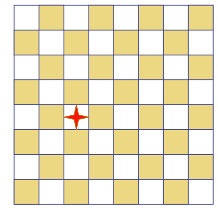
Constraint Satisfaction Problems

Russell and Norvig: Chapter 5

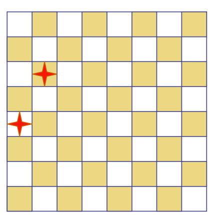
CMSC 421 – Fall 2003

Intro Example: 8-Queens



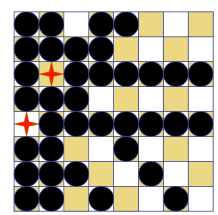
- Purely generate-and-test
- The "search" tree is only used to enumerate all possible 64^8 combinations

Intro Example: 8-Queens



Another form of generate-and-test, with no redundancies → "only" 8^8 combinations

Intro Example: 8-Queens



What is Needed?

- Not just a successor function and goal test
- But also a means to propagate the constraints imposed by one queen on the others and an early failure test
- → Explicit representation of constraints and constraint manipulation algorithms

Constraint Satisfaction Problem

- Set of variables $\{X_1, X_2, \ldots, X_n\}$
- Each variable X_i has a domain D_i of possible values
- Usually D_i is discrete and finite
- Set of constraints $\{C_1, C_2, \ldots, C_p\}$
- Each constraint C_k involves a subset of variables and specifies the allowable combinations of values of these variables

Constraint Satisfaction Problem

- Set of variables {X1, X2, …, Xn}
- Each variable Xi has a domain Di of possible values
- Usually Di is discrete and finite
- Set of constraints {C1, C2, …, Cp}
- Each constraint C_k involves a subset of variables and specifies the allowable combinations of values of these variables

- Assign a value to every variable such that all constraints are satisfied

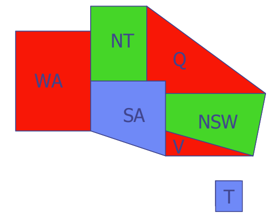
Example: 8-Queens Problem

- 64 variables X_{ij}, $i = 1$ to 8, $j = 1$ to 8
- Domain for each variable {yes,no}
- Constraints are of the forms:
  - X_{ij} = yes → X_{ik} = no for all $k = 1$ to 8, $k \neq j$
  - X_{ij} = yes → X_{kj} = no for all $k = 1$ to 8, $k \neq I$
  - Similar constraints for diagonals

Example: 8-Queens Problem

- 8 variables X_i, i = 1 to 8
- Domain for each variable {1,2,…,8}
- Constraints are of the forms:
  - $X_i = k \rightarrow X_j \neq k$ for all j = 1 to 8, $j \neq i$
  - Similar constraints for diagonals

Example: Map Coloring



- 7 variables {WA,NT,SA,Q,NSW,V,T}
- Each variable has the same domain {red, green, blue}
- No two adjacent variables have the same value:

WA≠NT, WA≠SA, NT≠SA, NT≠Q, SA≠Q, SA≠NSW, SA≠V,Q≠NSW, NSW≠V

Example: Street Puzzle

1 2 3 4 5

Ni = {English, Spaniard, Japanese, Italian, Norwegian}
Ci = {Red, Green, White, Yellow, Blue}
Di = {Tea, Coffee, Milk, Fruit-juice, Water}
Ji = {Painter, Sculptor, Diplomat, Violonist, Doctor}
Ai = {Dog, Snails, Fox, Horse, Zebra}

Example: Street Puzzle

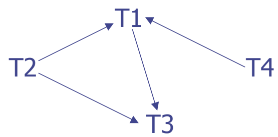
1 2 3 4 5

Ni = {English, Spaniard, Japanese, Italian, Norwegian}
Ci = {Red, Green, White, Yellow, Blue}
Di = {Tea, Coffee, Milk, Fruit-juice, Water}
Ji = {Painter, Sculptor, Diplomat, Violonist, Doctor}
Ai = {Dog, Snails, Fox, Horse, Zebra}

The Englishman lives in the Red house
The Spaniard has a Dog
The Japanese is a Painter
The Italian drinks Tea
The Norwegian lives in the first house on the left
The owner of the Green house drinks Coffee
The Green house is on the right of the White house
The Sculptor breeds Snails
The Diplomat lives in the Yellow house
The owner of the middle house drinks Milk
The Norwegian lives next door to the Blue house
The Violonist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

Who owns the Zebra?
Who drinks Water?

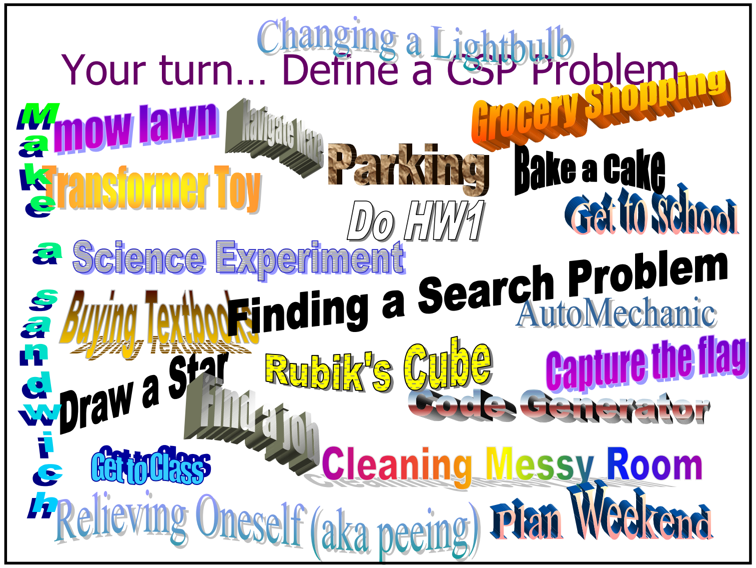
Example: Task Scheduling



T1 must be done during T3
T2 must be achieved before T1 starts
T2 must overlap with T3
T4 must start after T1 is complete

- Are the constraints compatible?
- Find the temporal relation between every two tasks

Your turn... Define a CSP Problem



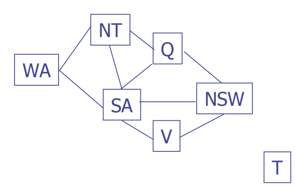
Constraint Satisfaction Problem

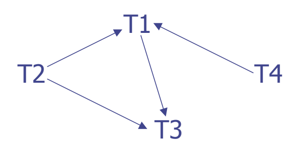
- Set of variables {X1, X2, …, Xn}
- Each variable Xi has a domain Di of possible values
- Usually Di is discrete and finite
- Set of constraints {C1, C2, …, Cp}
- Each constraint C_k involves a subset of variables and specifies the allowable combinations of values of these variables

- Assign a value to every variable such that all constraints are satisfied

Constraint Graph

Binary constraints





Two variables are adjacent or neighbors if they are connected by an edge or an arc

CSP as a Search Problem

- Initial state: empty assignment
- Successor function: a value is assigned to any unassigned variable, which does not conflict with the currently assigned variables
- Goal test: the assignment is complete
- Path cost: irrelevant

CSP as a Search Problem

- Initial state: empty assignment
- Successor function: a value is assigned to any unassigned variable, which does not conflict with the currently assigned variables
- Goal test: the assignment is complete
- Path cost: irrelevant

n variables of domain size d
$\rightarrow$ $O(d^n)$ distinct complete assignments

Remark

- Finite CSP include 3SAT as a special case
- 3SAT is known to be NP-complete
- So, in the worst-case, we cannot expect to solve a finite CSP in less than exponential time

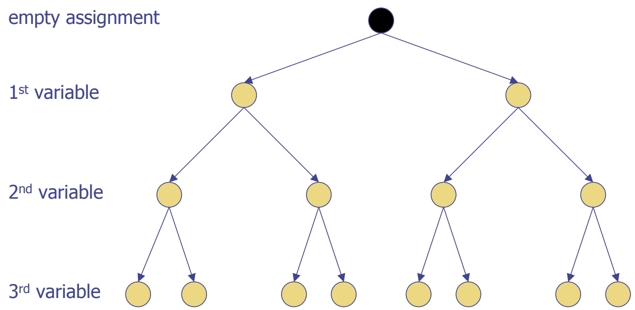
Commutativity of CSP

The order in which values are assigned to variables is irrelevant to the final assignment, hence:

1. Generate successors of a node by considering assignments for only one variable

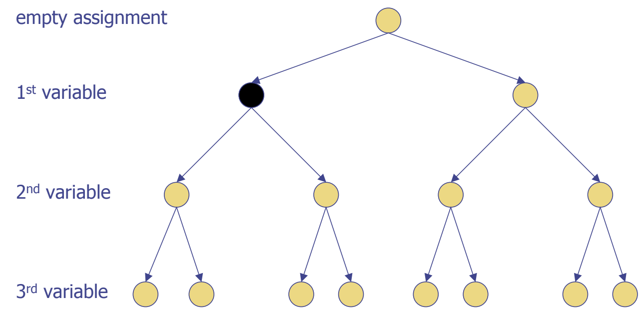
2. Do not store the path to node

→ Backtracking Search



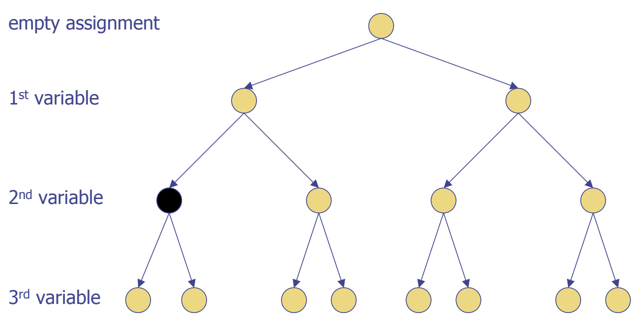
Assignment = {}

→ Backtracking Search



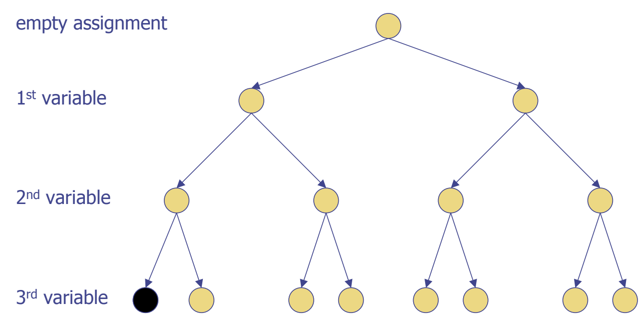
Assignment = {(var1=v11)}

→ Backtracking Search



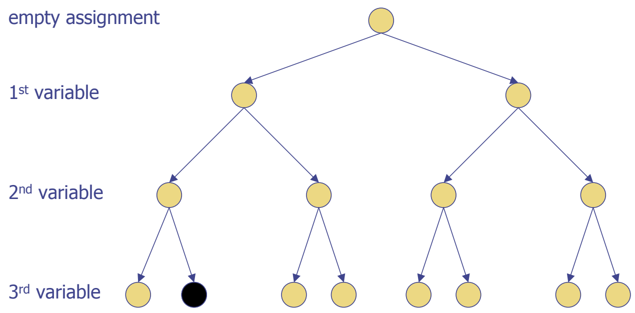
Assignment = {(var1=v11),(var2=v21)}

→ Backtracking Search



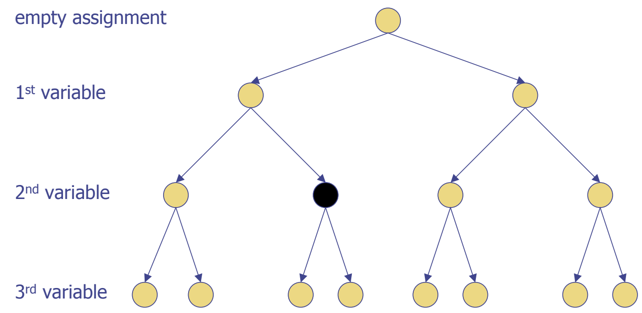
Assignment = {(var1=v11),(var2=v21),(var3=v31)}

→ Backtracking Search



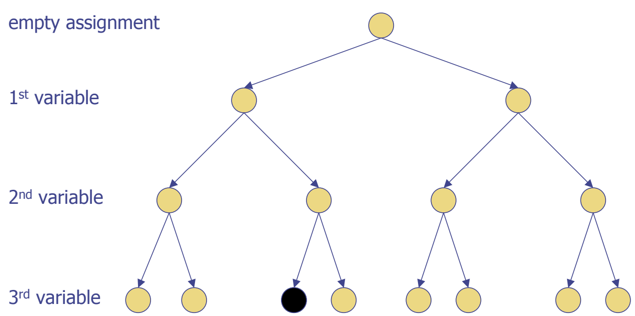
Assignment = {(var1=v11),(var2=v21),(var3=v32)}

→ Backtracking Search



Assignment = {(var1=v11),(var2=v22)}

→ Backtracking Search



Assignment = {(var1=v11),(var2=v22),(var3=v31)}

Backtracking Algorithm

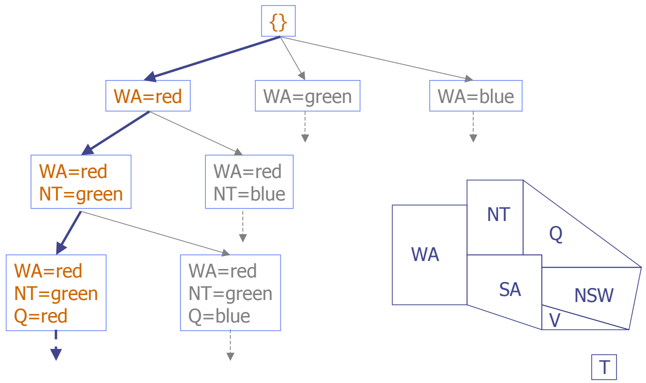
CSP-BACKTRACKING({})

CSP-BACKTRACKING(a)

partial assignment of variables

- If a is complete then return a
- X ← select unassigned variable
- D ← select an ordering for the domain of X
- For each value v in D do
  - If v is consistent with a then
    - Add (X= v) to a
    - result ← CSP-BACKTRACKING(a)
    - If result ≠ *failure* then return result
- Return *failure*

Map Coloring



Questions

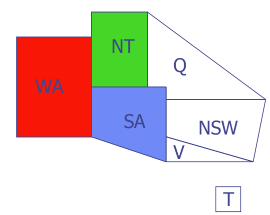
1. Which variable X should be assigned a value next?
2. In which order should its domain D be sorted?

Questions

1. Which variable X should be assigned a value next?
2. In which order should its domain D be sorted?
3. What are the implications of a partial assignment for yet unassigned variables?

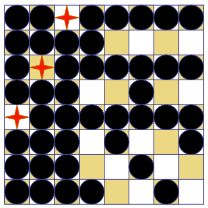
Choice of Variable

- Map coloring



Choice of Variable

- 8-queen

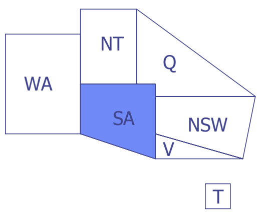


Choice of Variable

Minimum remaining values (MRV)/Most-constrained-variable heuristic:

Select a variable with the fewest remaining values

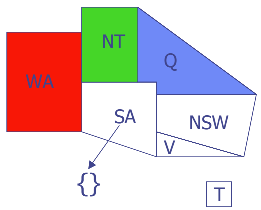
Choice of Variable



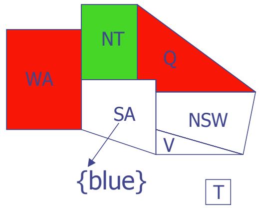
Degree Heuristic/Most-constraining-variable heuristic:

Select the variable that is involved in the largest number of constraints on other unassigned variables

Choice of Value



Choice of Value

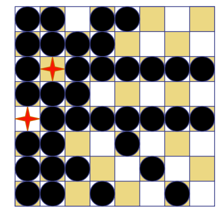


Least-constraining-value heuristic:

Prefer the value that leaves the largest subset of legal values for other unassigned variables

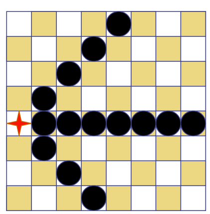
Constraint Propagation ...

... is the process of determining how the possible values of one variable affect the possible values of other variables

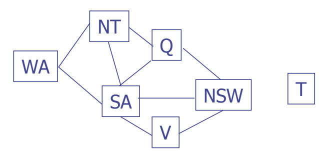


Forward Checking

After a variable X is assigned a value v, look at each unassigned variable Y that is connected to X by a constraint and deletes from Y's domain any value that is inconsistent with v

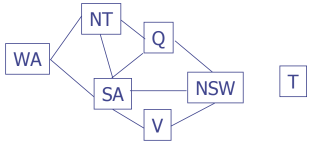


Map Coloring



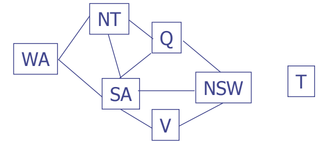
WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB

Map Coloring



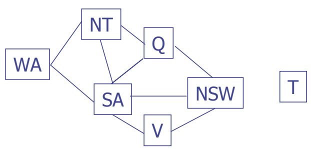
WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB

Map Coloring



WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	B	G	RB	RGB	B	RGB

Map Coloring



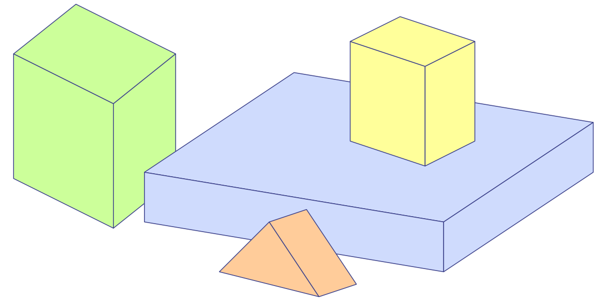
Impossible assignments that forward checking do not detect

WA	NT	Q				
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	B	G	RB	RGB	B	RGB
R	B	G	R	B		RGB

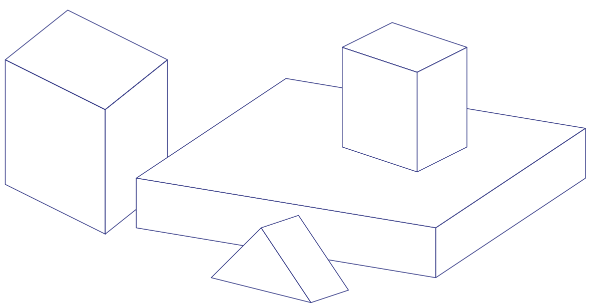
→ constraint propagation

Early Application: Edge Labeling in Computer Vision

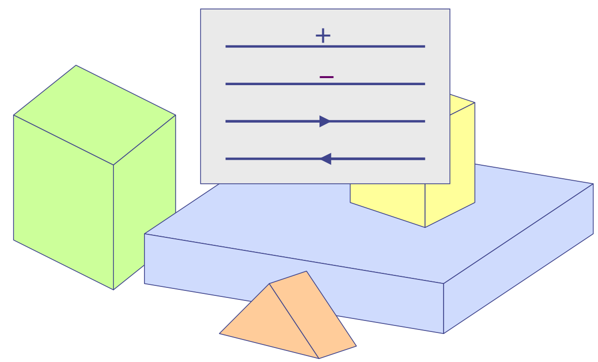
Edge Labeling



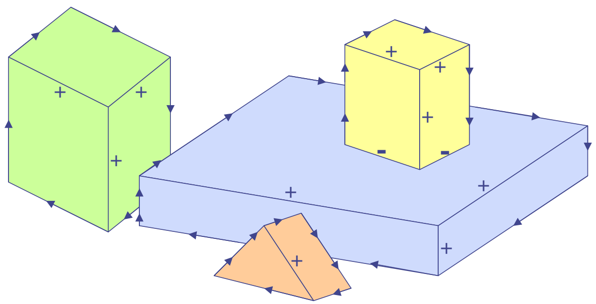
Edge Labeling



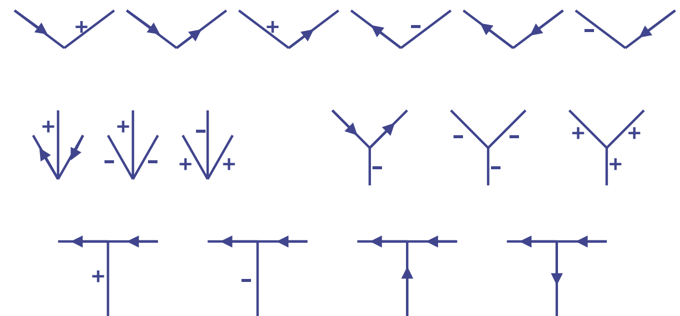
Edge Labeling



Edge Labeling



Junction Label Sets

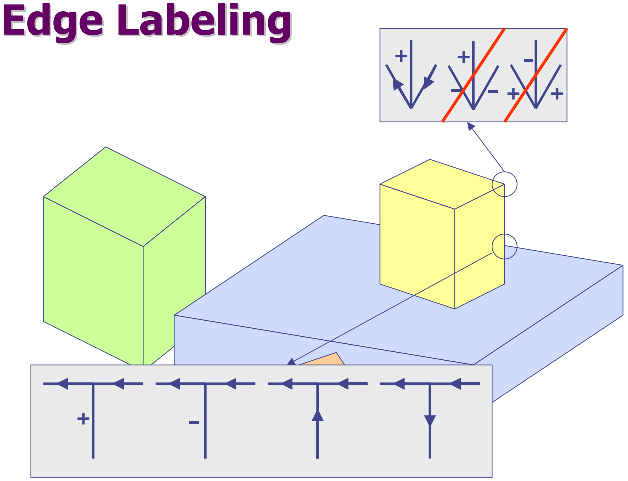


(Waltz, 1975; Mackworth, 1977)

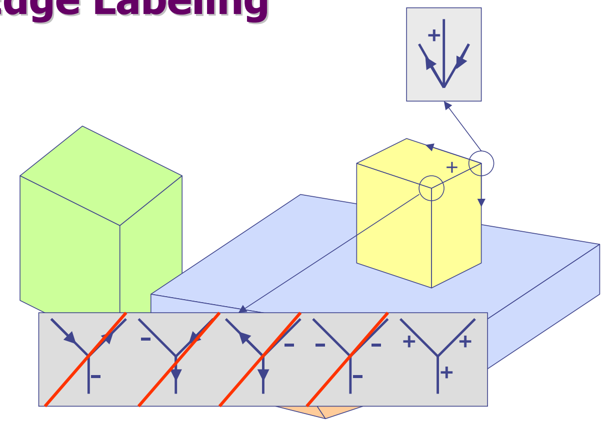
Edge Labeling as a CSP

- A variable is associated with each junction
- The domain of a variable is the label set of the corresponding junction
- Each constraint imposes that the values given to two adjacent junctions give the same label to the joining edge

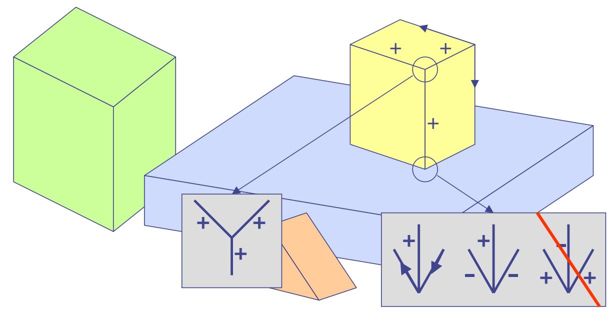
Edge Labeling



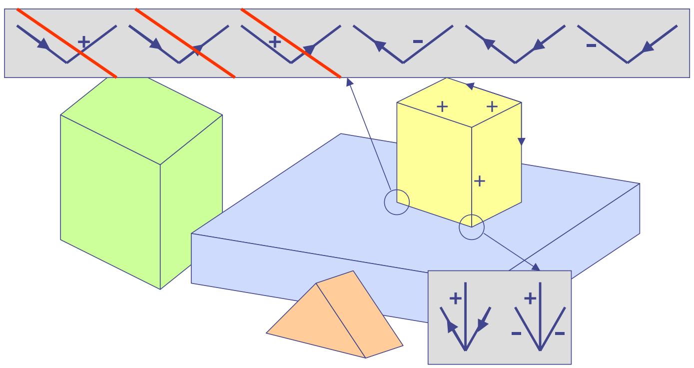
Edge Labeling



Edge Labeling



Edge Labeling



Removal of Arc Inconsistencies

REMOVE-ARC-INCONSISTENCIES(J,K)

- removed ← *false*
- X ← label set of J
- Y ← label set of K
- For every label y in Y do
  - If there exists no label x in X such that the constraint (x,y) is satisfied then
    - Remove y from Y
    - If Y is empty then contradiction ← *true*
    - removed ← *true*
- Label set of K ← Y
- Return removed

CP Algorithm for Edge Labeling

- Associate with every junction its label set
- contradiction ← *false*
- Q ← stack of all junctions
- while Q is not empty and not contradiction do
  - J ← UNSTACK(Q)
  - For every junction K adjacent to J do
    - If REMOVE-ARC-INCONSISTENCIES(J,K) then
      - STACK(K,Q)

(Waltz, 1975; Mackworth, 1977)

General CP for Binary Constraints

Algorithm AC

- contradiction ← *false*
- Q ← stack of all variables
- while Q is not empty and not contradiction do
  - X ← UNSTACK(Q)
  - For every variable Y adjacent to X do
    - If REMOVE-ARC-INCONSISTENCIES(X,Y) then
      - STACK(Y,Q)

General CP for Binary Constraints

Algorithm AC

- contradiction ← *false*
- Q ← stack of all variables
- while Q is not empty and not contradiction do
  - X ← UNSTACK(Q)
  - For every variable Y adjacent to X do
    - If REMOVE-ARC-INCONSISTENCY(X,Y) then
      - STACK(Y,Q)

REMOVE-ARC-INCONSISTENCY(X,Y)

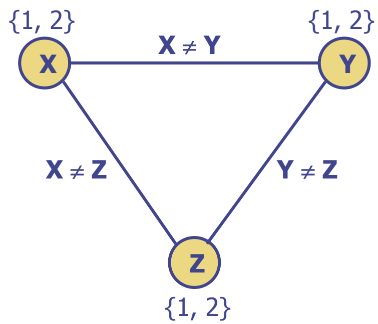
- removed ← *false*
- For every value y in the domain of Y do
  - If there exists no value x in the domain of X such that the constraints on (x,y) is satisfied then
    - Remove y from the domain of Y
    - If Y is empty then contradiction ← *true*
    - removed ← *true*
- Return removed

Complexity Analysis of AC3

- n = number of variables
- d = number of values per variable
- s = maximum number of constraints on a pair of variables
- Each variables is inserted in Q up to d times
- REMOVE-ARC-INCONSISTENCY takes $O(d^2)$ time
- CP takes $O(n\ s\ d^3)$ time

Is AC3 All What is Needed?

NO!



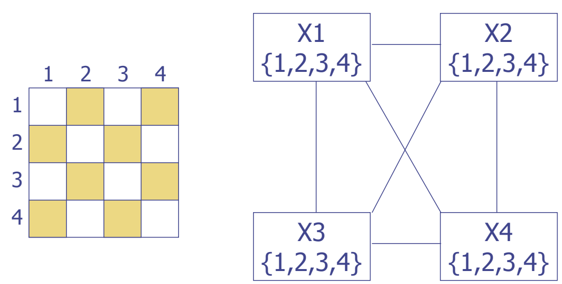
Solving a CSP

Interweave constraint propagation, e.g.,
- forward checking
- AC3

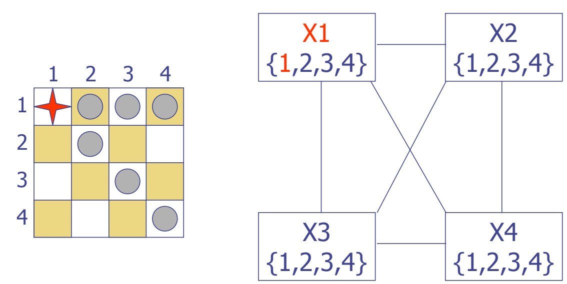
and backtracking

+ Take advantage of the CSP structure

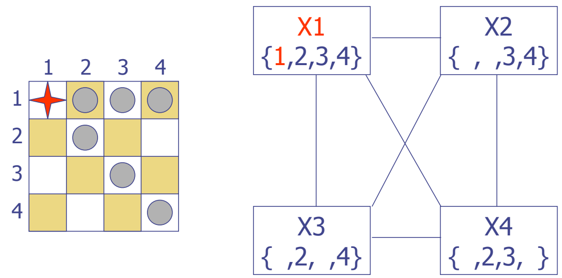
4-Queens Problem



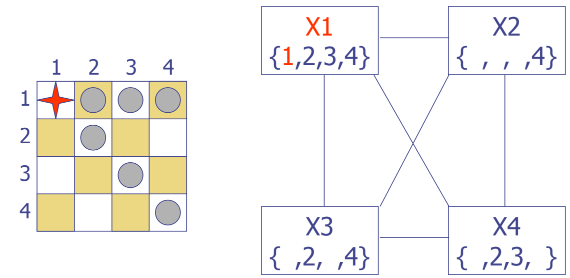
4-Queens Problem



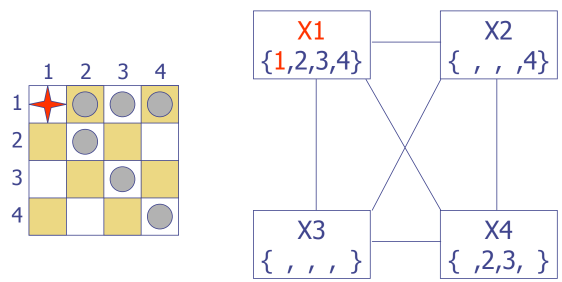
4-Queens Problem



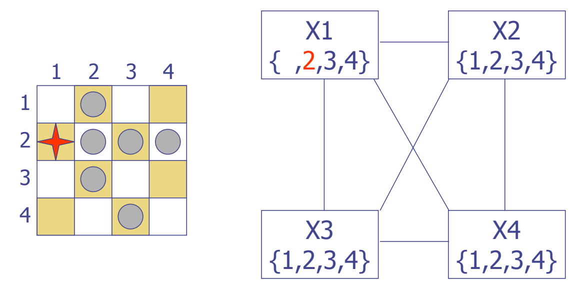
4-Queens Problem



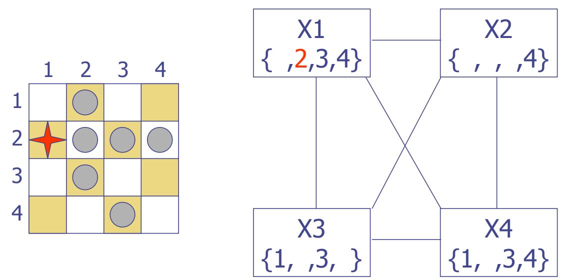
4-Queens Problem



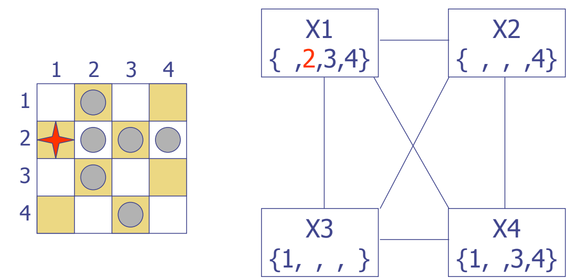
4-Queens Problem



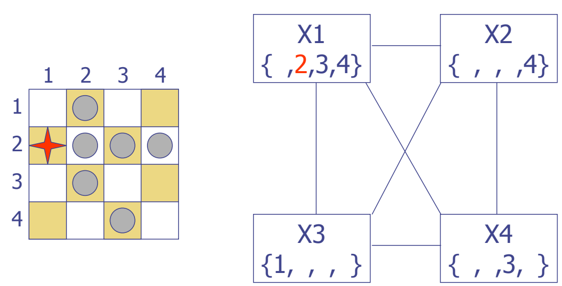
4-Queens Problem



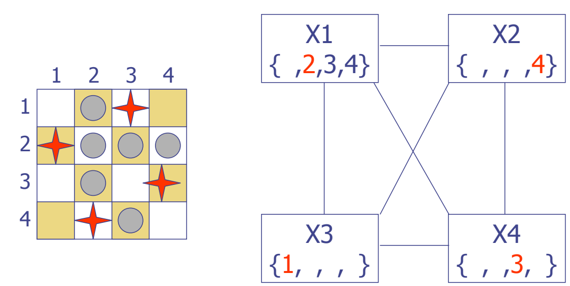
4-Queens Problem



4-Queens Problem

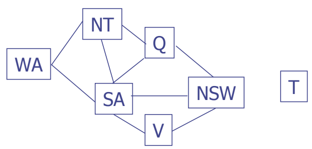


4-Queens Problem



Structure of CSP

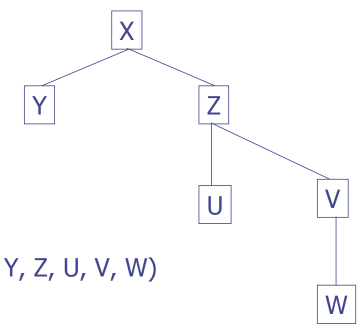
- If the constraint graph contains multiple components, then one independent CSP per component



Structure of CSP

- If the constraint graph contains multiple components, then one independent CSP per component
- If the constraint graph is a tree (no loop), then the CSP can be solved efficiently

Constraint Tree

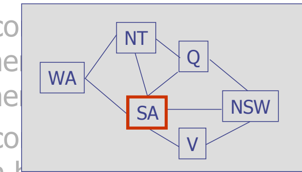


→ (X, Y, Z, U, V, W)

Constraint Tree

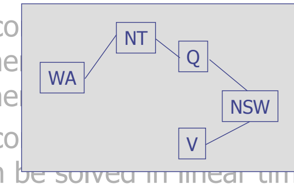
- Order the variables from the root to the leaves → $(X_1, X_2, \ldots, X_n)$
- For j = n, n-1, ..., 2 do
  REMOVE-ARC-INCONSISTENCY(X_j, X_i)
  where X_i is the parent of X_j
- Assign any legal value to X_1
- For j = 2, ..., n do
  - assign any value to X_j consistent with the value assigned to X_i, where X_i is the parent of X_j

Structure of CSP



- If the co... s multiple compone... dent CSP per compone...
- If the co... e, then the CSP can be solved efficiently
- Whenever a variable is assigned a value by the backtracking algorithm, propagate this value and remove the variable from the constraint graph

Structure of CSP



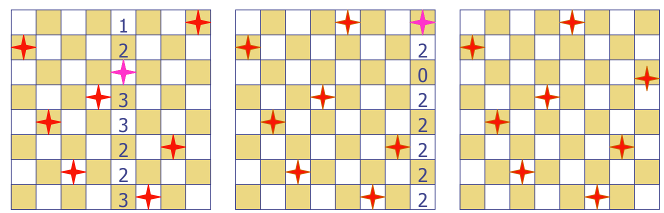
- If the co... s multiple compone... dent CSP per compone...
- If the co... e, then the CSP can be solved in linear time
- Whenever a variable is assigned a value by the backtracking algorithm, propagate this value and remove the variable from the constraint graph

Over-Constrained Problems

Weaken an over-constrained problem by:

- Enlarging the domain of a variable
- Loosening a constraint
- Removing a variable
- Removing a constraint

Local Search for CSP



Pick initial complete assignment (at random)
Repeat

- Pick a conflicted variable var (at random)
- Set the new value of var to minimize the number of conflicts
- If the new assignment is not conflicting then return it

(min-conflicts heuristics)

Remark

- Local search with min-conflict heuristic works extremely well for million-queen problems
- The reason: Solutions are densely distributed in the $O(n^n)$ space, which means that on the average a solution is a few steps away from a randomly picked assignment

Infinite-Domain CSP

- Variable domain is the set of the integers (discrete CSP) or of the real numbers (continuous CSP)
- Constraints are expressed as equalities and inequalities
- Particular case: Linear-programming problems

When to Use CSP Techniques?

- When the problem can be expressed by a set of variables with constraints on their values
- When constraints are relatively simple (e.g., binary)
- When constraints propagate well (AC3 eliminates many values)
- When the solutions are "densely" distributed in the space of possible assignments

Applications

- CSP techniques allow solving very complex problems
- Numerous applications, e.g.:
  - Crew assignments to flights
  - Management of transportation fleet
  - Flight/rail schedules
  - Task scheduling in port operations
  - Design
  - Brain surgery
- See **`www.ilog.com`**

→ Constraint Programming

"Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it."

Eugene C. Freuder, Constraints, April 1997

Summary

- Backtrack Search
- Variable and value ordering
- Constraint propagation
- Edge labeling in Computer Vision
- Local Search