Constraint Satisfaction Problems

Blake VanBerlo

Lecture 4

Readings: RN 6.1 - 6.3. PM 4.1 - 4.4.

Outline

Learning Goals

By the end of the lecture, you should be able to

- Formulate a real-world problem as a constraint satisfaction problem.
- Trace the execution of the backtracking search algorithm.
- Verify whether a constraint is arc-consistent.
- Trace the execution of the AC-3 arc consistency algorithm.
- Trace the execution of the backtracking search algorithm with arc consistency.

Learning Goals

Examples of CSP Problems

Formulating a CSP

Solving a CSP

Revisiting the Learning goals

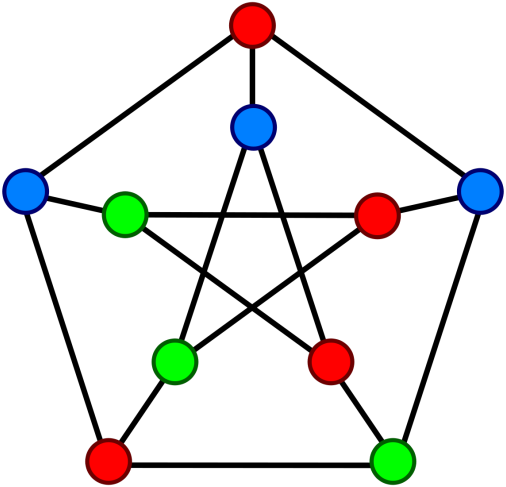
Evacuation Planning

- Instruct residents to follow a route at a given time.
- Two challenges:
  (1) deploy enough resources to give instructions.
  (2) ensure that the population comply with the instructions.
- Applied in a real-life case study and
  generated schedules close to the optimal ones from prior work.
- Even, C., Schutt, A., & Van Hentenryck, P. (2015).
  A constraint programming approach for non-preemptive evacuation scheduling.
  https://arxiv.org/pdf/1505.02487.pdf.

Crossword Puzzles

T	A	O	S		S	P	E	C	S		S	E	P	T
E	G	G	Y		A	L	A	A	P		I	L	I	A
T	H	R	E	E	P	E	N	N	Y		D	H	A	K
H	A	E		X	E	B	E	C		S	E	I	N	E
			V	O	L		D	H	O	W	S			
S	W	E	E	N	E	Y		A	N	A	T	M	A	N
P	R	Y	S		S	O	N		E	P	E	I	R	A
L	Y	R	I	C		N	O	S		S	P	A	E	R
I	L	I	C	E	S		R	U	C		P	O	C	K
T	Y	R	A	N	E	D		R	O	P	E	W	A	Y
			T	S	A	R	S		C	U	D			
B	A	G	I	E		O	I	N	K	S		T	S	K
O	L	E	O		S	W	E	E	P	S	T	A	K	E
Y	A	R	N		E	N	V	O	I		A	B	U	T
G	Y	M	S		I	D	E	N	T		G	U	G	A

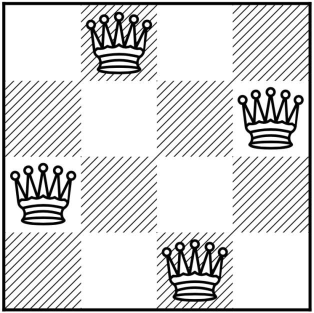
Graph Colouring Problem



Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

4-Queens Problem



Learning Goals

Examples of CSP Problems

Formulating a CSP

Solving a CSP

Revisiting the Learning goals

Internal Structure of States

- Search algorithms are unaware of the internal structure of states.

  $\rightarrow$ Generate successors of a state. Test whether a state is a goal.

- However, knowing a state's internal structure can help.

$\rightarrow$ 4-queens: Consider a state with 2 queens in the same row. A search algorithm only knows that this is not a goal and will keep searching. But in fact, this is a dead end and we should backtrack.

Let's model the internal structure of states.

Defining a CSP

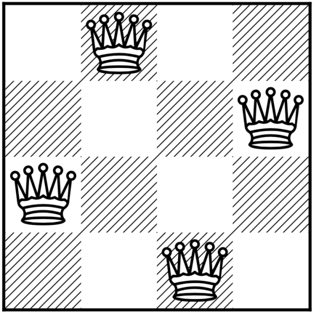
Each state contains

- A set X of variables: $\{X_1, X_2, ..., X_n\}$.
- A set D of domains: D_i is the domain for variable X_i, $\forall i$.
- A set C of constraints specifying allowable value combinations.

A solution is an assignment of values to all the variables that satisfy all the constraints.

$\rightarrow$ More efficient than search because the constraints will help us eliminate large portions of the search space.

Example: 4-Queens Problem



4-Queens: State Definition in a CSP

4-Queens: State Definition in a CSP

- Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i, where $i \in \{0, 1, 2, 3\}$.

  Assume that exactly one queen is in each column.

4-Queens: State Definition in a CSP

- Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i, where $i \in \{0, 1, 2, 3\}$.

  Assume that exactly one queen is in each column.

- Domains: $D_{x_i} = \{0, 1, 2, 3\}$ for all x_i.

4-Queens: State Definition in a CSP

- Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i, where $i \in \{0, 1, 2, 3\}$.

  Assume that exactly one queen is in each column.

- Domains: $D_{x_i} = \{0, 1, 2, 3\}$ for all x_i.

- Constraints:

  No pair of queens are in the same row or diagonal.

  $(\forall i(\forall j((i \neq j) \rightarrow ((x_i \neq x_j) \wedge (|x_i - x_j| \neq |i - j|)))))$

  For example, $((x_0 \neq x_1) \wedge (|x_0 - x_1| \neq 1))$

Q: Constraints for 4-Queens Problem

Q: Given the definitions of variables and their domains for the 4-queens problems, which constraints do we need to define?

(A) No two queens can be in the same row.

(B) No two queens can be in the same column.

(C) No two queens can be in the same diagonal.

(D) Two of (A), (B), and (C).

(E) All of (A), (B), and (C).

Q: Constraints for 4-Queens Problem

Q: Given the definitions of variables and their domains for the 4-queens problems, which constraints do we need to define?

(A) No two queens can be in the same row.

(B) No two queens can be in the same column.

(C) No two queens can be in the same diagonal.

(D) Two of (A), (B), and (C).

(E) All of (A), (B), and (C).

$\rightarrow$ Correct Answer: (D) Two of A, B, and C.
No need to specify column constraint. Already defined separate variables for each column. Implicitly saying that we put one queen in each column.

Defining Constraints

There are two ways of defining a constraint.

- The list/table format:
  Give a list/table of values of the variables that satisfy the constraints.
- The function/formula format:
  Give a function/formula, which returns/is true if the values of the variables satisfy the constraint.

$\rightarrow$ In an implementation, the second format is a function, which returns true if the values satisfy the constraint.

Q: Defining Constraints as a Table

Q: Suppose that we use a 2-column table to encode the following constraint. In each row of the table, the two values of x_0 and x_2 satisfy the constraint.

> *The two queens in columns 0 and 2 are not in the same row or diagonal.*

How many rows are there in this table?

(A) Less than 8

(B) 8

(C) 9

(D) 10

(E) More than 10

Q: Defining Constraints as a Table

Q: Suppose that we use a 2-column table to encode the following constraint. In each row of the table, the two values of x_0 and x_2 satisfy the constraint.

> *The two queens in columns 0 and 2 are not in the same row or diagonal.*

How many rows are there in this table?

(A) Less than 8

(B) 8

(C) 9

(D) 10

(E) More than 10

$\rightarrow$ Correct Answer: 8

Q: Defining Constraints as a Formula

Q: How should we encode the following constraint as a propositional formula?

The two queens in columns 0 and 2 are not in the same row or diagonal.

(A) $(x_0 \neq x_2)$

(B) $((x_0 \neq x_2) \wedge ((x_0 - x_2) \neq 1))$

(C) $((x_0 \neq x_2) \wedge ((x_0 - x_2) \neq 2))$

(D) $((x_0 \neq x_2) \wedge (|x_0 - x_2| \neq 1))$

(E) $((x_0 \neq x_2) \wedge (|x_0 - x_2| \neq 2))$

Q: Defining Constraints as a Formula

Q: How should we encode the following constraint as a propositional formula?

The two queens in columns 0 and 2 are not in the same row or diagonal.

(A) $(x_0 \neq x_2)$

(B) $((x_0 \neq x_2) \wedge ((x_0 - x_2) \neq 1))$

(C) $((x_0 \neq x_2) \wedge ((x_0 - x_2) \neq 2))$

(D) $((x_0 \neq x_2) \wedge (|x_0 - x_2| \neq 1))$

(E) $((\mathbf{x_0} \neq \mathbf{x_2}) \wedge (|\mathbf{x_0} - \mathbf{x_2}| \neq \mathbf{2}))$

$\rightarrow$ Correct Answer: (E) $((x_0 \neq x_2) \wedge (|x_0 - x_2| \neq 2))$

Expressing Constraints

- As a propositional formula:

$$((x_0 \neq x_2) \wedge (|x_0 - x_2| \neq 2))$$

- As a table of allowable combinations of values

x_0	x_1
0	1
0	3
1	0
1	2
2	1
2	3
3	0
3	2

4-Queens Incremental CSP Formulation

4-Queens Incremental CSP Formulation

- State: one queen per column in the leftmost k columns with no pair of queens attacking each other.
  - Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i, where $i \in \{0, 1, 2, 3, \_\}$. Exactly one queen is in each column. $x_i = \_$ denotes that column i does not have a queen.
  - Domains: $D_{x_i} = \{0, 1, 2, 3\}$ for all x_i.
  - Constraints: No pair of queens are in the same row or diagonal.

4-Queens Incremental CSP Formulation

- State: one queen per column in the leftmost k columns with no pair of queens attacking each other.
  - Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i, where $i \in \{0, 1, 2, 3, \_\}$. Exactly one queen is in each column. $x_i = \_$ denotes that column i does not have a queen.
  - Domains: $D_{x_i} = \{0, 1, 2, 3\}$ for all x_i.
  - Constraints: No pair of queens are in the same row or diagonal.
- Initial state: the empty board, that is, _ _ _ _.

4-Queens Incremental CSP Formulation

- State: one queen per column in the leftmost k columns with no pair of queens attacking each other.
  - Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i, where $i \in \{0, 1, 2, 3, \_\}$. Exactly one queen is in each column. $x_i = \_$ denotes that column i does not have a queen.
  - Domains: $D_{x_i} = \{0, 1, 2, 3\}$ for all x_i.
  - Constraints: No pair of queens are in the same row or diagonal.
- Initial state: the empty board, that is, _ _ _ _.
- Goal state: 4 queens on the board. No pair of queens are attacking each other. For example, 2 0 3 1 is a goal state.

4-Queens Incremental CSP Formulation

- State: one queen per column in the leftmost k columns with no pair of queens attacking each other.
  - Variables: x_0, x_1, x_2, x_3 where x_i is the row position of the queen in column i, where $i \in \{0, 1, 2, 3, \_\}$. Exactly one queen is in each column. $x_i = \_$ denotes that column i does not have a queen.
  - Domains: $D_{x_i} = \{0, 1, 2, 3\}$ for all x_i.
  - Constraints: No pair of queens are in the same row or diagonal.
- Initial state: the empty board, that is, _ _ _ _.
- Goal state: 4 queens on the board. No pair of queens are attacking each other. For example, 2 0 3 1 is a goal state.
- Successor function: add a queen to the leftmost empty column such that it is not attacked by any other existing queen. For example, 0 _ _ _ has two successors 0 2 _ _ and 0 3 _ _.

Backtracking Search

Algorithm 1 BACKTRACK(assignment, csp)

```
if assignment is complete then return assignment
Let var be an unassigned variable
for every value in the domain of var do
    if adding {var = value} satisfies every constraint then
        add {var = value} to assignment
        result ← BACKTRACK(assignment, csp)
        if result ≠ failure then return result
    remove {var = value} from assignment if it was added
return failure
```

Solve 4-Queens using Backtracking Search

Step 0:

– – – –

	x_0	x_1	x_2	x_3
0				
1				
2				
3				

Solve 4-Queens using Backtracking Search

Step 1:

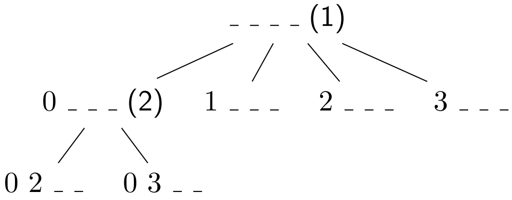
_ _ _ _ (1)

0 _ _ _   1 _ _ _   2 _ _ _   3 _ _ _

	x_0	x_1	x_2	x_3
0				
1				
2				
3				

Solve 4-Queens using Backtracking Search

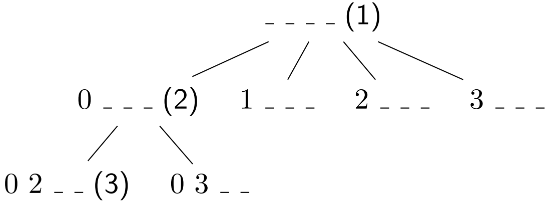
Step 2:



	x_0	x_1	x_2	x_3
0	Q	X	X	X
1	X	X		
2	X		X	
3	X			X

Solve 4-Queens using Backtracking Search

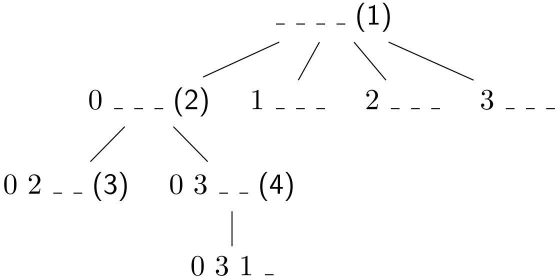
Step 3:



	x_0	x_1	x_2	x_3
0	Q	X	X	X
1	X	X	X	
2	X	Q	X	
3	X	X	X	X

Solve 4-Queens using Backtracking Search

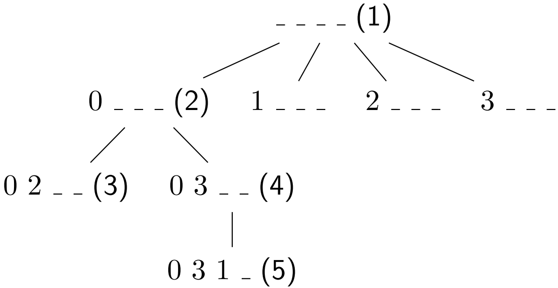
Step 4:



	x_0	x_1	x_2	x_3
0	Q	X	X	X
1	X	X		X
2	X	X	X	
3	X	Q	X	X

Solve 4-Queens using Backtracking Search

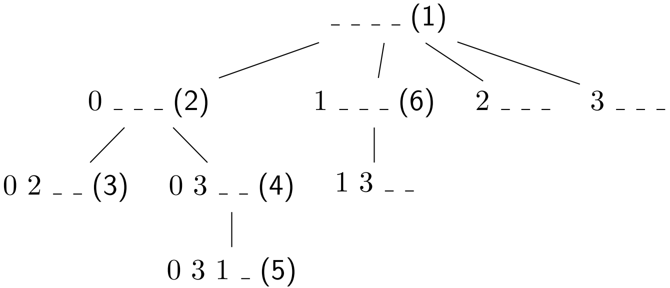
Step 5:



	x_0	x_1	x_2	x_3
0	Q	X	X	X
1	X	X	Q	X
2	X	X	X	X
3	X	Q	X	X

Solve 4-Queens using Backtracking Search

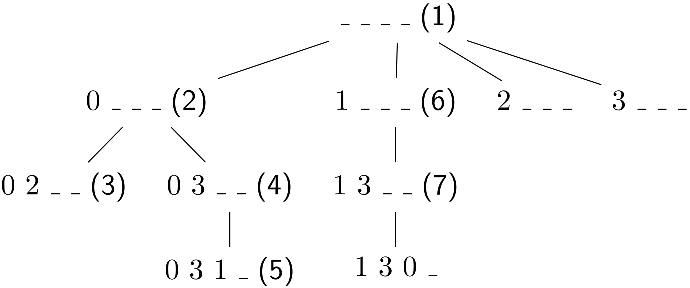
Step 6:



	x_0	x_1	x_2	x_3
0	X	X		
1	Q	X	X	X
2	X	X		
3	X		X	

Solve 4-Queens using Backtracking Search

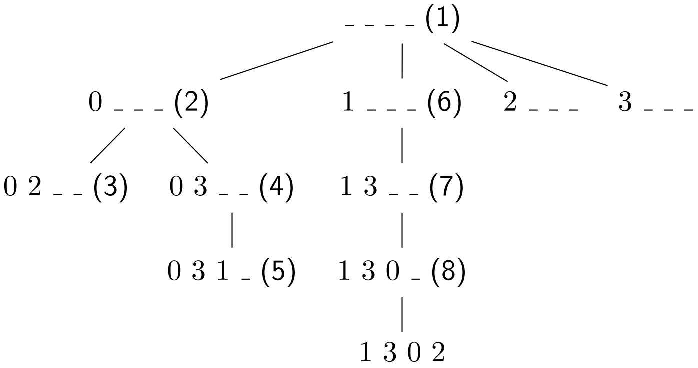
Step 7:



	x_0	x_1	x_2	x_3
0	X	X		
1	Q	X	X	X
2	X	X	X	
3	X	Q	X	X

Solve 4-Queens using Backtracking Search

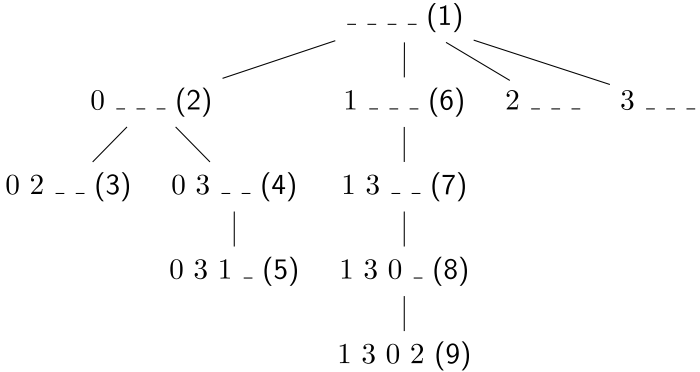
Step 8:



	x_0	x_1	x_2	x_3
0	X	X	Q	X
1	Q	X	X	X
2	X	X	X	
3	X	Q	X	X

Solve 4-Queens using Backtracking Search

Step 9:



	x_0	x_1	x_2	x_3
0	X	X	Q	X
1	Q	X	X	X
2	X	X	X	Q
3	X	Q	X	X

The Idea of Arc Consistency

$x_0 = 0$ and $x_1 = 2$ do not lead to a solution. Why?

	x_0	x_1	x_2	x_3
0	Q	X	X	X
1	X	X	X	
2	X	Q	X	X
3	X	X	X	X

Handling Different Types of Constraints

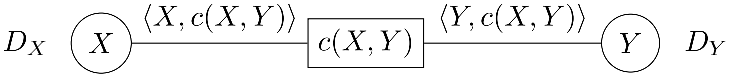
- Consider binary constraints only.
- How should we handle unary constraints?

  $\rightarrow$ Remove invalid values from variable domain
- How should be handle constraints involving 3 or more variables?

  $\rightarrow$ Convert them to binary constraints

Notation for an Arc

- X and Y are two variables. $c(X,Y)$ is a binary constraint.



- $\langle X, c(X,Y)\rangle$ denotes an arc.
  X is the *primary variable* and Y is the *secondary variable*.

The Arc Consistency Definition

Definition (Arc Consistency)

An arc $\langle X, c(X, Y) \rangle$ is arc-consistent if and only if for every value $v \in D_X$, there is a value $w \in D_Y$ such that (v, w) satisfies the constraint $c(X, Y)$.

Q: Applying The Arc Consistency Definition

Q: Consider the constraint $X < Y$. Let $D_X = \{1, 2\}$ and $D_Y = \{1, 2\}$. Is the arc $\langle X, X < Y \rangle$ consistent?

(A) Yes.

(B) No.

(C) Maybe?

(D) I don't know.

Q: Applying The Arc Consistency Definition

Q: Consider the constraint $X < Y$. Let $D_X = \{1, 2\}$ and $D_Y = \{1, 2\}$. Is the arc $\langle X, X < Y \rangle$ consistent?

(A) Yes.

(B) **No.**

(C) Maybe?

(D) I don't know.

→ Correct answer is (B) No. We can remove 2 from D_X.

The AC-3 Arc Consistency Algorithm

Algorithm 2 The AC-3 Algorithm

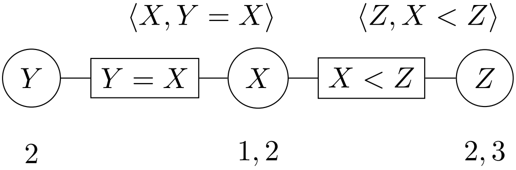
```
put every arc in the set S.
while S is not empty do
    select and remove ⟨X, c(X, Y)⟩ from S
    remove every value in D_X that doesn't have a value in D_Y that satisfies the constraint c(X, Y)
    if D_X was reduced then
        if D_X is empty then return false
        for every Z ≠ Y, add ⟨Z, c'(Z, X)⟩ to S
return true
```

Why do we need to add arcs back to S?

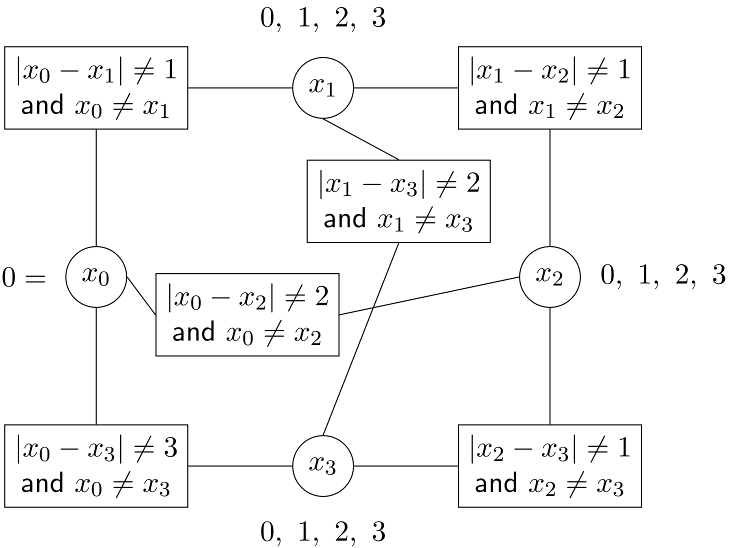
Q: After reducing a variable's domain,
we may need to add arcs back to S. Why?

A: Reducing a variable's domain may
cause a previously consistent arc to become inconsistent.

Example:



Trace the AC-3 Algorithm on 4-Queens Problem



Properties of the AC-3 Algorithm

- Does the order of removing arcs matter?

Properties of the AC-3 Algorithm

- Does the order of removing arcs matter?

  $\rightarrow$ No.

Properties of the AC-3 Algorithm

- Does the order of removing arcs matter?

  $\rightarrow$ No.

- Three possible outcomes of the arc consistency algorithm:

Properties of the AC-3 Algorithm

- Does the order of removing arcs matter?

  $\rightarrow$ No.

- Three possible outcomes of the arc consistency algorithm:

  $\rightarrow$

  1. A domain is empty. No solution.
  2. Every domain has 1 value left. Found the solution without search.
  3. Every domain has at least 1 value left and some domain has multiple values left. Need search to find a solution.

Properties of the AC-3 Algorithm

- Is AC-3 guaranteed to terminate?

  → Yes.

Properties of the AC-3 Algorithm

- Is AC-3 guaranteed to terminate?

  $\rightarrow$ Yes.

- What is the complexity of AC-3?

Properties of the AC-3 Algorithm

- Is AC-3 guaranteed to terminate?

  $\rightarrow$ Yes.

- What is the complexity of AC-3? $\rightarrow$ Time complexity: $O(cd^3)$.
  n variables, c binary constraints, and the size of each domain is at most d. Each arc (X_k, X_i) can be added to the queue at most d times because we can delete at most d values from X_i. Checking consistency of each arc can be done in $O(d^2)$ time.

Combining Backtracking and Arc Consistency

1. Perform backtracking search.
2. After each value assignment, perform arc consistency.
3. If a domain is empty, terminate and return no solution.
4. If a unique solution is found, return the solution.
5. Otherwise, continue with backtracking search
   on the unassigned variables.

Solving 4-Queens Problem with Backtracking and AC-3

Revisiting the Learning Goals

By the end of the lecture, you should be able to

- Formulate a real-world problem as a constraint satisfaction problem.
- Trace the execution of the backtracking search algorithm.
- Verify whether a constraint is arc-consistent.
- Trace the execution of the AC-3 arc consistency algorithm.
- Trace the execution of the backtracking search algorithm with arc consistency.