Constraint Satisfaction

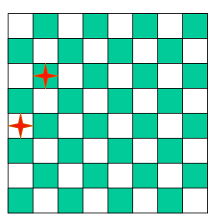
Russell & Norvig Ch. 5

Overview

- Constraint satisfaction offers a powerful problem-solving paradigm
  - View a problem as a **set of variables** to which we have to assign **values** that satisfy a number of **problem-specific constraints**.
  - Constraint programming, constraint satisfaction problems (CSPs), constraint logic programming…
- Algorithms for CSPs
  - Backtracking (systematic search)
  - Constraint propagation (k-consistency)
  - Variable and value ordering heuristics
  - Backjumping and dependency-directed backtracking

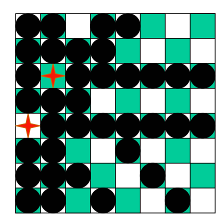
Motivating example: 8 Queens

Place 8 queens on a chess board such
That none is attacking another.



Generate-and-test, with no redundancies → "only" 8^8 combinations

Motivating example: 8-Queens



What more do we need for 8 queens?

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

5

Informal definition of CSP

- CSP = Constraint Satisfaction Problem, given
  (1) a finite set of variables
  (2) each with a domain of possible values (often finite)
  (3) a set of constraints that limit the values the variables can take on
- A **solution** is an assignment of a value to each variable such that the constraints are all satisfied.
- Tasks might be to decide if a solution exists, to find a solution, to find all solutions, or to find the "best solution" according to some metric (objective function).

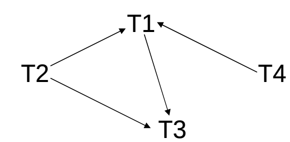
6

Example: 8-Queens Problem

- 8 variables Xi, i = 1 to 8 where Xi is the row number of queen in column i.
- Domain for each variable {1,2,…,8}
- Constraints are of the forms:
  - Xi = k → Xj ≠ k for all j = 1 to 8, j≠i
  - Xi = ki, Xj = kj →|i - j|≠| ki - kj| for all j = 1 to 8, j≠i

7

Example: Task Scheduling

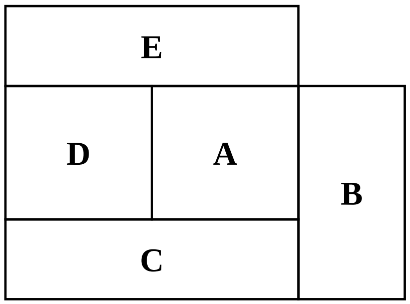


T1 must be done during T3
T2 must be achieved before T1 starts
T2 must overlap with T3
T4 must start after T1 is complete

8

Example: Map coloring

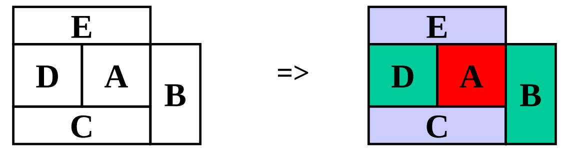
Color the following map using three colors (red, green, blue) such that no two adjacent regions have the same color.



9

Map coloring

- Variables: A, B, C, D, E all of domain RGB
- Domains: RGB = {red, green, blue}
- Constraints: $A \neq B$, $A \neq C$, $A \neq E$, $A \neq D$, $B \neq C$, $C \neq D$, $D \neq E$
- One solution: A=red, B=green, C=blue, D=green, E=blue



10

Brute Force methods

- Finding a solution by a brute force search is easy
  - **Generate and test** is a weak method
  - Just generate potential combinations and test each
- Potentially very inefficient
  - With n variables each of which can have one of three values, there are 3^n possible solutions to check.
- There are about 190 countries in the world today, which we can color using four colors.
- 4^{190} is a bit number!

```
solve(A,B,C,D,E) :-
  color(A),
  color(B),
  color(C),
  color(D),
  color(E),
  not(A=B),
  not(A=B),
  not(B=C),
  not(A=C),
  not(C=D),
  not(A=E),
  not(C=D).

color(red).
color(green).
color(blue).
```

11

Example: SATisfiability

- Given a set of propositions containing variables, find an assignment of the variables to {false,true} that satisfies them.
- For example, the clauses:
  - $(A \vee B \vee \neg C) \wedge (\neg A \vee D)$
  - (equivalent to $(C \rightarrow A) \vee (B \wedge D \rightarrow A)$

  are satisfied by

  A = false, B = true, C = false, D = false
- 3SAT is known to be NP-complete; in the worst case, solving CSP problems requires exponential time

12

Real-world problems

CSPs are a good match for many practical problems that arise in the real world

- Scheduling
- Temporal reasoning
- Building design
- Planning
- Optimization/satisfaction
- Vision
- Graph layout
- Network management
- Natural language processing
- Molecular biology / genomics
- VLSI design

13

Constraint network/graph

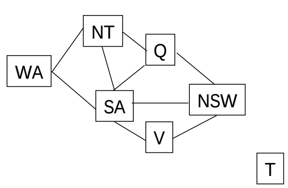
A constraint network (CN) consists of

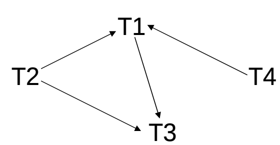
- a set of variables $X = \{x_1, x_2, \ldots x_n\}$
  - each with an associated domain of values $\{d_1, d_2, \ldots d_n\}$.
  - the domains are typically finite
- a set of constraints $\{c_1, c_2 \ldots c_m\}$ where
  - each defines a predicate which is a relation over a particular subset of X.
  - e.g., C_i involves variables $\{X_{i1}, X_{i2}, \ldots X_{ik}\}$ and defines the relation $R_i \subseteq D_{i1} \times D_{i2} \times \ldots D_{ik}$
- **Unary** constraint: only involves one variable
- **Binary** constraint: only involves two variables

14

Constraint Network/Graph

Binary constraints





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

15

Formal definition of a CN

- Instantiations
  - An **instantiation** of a subset of variables S is an assignment of a value in its domain to each variable in S
  - An instantiation is **legal** iff it does not violate any constraints.
- A **solution** is an instantiation of all of the variables in the network.

16

Typical tasks for CSP

- Solutions:
  - Does a solution exist?
  - Find one solution
  - Find all solutions
  - Given a metric on solutions, find the best one
  - Given a partial instantiation, do any of the above
- Transform the CN into an equivalent CN that is easier to solve.

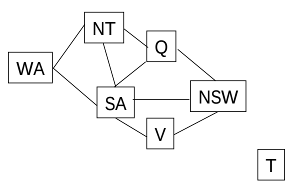
17

Binary CSP

- A **binary CSP** is a CSP where all constraints are binary or unary
- Any non-binary CSP can be converted into a binary CSP by introducing additional variables
- A binary CSP can be represented as a **constraint graph**, which has a node for each variable and an arc between two nodes if and only there is a constraint involving the two variables
  - Unary constraints appear as self-referential arcs

18

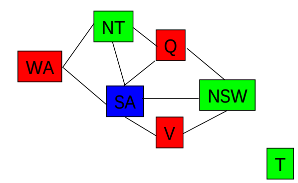
A running example: coloring Australia



- 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**

19

A running example: coloring Australia



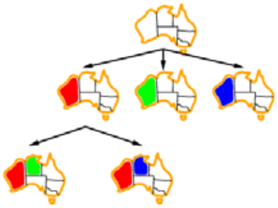
- Solutions are complete and consistent assignments
- One of several solutions
- Note that for generality, constraints can be expressed as relations, e.g., WA ≠ NT is
  (WA,NT) in {(red,green), (red,blue), (green,red), (green,blue), (blue,red),(blue,green)}

20

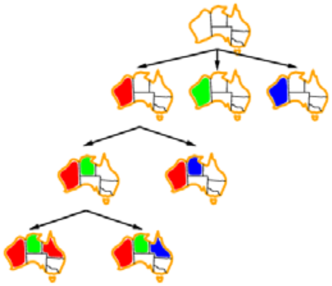
Backtracking example

Backtracking example

Backtracking example



Backtracking example



Basic Backtracking Algorithm

CSP-BACKTRACKING(PartialAssignment a)

- If a is complete then return a
- X ← select an 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
  - Remove (X= v) from a
- Return *failure*

Start with CSP-BACKTRACKING({})

Note: this is depth first search; can solve n-queens problems for n ~ 25

25

Problems with backtracking

- Thrashing: keep repeating the same failed variable assignments
  - Consistency checking can help
  - Intelligent backtracking schemes can also help
- Inefficiency: can explore areas of the search space that aren't likely to succeed
  - Variable ordering can help

26

Improving backtracking efficiency

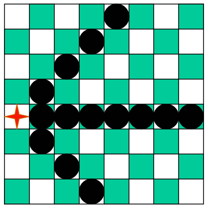
Here are some standard techniques to improve the efficiency of backtracking

- Can we detect inevitable failure early?
- Which variable should be assigned next?
- In what order should its values be tried?

27

Forward Checking

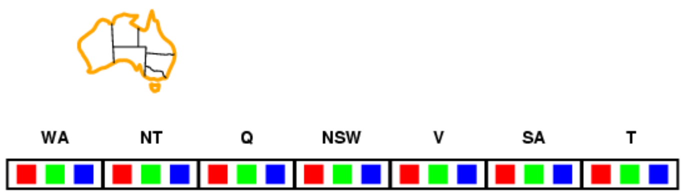
After a variable X is assigned a value v, look at each unassigned variable Y connected to X by a constraint and delete from Y's domain values inconsistent with v



Using forward checking and backward checking roughly doubles the size of N-queens problems that can be practically solved

28

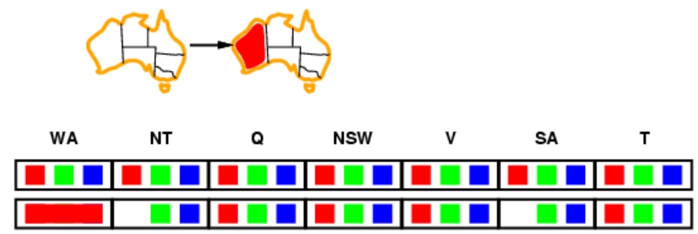
Forward checking



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

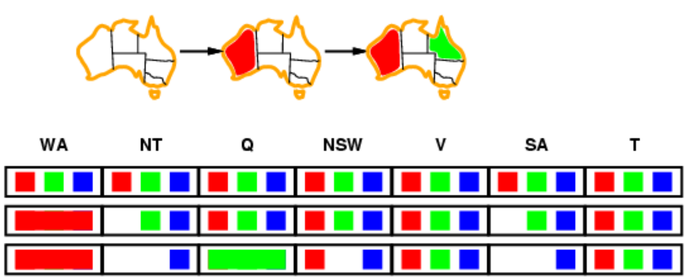
29

Forward checking



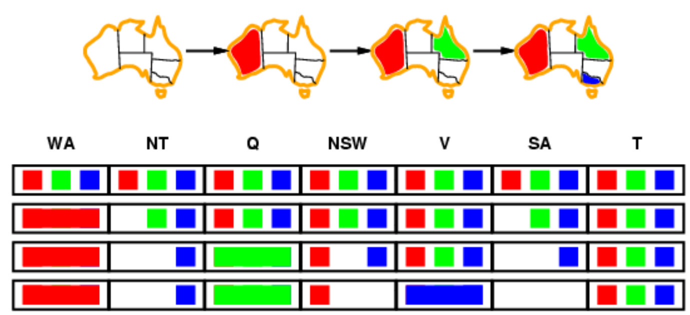
30

Forward checking



31

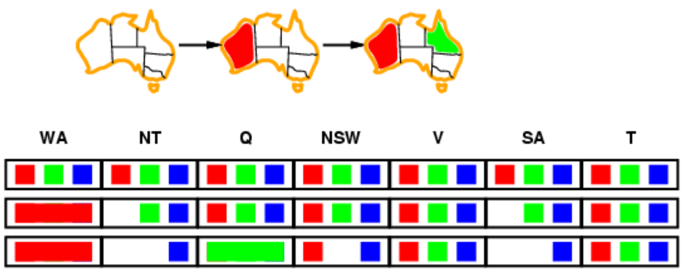
Forward checking



32

Constraint propagation

- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures.
- NT and SA cannot both be blue!



Definition: Arc consistency

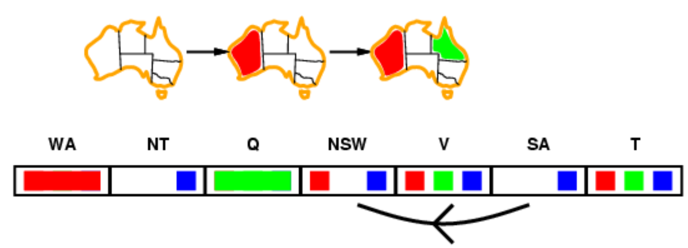
- A constraint C_xy is said to be arc consistent w.r.t. x if for each value v of x there is an allowed value of y.
- Similarly, we define that C_xy is arc consistent w.r.t. y.
- A binary CSP is arc consistent iff every constraint C_xy is arc consistent wrt x as well as wrt y.
- When a CSP is not arc consistent, we can make it arc consistent, e.g. by using AC3.
  - This is also called “enforcing arc consistency”.

Arc Consistency Example

- Let domains be
  - D_x = {1, 2, 3}, D_y = {3, 4, 5, 6}
- A constraint
  - C_xy = {(1,3), (1,5), (3,3), (3,6)}
- C_xy is not arc consistent w.r.t. x, neither w.r.t. y. By enforcing arc consistency, we get reduced domains
  - D’_x = {1, 3}, D’_y={3, 5, 6}

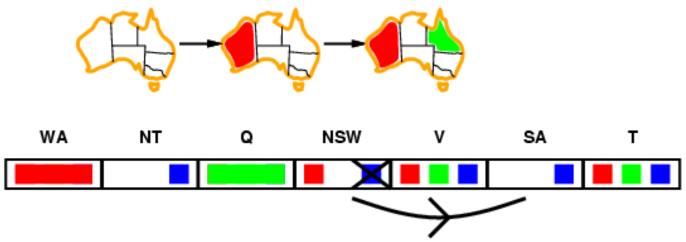
Arc consistency

- Simplest form of propagation makes each arc consistent
- *X* $\rightarrow$ *Y* is consistent iff for every value *x* of *X* there is some allowed *y*

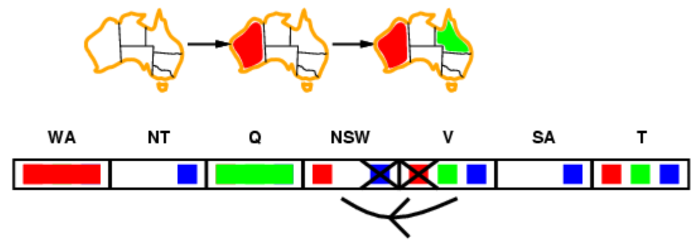


Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff for every value x of X there is some allowed y



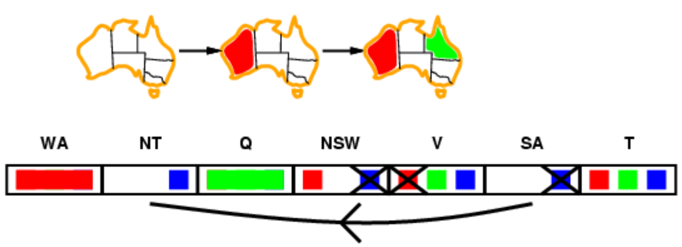
Arc consistency



If X loses a value, neighbors of X need to be rechecked

Arc consistency

- Arc consistency detects failure earlier than forward checking
- Can be run as a preprocessor or after each assignment



General CP for Binary Constraints

Algorithm AC3

- 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
      - If Y's domain is non-empty then STACK(Y,Q)
      - Else return false

Complexity of AC3

- e = number of constraints (edges)
- d = number of values per variable
- Each variables is inserted in Q up to d times
- REMOVE-ARC-INCONSISTENCY takes $O(d^2)$ time
- CP takes $O(ed^3)$ time

41

Improving backtracking efficiency

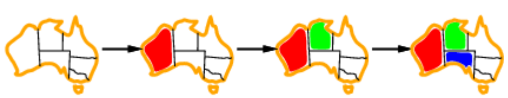
- Here are some standard techniques to improve the efficiency of backtracking
  - Can we detect inevitable failure early?
  - Which variable should be assigned next?
  - In what order should its values be tried?
- Combining constraint propagation with these heuristics makes 1000 N-queen puzzles feasible

42

Most constrained variable

- Most constrained variable:

  choose the variable with the fewest legal values

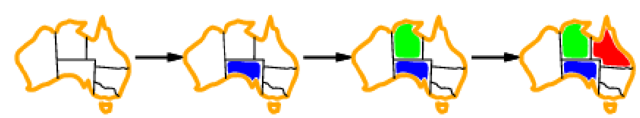


- a.k.a. **minimum remaining values (MRV)** heuristic

43

Most constraining variable

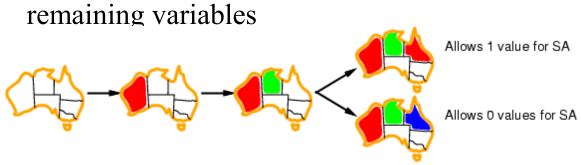
- Tie-breaker among most constrained variables
- Most constraining variable:
  - choose variable involved in largest # of constraints on remaining variables



44

Least constraining value

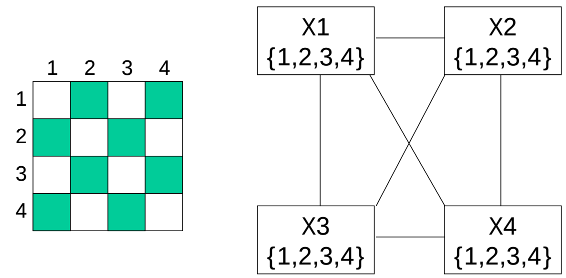
- Given a variable, choose least constraining value:
  - the one that rules out the fewest values in the remaining variables



- Combining these heuristics makes 1000 queens feasible

45

Is AC3 Alone Sufficient?

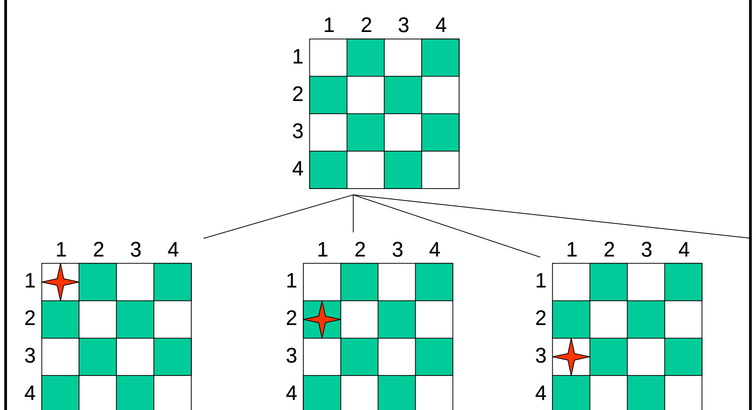


46

Solving a CSP still requires search

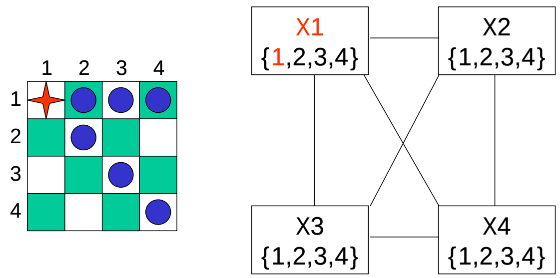
- Search:
  - can find good solutions, but must examine non-solutions along the way
- Constraint Propagation:
  - can rule out non-solutions, but this is not the same as finding solutions:
- Interweave constraint propagation and search
  - Perform constraint propagation at each search step.

47

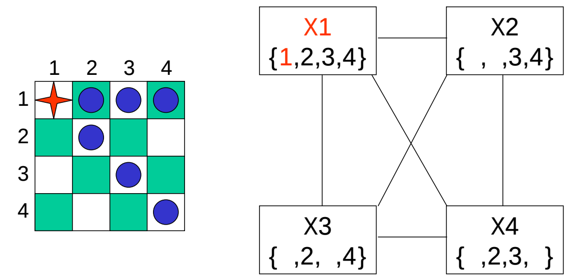


48

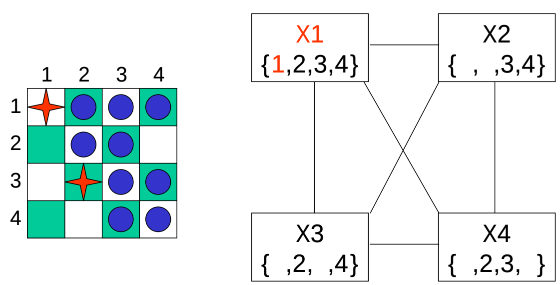
4-Queens Problem



4-Queens Problem

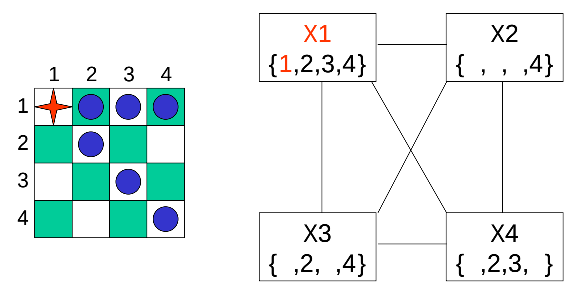


4-Queens Problem



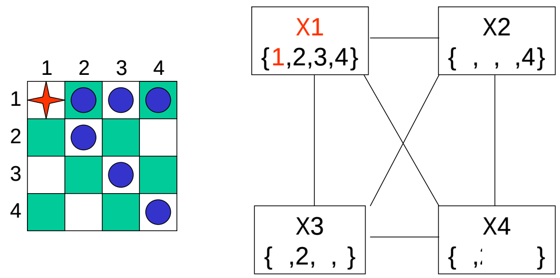
X2=3 eliminates { X3=2, X3=3, X3=4 }
⇒ inconsistent!

4-Queens Problem



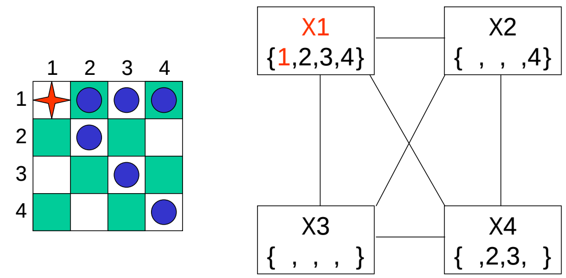
X2=4 ⇒ X3=2, which eliminates { X4=2, X4=3}
⇒ inconsistent!

4-Queens Problem

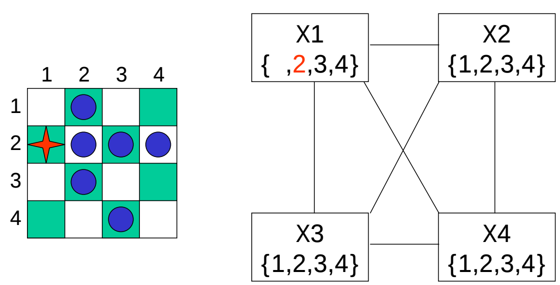


X3=2 eliminates { X4=2, X4=3}
⇒ inconsistent!

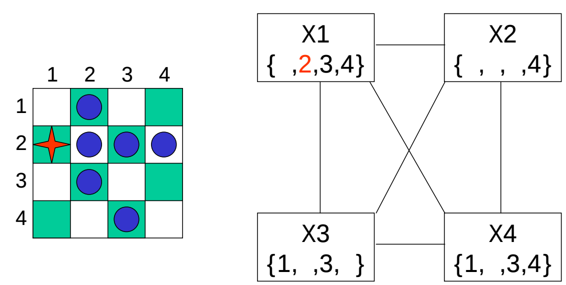
4-Queens Problem



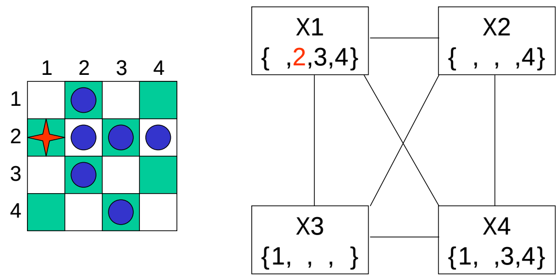
4-Queens Problem



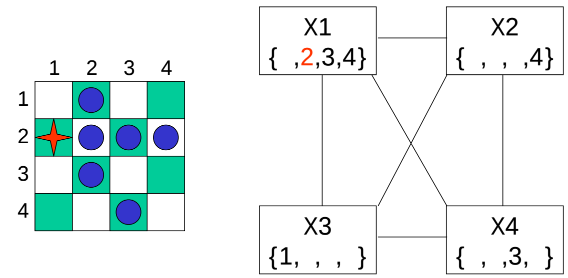
4-Queens Problem



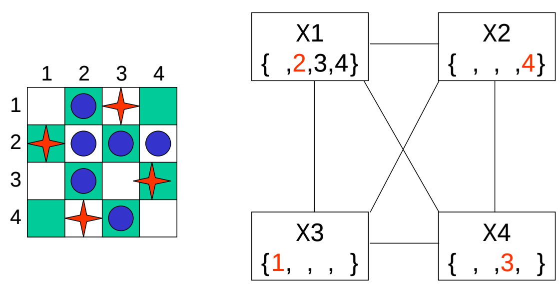
4-Queens Problem



4-Queens Problem



4-Queens Problem

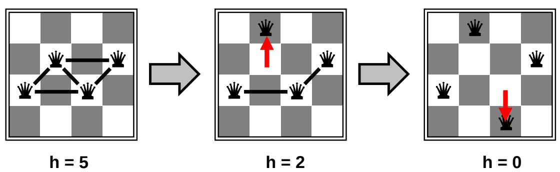


Local search for constraint problems

- Remember local search?
- A version of local search exists for constraint problems
- Basic idea:
  - generate a random "solution"
  - Use metric of "number of conflicts"
  - Keep modifying solution by reassigning one variable at a time to decrease metric until a solution is found or no modification improves it
- Has all the features and problems of local search

Min Conflict Example

- **States:** 4 Queens, 1 per column
- **Operators:** Move queen in its column
- ***Goal* test:** No attacks
- **Evaluation metric:** Total number of attacks

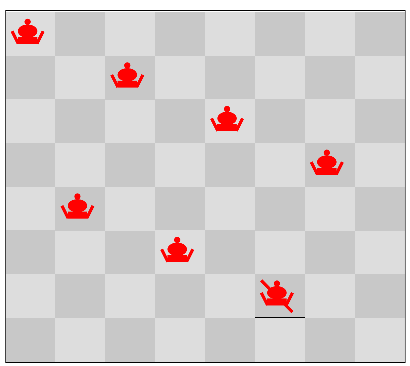


Basic Local Search Algorithm

Assign a domain value d_i to each variable v_i
while no solution and not stuck and not timed out
    bestCost $\leftarrow \infty$; bestList $\leftarrow \emptyset$;
    for each variable v_i | Cost(Value(v_i)) > 0
        for each domain value d_i of v_i
            if Cost(d_i) < bestCost
                bestCost $\leftarrow$ Cost(d_i); bestList $\leftarrow d_i$;
            else if Cost(d_i) = bestCost
                bestList $\leftarrow$ bestList $\cup\ d_i$
    Take a randomly selected move from bestList

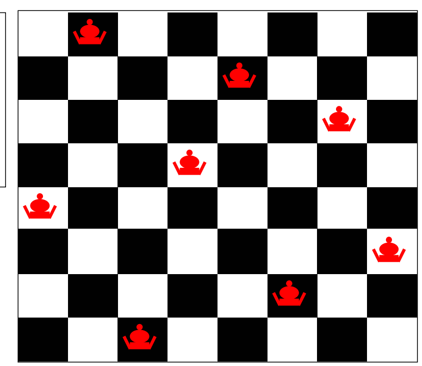
Eight Queens using Backtracking

Undo move for Queen 7
and so on...

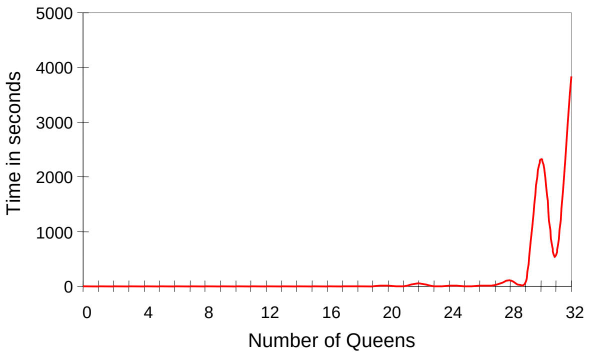


Eight Queens using Local Search

Answer Found

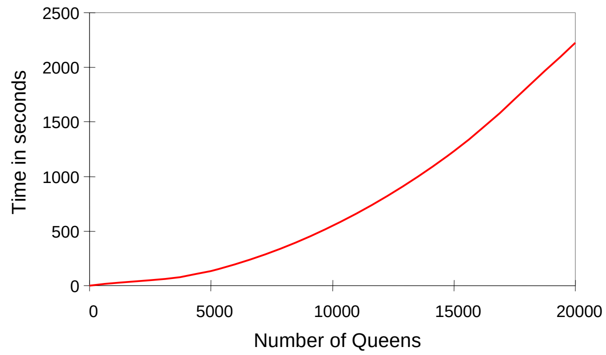


Backtracking Performance



65

Local Search Performance



66

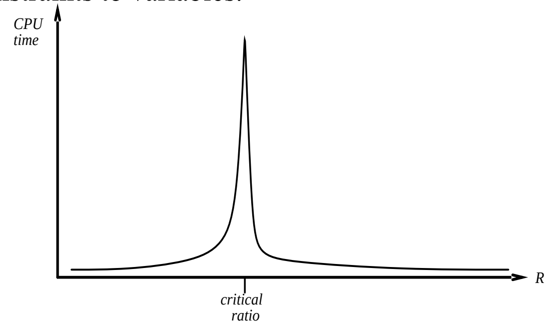
Min Conflict Performance

- Performance depends on quality and informativeness of initial assignment; inversely related to distance to solution
- Min Conflict often has astounding performance.
- For example, it's been shown to solve arbitrary size (in the millions) N-Queens problems in constant time.
- This appears to hold for arbitrary CSPs with the caveat…

67

Min Conflict Performance

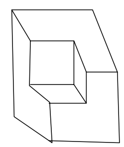
Except in a certain critical range of the ratio constraints to variables.



68

A famous example: Labeling line drawings

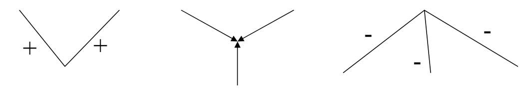
- Waltz labeling algorithm – one of the earliest CSP applications
  - Convex interior lines are labeled as +
  - Concave interior lines are labeled as –
  - Boundary lines are labeled as ⟶
- There are 208 labeling (most of which are impossible)
- Here are the 18 legal labeling:



69

Labeling line drawings II

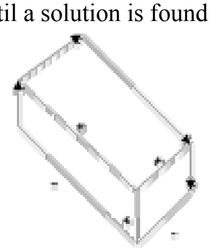
- Here are some illegal labelings:



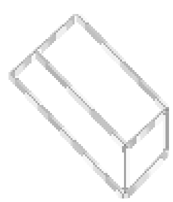
70

Labeling line drawings (cont.)

- Waltz labeling algorithm: Propagate constraints repeatedly until a solution is found



A solution for one labeling problem



A labeling problem with no solution

71

K-consistency

- K- consistency generalizes the notion of arc consistency to sets of more than two variables.
  - A graph is K-consistent if, for legal values of any K-1 variables in the graph, and for any Kth variable V_k, there is a legal value for V_k
- Strong K-consistency = J-consistency for all J<=K
- **Node consistency = strong 1-consistency**
- **Arc consistency = strong 2-consistency**
- Path consistency = strong 3-consistency

72

Why do we care?

1. If we have a CSP with N variables that is known to be **strongly N-consistent**, we can solve it **without backtracking**
2. For any CSP that is **strongly K-consistent**, if we find an **appropriate variable ordering** (one with "small enough" branching factor), we can solve the CSP **without backtracking**

73

Intelligent backtracking

- **Backjumping**: if V_j fails, jump back to the variable V_i with greatest i such that the constraint (V_i, V_j) fails (i.e., most recently instantiated variable in conflict with V_i)
- **Backchecking**: keep track of incompatible value assignments computed during backjumping
- **Backmarking**: keep track of which variables led to the incompatible variable assignments for improved backchecking

74

Challenges for constraint reasoning

- What if not all constraints can be satisfied?
  - Hard vs. soft constraints
  - Degree of constraint satisfaction
  - Cost of violating constraints
- What if constraints are of different forms?
  - Symbolic constraints
  - Numerical constraints [constraint solving]
  - Temporal constraints
  - Mixed constraints

75

Challenges for constraint reasoning

- What if constraints are represented intensionally?
  - Cost of evaluating constraints (time, memory, resources)
- What if constraints, variables, and/or values change over time?
  - Dynamic constraint networks
  - Temporal constraint networks
  - Constraint repair
- What if you have multiple agents or systems involved in constraint satisfaction?
  - Distributed CSPs
  - Localization techniques

76