

Constraint Satisfaction

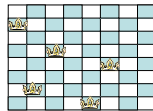
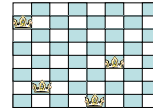
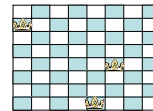
CS 486/686
May 17, 2005
University of Waterloo

Outline

- What are CSPs?
- Standard search and CSPs
- Improvements
 - Backtracking
 - Backtracking + heuristics
 - Forward checking

Introduction

- In the last couple of lectures we have been solving problems by searching in a space of states
 - Treating states as black boxes, ignoring any structure inside them
 - Using problem-specific routines
- Today we study problems where the state structure is important



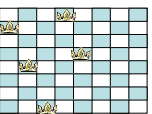
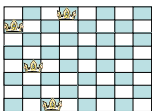
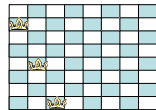
- **States:** all arrangements of 0,1,..., or 8 queens on the board

- **Initial state:** 0 queens on the board

- **Successor function:** Add a queen to the board

- **Goal test:** 8 queens on the board with no two of them attacking each other

$64 \times 63 \times \dots \times 57 \approx 3 \times 10^{14}$ states



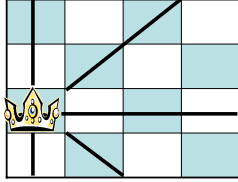
- **States:** all arrangements k queens ($0 \leq k \leq 8$), one per column in the leftmost k columns, with no queen attacking another
- **Initial state:** 0 queens on the board
- **Successor function:** Add a queen to the leftmost empty column such that it is not attacked
- **Goal test:** 8 queens on the board

2057 States

Introduction

- Earlier search methods studied often make choices in an arbitrary order
- In many problems the same state can be reached independent of the order in which the moves are chosen (commutative actions)
- Can we solve problems efficiently by being smart in the order in which we take actions?

4-queens Constraint Propagation

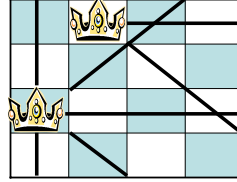


Place a queen in a square
Remove conflicting squares from
consideration

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

7

4-queens Constraint Propagation

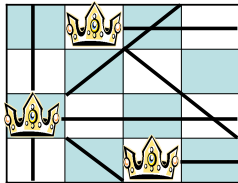


Place a queen in a square
Remove conflicting squares from
consideration

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

8

4-queens Constraint Propagation

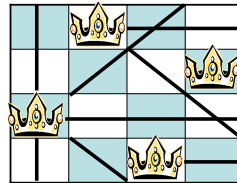


Place a queen in a square
Remove conflicting squares from
consideration

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

9

4-queens Constraint Propagation



Place a queen in a square
Remove conflicting squares from
consideration

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

10

CSP Definition

- A **constraint satisfaction problem** (CSP) is defined by $\{V, D, C\}$ where
 - $V = \{V_1, V_2, \dots, V_n\}$ is a set of variables
 - $D = \{D_1, \dots, D_n\}$ is the set of **domains**, D_i is the domain of possible values for variable V_i
 - $C = \{C_1, \dots, C_m\}$ is the set of **constraints**
 - Each constraint involves some subset of the variables and specifies the allowable combinations of values for that subset

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

11

CSP Definition

- A state is an **assignment** of values to some or all of the variables
 - $\{V_i = x_i, V_j = x_j, \dots\}$
- An assignment is **consistent** if it does not violate any constraints
- A **solution** is a complete, consistent assignment ("hard constraints")
 - Some CSPs also require an objective function to be optimized ("soft constraints")

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

12

Example 1: 8-Queens

- 64 variables V_{ij} , $i=1$ to 8, $j=1$ to 8
- Domain of each variable is $\{0,1\}$
- Constraints
 - $V_{ij}=1 \rightarrow V_{ik}=0$ for all $k \neq j$
 - $V_{ij}=1 \rightarrow V_{kj}=0$ for all $k \neq i$
 - Similar constraint for diagonals
 - $\sum_{i,j} V_{ij}=8$

Binary constraints
relate two variables

13

CS486/686 Lecture Slides © 2005, K. Larsen and P. Puggart

Example 2 - 8 queens

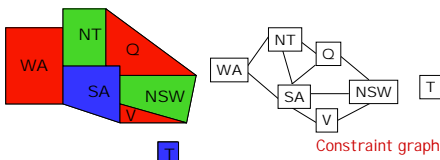
- 8 variables V_i , $i=1$ to 8
- Domain of each variable is $\{1,2,\dots,8\}$
- Constraints
 - $V_i=k \rightarrow V_j \neq k$ for all $j \neq i$
 - Similar constraints for diagonals

Binary constraints
relate two variables

14

CS486/686 Lecture Slides © 2005, K. Larsen and P. Puggart

Example 3 - Map Coloring



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

15

CS486/686 Lecture Slides © 2005, K. Larsen and P. Puggart

Example from R and N, Annotations from Stanford CS121

Example 4 - 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 owner of the White 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?

16

CS486/686 Lecture Slides © 2005, K. Larsen and P. Puggart

Example from R and N, Annotations from Stanford CS121

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 $\rightarrow (N_i = \text{English}) \leftrightarrow (C_i = \text{Red})$
- The Spaniard has a Dog
- The Japanese is a Painter $\rightarrow (N_i = \text{Japanese}) \leftrightarrow (J_i = \text{Painter})$
- The Italian drinks Tea
- The Norwegian lives in the first house on the left $\rightarrow (N_i = \text{Norwegian})$
- 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 $\left\{ \begin{array}{l} (C_i = \text{White}) \leftrightarrow (C_{i+1} = \text{Green}) \\ (C_5 \neq \text{White}) \\ (C_1 \neq \text{Green}) \end{array} \right.$
- 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

17

CS486/686 Lecture Slides © 2005, K. Larsen and P. Puggart

Example from R and N, Annotations from Stanford CS121

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 $\rightarrow (N_i = \text{English}) \leftrightarrow (C_i = \text{Red})$
- The Spaniard has a Dog
- The Japanese is a Painter $\rightarrow (N_i = \text{Japanese}) \leftrightarrow (J_i = \text{Painter})$
- The Italian drinks Tea
- The Norwegian lives in the first house on the left $\rightarrow (N_i = \text{Norwegian})$
- 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 $\left\{ \begin{array}{l} (C_i = \text{White}) \leftrightarrow (C_{i+1} = \text{Green}) \\ (C_5 \neq \text{White}) \\ (C_1 \neq \text{Green}) \end{array} \right.$
- 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

CS486/686 Lecture Slides © 2005, K. Larsen and P. Puggart

Example from R and N, Annotations from Stanford CS121

Street Puzzle

1
2
3
4
5

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

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

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

19

Example from R and N, Annotations from Stanford CS121

Street Puzzle

1
2
3
4
5

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

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 $\rightarrow N_1 = \text{Norwegian}$
 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 $\rightarrow D_3 = \text{Milk}$
 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

20

Example from R and N, Annotations from Stanford CS121

Street Puzzle

1
2
3
4
5

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

The Englishman lives in the Red house $\rightarrow C_1 = \text{Red}$
 The Spaniard has a Dog $\rightarrow A_1 \neq \text{Dog}$
 The Japanese is a Painter
 The Italian drinks Tea
 The Norwegian lives in the first house on the left $\rightarrow N_1 = \text{Norwegian}$
 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 $\rightarrow D_3 = \text{Milk}$
 The Norwegian lives next door to the Blue house
 The Violinist drinks Fruit juice $\rightarrow J_2 = \text{Violinist}$
 The Fox is in the house next to the Doctor's
 The Horse is next to the Diplomat's

21

Example 5 - 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?

22

Example 6 - 3-Sat

- n Boolean variables, V_1, \dots, V_n
- K constraints of the form $V_i \vee V_j \vee V_k$ where V_i is either true or false
- NP-complete
 - Recall GSAT and WALKSAT

23

Properties of CSPs

- Types of variables
 - Discrete and finite
 - Map colouring, 8-queens, boolean CSPs
 - Discrete variables with infinite domains
 - Scheduling jobs in a calendar
 - Require a **constraint language** ($Job_{i+3} \leq Job_i$)
 - Continuous domains
 - Scheduling on the Hubble telescope
 - **Linear programming**

24

Properties of CSPs

- Types of constraints
 - Unary** constraint restricts a variable to a single value
 - Queensland=Blue, SA≠Green
 - Binary** constraints relates two variables
 - SA≠NSW
 - Can use a constraint graph to represent CSPs with only binary constraints
 - Higher order constraints involve three or more variables
 - AllDiff(V_1, \dots, V_n)
 - Can use a constraint hypergraph to represent the problem

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

25

CSPs and search

- N variables V_1, \dots, V_n
- Valid assignment: $\{V_k = x_k\}$ for $0 \leq k \leq n$ such that values satisfy constraints on the variables
- States: valid assignments
- Initial state: empty assignment
- Successor:
 - $\{V_1 = x_1, \dots, V_k = x_k\} \rightarrow \{V_1 = x_1, \dots, V_k = x_k, V_{k+1} = x_{k+1}\}$
- Goal test: complete assignment
- If all domains all have size d , then there are $O(d^n)$ complete assignments

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

26

CSPs and commutativity

- CSPs are commutative!
 - The order of application of any given set of actions has no effect on the outcome
 - When assigning values to variables we reach the same partial assignment, no matter the order
 - All CSP search algorithms generate successors by considering possible assignments for only a **single variable at each node in the search tree**

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

27

CSPs and commutativity

- 3 variables V_1, V_2, V_3
- Let the current assignment be
 - $A = \{V_1 = x_1\}$
- Pick variable 3
- Let domain of V_3 be $\{a, b, c\}$
- The successors of A are
 - $\{V_1 = x_1, V_3 = a\}$
 - $\{V_1 = x_1, V_3 = b\}$
 - $\{V_1 = x_1, V_3 = c\}$

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

28

Backtracking Search

```

function BACKTRACKING-SEARCH(csp) returns a solution, or failure
return RECURSIVE-BACKTRACKING({}, csp)
function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(Variables[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment according to Constraints[csp] then
            add { var = value } to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove { var = value } from assignment
    return failure
    
```

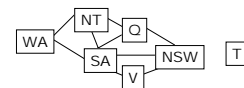
Depth first search which chooses values for one variable at a time
Backtracks when a variable has no legal values to assign

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

29

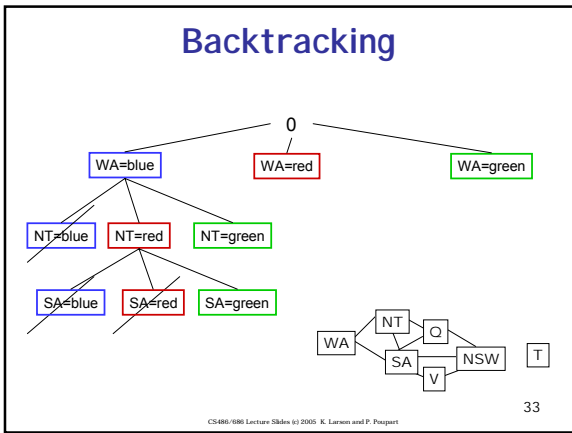
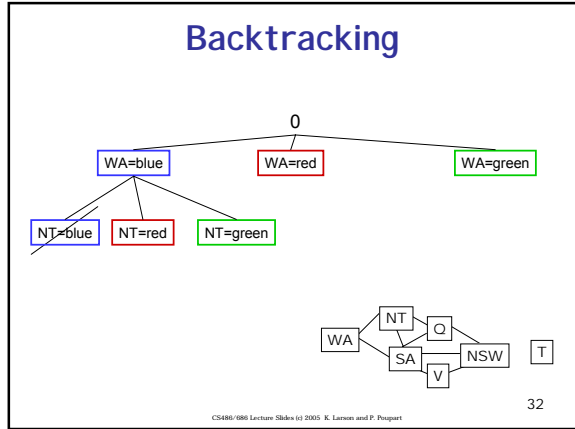
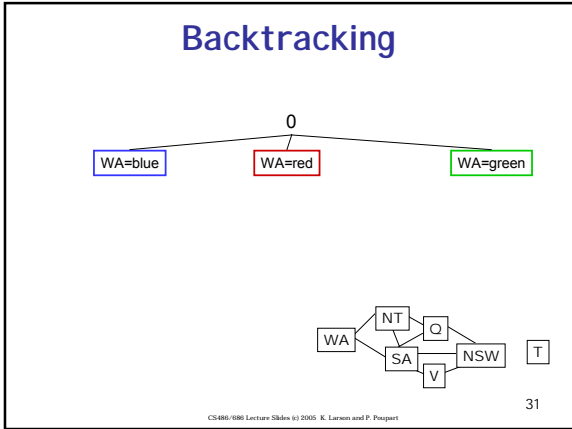
Backtracking

0



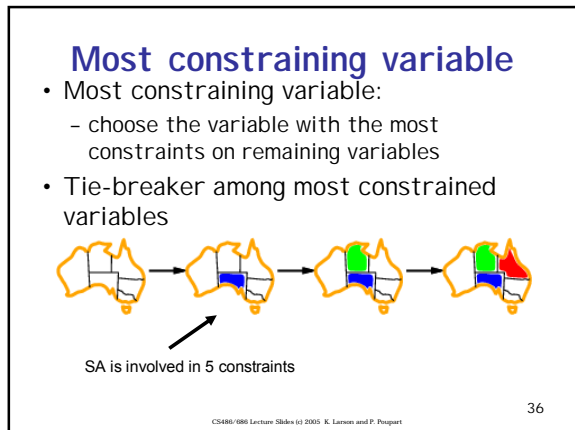
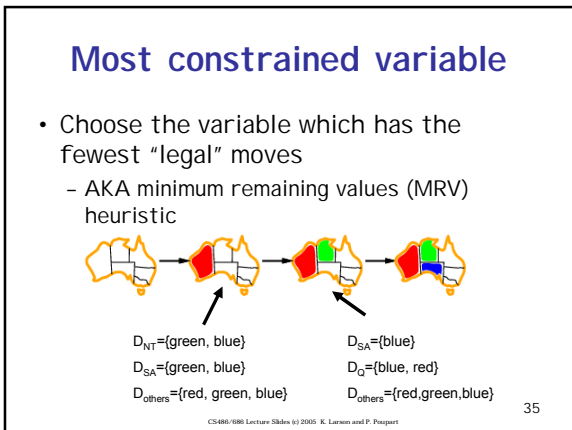
CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

30



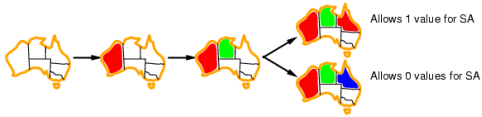
Backtracking and efficiency

- Backtracking search is an uninformed search method
 - Not very efficient
- We can do better by thinking about the following questions
 - Which variable should be assigned next?
 - In which order should its values be tried?
 - Can we detect inevitable failure early (and avoid the same failure in other paths)?



Least-constraining value

- Given a variable, choose the least constraining value:
 - the one that rules out the fewest values in the remaining variables



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

37

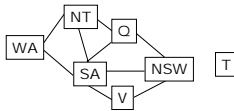
Forward checking

- The third question was
 - Is there a way to detect failure early?
- Forward checking
 - Keep track of remaining legal values for unassigned variables
 - Terminate search when any variable has no legal values

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

38

Forward Checking in Map Coloring

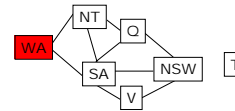


WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

39

Forward Checking in Map Coloring



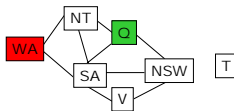
WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB

Forward checking removes the value Red of NT and of SA

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

40

Forward Checking in Map Coloring

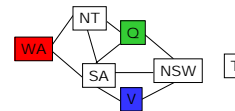


WA	NT	Q	NSW	V	SA	T
RGB	RGB	RGB	RGB	RGB	RGB	RGB
R	GB	RGB	RGB	RGB	GB	RGB
R	GB	G	RB	RGB	GB	RGB

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

41

Forward Checking in 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
R	B	G	RB	B	GB	RGB

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

42

Forward Checking in Map Coloring

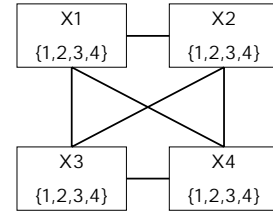
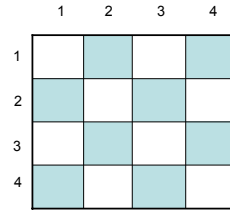
Empty set: the current assignment
 $\{(WA \leftarrow R), (Q \leftarrow G), (V \leftarrow B)\}$
 does not lead to a solution

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
R	B	G	R	B	B	RGB

CS486/686 Lecture Slides © 2005, K. Larsen and P. Pongratz

43

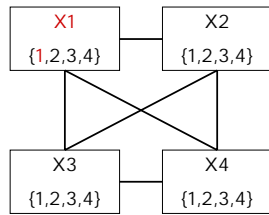
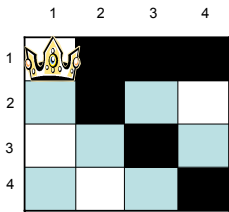
Example: 4 Queens



CS486/686 Lecture Slides © 2005, K. Larsen and P. Pongratz

44

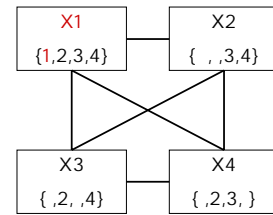
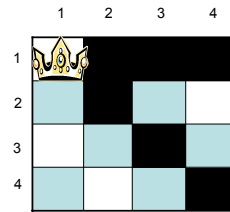
Example: 4 Queens



CS486/686 Lecture Slides © 2005, K. Larsen and P. Pongratz

45

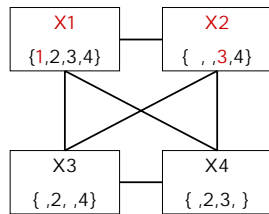
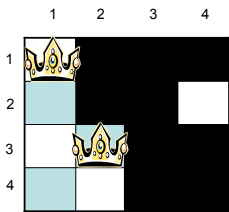
Example: 4 Queens



CS486/686 Lecture Slides © 2005, K. Larsen and P. Pongratz

46

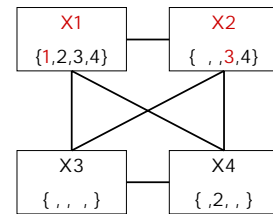
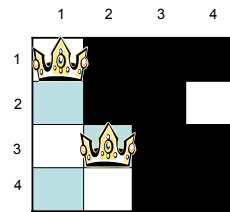
Example: 4 Queens



CS486/686 Lecture Slides © 2005, K. Larsen and P. Pongratz

47

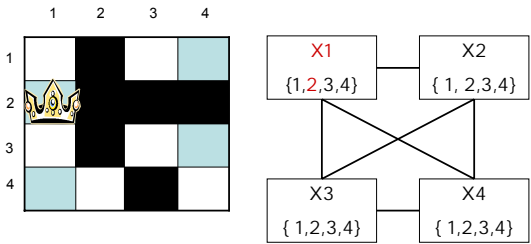
Example: 4 Queens



CS486/686 Lecture Slides © 2005, K. Larsen and P. Pongratz

48

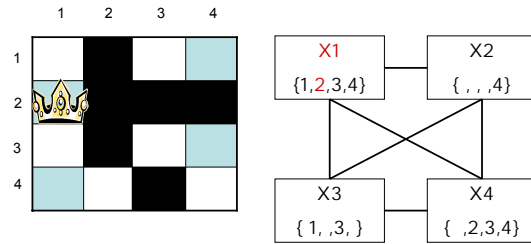
Example: 4 Queens



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

49

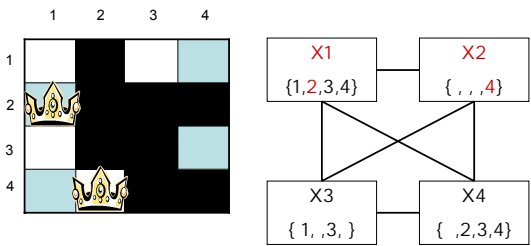
Example: 4 Queens



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

50

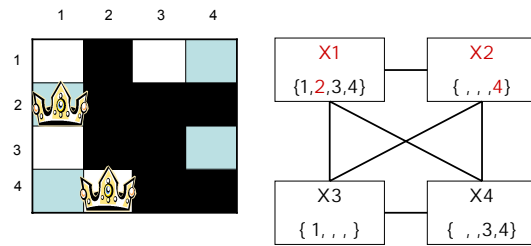
Example: 4 Queens



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

51

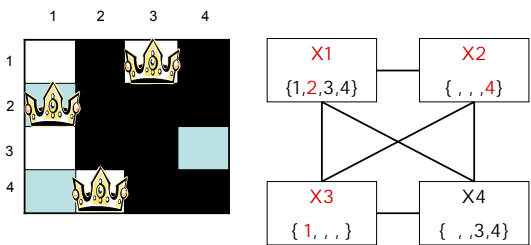
Example: 4 Queens



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

52

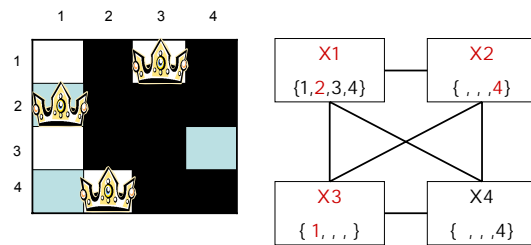
Example: 4 Queens



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

53

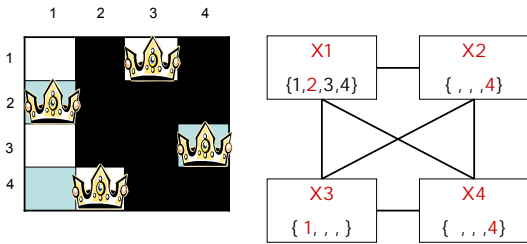
Example: 4 Queens



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

54

Example: 4 Queens



CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

55

Summary

- What you should know
 - How to formalize problems as CSPs
 - Backtracking search
 - Heuristics
 - Variable ordering
 - Value ordering
 - Forward checking

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

56

Next class

- Adversarial search
 - Russell and Norvig, Chapter 6

CS486/686 Lecture Slides (c) 2005, K. Larsen and P. Poupart

57