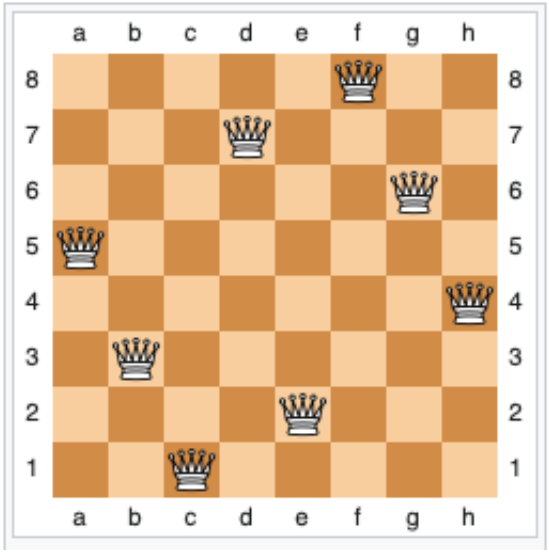
Constraint Satisfaction Problems

Musad Haque
03-Nov-2023

Motivating Example

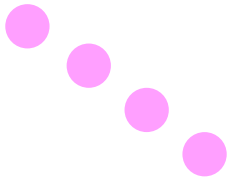
- n-queens: place n queens on an nxn board so that no two queens are attacking each other



https://en.wikipedia.org/wiki/Eight_queens_puzzle

Motivating Example

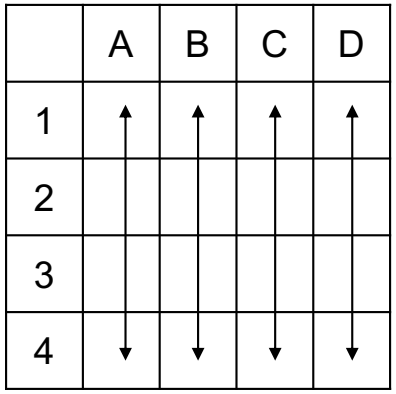
- 4-queens:



	A	B	C	D
1				
2				
3				
4				

Motivating Example

- Imagine a queen is assigned to each column and we have to figure out which row to place it in



Just to make our lives easier, we're restricting the degrees of freedom for a piece (rows only vs rows AND columns)

Terminology

4-queens:

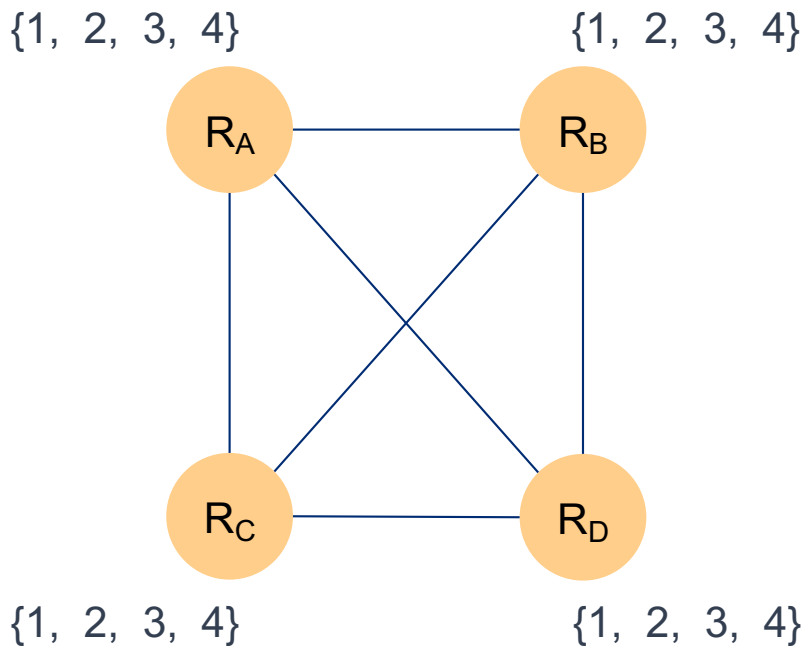
- **X** is a set of variables — $\{R_A, R_B, R_C, R_D\}$
- **D** is the domain set — {1, 2, 3, 4}
- **C** is the set of constraints — R_i != R_j and Diag(R_i, R_j) not allowed

- An **assignment** is a particular setting of some or all the variables to values
- An assignment is **consistent** if it doesn't violate any constraints
- An assignment is **complete** if all variables have values

- A **solution** is a complete, consistent assignment

Terminology

- **Constraint Graph** consists a set of **nodes** (variables), and **edges** (between variables that share constraints)

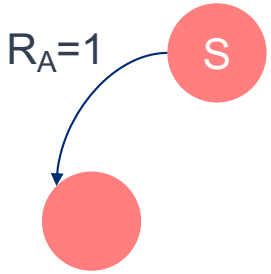


In n-queens, all variables share constraints, so we have a complete graph

Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)

	A	B	C	D
1	●			
2				
3				
4				

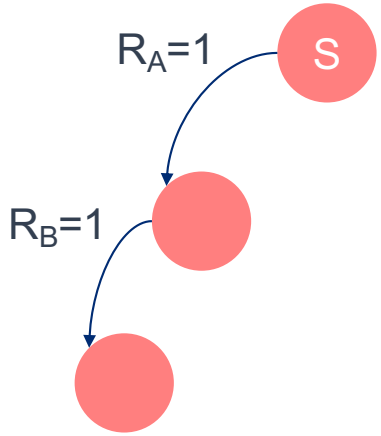


Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)
- Next, R_B=1

We immediately have a problem. We've violated a constraint. We need to backtrack to the last place we made a decision.

	A	B	C	D
1	●	●		
2				
3				
4				

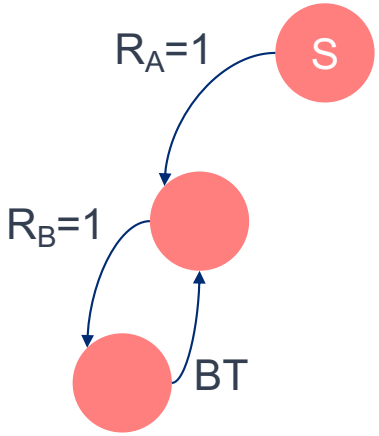


Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)
- Next, R_B=1

We immediately have a problem. We've violated a constraint. We need to backtrack to the last place we made a decision.

	A	B	C	D
1	●	●		
2				
3				
4				

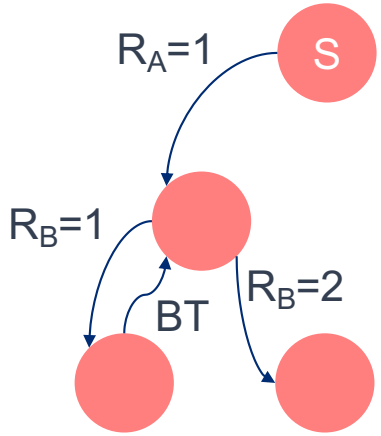


Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)
- Try, R_B=2

Violated constraint. Backtrack (BT) again.

	A	B	C	D
1	●			
2		●		
3				
4				

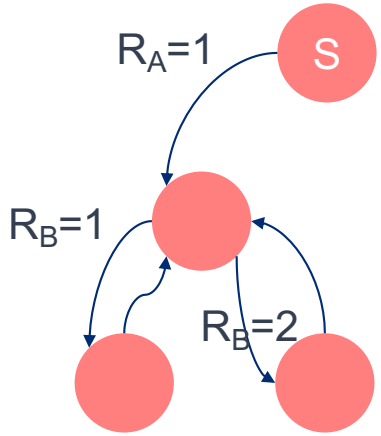


Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)
- Try, R_B=2

Violated constraint. Backtrack (BT) again.

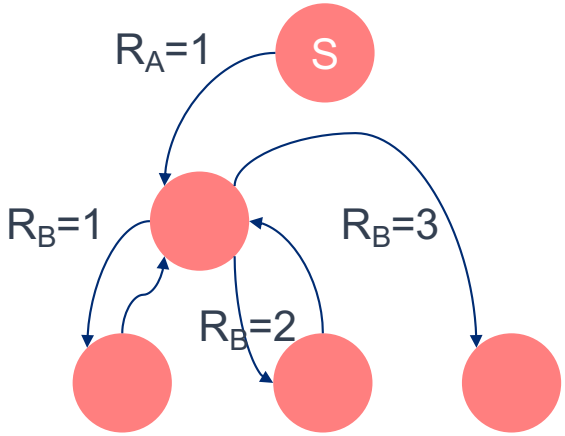
	A	B	C	D
1	●			
2		●		
3				
4				



Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)
- Try, R_B=3

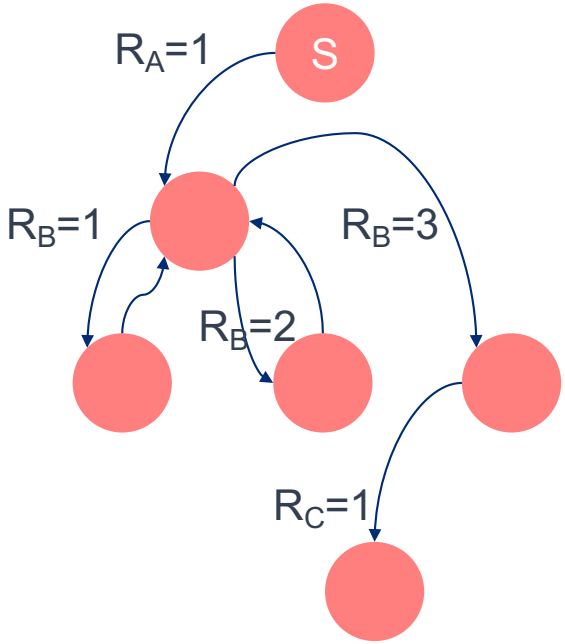
	A	B	C	D
1	●			
2				
3		●		
4				



Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)
- Next, R_C=1

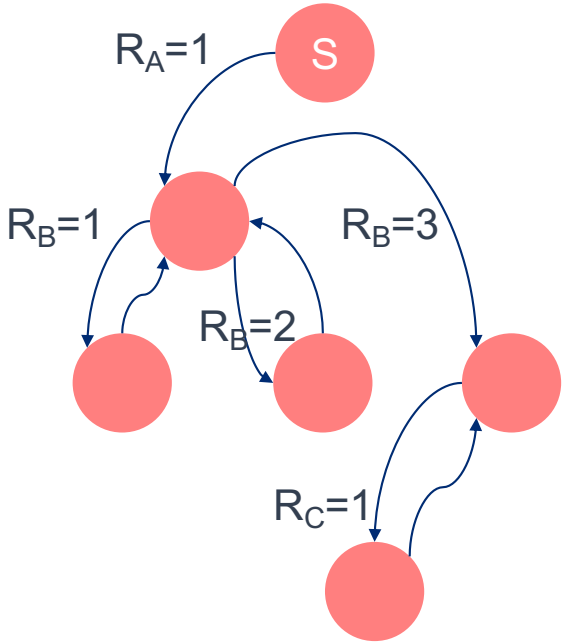
	A	B	C	D
1	●		●	
2				
3		●		
4				



Feature #1: Backtracking

- Start with R_A=1
- (first queen, first available position)
- Next, R_C=1

	A	B	C	D
1	●		●	
2				
3		●		
4				



Ran into an inconsistent assignment, backtrack (BT) again.

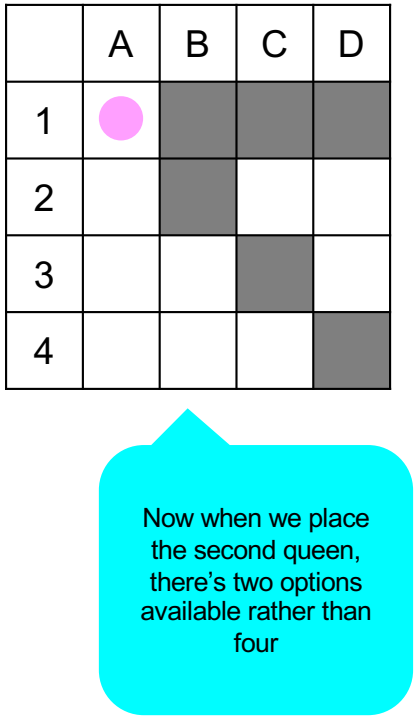
Can we be smarter?

Yes, Forward Check (FC).

After setting a value, remove it from the domain of other variables, so we spend less time backtracking.

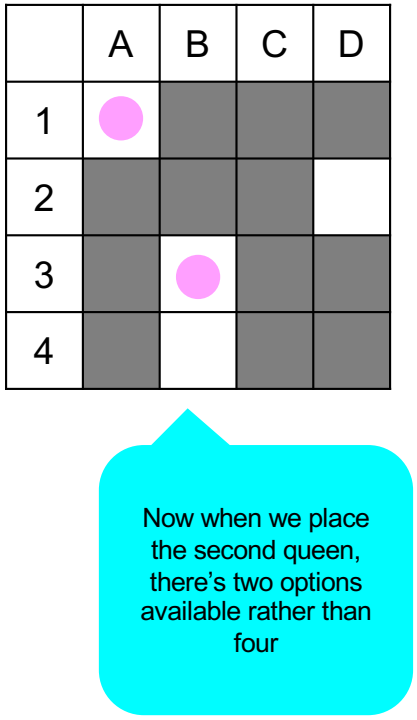
Feature #2: Forward Checking

- Start with R_A=1
- Then, remove values from other variables by enforcing the constraints



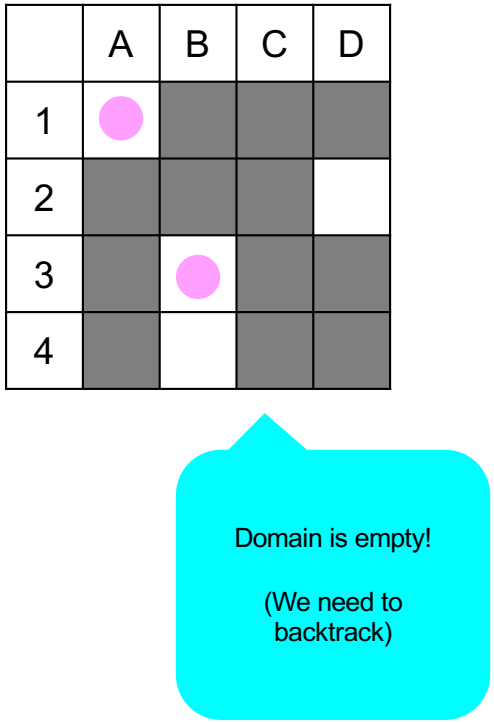
Feature #2: Forward Checking

- Start with R_A=1
- Then, remove values from other variables by enforcing the constraints
- Set R_B=3
- Remove values from other variables

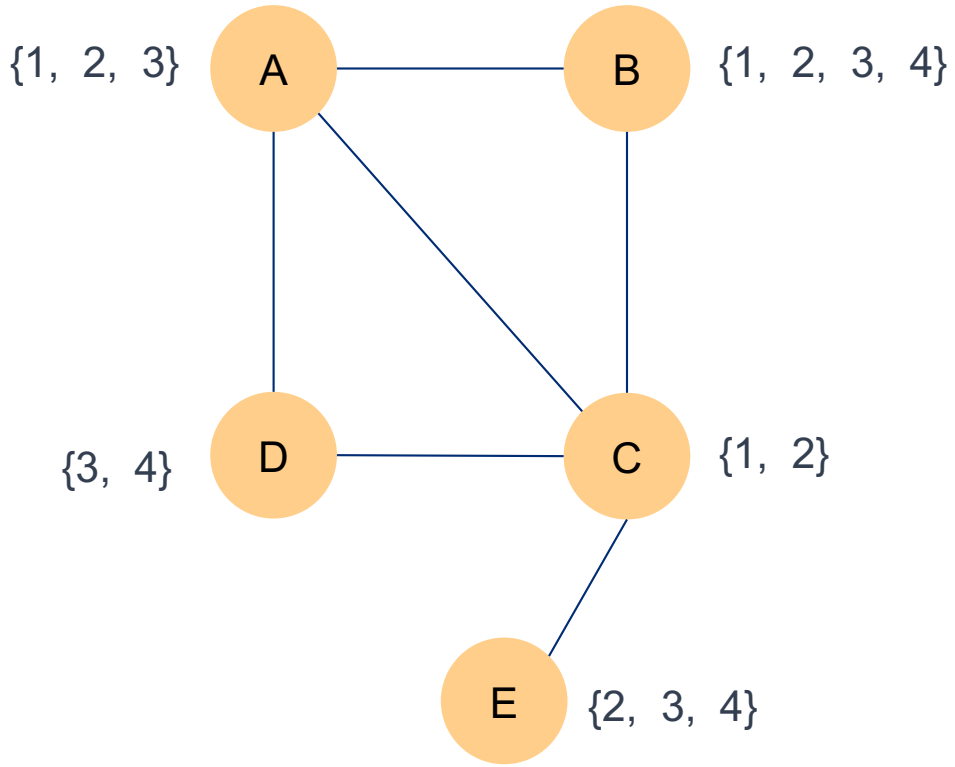


Feature #2: Forward Checking

- Start with R_A=1
- Then, remove values from other variables by enforcing the constraints
- Set R_B=3
- Remove values from other variables
- Move on to R_C

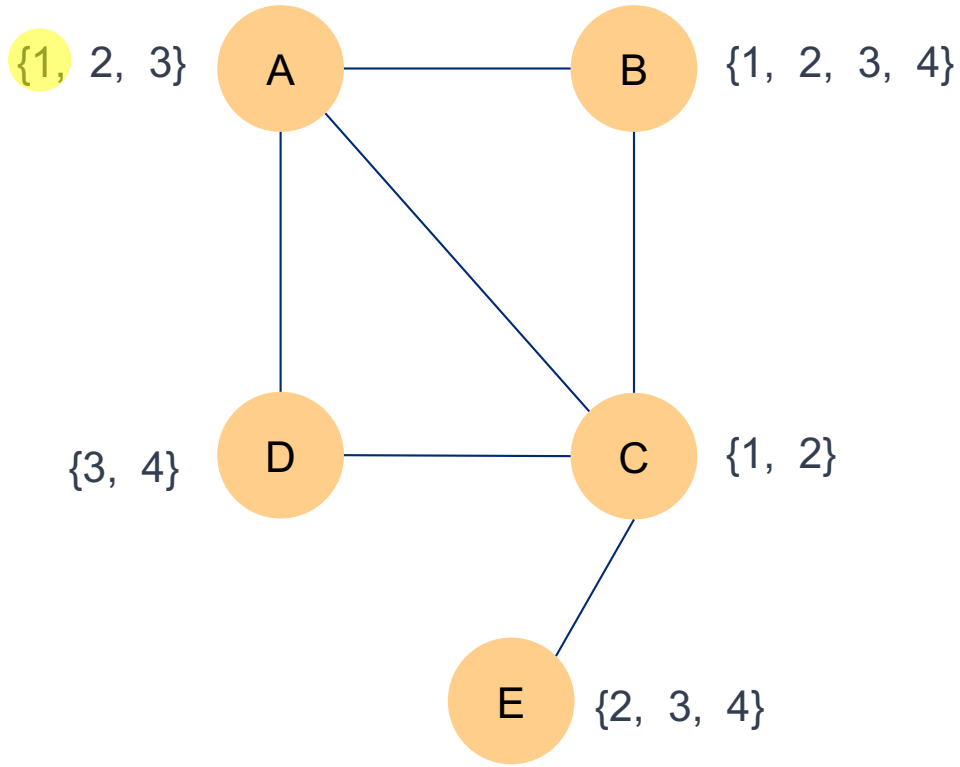


New Example



New Example

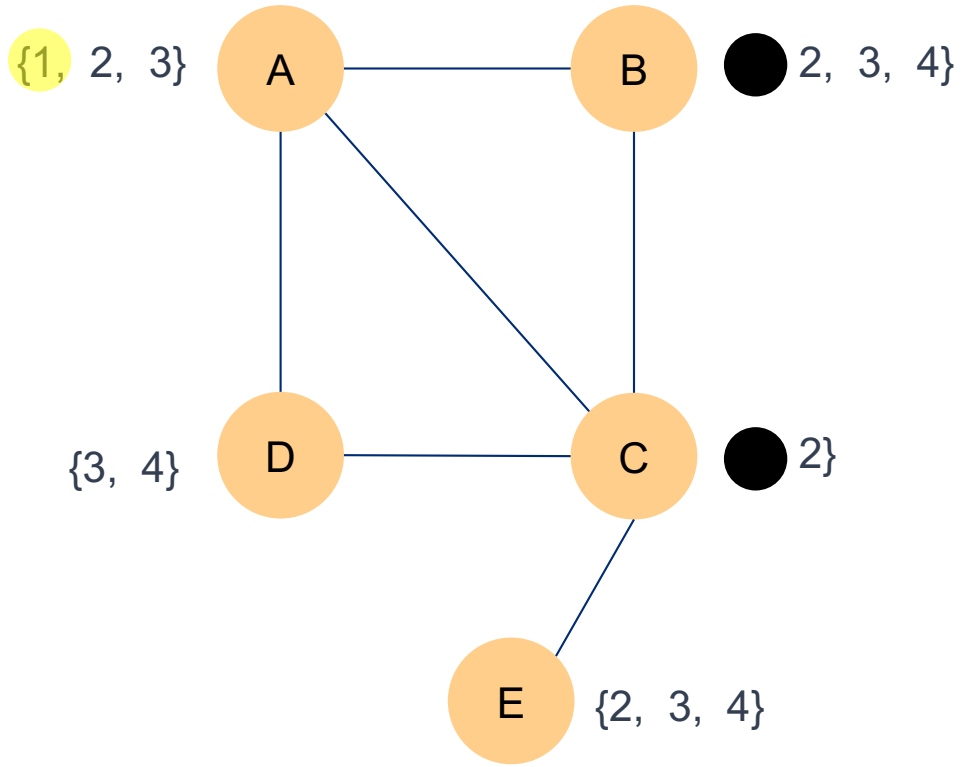
1. Pick A, assign 1



New Example

1. Pick A, assign 1

FC

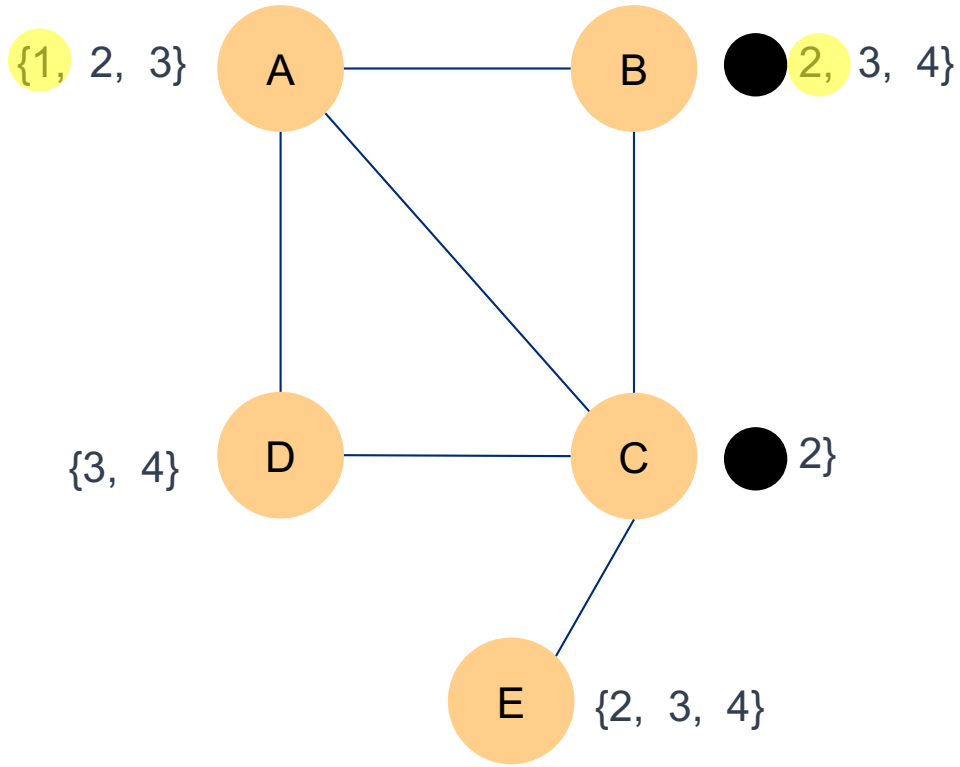


New Example

1. Pick A, assign 1

FC

2. Pick B, assign 2



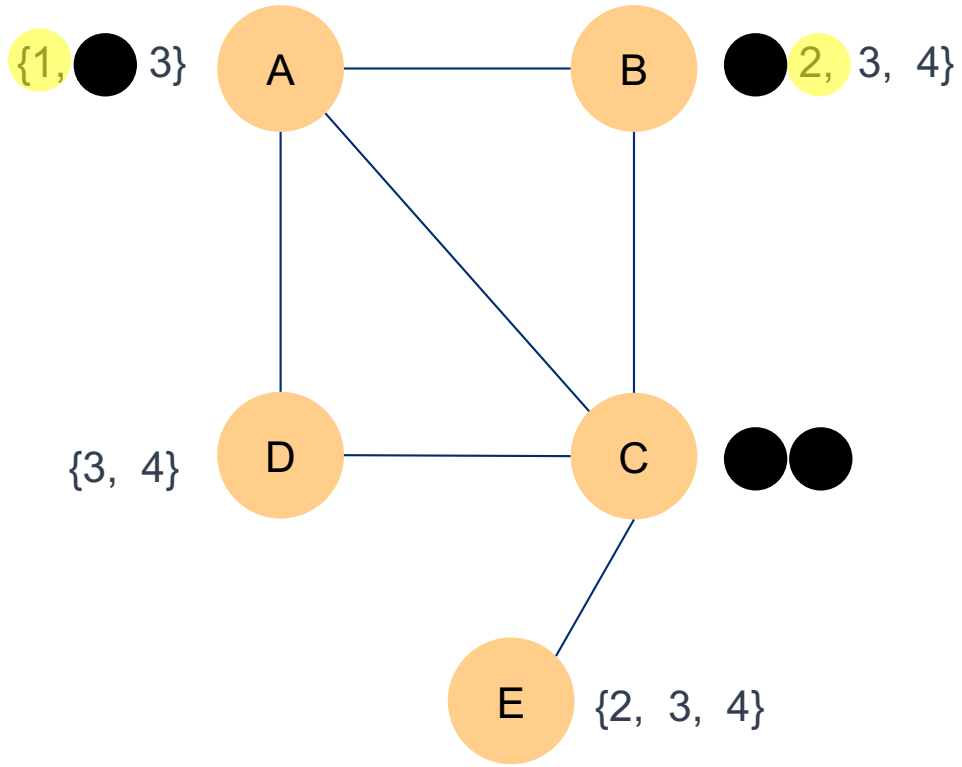
New Example

1. Pick A, assign 1

FC

2. Pick B, assign 2

FC



New Example

1. Pick A, assign 1

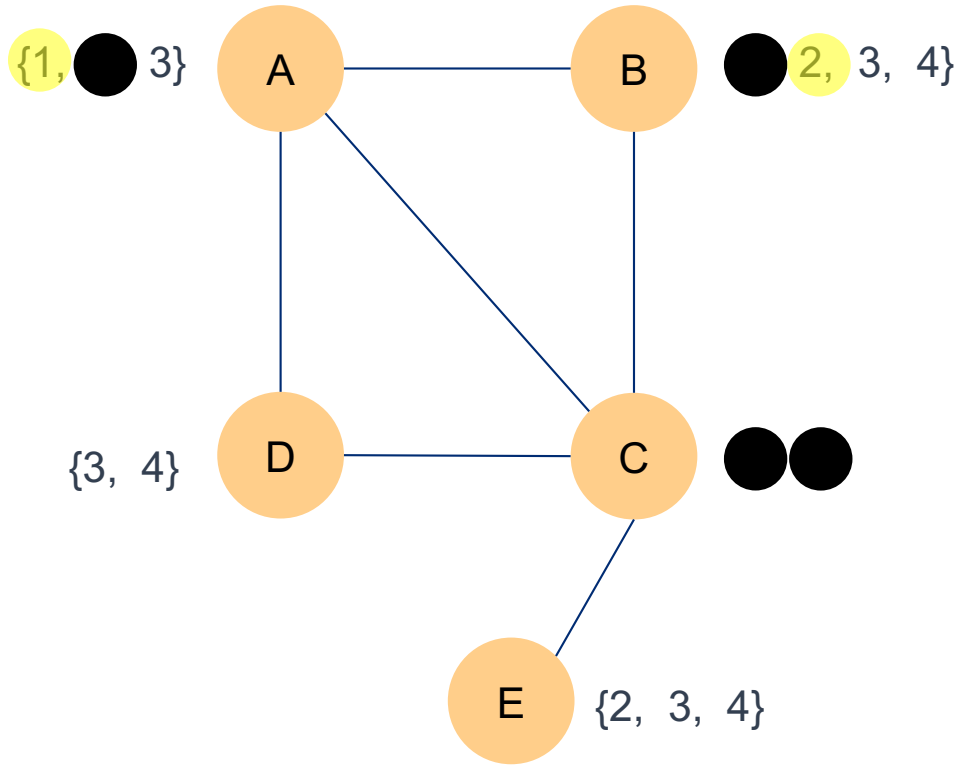
FC

2. Pick B, assign 2

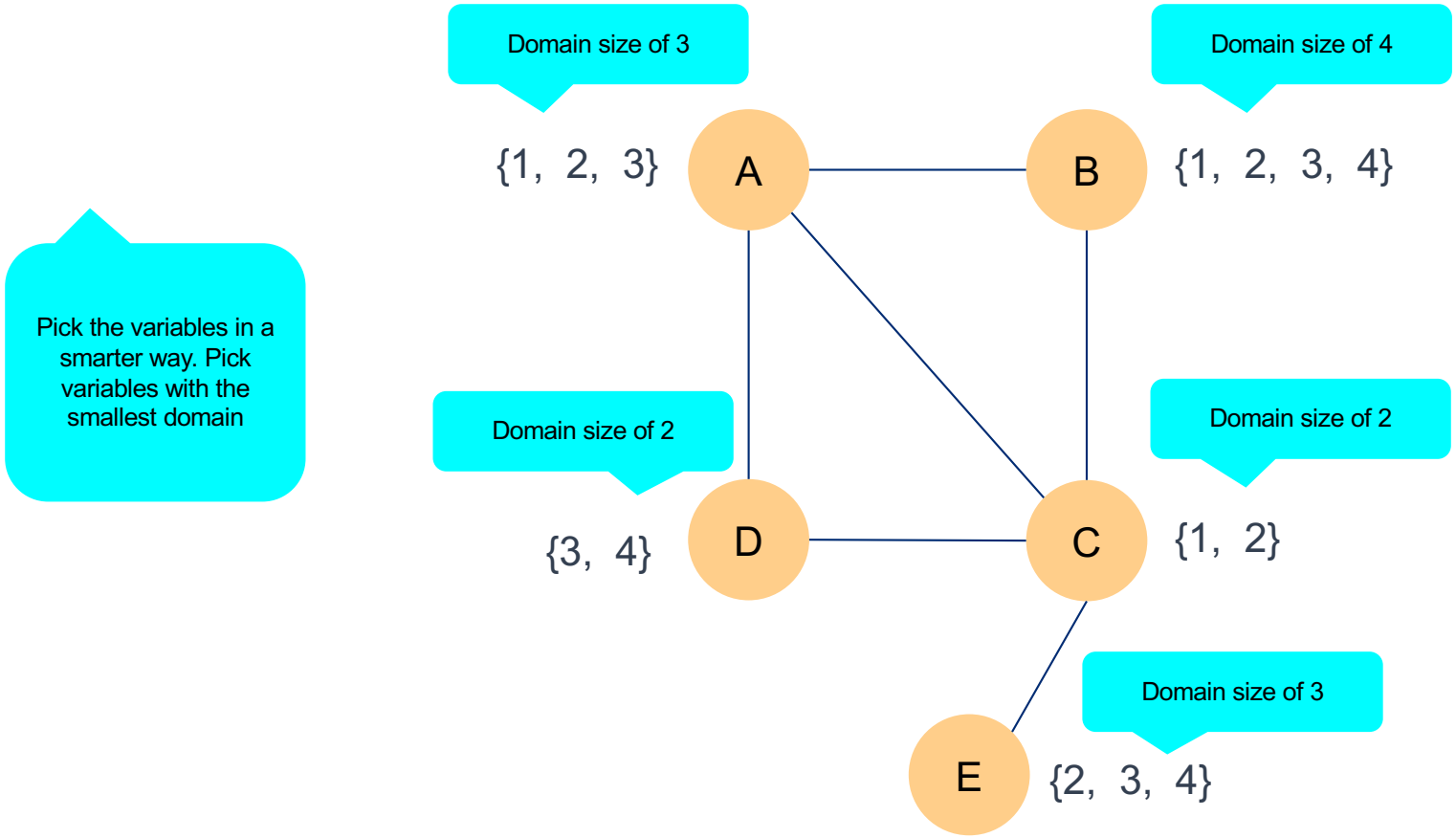
FC

3. Pick C

Domain empty, so backtrack

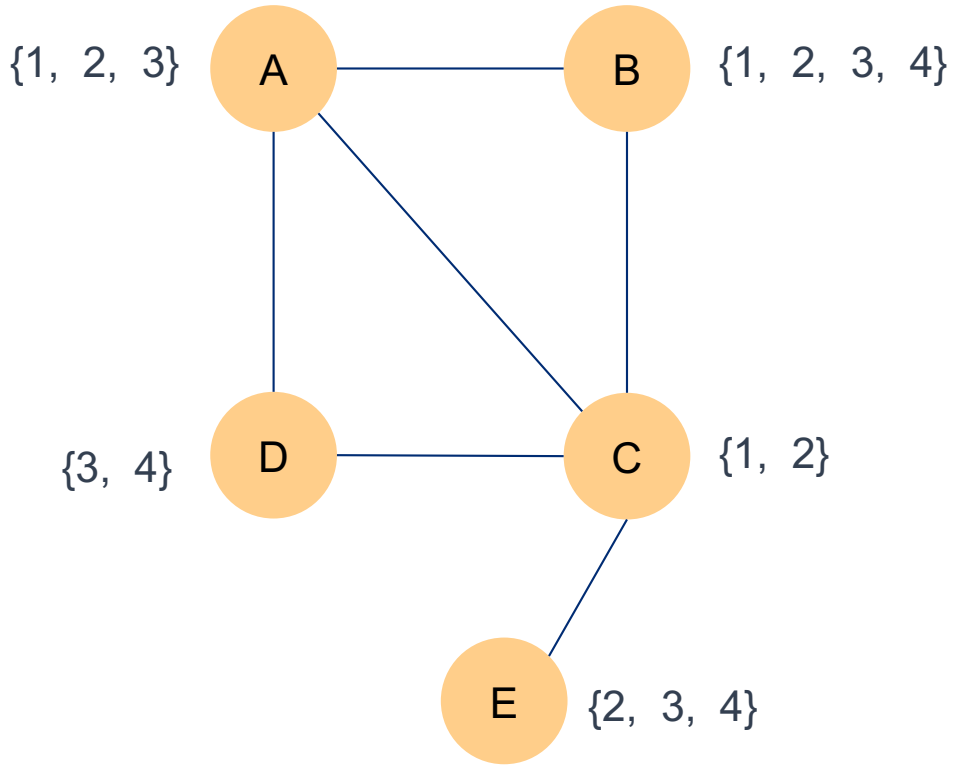


Feature #3a: Minimum Remaining Values



Feature #3a: Minimum Remaining Values

1. Pick either D or C

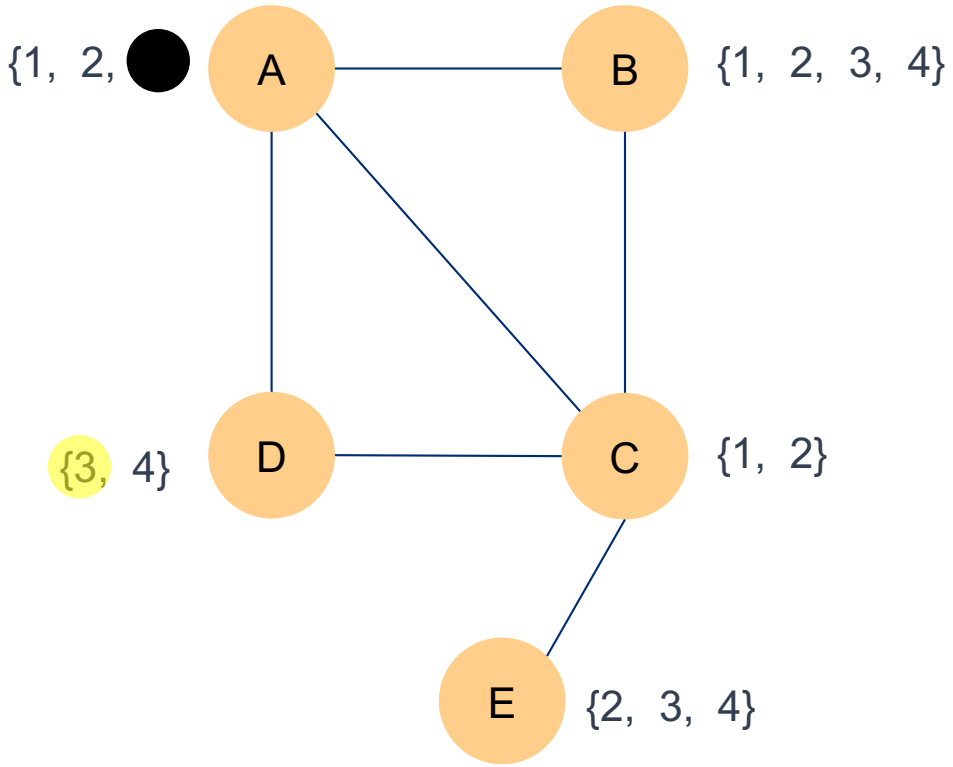


Feature #3a: Minimum Remaining Values

1. Pick D, assign 3

FC

2. Pick either A or C. They have a domain size of 2.



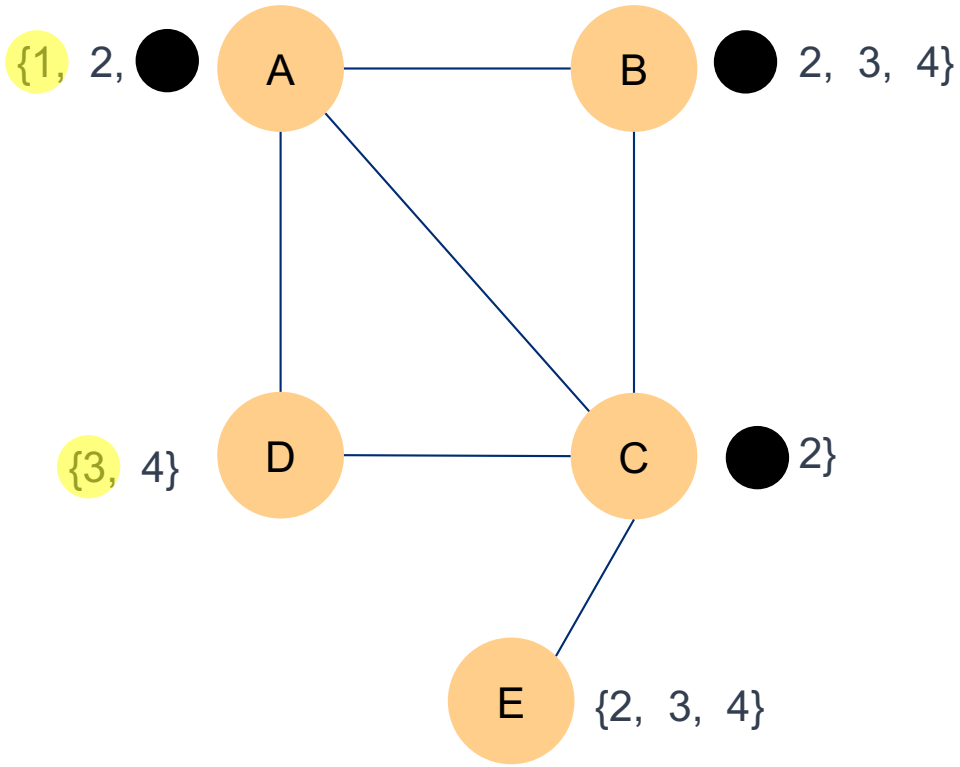
Feature #3a: Minimum Remaining Values

1. Pick D, assign 3

FC

2. Pick A, assign 1

FC



Feature #3a: Minimum Remaining Values

1. Pick D, assign 3

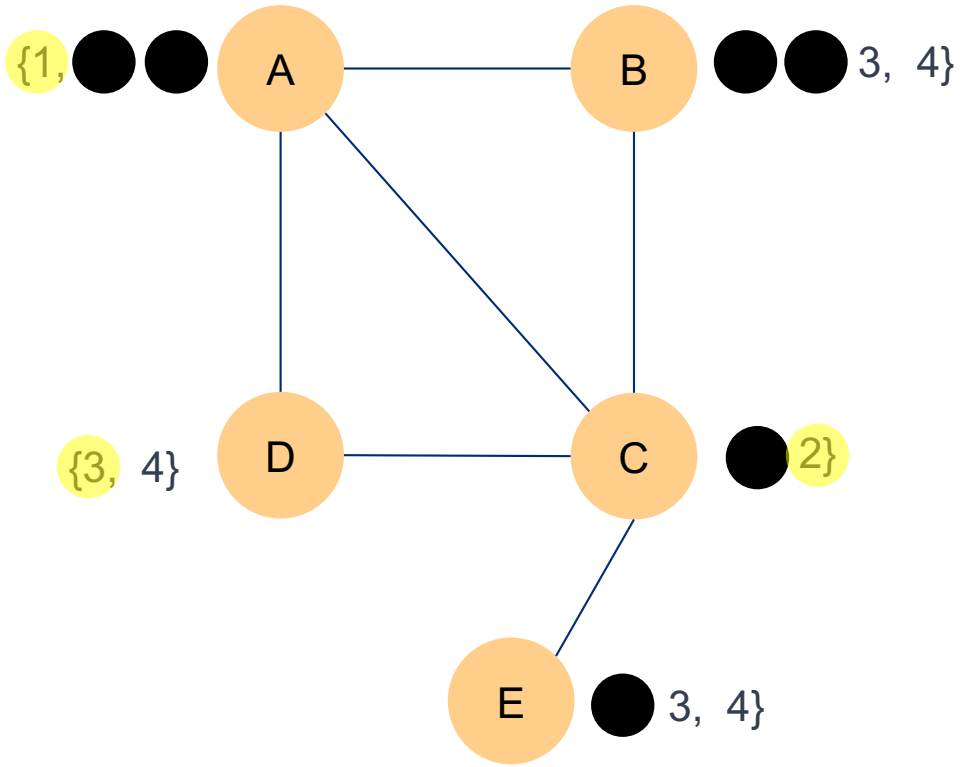
FC

2. Pick A, assign 1

FC

3. Pick C, assign 2

FC



Feature #3a: Minimum Remaining Values

1. Pick D, assign 3

FC

2. Pick A, assign 1

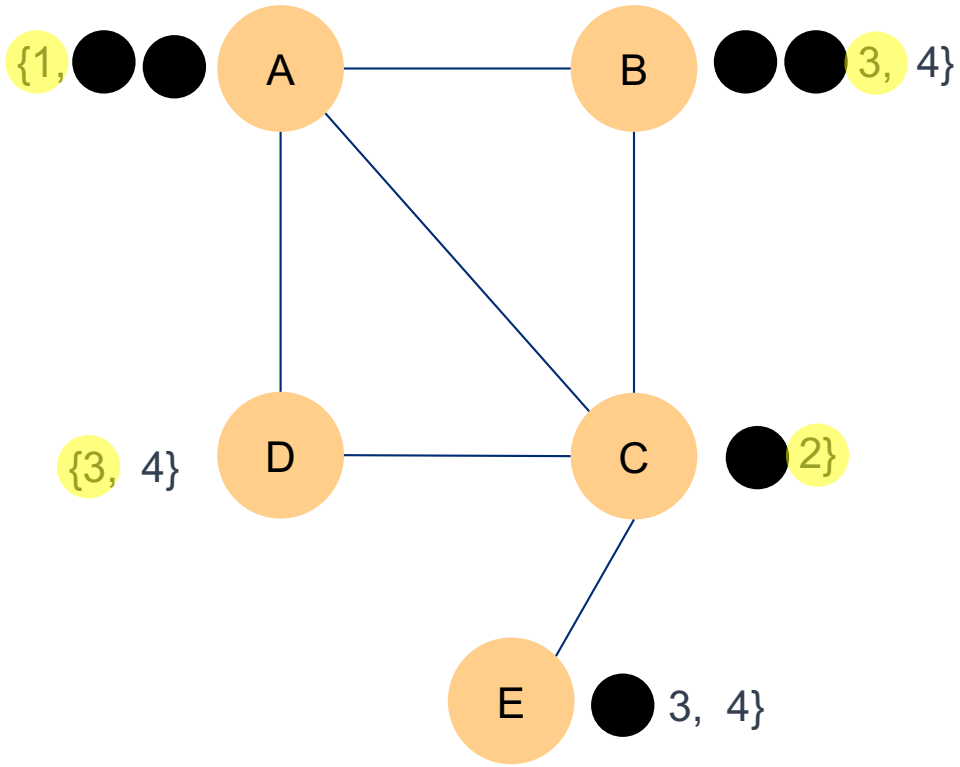
FC

3. Pick C, assign 2

FC

4. Pick B, assign 3

FC



Feature #3a: Minimum Remaining Values

1. Pick D, assign 3

FC

2. Pick A, assign 1

FC

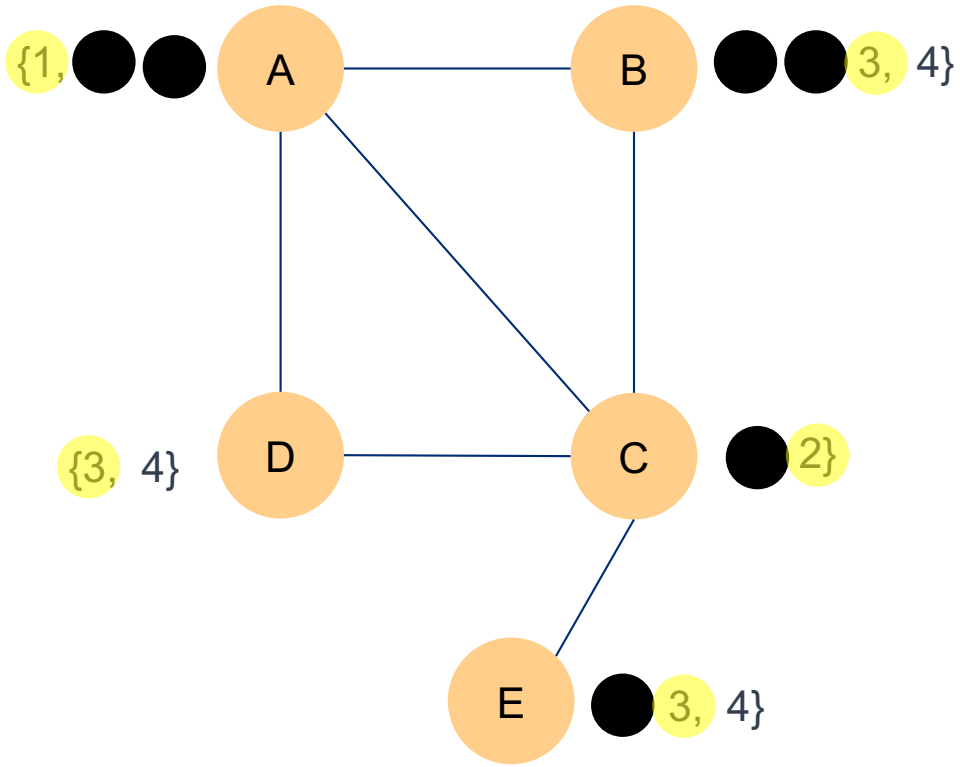
3. Pick C, assign 2

FC

4. Pick B, assign 3

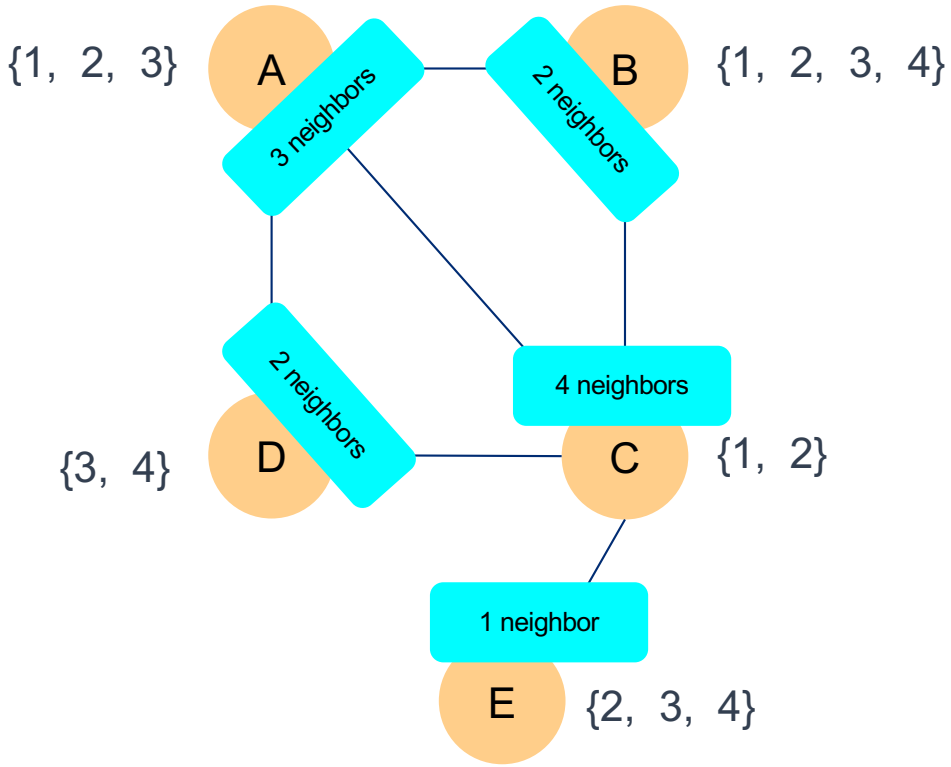
FC

5. Pick E, assign 3



Feature #3b: Degree Heuristic

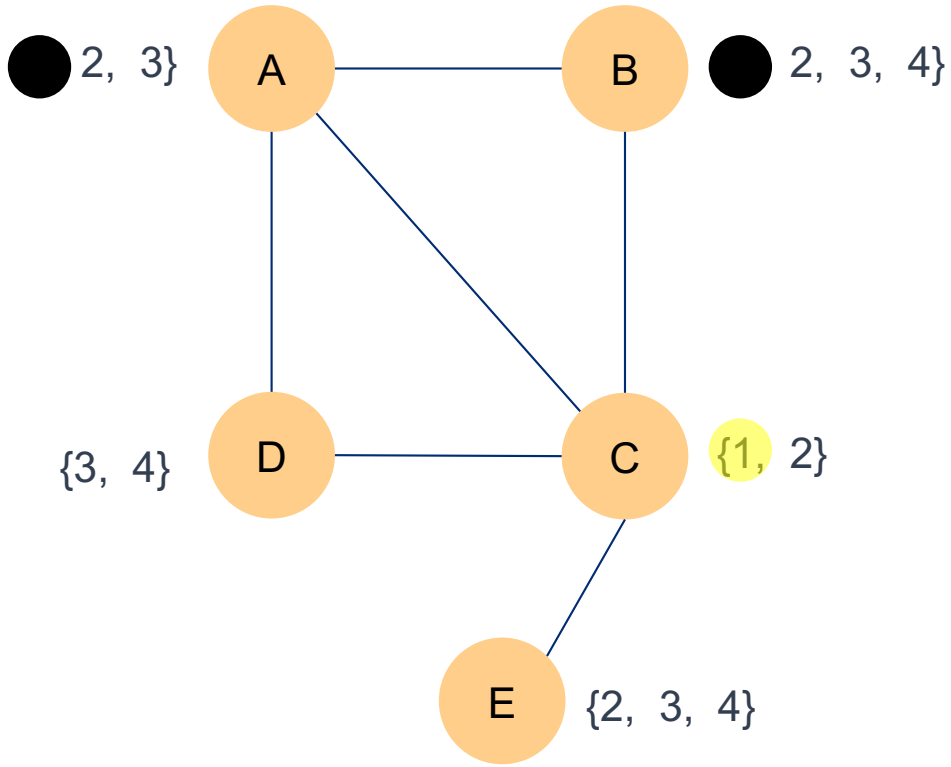
Pick variables in order of degree: largest first. Degree (or number of neighbors) of a variable doesn't change.



Feature #3b: Degree Heuristic

1. Pick C, assign 1

FC



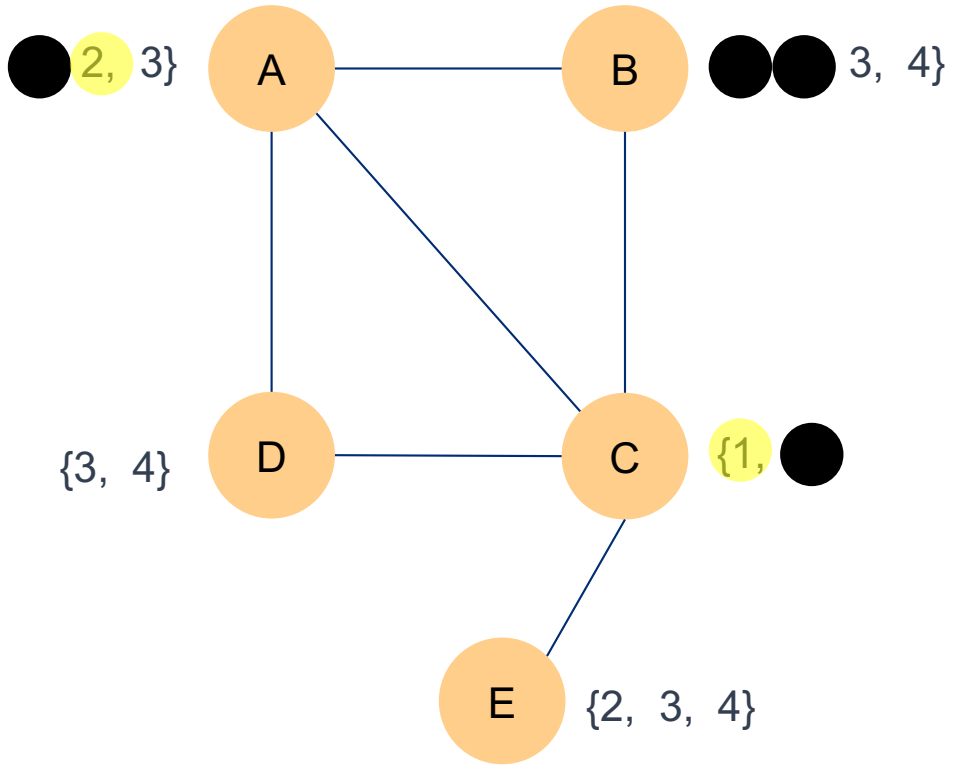
Feature #3b: Degree Heuristic

1. Pick C, assign 1

FC

2. Pick A, assign 2

FC



Feature #3b: Degree Heuristic

1. Pick C, assign 1

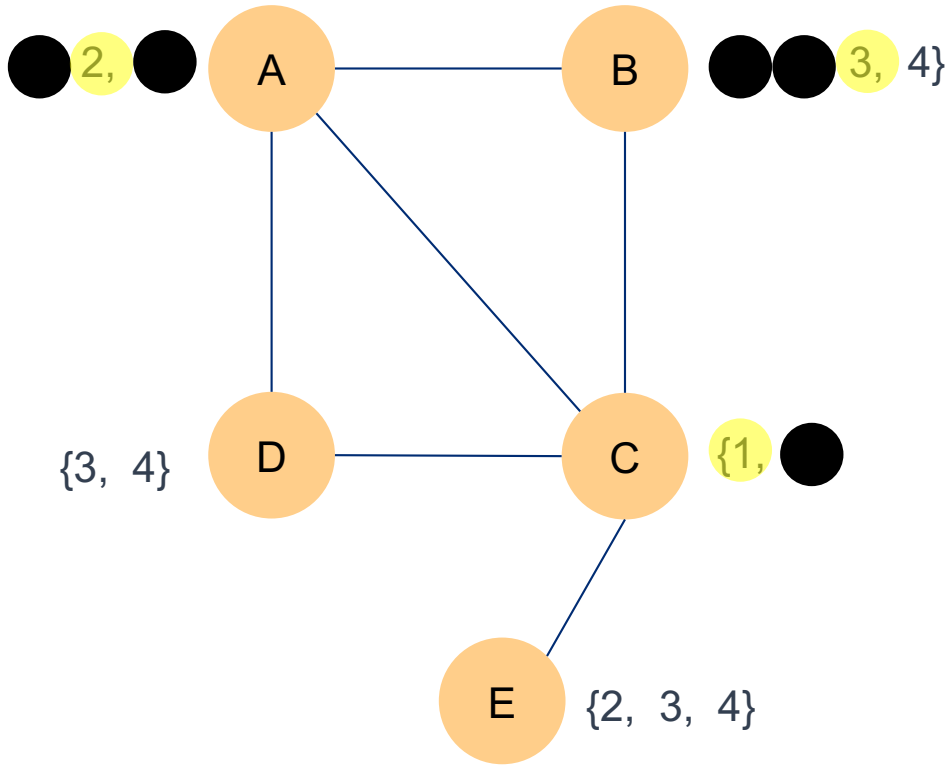
FC

2. Pick A, assign 2

FC

3. Pick B, assign 3

FC



Feature #3b: Degree Heuristic

1. Pick C, assign 1

FC

2. Pick A, assign 2

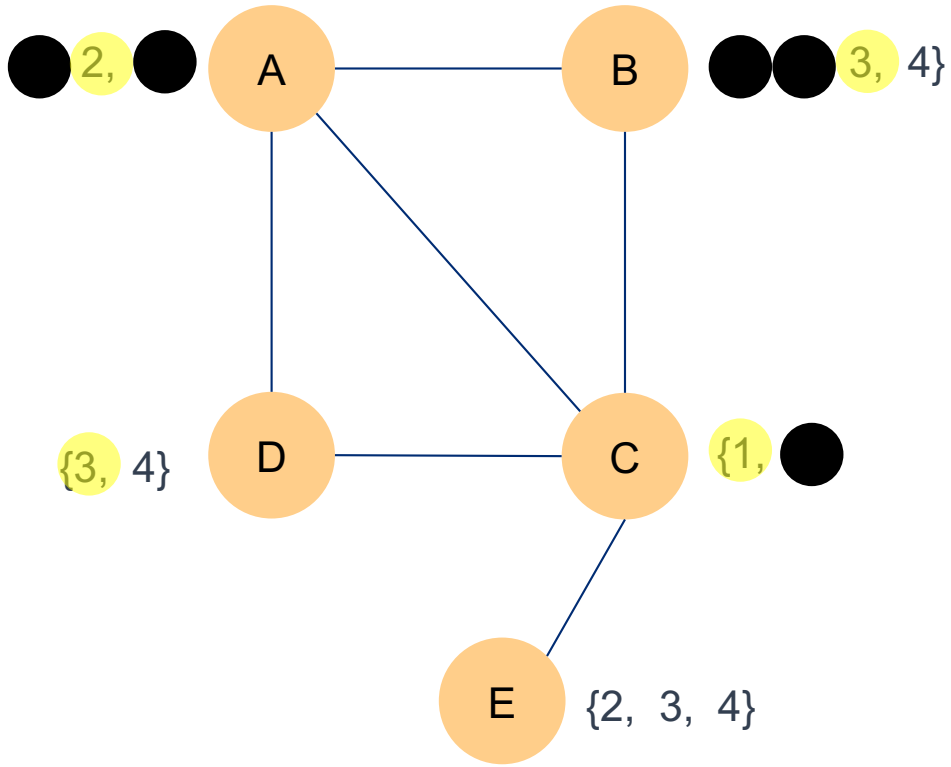
FC

3. Pick B, assign 3

FC

4. Pick D, assign 3

FC



Feature #3b: Degree Heuristic

1. Pick C, assign 1

FC

2. Pick A, assign 2

FC

3. Pick B, assign 3

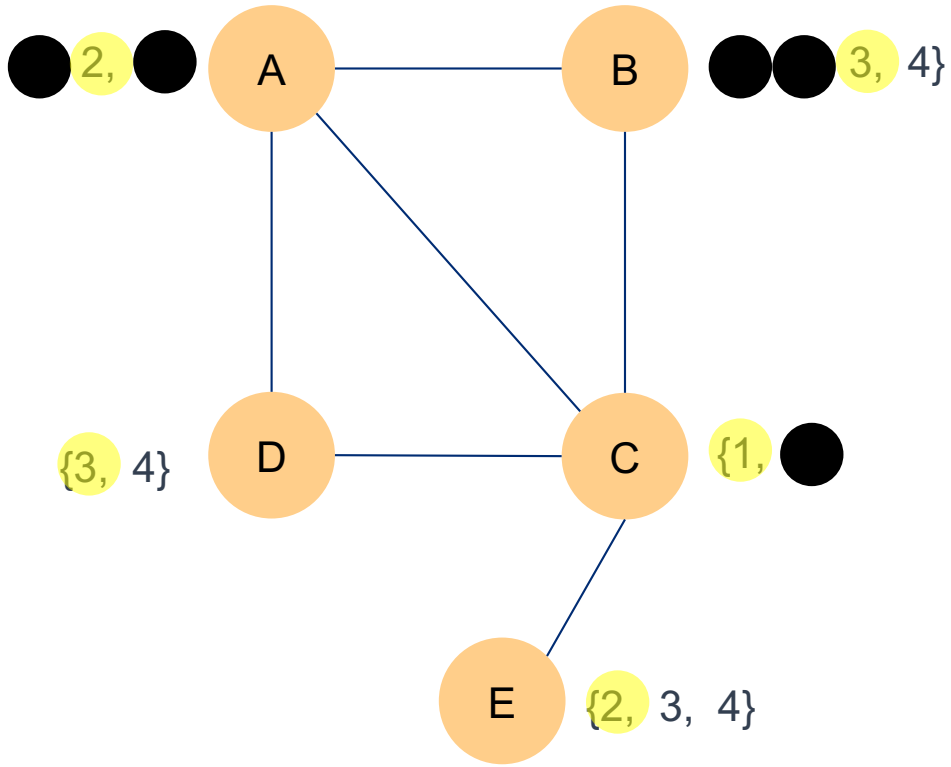
FC

4. Pick D, assign 3

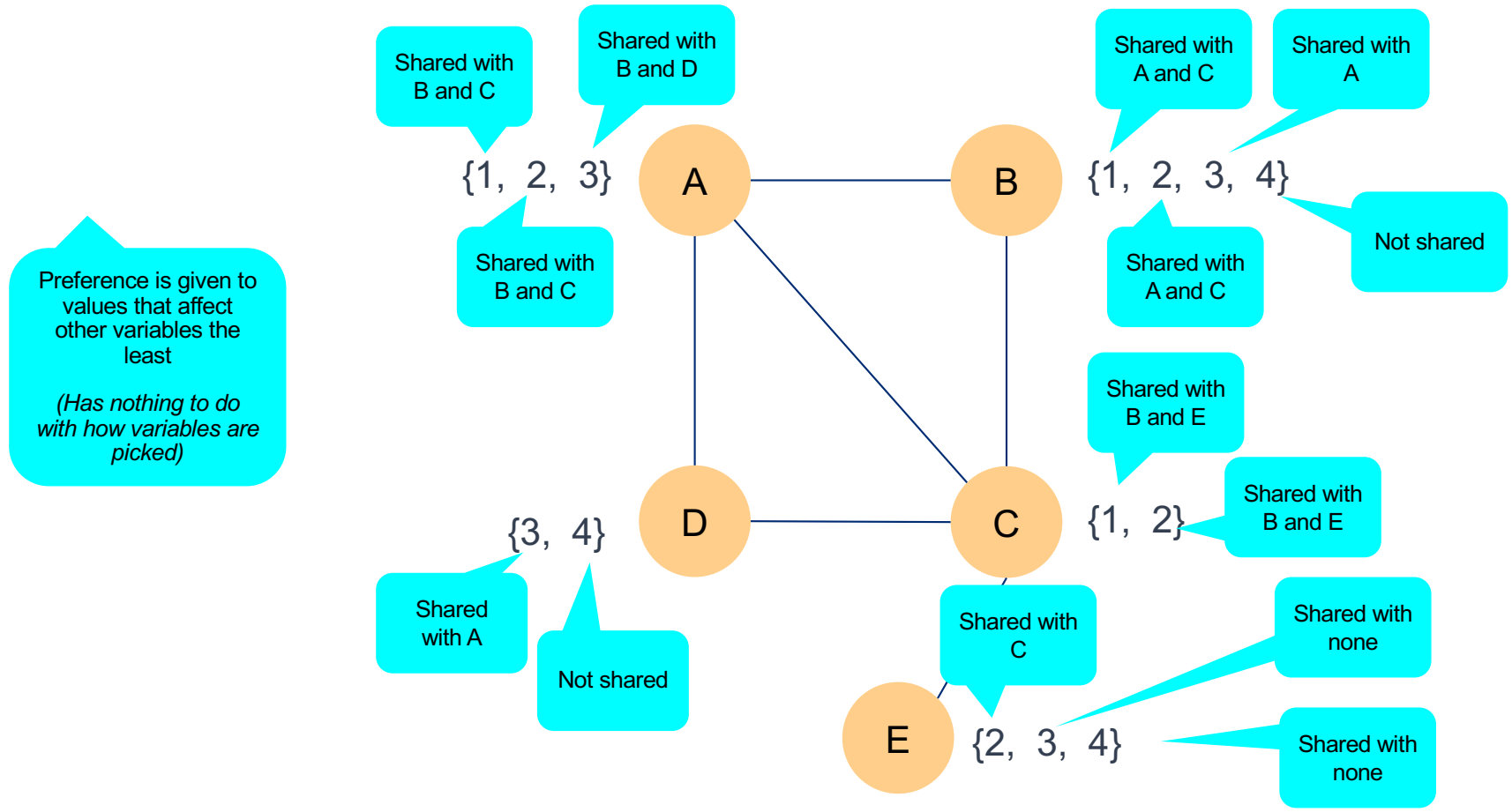
FC

5. Pick E, assign 2

FC



Feature #4: Least Constrained Values



Feature #4: Least Constrained Values

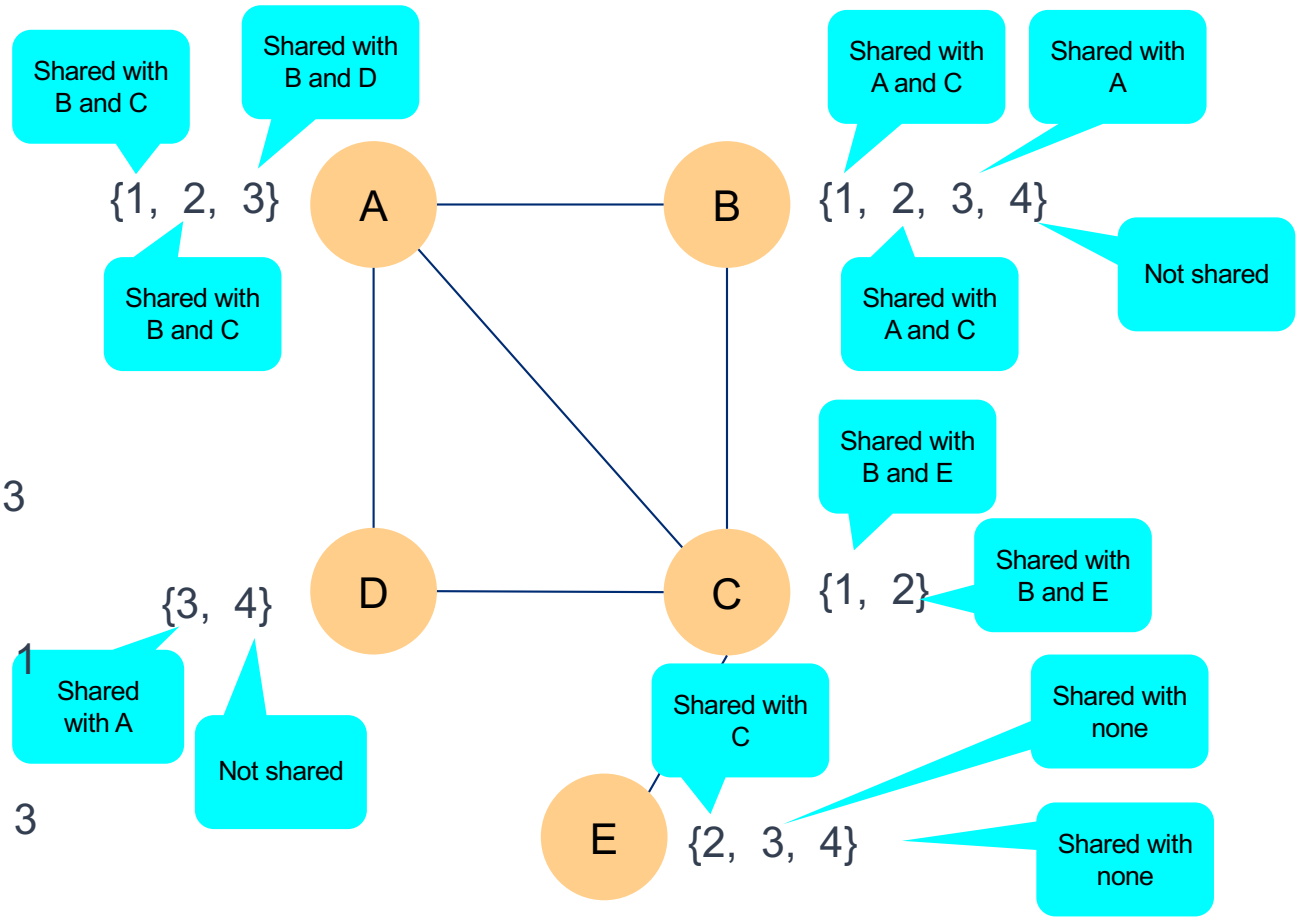
If you pick D, preference is 4 to 3

If you pick A, no preference between 1, 2, or 3

If you pick E, preference is either 3 or 4 then 2.

If you pick C, no preference between 1 or 2

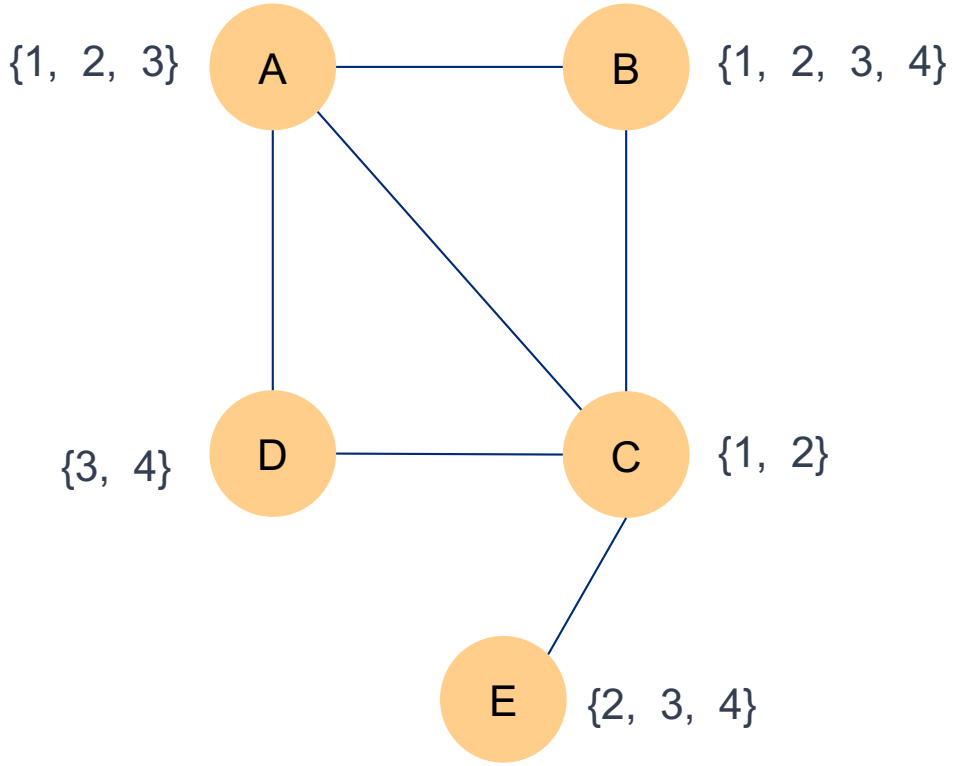
If you pick B, preference is 4 then 3 then either 1 or 2



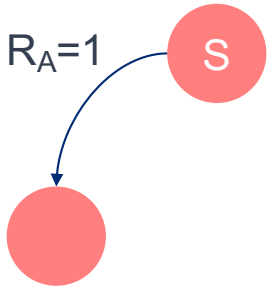
Conflict-Directed “Backjumping”

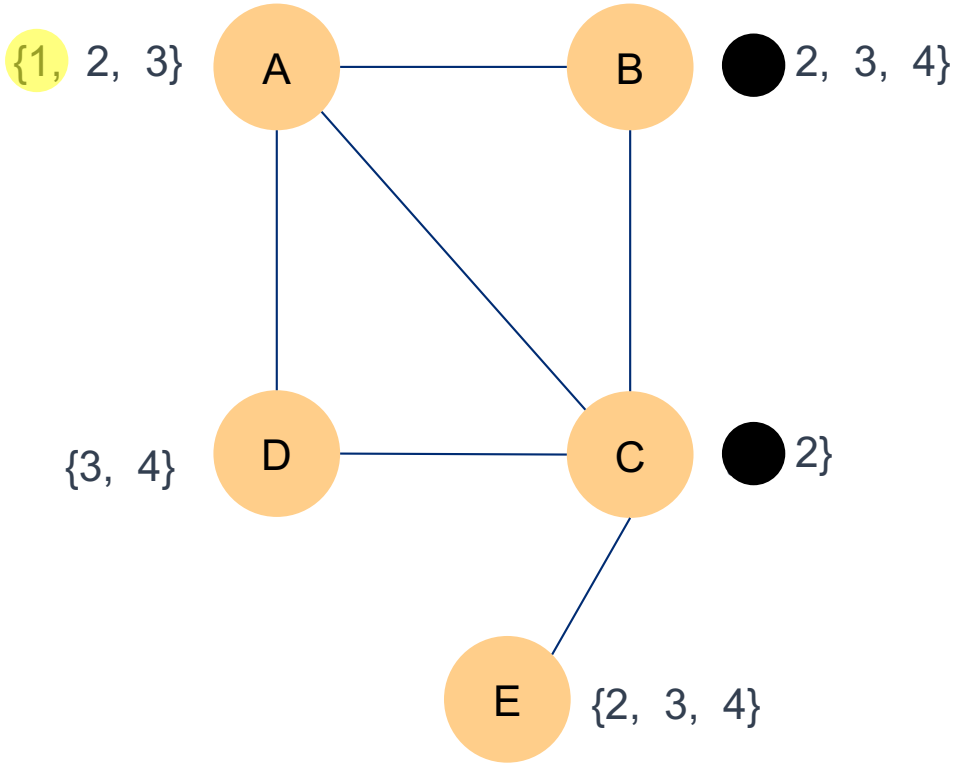
S

Let’s imagine the order of variables is A, B, E, D, and C

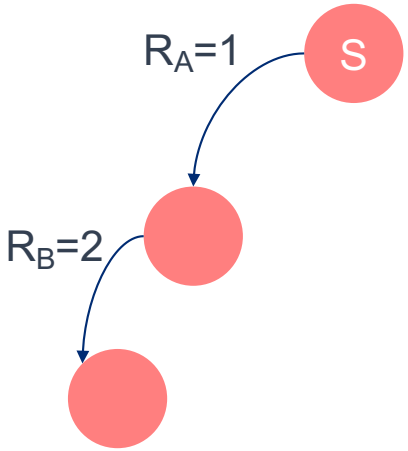


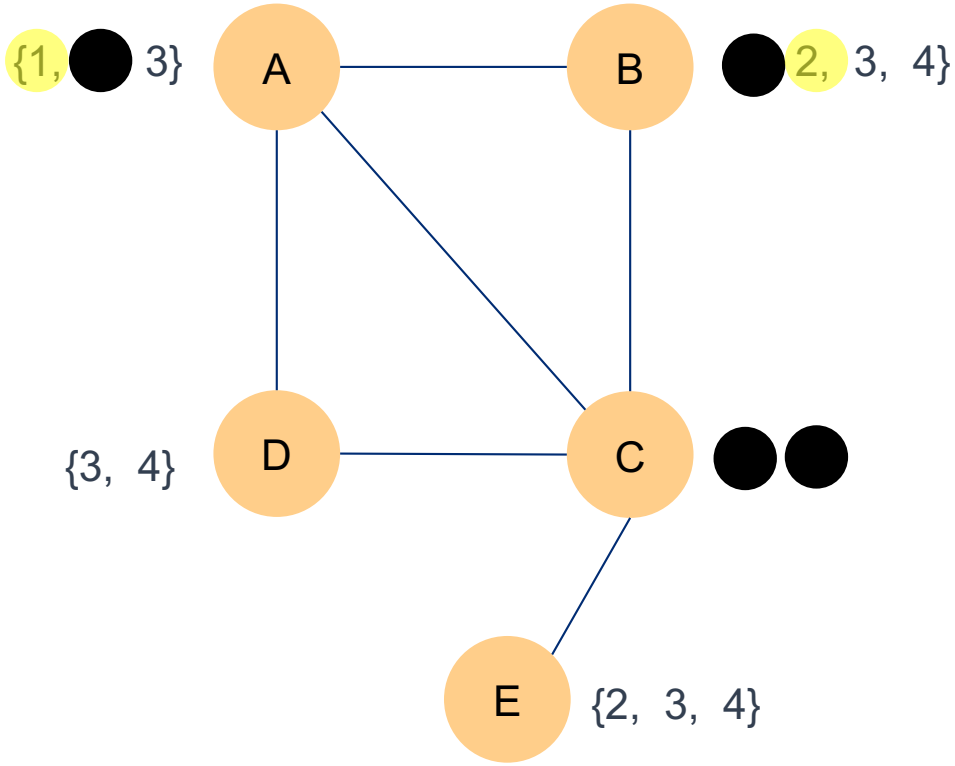
Conflict-Directed “Backjumping”



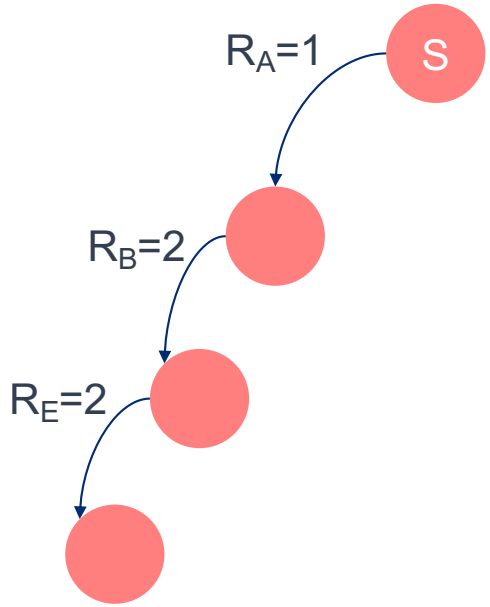


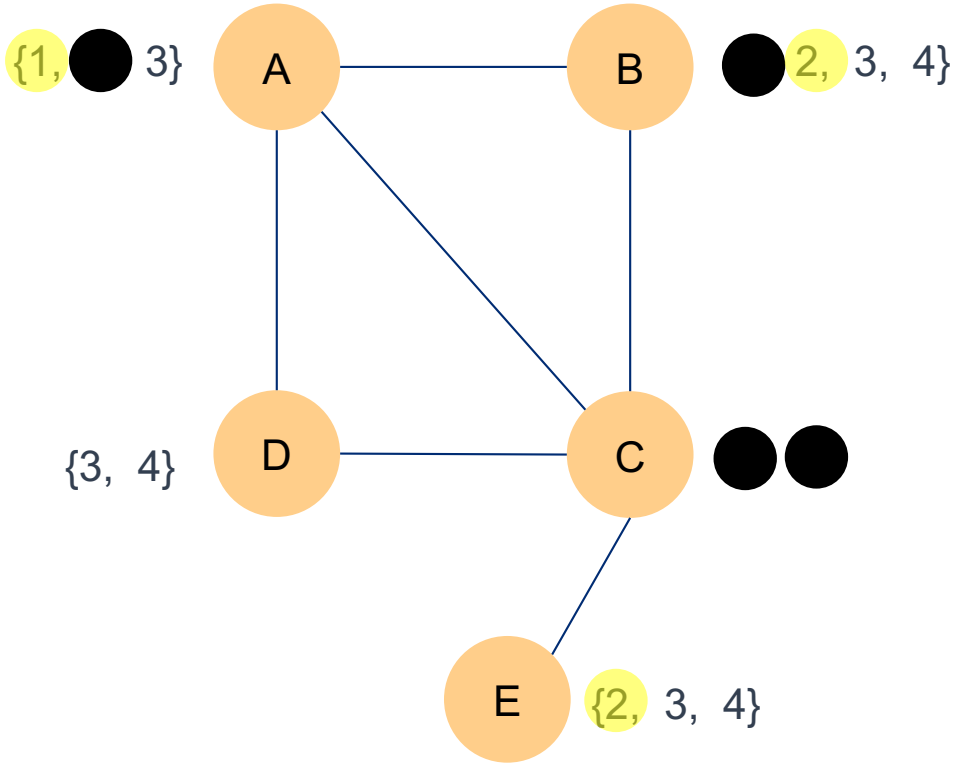
Conflict-Directed “Backjumping”



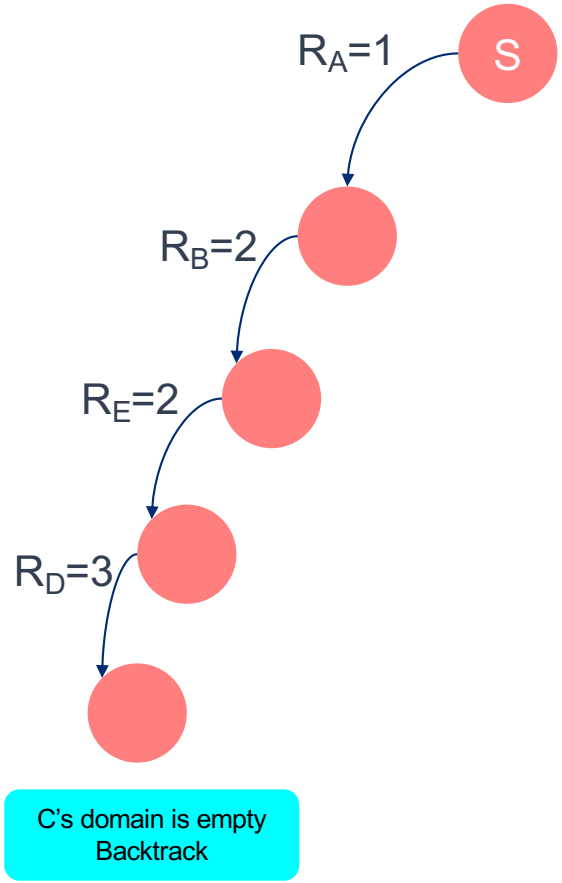


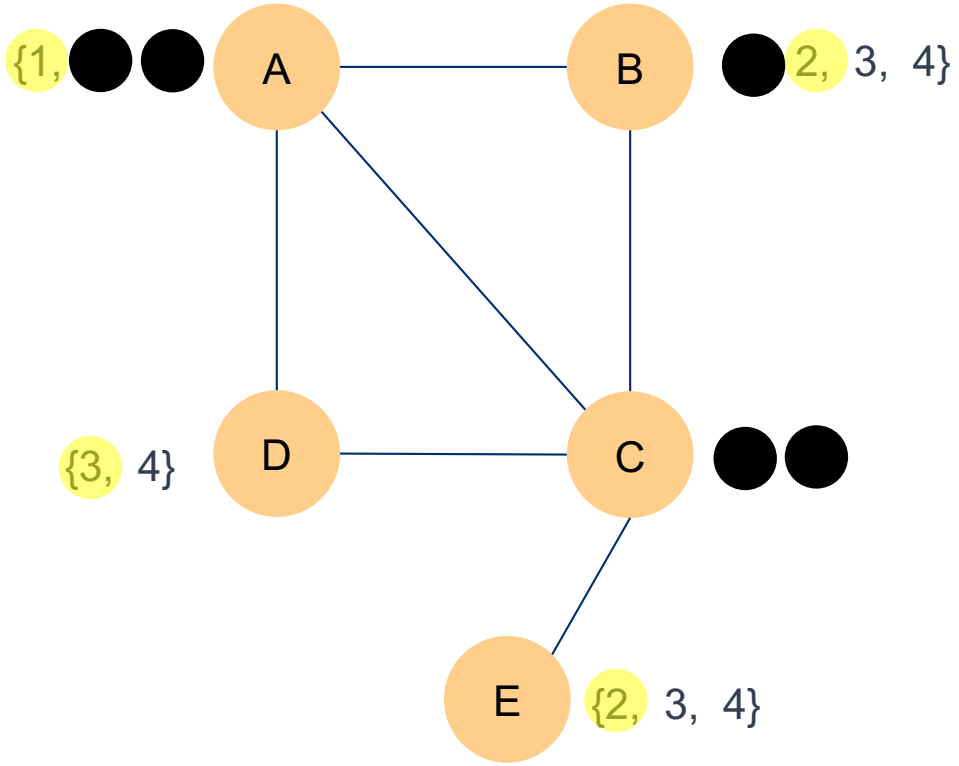
Conflict-Directed “Backjumping”



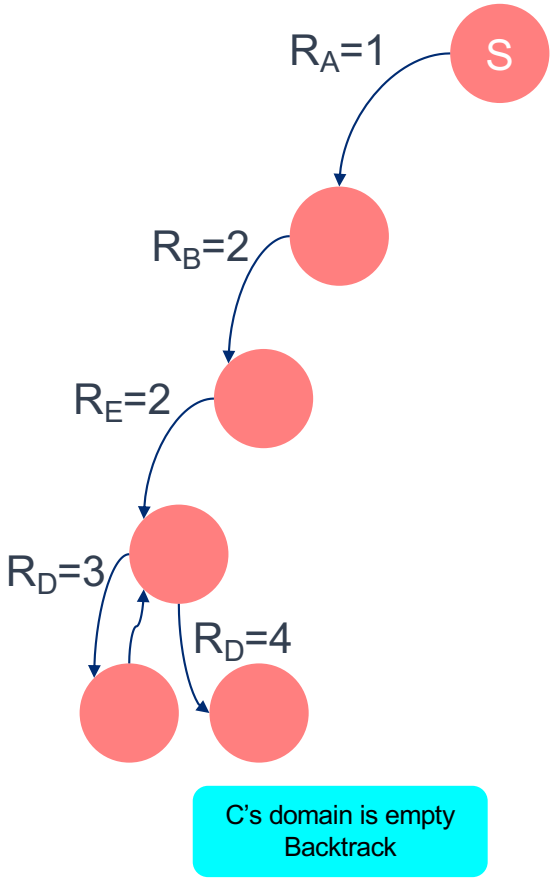


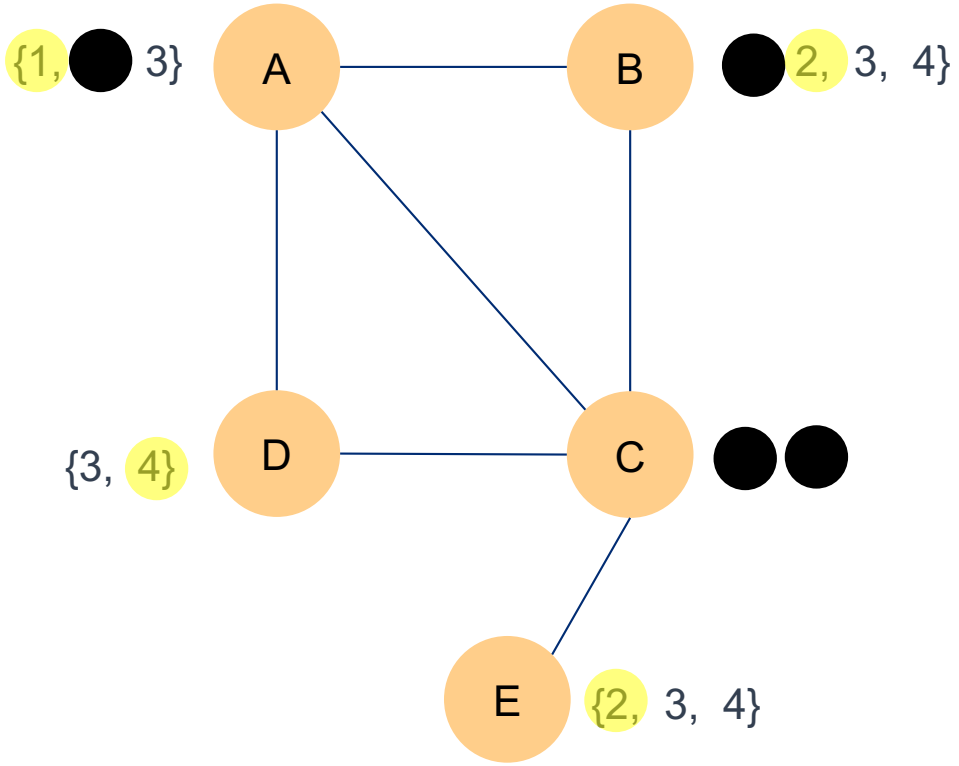
Conflict-Directed “Backjumping”



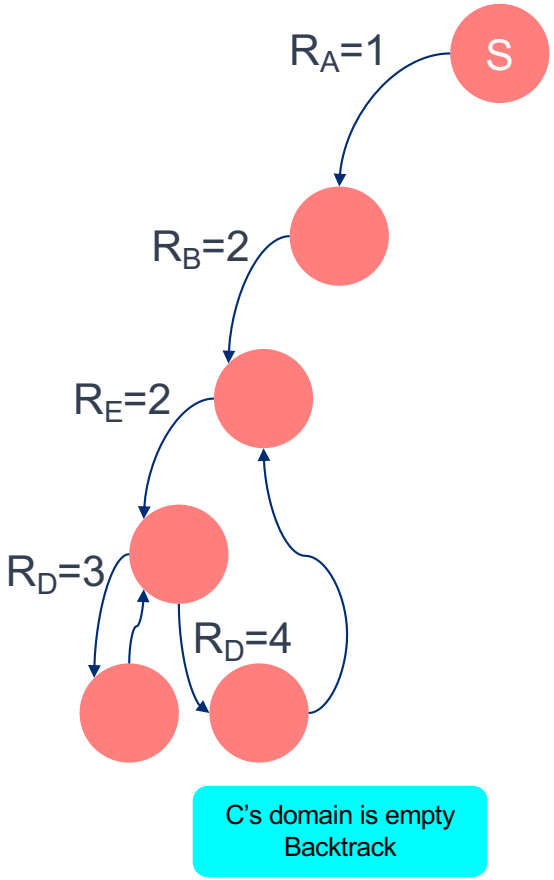


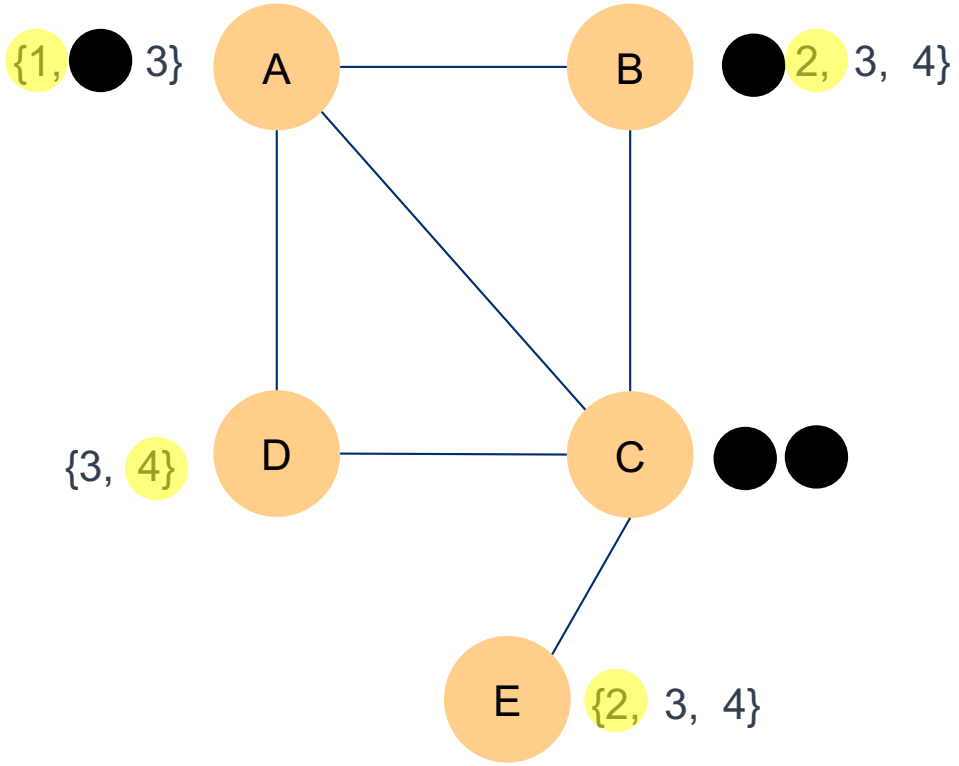
Conflict-Directed “Backjumping”



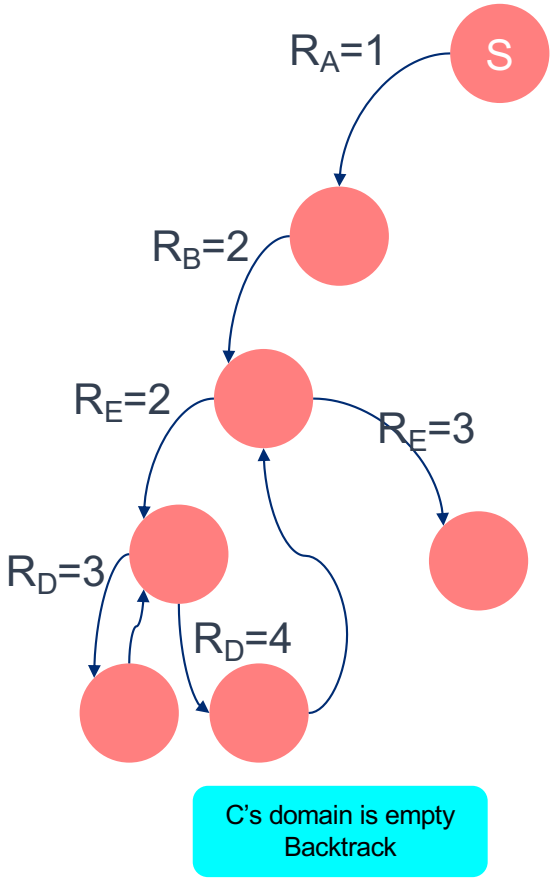


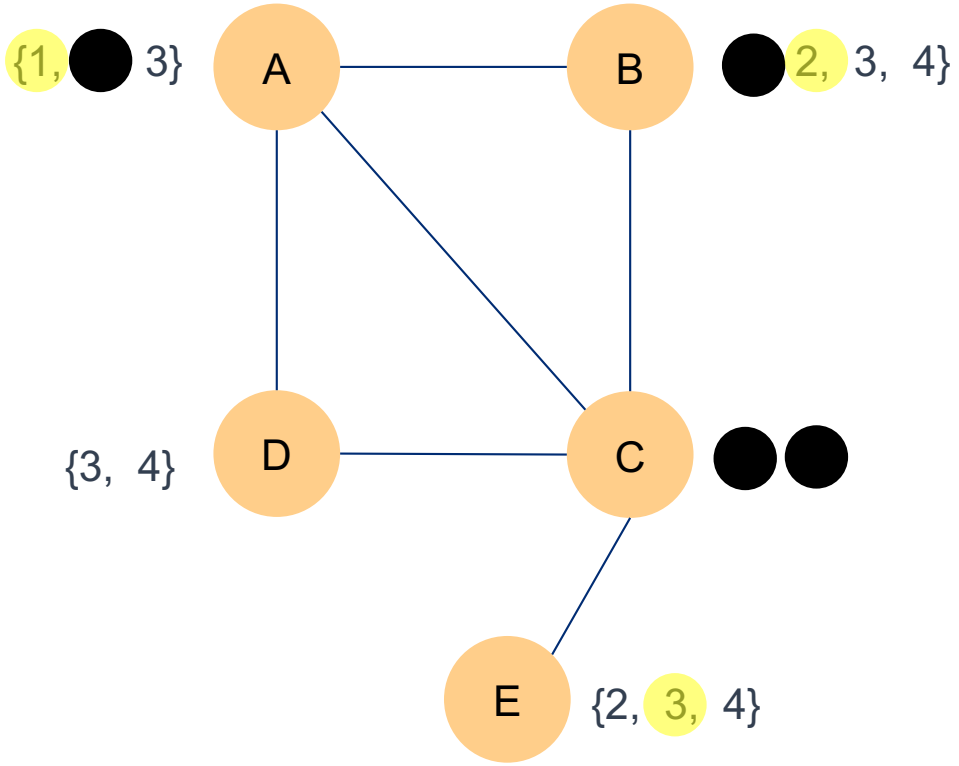
Conflict-Directed “Backjumping”



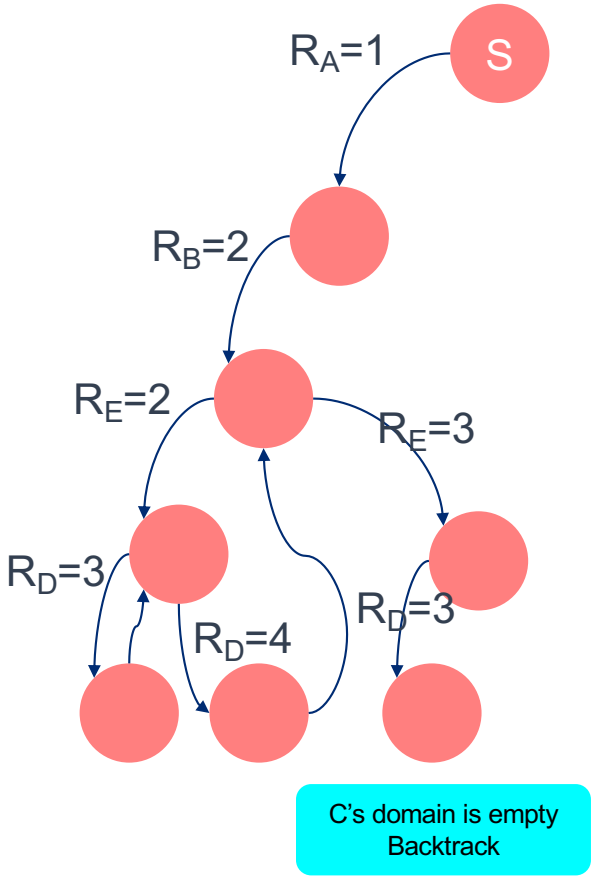


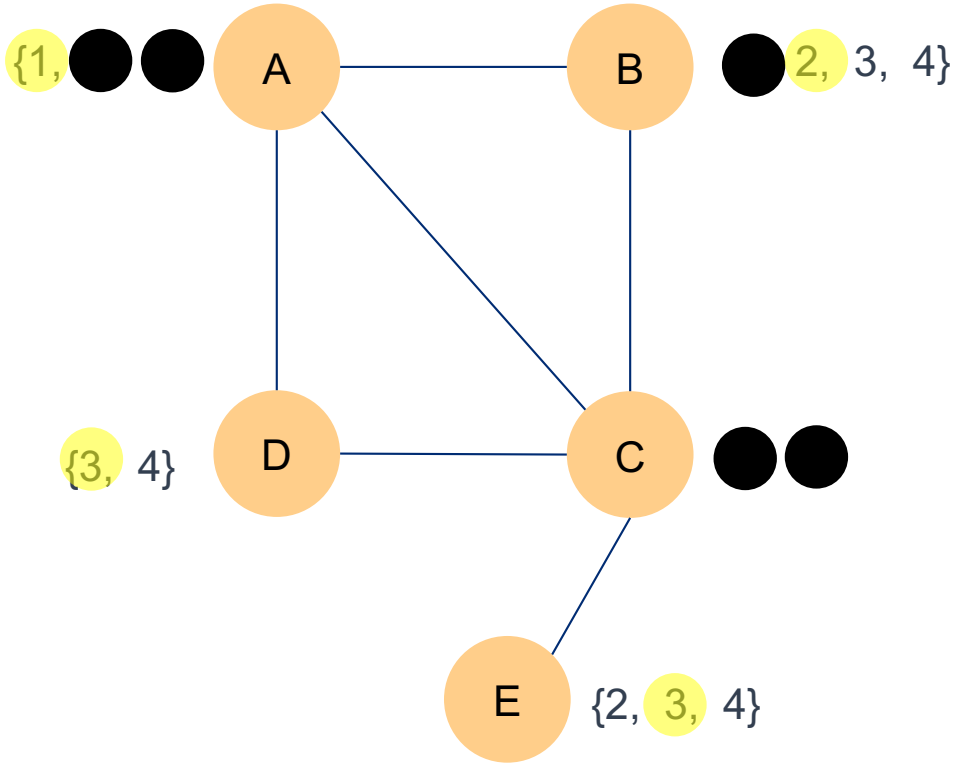
Conflict-Directed “Backjumping”



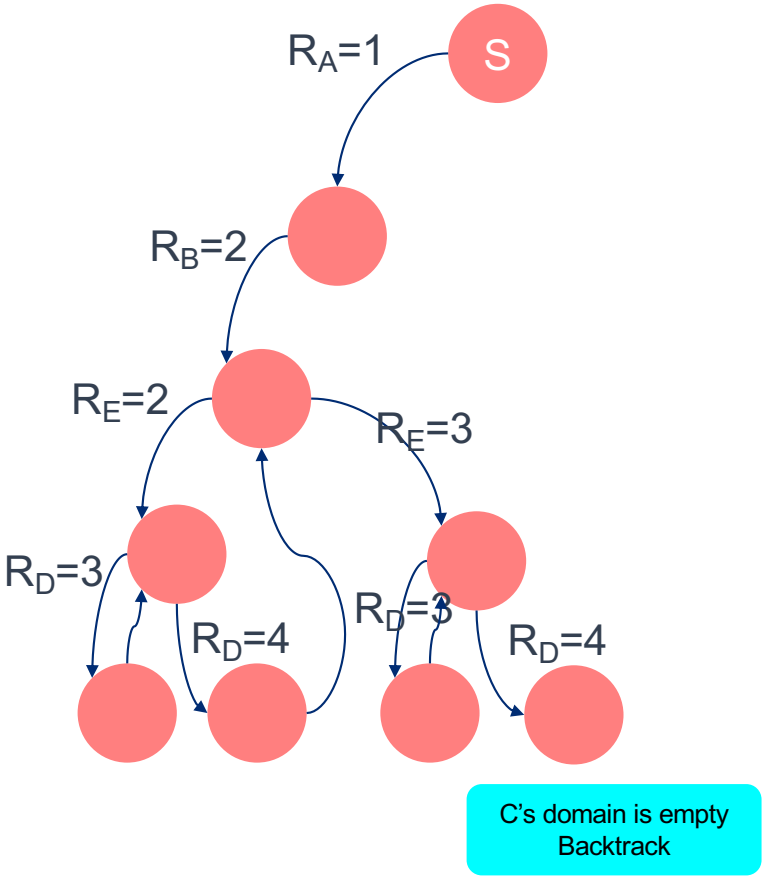


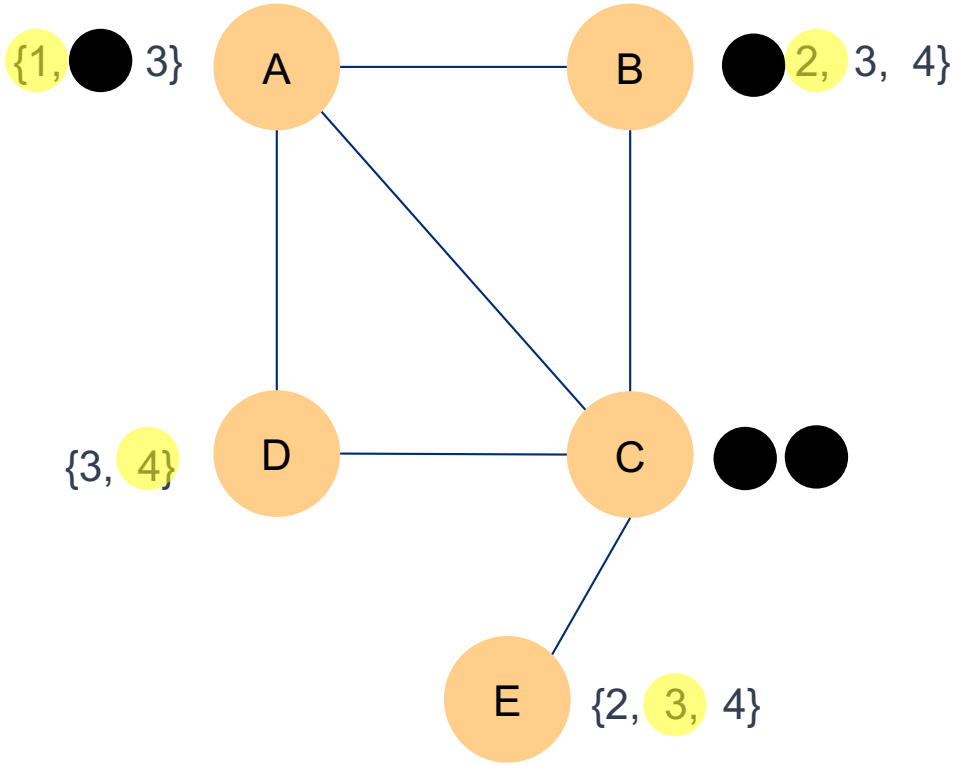
Conflict-Directed “Backjumping”



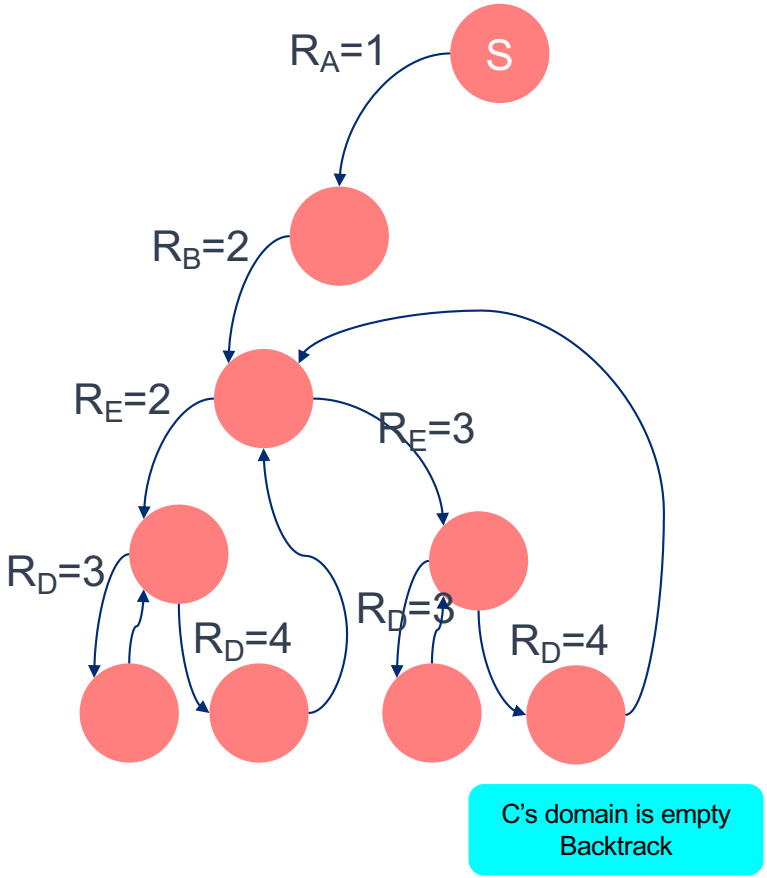


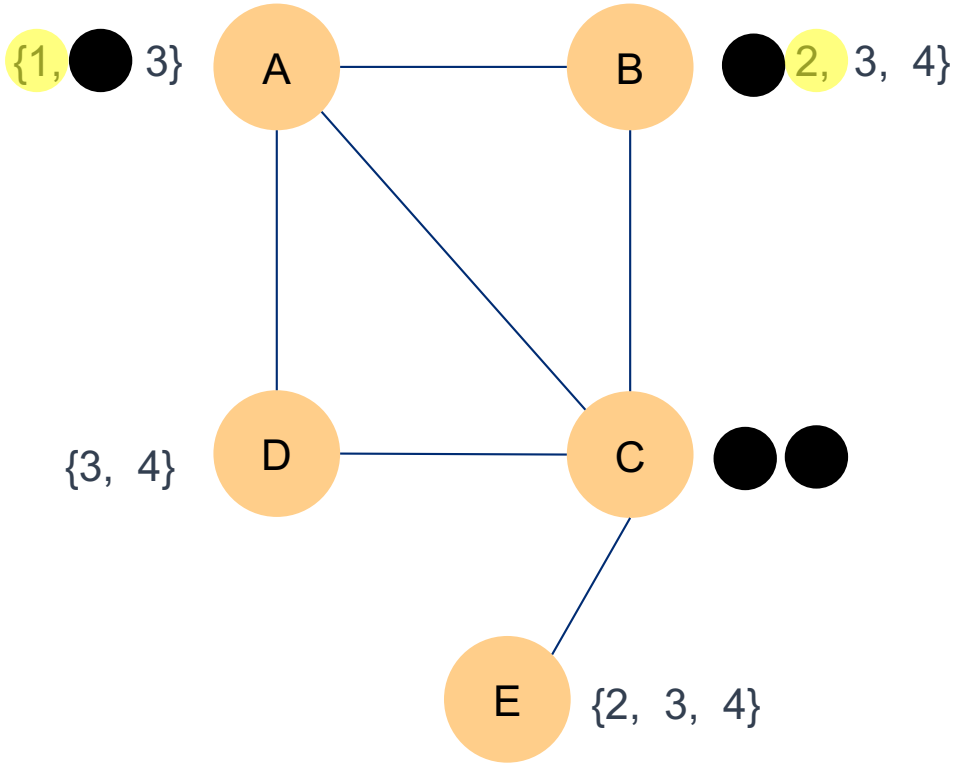
Conflict-Directed “Backjumping”



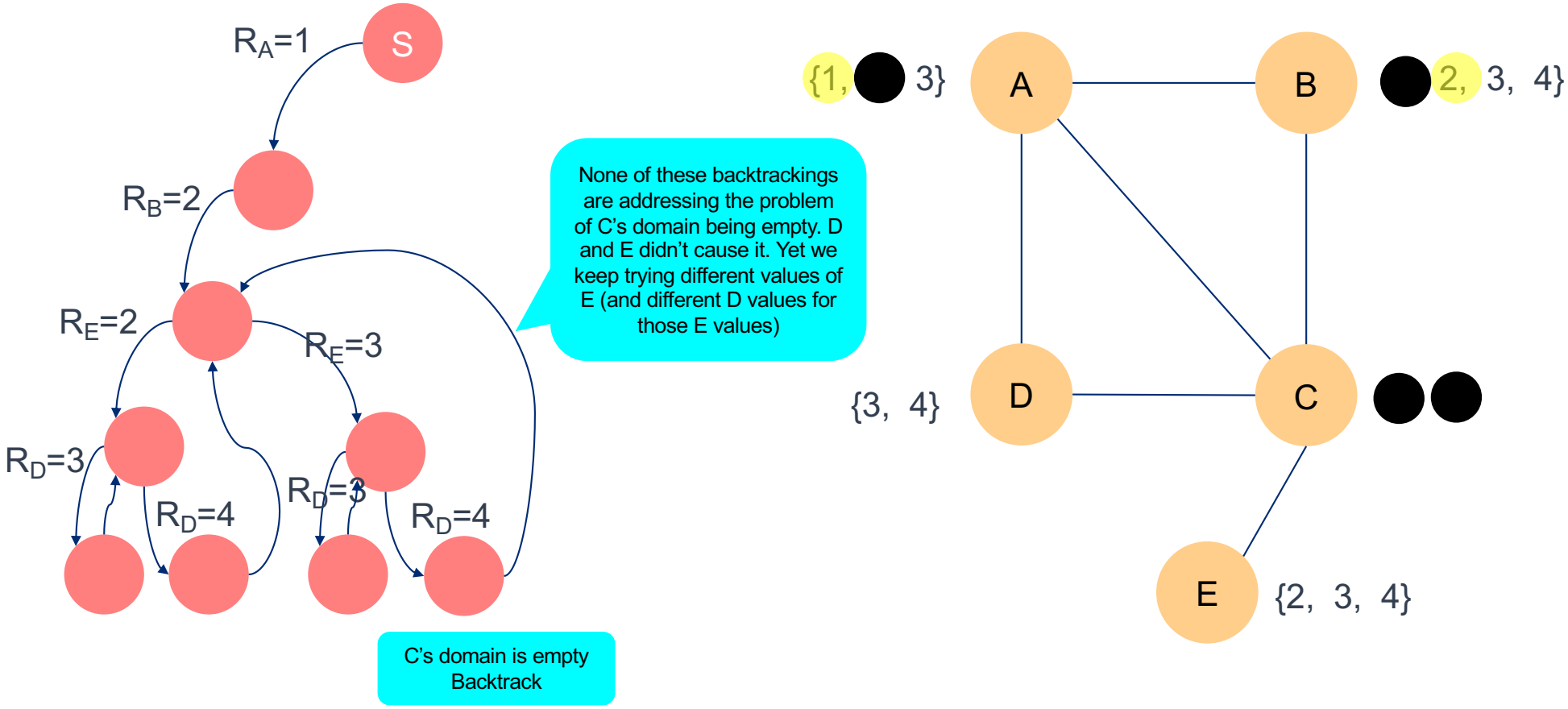


Conflict-Directed “Backjumping”

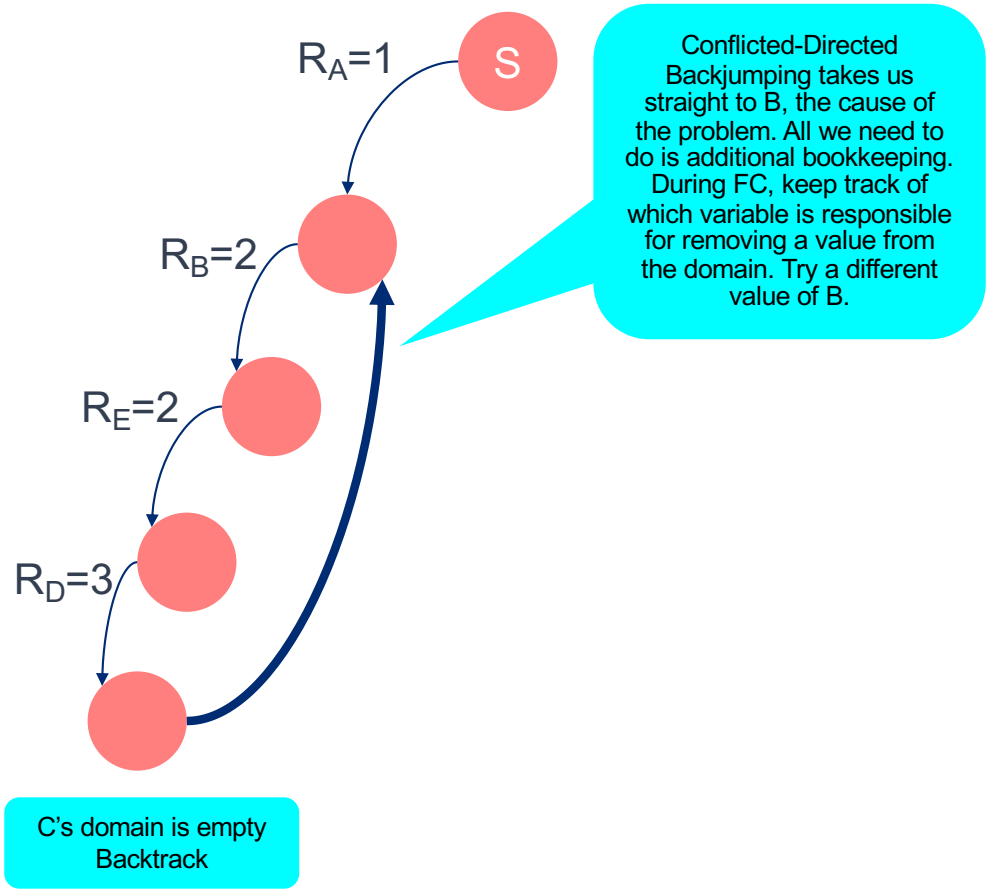


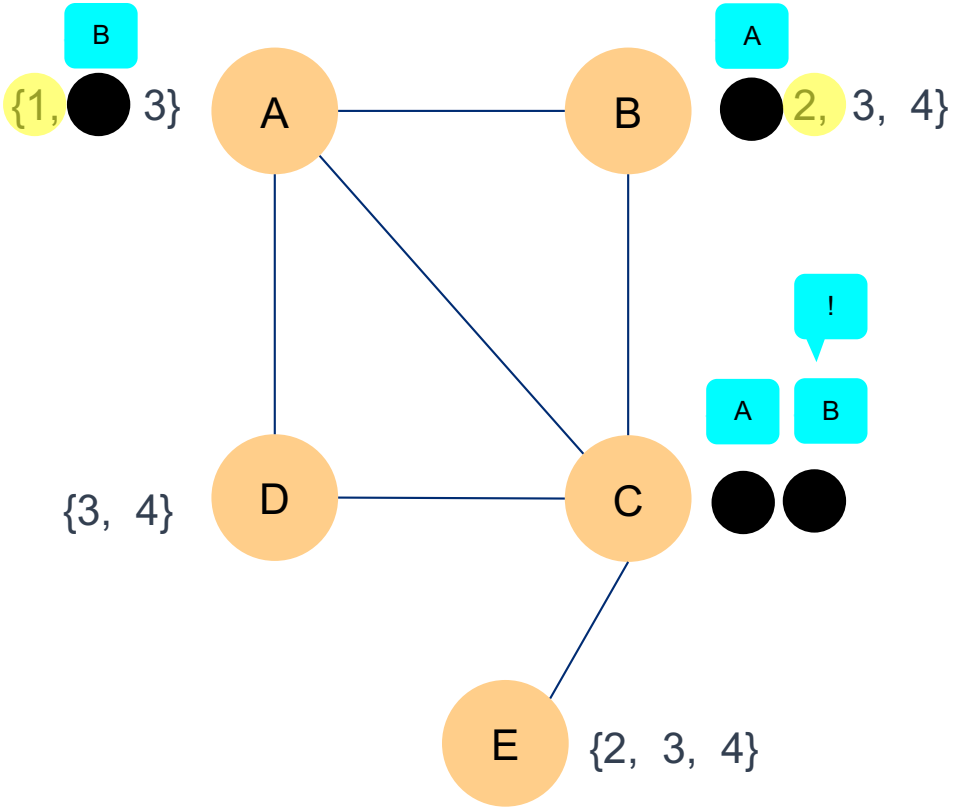


Conflict-Directed “Backjumping”



Conflict-Directed “Backjumping”





A Note on Applications

- Scheduling problems
- Puzzles
- Problems where there's contention (as long as two variables share a constraint)
- Global constraints are typically of the nature "all different"
- Resource constraint problems are of the form "at least" (consumption less than 5W, for instance)
- When to exit while backtracking?
  - Be mindful of: 1) finding *the* solution, 2) finding *a* solution, or 3) finding *all* solutions.

JOHNS HOPKINS

APPLIED PHYSICS LABORATORY