# Constraint Satisfaction Problems (CSP)

(Where we postpone making difficult decisions until they become easy to make)

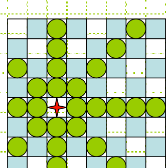
R&N: Chap. 5

Slides from Jean-Claude Latombe at Stanford University (used with permission)

## What we will try to do ...

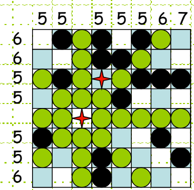
- Search techniques make choices in some order which often is arbitrary. Often little state information is available to make each of them (states are "black boxes")
- In many problems, the same states can be reached independent of the order in which choices are made ("commutative" actions)
- Can we solve such problems more efficiently by picking the order appropriately? Can we even avoid making any choice? Do we have all the information needed?

## Constraint Propagation



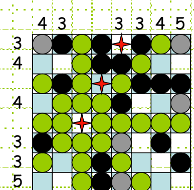
- Place a queen in a square
- Remove the attacked squares from future consideration

## Constraint Propagation



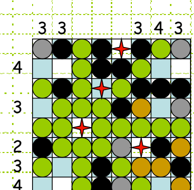

- Count the number of non-attacked squares in every row and column
- Place a queen in a row or column with minimum number
- Remove the attacked squares from future consideration

## Constraint Propagation

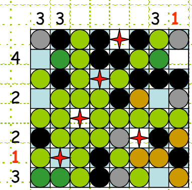



- Repeat

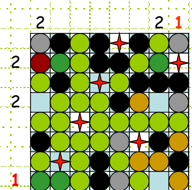
## Constraint Propagation



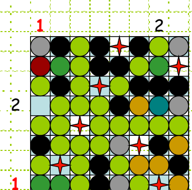
## Constraint Propagation



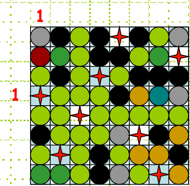

## Constraint Propagation



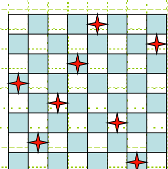

## Constraint Propagation



## Constraint Propagation




## Constraint Propagation



## What do we need?

- More than just a successor function and a goal test
- We also need:
  - A means to propagate the constraints imposed by one queen's position on the positions of the other queens
  - An early failure test

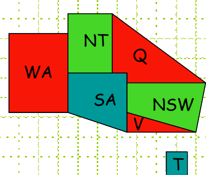
→ Explicit representation of constraints

→ Constraint propagation algorithms

## Constraint Satisfaction Problem (CSP)

- 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 finite
- Set of constraints $\{C_1, C_2, \ldots, C_p\}$
- Each constraint relates a subset of variables by specifying the valid combinations of their values
- Goal: Assign a value to every variable such that all constraints are satisfied

## 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

## 8-Queen Problem

- 8 variables $X_i$, i = 1 to 8
- The domain of each variable is: {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

All constraints are binary

## Street Puzzle

1 2 3 4 5

$N_i$ = {English, Spaniard, Japanese, Italian, Norwegian}
$C_i$ = {Red, Green, White, Yellow, Blue}
$D_i$ = {Tea, Coffee, Milk, Fruit-juice, Water}
$J_i$ = {Painter, Sculptor, Diplomat, Violinist, Doctor}
$A_i$ = {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 Violinist 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?

## Street Puzzle

1 2 3 4 5

$N_i$ = {English, Spaniard, Japanese, Italian, Norwegian}
$C_i$ = {Red, Green, White, Yellow, Blue}
$D_i$ = {Tea, Coffee, Milk, Fruit-juice, Water}
$J_i$ = {Painter, Sculptor, Diplomat, Violinist, Doctor}
$A_i$ = {Dog, Snails, Fox, Horse, Zebra}

$\forall i,j \in [1,5], i \neq j, N_i \neq N_j$
$\forall i,j \in [1,5], i \neq j, C_i \neq C_j$
...

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 Violinist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

## Street Puzzle

1 2 3 4 5

$N_i$ = {English, Spaniard, Japanese, Italian, Norwegian}
$C_i$ = {Red, Green, White, Yellow, Blue}
$D_i$ = {Tea, Coffee, Milk, Fruit-juice, Water}
$J_i$ = {Painter, Sculptor, Diplomat, Violinist, Doctor}
$A_i$ = {Dog, Snails, Fox, Horse, Zebra}

The Englishman lives in the Red house → $(N_i = \text{English}) \Leftrightarrow (C_i = \text{Red})$
The Spaniard has a Dog
The Japanese is a Painter → $(N_i = \text{Japanese}) \Leftrightarrow (J_i = \text{Painter})$
The Italian drinks Tea
The Norwegian lives in the first house on the left → $(N_1 = \text{Norwegian})$
The owner of the Green house drinks Coffee
The Green house is on the right of the White house → $\{(C_i = \text{White}) \Leftrightarrow (C_{i+1} = \text{Green}),\ (C_5 \neq \text{White}),\ (C_1 \neq \text{Green})$
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 → left as an exercise
The Violinist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

## Street Puzzle

1 2 3 4 5

$N_i$ = {English, Spaniard, Japanese, Italian, Norwegian}
$C_i$ = {Red, Green, White, Yellow, Blue}
$D_i$ = {Tea, Coffee, Milk, Fruit-juice, Water}
$J_i$ = {Painter, Sculptor, Diplomat, Violinist, Doctor}
$A_i$ = {Dog, Snails, Fox, Horse, Zebra}

The Englishman lives in the Red house → $(N_i = English) \Leftrightarrow (C_i = Red)$
The Spaniard has a Dog
The Japanese is a Painter → $(N_i = Japanese) \Leftrightarrow (J_i = Painter)$
The Italian drinks Tea
The Norwegian lives in the first house on the left → $(N_1 = Norwegian)$
The owner of the Green house drinks Coffee
The Green house is on the right of the White house → $(C_i = White) \Leftrightarrow (C_{i+1} = Green)$, $(C_5 \neq White)$, $(C_1 \neq Green)$
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 Violinist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

unary constraints

## Street Puzzle

1 2 3 4 5

$N_i$ = {English, Spaniard, Japanese, Italian, Norwegian}
$C_i$ = {Red, Green, White, Yellow, Blue}
$D_i$ = {Tea, Coffee, Milk, Fruit-juice, Water}
$J_i$ = {Painter, Sculptor, Diplomat, Violinist, Doctor}
$A_i$ = {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 → $N_1$ = Norwegian
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 → $D_3$ = Milk
The Norwegian lives next door to the Blue house
The Violinist drinks Fruit juice
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

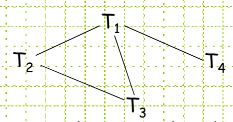
## Street Puzzle

1 2 3 4 5

$N_i$ = {English, Spaniard, Japanese, Italian, Norwegian}
$C_i$ = {Red, Green, White, Yellow, Blue}
$D_i$ = {Tea, Coffee, Milk, Fruit-juice, Water}
$J_i$ = {Painter, Sculptor, Diplomat, Violinist, Doctor}
$A_i$ = {Dog, Snails, Fox, Horse, Zebra}

The Englishman lives in the Red house → $C_1 \neq$ Red
The Spaniard has a Dog → $A_1 \neq$ Dog
The Japanese is a Painter
The Italian drinks Tea
The Norwegian lives in the first house on the left → $N_1$ = Norwegian
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 → $D_3$ = Milk
The Norwegian lives next door to the Blue house
The Violinist drinks Fruit juice → $J_3 \neq$ Violinist
The Fox is in the house next to the Doctor's
The Horse is next to the Diplomat's

## Task Scheduling




Four tasks $T_1$, $T_2$, $T_3$, and $T_4$ are related by time constraints:

- $T_1$ must be done during $T_3$
- $T_2$ must be achieved before $T_1$ starts
- $T_2$ must overlap with $T_3$
- $T_4$ must start after $T_1$ is complete

- Are the constraints compatible?
- What are the possible time relations between two tasks?
- What if the tasks use resources in limited supply?

How to formulate this problem as a CSP?

## 3-SAT

- n Boolean variables $u_1, \ldots, u_n$
- p constrains of the form

  $$u_i^* \vee u_j^* \vee u_k^* = 1$$

  where $u^*$ stands for either u or $\neg u$
- Known to be NP-complete

## Finite vs. Infinite CSP

- Finite CSP: each variable has a finite domain of values
- Infinite CSP: some or all variables have an infinite domain

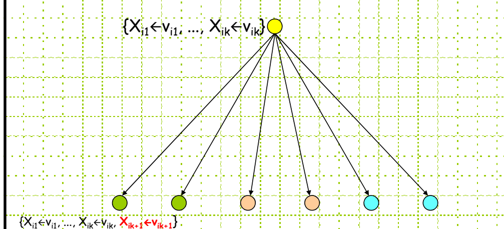
  E.g., linear programming problems over the reals:

  for $i = 1, 2, \ldots, p : a_{i,1}x_1+a_{i,2}x_2+\ldots+a_{i,n}x_n = a_{i,0}$

  for $j = 1, 2, \ldots, q : b_{j,1}x_1+b_{j,2}x_2+\ldots+b_{j,n}x_n \leq b_{j,0}$
- We will only consider finite CSP

# CSP as a Search Problem

- n variables $X_1, \ldots, X_n$
- Valid assignment: $\{X_{i1} \leftarrow v_{i1}, \ldots, X_{ik} \leftarrow v_{ik}\}$, $0 \le k \le n$, such that the values $v_{i1}, \ldots, v_{ik}$ satisfy all constraints relating the variables $X_{i1}, \ldots, X_{ik}$
- Complete assignment: one where $k = n$
  [if all variable domains have size d, there are $O(d^n)$ complete assignments]
- States: valid assignments
- Initial state: empty assignment {}, i.e. $k = 0$
- Successor of a state:
  $\{X_{i1} \leftarrow v_{i1}, \ldots, X_{ik} \leftarrow v_{ik}\} \rightarrow \{X_{i1} \leftarrow v_{i1}, \ldots, X_{ik} \leftarrow v_{ik}, X_{ik+1} \leftarrow v_{ik+1}\}$
- Goal test: $k = n$




r = n−k variables with s values → r×s branching factor

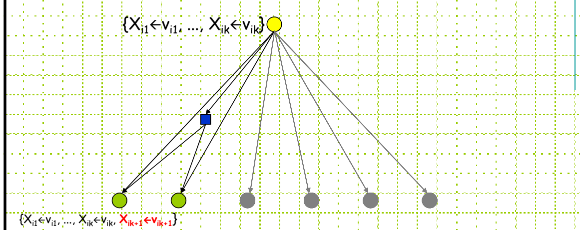
# A Key property of CSP: Commutativity

The order in which variables are assigned values has no impact on the reachable complete valid assignments

Hence:

1) One can expand a node N by first selecting **one** variable X not in the assignment A associated with N and then assigning every value v in the domain of X
   [→ big reduction in branching factor]

- 4 variables $X_1, \ldots, X_4$
- Let the valid assignment of N be:
  $A = \{X_1 \leftarrow v_1, X_3 \leftarrow v_3\}$
- For example pick variable $X_4$
- Let the domain of $X_4$ be $\{v_{4,1}, v_{4,2}, v_{4,3}\}$
- The successors of A are all the valid assignments among:
  $\{X_1 \leftarrow v_1, X_3 \leftarrow v_3, X_4 \leftarrow v_{4,1}\}$
  $\{X_1 \leftarrow v_1, X_3 \leftarrow v_3, X_4 \leftarrow v_{4,2}\}$
  $\{X_1 \leftarrow v_1, X_3 \leftarrow v_3, X_4 \leftarrow v_{4,2}\}$




**r = n-k variables with s values → r×s branching factor**

The depth of the solutions in the search tree is un-changed (n)

## A Key property of CSP: Commutativity

The order in which variables are assigned values has no impact on the reachable complete valid assignments

Hence:

1) One can expand a node N by first selecting **one** variable X not in the assignment A associated with N and then assigning every value v in the domain of X
   [→ big reduction in branching factor]
2) One need not store the path to a node
   → Backtracking search algorithm

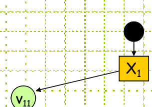
## Backtracking Search

Essentially a simplified depth-first algorithm using recursion

## Backtracking Search
(3 variables)

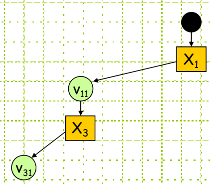
Assignment = {}

## Backtracking Search
(3 variables)



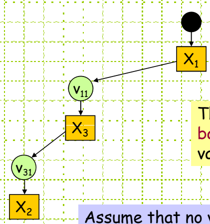

Assignment = $\{(X_1, v_{11})\}$

## Backtracking Search
### (3 variables)




Assignment = {($X_1$,$v_{11}$), ($X_3$,$v_{31}$)}

## Backtracking Search
### (3 variables)

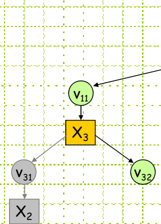



Then, the search algorithm backtracks to the previous ($X_3$) variable and tries another value

Assume that no value of $X_2$ leads to a valid assignment

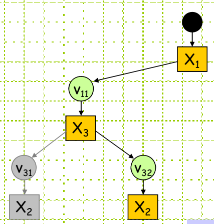
Assignment = {($X_1$,$v_{11}$), ($X_3$,$v_{31}$)}

## Backtracking Search
### (3 variables)




Assignment = {($X_1$,$v_{11}$), ($X_3$,$v_{32}$)}

# Backtracking Search
## (3 variables)

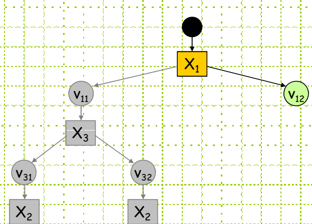



The search algorithm backtracks to the previous variable ($X_3$) and tries another value. But assume that $X_3$ has only two possible values. The algorithm backtracks to $X_1$

Assume again that no value of $X_2$ leads to a valid assignment

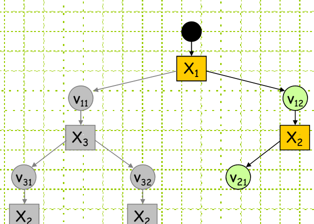
Assignment = {($X_1$,$v_{11}$), ($X_3$,$v_{32}$)}

# Backtracking Search
## (3 variables)



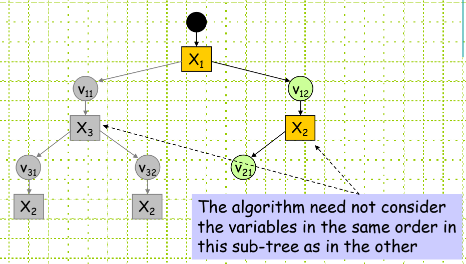

Assignment = {($X_1$,$v_{12}$)}

# Backtracking Search
## (3 variables)



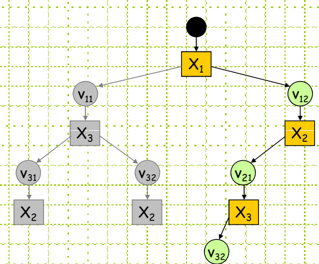

Assignment = {($X_1$,$v_{12}$), ($X_2$,$v_{21}$)}

## Backtracking Search
(3 variables)



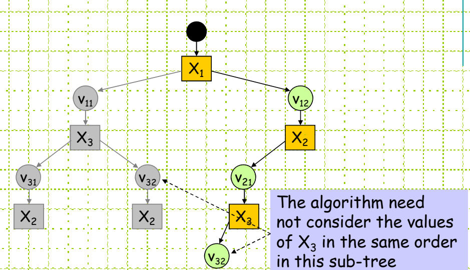

Assignment = {($X_1$,$v_{12}$), ($X_2$,$v_{21}$)}

## Backtracking Search
(3 variables)



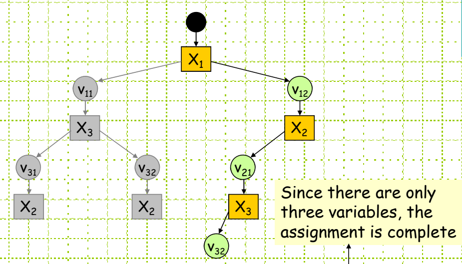

Assignment = {($X_1$,$v_{12}$), ($X_2$,$v_{21}$), ($X_3$,$v_{32}$)}

## Backtracking Search
(3 variables)




Assignment = {($X_1$,$v_{12}$), ($X_2$,$v_{21}$), ($X_3$,$v_{32}$)}

## Backtracking Search
### (3 variables)




Assignment = {$(X_1,v_{12})$, $(X_2,v_{21})$, $(X_3,v_{32})$}

## Backtracking Algorithm

CSP-BACKTRACKING(A)

1. If assignment A is complete then return A
2. X ← select a variable not in A
3. D ← select an ordering on the domain of X
4. For each value v in D do
   a. Add (X←v) to A
   b. If A is valid then
      i. result ← CSP-BACKTRACKING(A)
      ii. If result ≠ failure then return result
   c. Remove (X←v) from A
5. Return failure

Call CSP-BACKTRACKING({})