  Then If (State satisfies Constraints)
       Then Return(State)
       Else Return(Failure)
  Else 1. Let V = First(Variables).
       2. Let D = First(Domains).
       3. Let V-Rest = Rest(Variables).
       4. Let D-Rest = Rest(Domains).
       5. Return(Successors(State,V,D,V-Rest,D-Rest,Constraints)).
```

Simple Backtracking

```
Successors(State,Variable,Domain,V-Rest,D-Rest,Constraints)
If (Domain is empty) Then Return(Failure)
  Else 1. Let W = First(Domain).
       2. Let New-State = Assign(V,W,State).
       3. Let Answer = Search(New-State,V-Rest,D-Rest,Constraints).
       4. If (Answer is not Failure)
            Then Return(Answer)
            Else a. Let Remaining = Rest(Domain).
                 b. Return(Successors(State,
                                      Variable,
                                      Remaining,
                                       V-Rest,
                                        D-Rest,
                                      Constraints)).
```


Simple Backtracking: 4-Queens



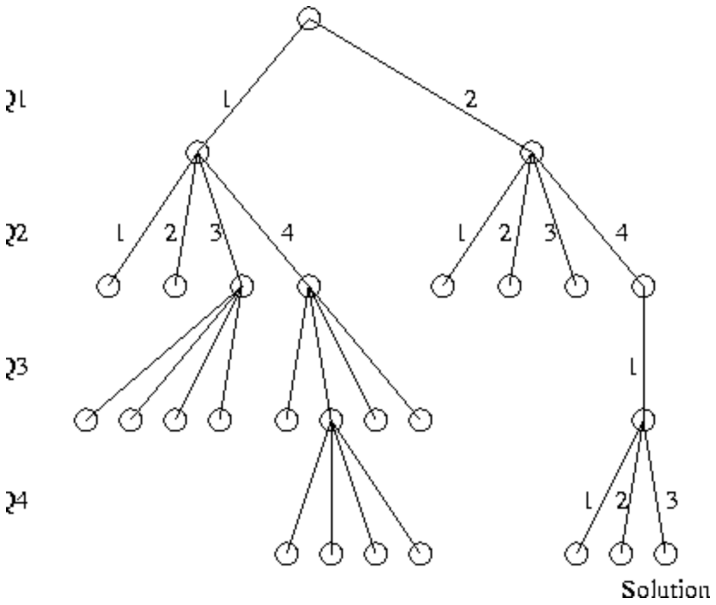
Backtracking with Early Pruning

Search(State, Variables, Domains, Constraints):

1. **Let Ready-Constraints = Constraints ready for evaluation in State.**
2. **If (State does not satisfy Ready-Constraints)**
   Then Return(Failure)

   Else If (Variables is empty)
   Then Return(State)
   Else 1. Let V = First(Variables).
   2. Let D = First(Domains).
   3. Let V-Rest = Rest(Variables).
   4. Let D-Rest = Rest(Domains).
   5. Return(Successors(State,V,D,V-Rest,D-Rest,Constraints)).

Backtracking with Early Pruning on 4-Queens



Constraint Propagation

- **Achieves ``local consistency''.**
- **Various types of local consistency:**
  - **Node consistency.**
  - **Arc consistency.**
  - **Directional arc consistency.**
  - **...More ...**

Node Consistency

- **A CSP is ``node consistent'' if for every variable xi and every value v ∈ Di, the value v satisfies the unary constraint on variable xi**
- **Achieve node consistency by pre-processing unary constraints.**

Directional Arc Consistency

- **A (binary) CSP is ``directionally arc consistent'' w.r.t. an ordering of variables x1 … xn if**
  For all xi, xj with i < j,
  For all vi in Di there exists a Vj in Dj s.t.
  Cij(vi, vj) is satisfied

Arc Consistency

- **A (binary) CSP is ``arc consistent'' if for all orderings it is directionally arc consistent.**

Forward Checking

- **Each time a variable vi is assigned a value, restrict the domains of unassigned variables to achieve:**
  - **Arc consistency between vi and each vj j>i**
  - **(I.e., Consistency between current variable and all future variables.)**
- **Prune the search tree if any variable domain is reduced to the empty set.**

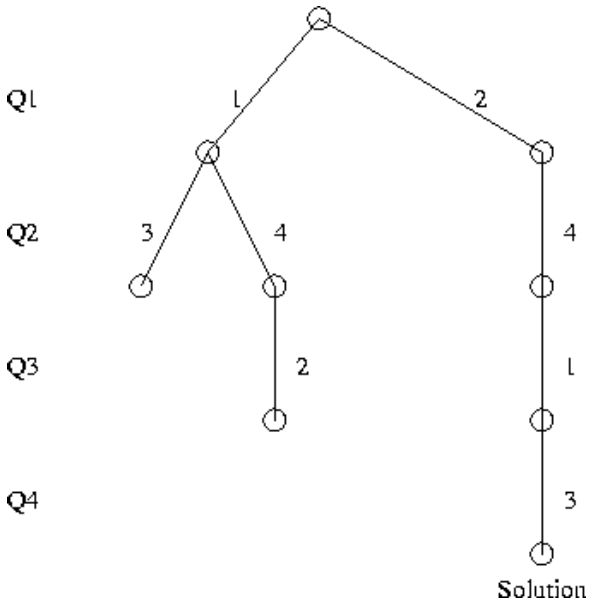
Forward Checking

```
Successors(State,Variable,Domain,V-Rest,D-Rest,Constraints)
 If (Domain is empty) Then Return(Failure)
  Else 1. Let W = First(Domain).
       2. Let New-State = Assign(V,W,State).
       3. Let Restricted = Check-Forward(V,W,V-Rest,D,Rest,Constraints)
       4. If (Restricted = Failure) Then Return(Failure)
      Else a. Let Answer = Search(New-State, V-Rest,
                                  Restricted, Constraints)
           b.  If (Answer is not Failure) Then Return(Answer)
           Else 1. Let Remaining = Rest(Domain).
                2. Return(Successors(State, Variable, Remaining, V-Rest,
                                     D-Rest, Constraints).
```

Forward Checking - Continued

Check-Forward(V,W,V-Rest,D-Rest,Constraints)
;;V is a variable just assigned a value.
;;W is the value it was assigned.
;;V-Rest is a list of unassigned variables. (Vi)
;;D-Rest is a list of domains of unassigned variables. (Di)
1. For each variable Vi on V-Rest do
a. Let C be the constraint between V and Vi.
b. Let Di be the ith member of D-Rest.
c. Let New-Di be the set of values X in Di such that C(W,X).
2. If (For any i, New-Di is empty)
Then Return(Failure)
Else Return(List of all New-Di's).

Forward Checking: 4-Queens



Full Lookahead

- **Each time a variable vi is assigned a value, restrict the domains of unassigned variables to achieve:**
  - **Arc consistency between vi and each vj, j>i**

    (I.e., Consistency between current variable and all future variables.)
  - **Arc consistency between each vk and vl, k > i, l > i**
  - **Prune the search tree if any variable domain is reduced to the empty set.**

Heuristics for CSPs

- **Variable Ordering Heuristics:**
  - **Smallest domain first.**
  - **Most constrained variable first.**
  - **Least constrained variable first.**
- **Value Ordering Heuristics:**
  - **Most constraining value first.**
  - **Least constraining value first.**

Humans in the Loop

- Human user selects a branch of the search tree.
- System performs full constraint propagation.
- Paradigm for interactive configuration problems:
  - E.g., Interactive house design.
  - E.g., Interactive student course scheduling.