Constraint Satisfaction Problems

1

Outline

- Constraint Satisfaction Problems (CSP)
- Backtracking search for CSPs
- Local search for CSPs

2

제한조건을 만족하는 것이 무엇인지 알아내는 것

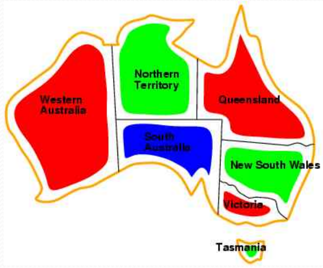
Constraint satisfaction problems (CSPs)

- Standard search problem:
  - state is a "black box“ – any data structure that supports successor function, heuristic function, and goal test
- CSP:
  - state is defined by variables X_i with values from domain D_i
  - goal test is a set of constraints specifying allowable combinations of values for subsets of variables
  - Allows useful general-purpose algorithms with more power than standard search algorithms

Example: Map-Coloring

- Variables *WA, NT, Q, NSW, V, SA, T*
- Domains D_i = {red,green,blue}
- Constraints: adjacent regions must have different colors
- e.g., WA ≠ NT, or
  (WA,NT) in {(red,green),(red,blue),(green,red),
  (green,blue),(blue,red),(blue,green)}

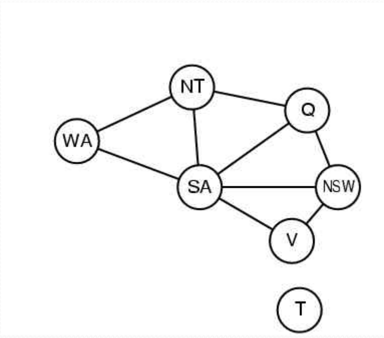
Example: Map-Coloring



- Solutions are complete and consistent assignments, e.g., WA = red, NT = green, Q = red, NSW = green,V = red, SA = blue, T = green

Constraint graph

- Binary CSP: each constraint relates two variables
- Constraint graph: nodes are variables, arcs are constraints



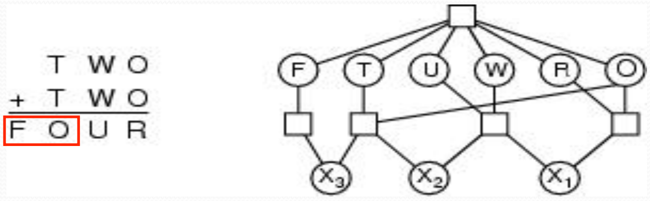
Varieties of CSPs

- Discrete variables
  - finite domains:
    - *n* variables, domain size *d* → $O(d^n)$ complete assignments
    - e.g., Boolean CSPs, includes Boolean satisfiability (NP-complete)
  - infinite domains:
    - integers, strings, etc.
    - e.g., job scheduling, variables are start/end days for each job
    - need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_3$
- Continuous variables
  - e.g., start/end times for Hubble Space Telescope observations
  - linear constraints solvable in polynomial time by linear programming

Varieties of constraints

- Unary constraints involve a single variable,
  - e.g., SA ≠ green
- Binary constraints involve pairs of variables,
  - e.g., SA ≠ WA
- Higher-order constraints involve 3 or more variables,
  - e.g., cryptarithmetic column constraints

Higher Order Constraint Example: Cryptarithmetic



- Variables: $F\ T\ U\ W\ R\ O\ X_1\ X_2\ X_3$
- Domains: $\{0,1,2,3,4,5,6,7,8,9\}$
- Constraints:
  - *alldifferent(F,T,U,W,R,O)*
  - $O + O = R + 10 \cdot X_1$
  - $X_1 + W + W = U + 10 \cdot X_2$
  - $X_2 + T + T = O + 10 \cdot X_3$
  - $X_3 = F,\ T \neq 0,\ F \neq 0$

9

Real-world CSPs

- Assignment problems
  - e.g., who teaches what class ?
- Timetabling problems
  - e.g., which class is offered when and where?
- Transportation scheduling
  - e.g., which car uses which road and when ?
- Factory scheduling
  - e.g., which job is performed when and about which part ?
- Notice that many real-world problems involve real-valued variables
  - e.g., "when" value can be a real number !

10

Standard search formulation for CSP

States are defined by the values assigned so far

- Initial state: the empty assignment { }
- Successor function: assign a value to an unassigned variable that does not conflict with current assignment

  → fail if no legal assignments

- Goal test: the current assignment is complete and consistent ? goal을 아는게 아니고 constraint를 하는 것

This is the same for all CSPs

Overall Complexity for CSP

1. Every solution appears at depth *n* with *n* variables
   → use depth-first search
2. Path is irrelevant, so can also use complete-state formulation
3. b = (n - l)d at depth l, hence $n! \cdot d^n$ leaves
   (d=domain size)

Backtracking search

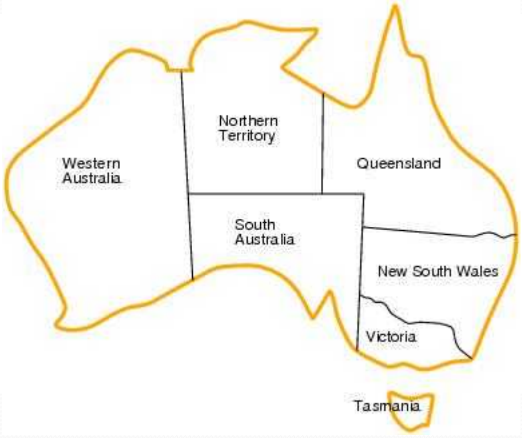
- Variable assignments are commutative
  - i.e., "WA = red then NT = green" is the same as "NT = green then WA = red"
- Only need to consider assignments to a single variable at each node
  → b = d and there are d^n leaves
- Depth-first search for CSPs with single-variable assignments is called backtracking search
- Backtracking search is the basic uninformed algorithm for CSPs
- Can solve *n*-queens for $n \approx 25$

Backtracking search

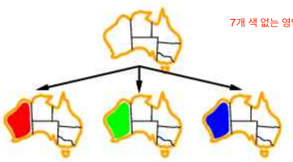
```
function BACKTRACKING-SEARCH( csp) returns a solution, or failure
   return RECURSIVE-BACKTRACKING({}, csp)

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 ≠ failue then return result
         remove { var = value } from assignment
   return failure
```

Backtracking example



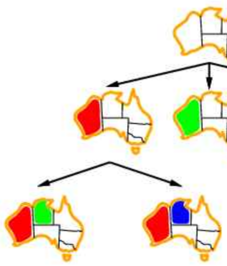
Backtracking example



7개 색 없는 영역. initial state

왼쪽부터 차례대로 집어넣는다고 가정.
빨 초 파

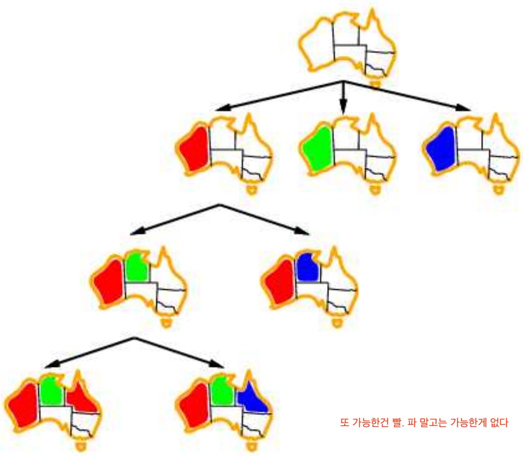
Backtracking example



인접한 것과 같은 색이 안되니까 가능한 옵션 2가지

17

Backtracking example



또 가능한건 빨. 파 말고는 가능한게 없다

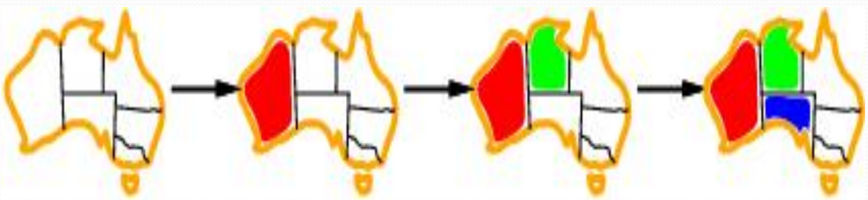
18

Improving backtracking efficiency

- Some general-purpose methods can give huge gains in speed:
  - Which variable should be assigned next?
  - In what order should its values be tried?
  - Can we detect inevitable failure early?
  - Can we take advantage of problem structure?

Most constrained variable variable이 적은 것 먼저 선택하자

- Most constrained variable:
  choose the variable with the fewest legal values



- minimum remaining values (MRV) heuristic

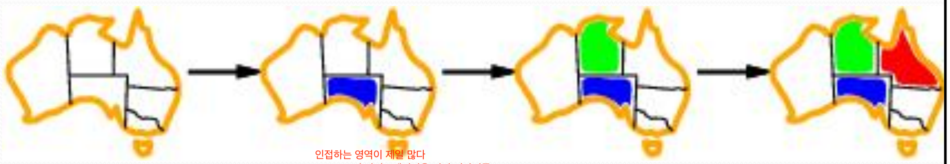
Most constraining variable

- Tie-breaker among most constrained variables

타이브레이킹 : 모스트 컨스테리인ㅇ 벨이ㅓ블 선택할 때, 그다음에 선택할 것은 뭘 할것이냐!

- Most constraining variable: 여러개있다
  - choose the variable with the most constraints on remaining variables

같은게있으면 variable 하나 선택하고 탐색하면
그다음영역 그다음영역 constraint가 최대로 나온다
그러면 그다음 나오는게 제일 작다…?? 먼말

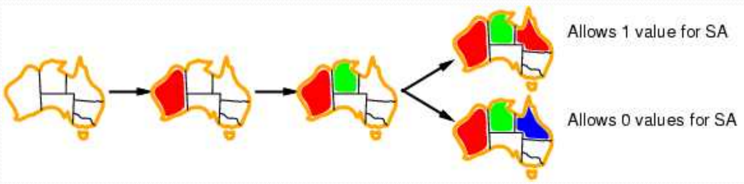


인접하는 영역이 제일 많다
m.c.v 쓰면 타이브레이킹을 써서 이자리를
선택

21

Least constraining value

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



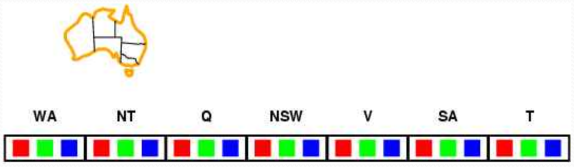
- Combining these heuristics makes 1000 queens feasible

22

3

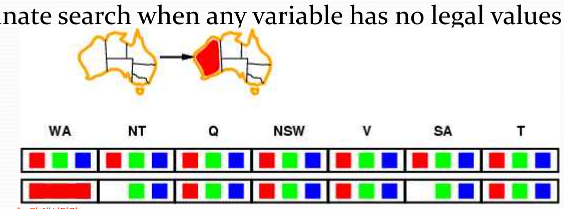
Forward checking

- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



Forward checking

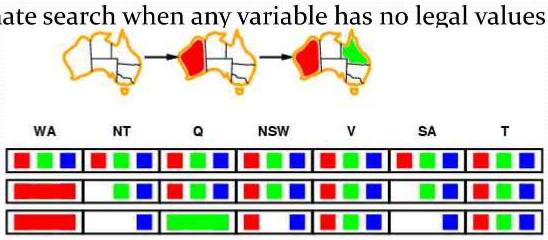
- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



초, 파 예선탈락

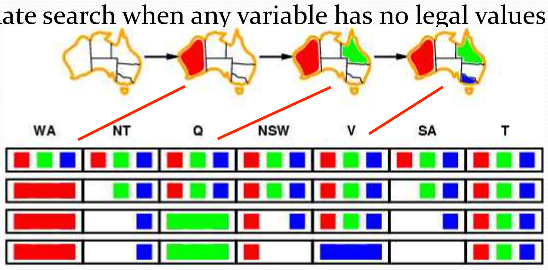
Forward checking

- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values

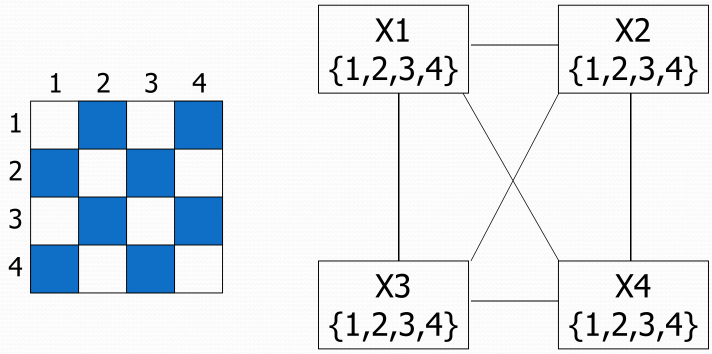


Forward checking

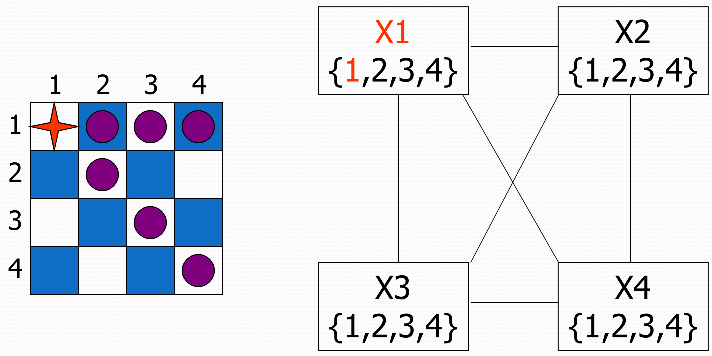
- Idea:
  - Keep track of remaining legal values for unassigned variables
  - Terminate search when any variable has no legal values



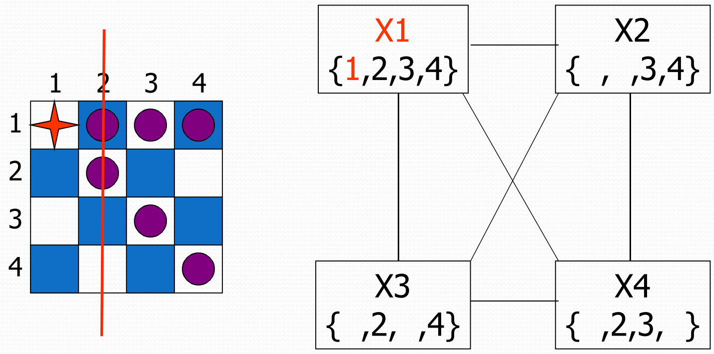
Example: 4-Queens Problem



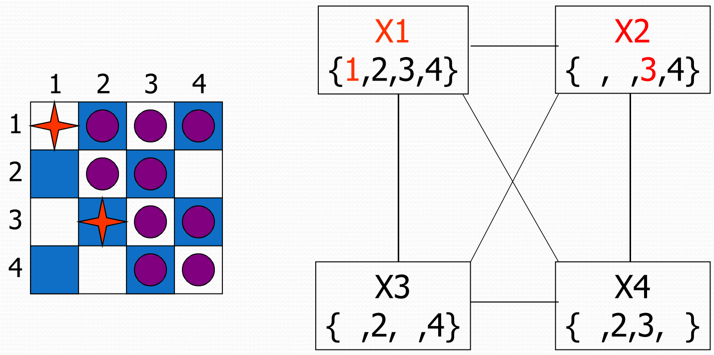
Example: 4-Queens Problem



Example: 4-Queens Problem

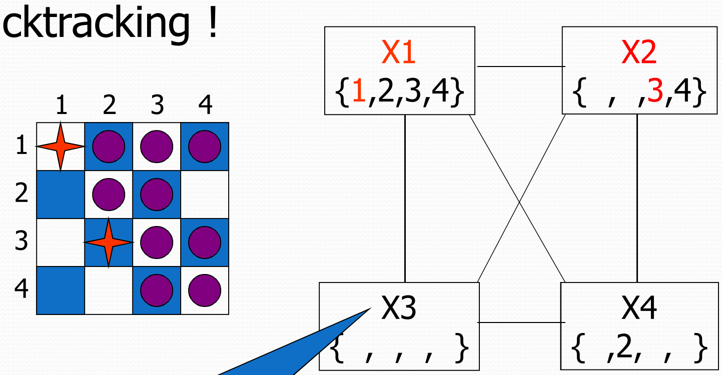


Example: 4-Queens Problem



Example: 4-Queens Problem

- Backtracking !

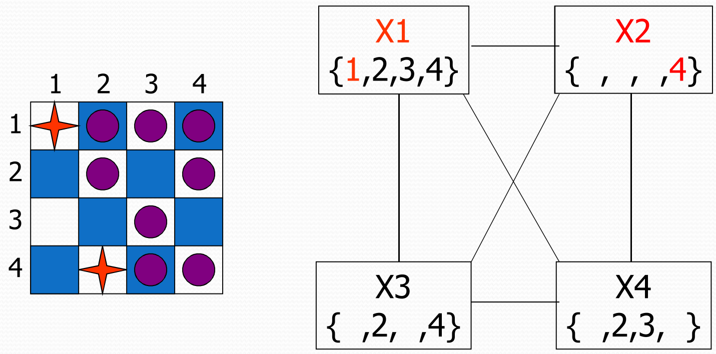


No legal value

Backtracking !

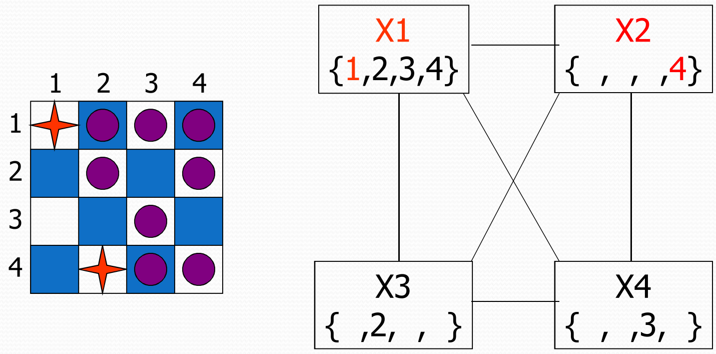
31

Example: 4-Queens Problem

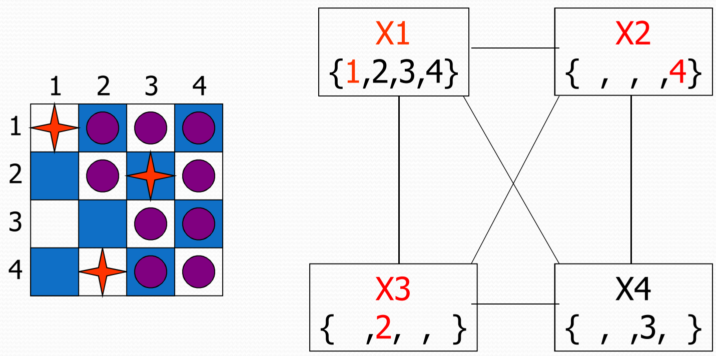


32

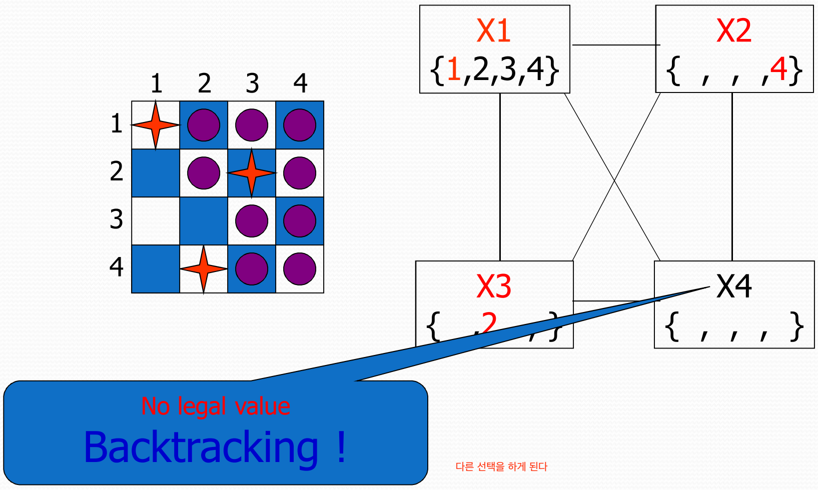
Example: 4-Queens Problem



Example: 4-Queens Problem



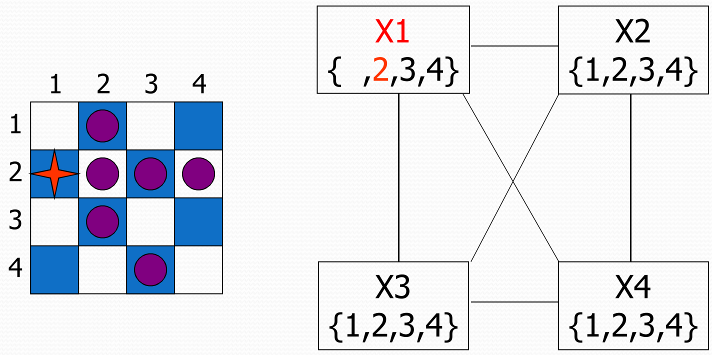
Example: 4-Queens Problem



다른 선택을 하게 된다

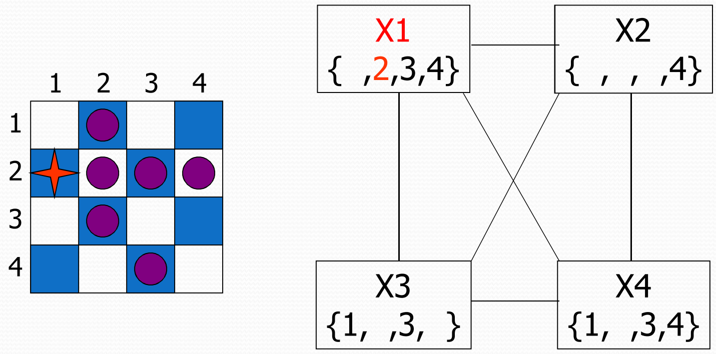
35

Example: 4-Queens Problem

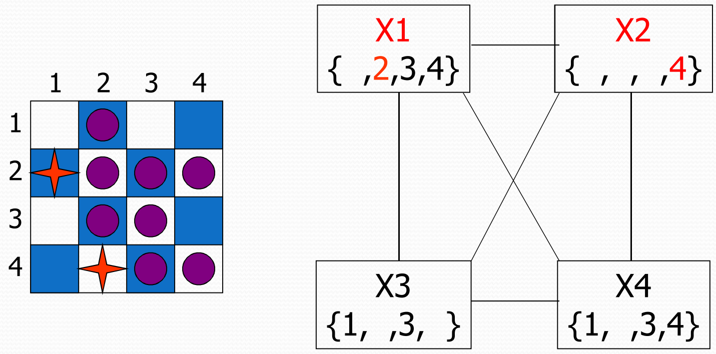


36

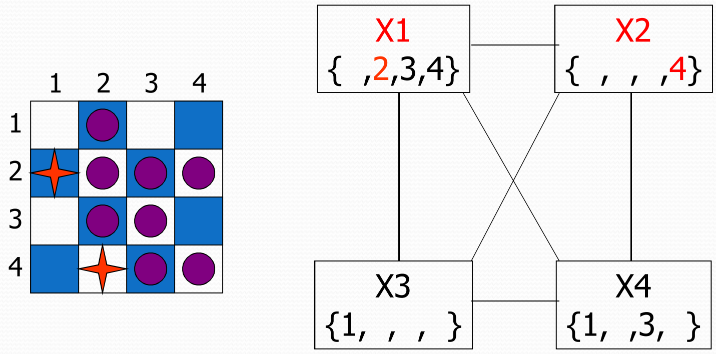
Example: 4-Queens Problem



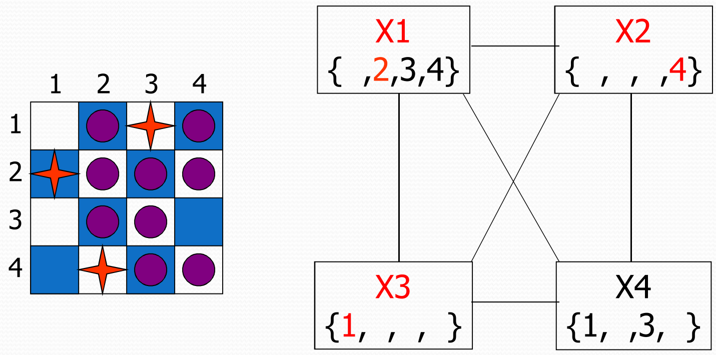
Example: 4-Queens Problem



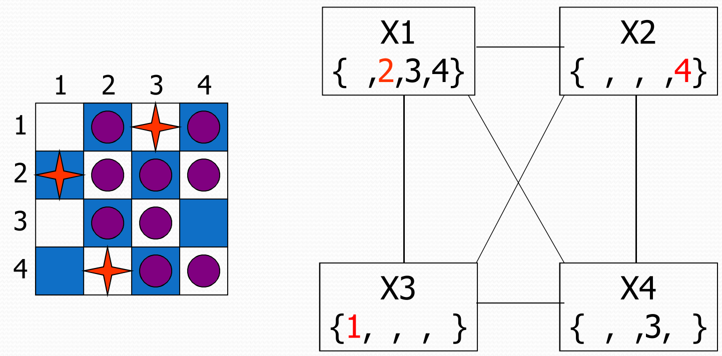
Example: 4-Queens Problem



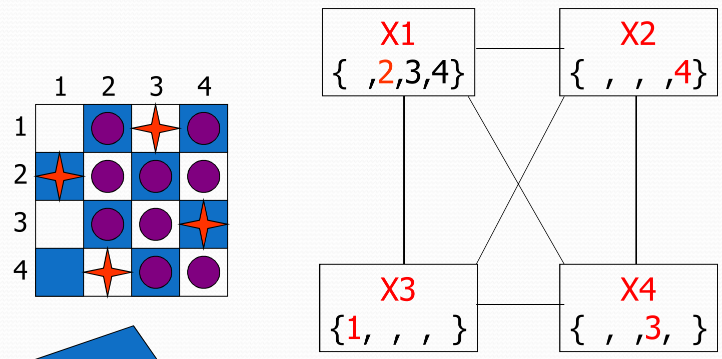
Example: 4-Queens Problem



Example: 4-Queens Problem



Example: 4-Queens Problem



"X1=2, X2=4, X3=1, X4=3"
Solution Found !

Local search for CSPs

- Hill-climbing, simulated annealing typically work with "complete" states, i.e., all variables assigned

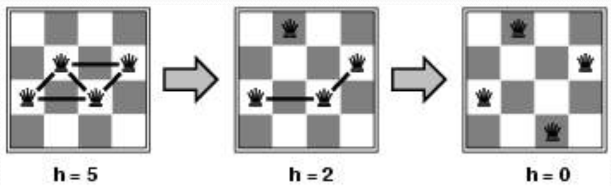
- To apply to CSPs:
  - allow states with unsatisfied constraints
  - operators reassign variable values

- Variable selection: randomly select any conflicted variable

- Value selection by min-conflicts heuristic:
  - choose value that violates the fewest constraints
  - i.e., hill-climb with $h(n)$ = total number of violated constraints

Example: 4-Queens

- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Actions: move queen in column
- Goal test: no attacks
- Evaluation: $h(n)$ = number of attacks



- Given random initial state, can solve n-queens in almost constant time for arbitrary n with high probability (e.g., n = 10,000,000)

Summary

- CSPs are a special kind of problem:
  - states defined by values of a fixed set of variables
  - goal test defined by constraints on variable values
- Backtracking = depth-first search with one variable assigned per node
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that guarantee later failure
- Constraint propagation (e.g., arc consistency) does additional work to constrain values and detect inconsistencies
- Iterative min-conflicts heuristic for hill climbing search is usually effective in practice