472 Recitation

Week 7

Constraint Satisfaction Problem

CSP formulation:

- A set of variables $\mathcal{X} = \{X_1, \dots, X_n\}$.
- A set of domains $\mathcal{D} = \{D_1, \dots, D_n\}$.
- A set of constraints $\mathcal{C} = \{C_1, \dots, C_m\}$ that specifies allowable combination of values.

Solve CSP:

- Constraint Propagation (arc-consistency)
- Backtracking

Backtracking

```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })                              Start with an empty assignment

function BACKTRACK(csp, assignment) returns a solution or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
  for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
     if value is consistent with assignment then
        add {var = value} to assignment
        inferences ← INFERENCE(csp, var, assignment)    // arc-, path-, or k-consistency
                                                         // forward checking, etc.
        if inferences ≠ failure then
           add inferences to csp
           result ← BACKTRACK(csp, assignment)           Recursive
           if result ≠ failure then return result
           remove inferences from csp
        remove {var = value} from assignment
  return failure
```

Variable & Value Selection

Variable Selection (fail-first)

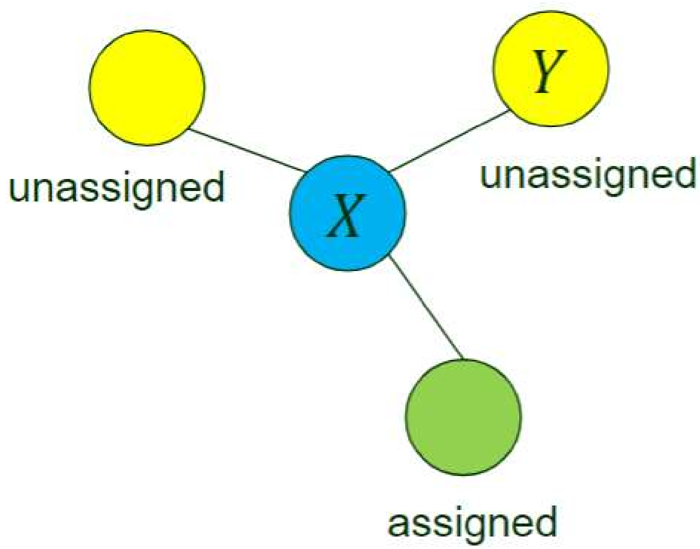
- *Minimum Remaining Values (MRV):* Choose the variable with the fewest“legal” values.
- Use the *degree heuristic* as a tie-breaker or at the start

Value Selection (fail-last)

- For the selected variable, choose its value that rules out the fewest choices for the neighboring variables in the constraint graph

Forward Checking

After assign a value for one variable X, modify the domain of the unassigned variables connected to X.



Assignment $X = v$

For every unassigned Y connected to X, delete any value from Y's domain that is inconsistent with v.

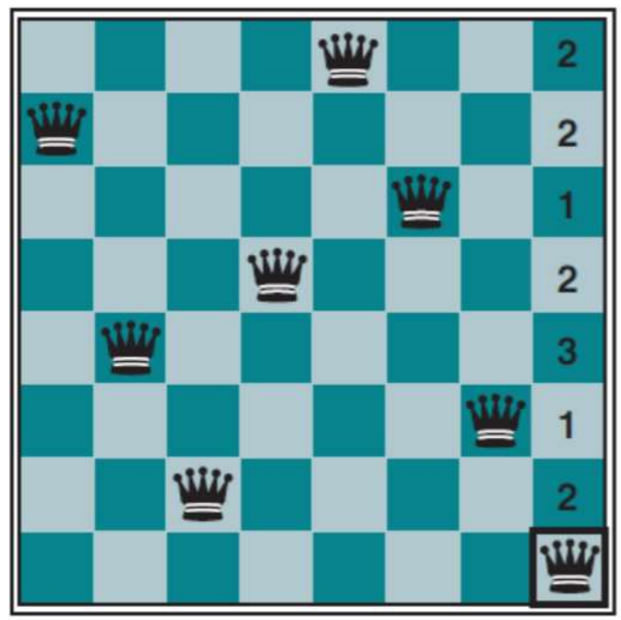
Local Search

- Every state corresponds to a complete assignment.
- Search changes the value of one variable at a time.

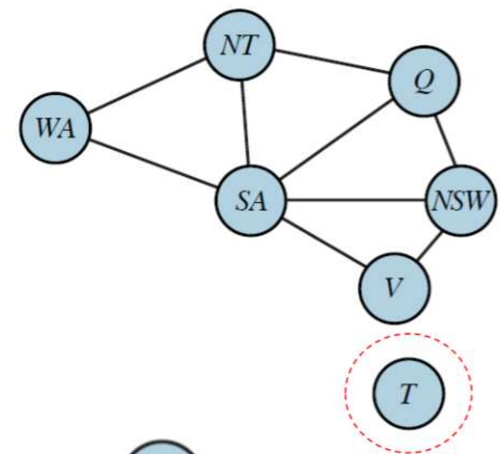
Back tracking starts
with empty assignment

Min-conflicts heuristic:

- Start with a complete assignment.
- Randomly choose a conflicted variable.
- Select the value that results in the least conflicts with other variables



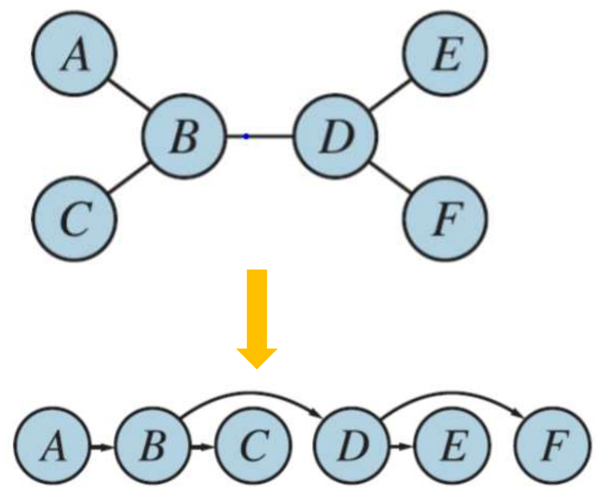
Structure of CSP



Independent subproblems

- Connected components in the constraint graph
- Each subproblem can be solved independently

Tree-Structured CSP:



- Generate a topological order of the variables $O(n)$
- Visit variables in the order and modify their domain based on arc-consistency $O(nd^2)$
- Visit variables again and assign values $O(n)$

Knowledge-based agents

Intelligent agents need *knowledge about the world* in order to carry out reasoning for good decision making.

function KB-AGENT(*percept*) **returns** an *action*
  persistent: KB, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(*percept*, t))
  action $\leftarrow$ ASK(KB, MAKE-ACTION-QUERY(t)) // asks what action // it should perform.
  TELL(KB, MAKE-ACTION-SENTENCE(*action*, t)) // tells what action // was chosen
  $t \leftarrow t + 1$
  return *action*

Inference: Derive new sentences from old

Logic

- A systematic study of rules of inference.
- A formal language for representing information such that conclusions can be drawn.

- *Syntax* – what expressions are legal (*well-formed* sentences)
- *Semantics* – what the "meanings" of sentences are.

Model m: assigns values to variables.

m *satisfies* a sentence α, or m is a model of α, if α is true in m.

$M(\alpha)$: set of all models of α.

$$\alpha \vDash \beta \quad \text{if and only if } M(\alpha) \subseteq M(\beta)$$

The sentence α entails the sentence β

Logic

Syntax	Semantics
If S, S_1 and S_2 are sentences:	

$\neg S$ is a sentence	negation
$S_1 \wedge S_2$ is a sentence	conjunction
$S_1 \vee S_2$ is a sentence	disjunction
$S_1 \implies S_2$ is a sentence	implication
$S_1 \iff S_2$ is a sentence	biconditional

$\neg S$	is true iff	S is false	
$S_1 \wedge S_2$	is true iff	S_1 is true and	S_2 is true
$S_1 \vee S_2$	is true iff	S_1 is true or	S_2 is true
$S_1 \Rightarrow S_2$	is true iff	S_1 is false or	S_2 is true
i.e.,	is false iff	S_1 is true and	S_2 is false
$S_1 \Leftrightarrow S_2$	is true iff	$S_1 \Rightarrow S_2$ is true and $S_2 \Rightarrow S_1$ is true	

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Inference – Model Checking

Q. Does $KB \models \neg P_{1,2}$?

Enumerate the models of KB and check if $\neg P_{1,2}$ is true in every model.

128 rows

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	<u>*true*</u>
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Validity: A sentence is valid if it is true in all models

Satisfiability:

- A sentence is satisfiable if it is true in, or satisfied by, some model.
- A sentence is unsatisfiable if it is false in all models

Inference Rules

Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

And-elimination

$$\frac{\alpha \wedge \beta}{\alpha}$$

$$\frac{\neg(\alpha \vee \beta)}{\neg\alpha \wedge \neg\beta} \quad \text{De Morgan}$$

$$\frac{\neg\alpha \wedge \neg\beta}{\neg\beta} \quad \text{and-elimination}$$

$$\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) \quad \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) \quad \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) \quad \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) \quad \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) \quad \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) \quad \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) \quad \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) \quad \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}$$