

CSC384
Constraint Satisfaction Problems
Part 2

Bahar Aameri & Sonya Allin

Winter 2020

Problems with Plain Backtracking

1	2	3						
						4	5	6
		7						
		8						
		9						

The backtracking search **won't** detect that the (3,3) cell has no possible value until all variables of the row/column/sub-square are assigned.

Constraint Propagation

- In CSPs, there might be variables that have **no possible value**, but BT **doesn't detect** this until it tries to **assign** them a value.

This leads to the idea of Constraint Propagation (or Domain Filtering).

Constraint Propagation: "looking ahead" at the yet unassigned variables in the search, trying to detect obvious failures.

"Obvious" means things we can **test/detect efficiently**.

- Even if it doesn't detect an obvious failure, it might be possible to **eliminate some parts** of the future search.

Constraint Propagation

- Propagation has to be applied **during the search**; potentially at **every node** of the search tree.
- Propagation itself is an inference step that needs some resources (in particular, **time**). If propagation is slow, this can slow the search down to the point where using propagation makes finding a solution take longer!
- Two main types of propagation: **Forward Checking** and **Generalized Arc Consistency**.

Constraint Propagation: Forward Checking

Forward Checking: An extension of [backtracking search](#).
Employs a [modest](#) amount of propagation (look ahead).

Intuition: When [instantiating](#) a variable V , do the following for [all constraints](#) C that have only [one uninstantiated](#) variable X remaining:

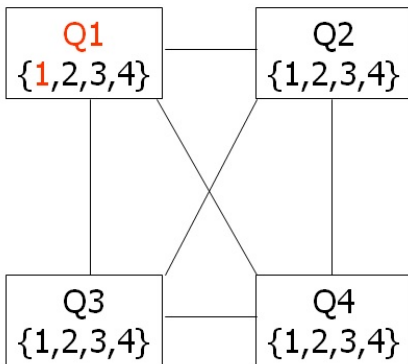
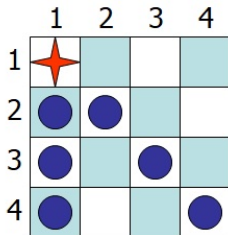
- [Check all](#) the values of X ;
- [Prune](#) those values that violate C .

















Undo the pruning when backtrack.

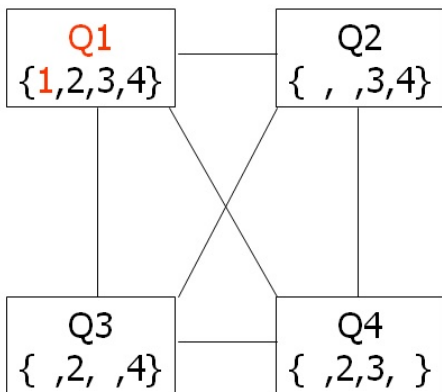
Forward Checking – Example

















Each of Q_1, \dots, Q_4 denotes a queen per row.

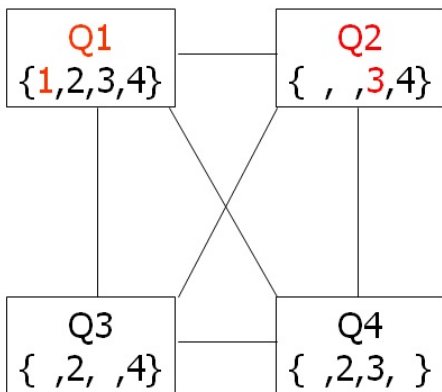
Forward checking prunes domains of Q_1, \dots, Q_4 based on binary constraints over Q_1, \dots, Q_4 .



















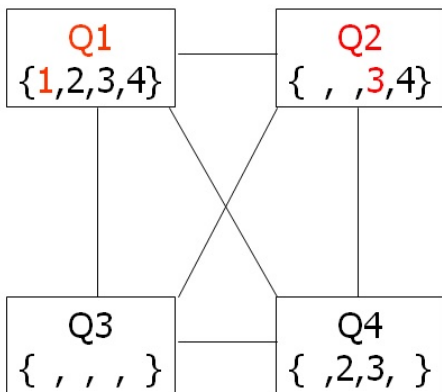
	1	2	3	4
1				
2				
3				
4				



















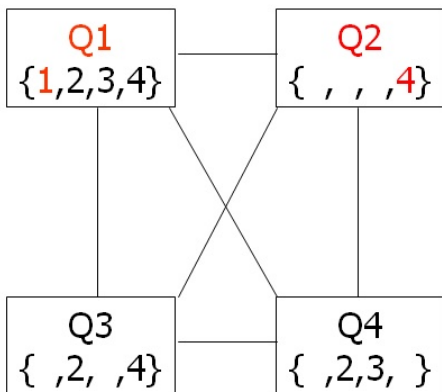
	1	2	3	4
1				
2				
3				
4				



















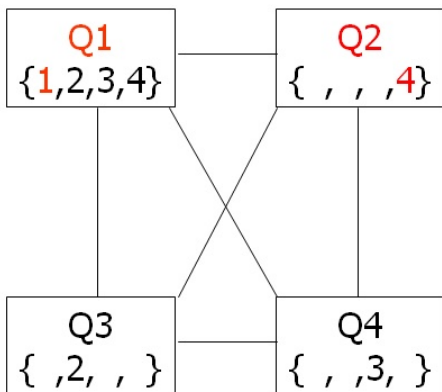
	1	2	3	4
1				
2				
3				
4				



















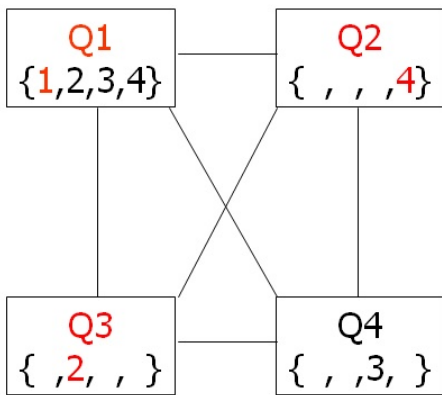
	1	2	3	4
1				
2				
3				
4				



















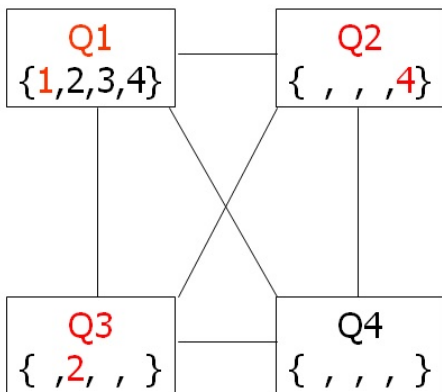
	1	2	3	4
1				
2				
3				
4				





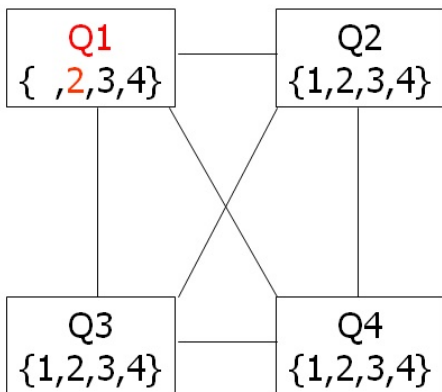
	1	2	3	4
1				
2				
3				
4				











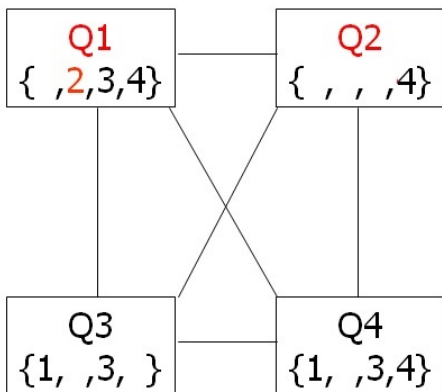
	1	2	3	4
1				
2				
3				
4				












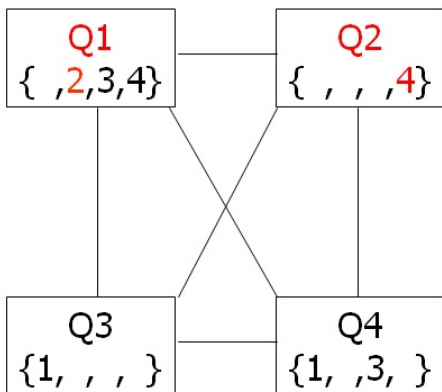
	1	2	3	4
1				
2				
3				
4				














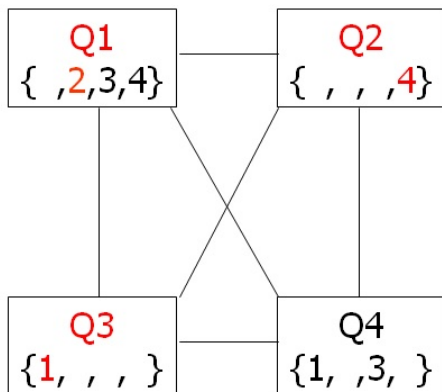
	1	2	3	4
1				
2				
3				
4				















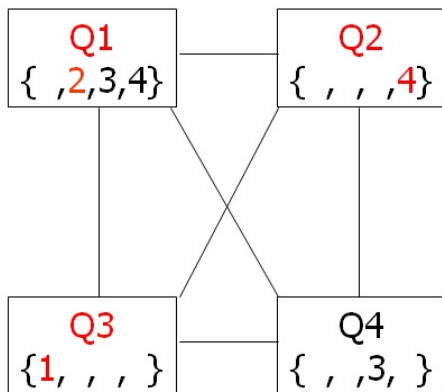
	1	2	3	4
1				
2				
3				
4				

















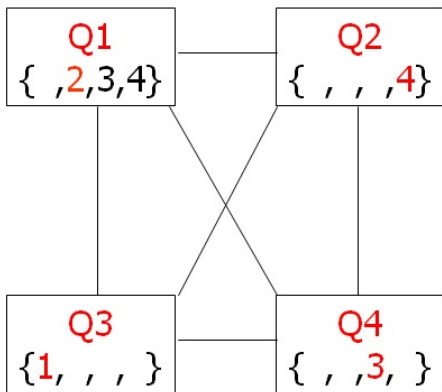
	1	2	3	4
1				
2				
3				
4				

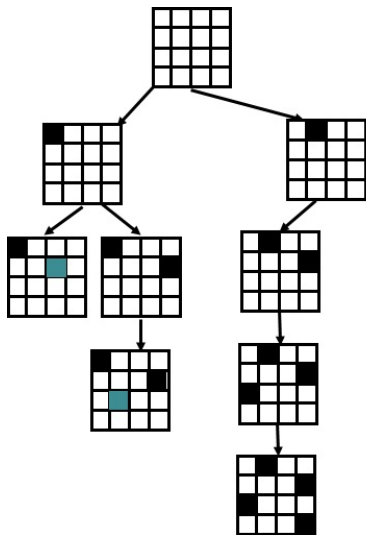


	1	2	3	4
1				
2				
3				
4				



	1	2	3	4
1				
2				
3				
4				





Backtracking Search: The Algorithm

```
def FCCheck(C,X):  
    // C is a constraint with all its variables already  
    // assigned, except for variable X.  
    1. for d := each member of CurDom(X):  
    2.     if making X = d together with previous assignments  
           to variables in the scope of C falsifies C:  
    3.         remove d from CurDom(X)  
    4. if CurDom[X] == {}:  
    5.     RETURN DWO    # Domain Wipe Out  
    6. RETURN ok
```

Backtracking Search: The Algorithm

```
def FC(Level):
1.  if all Variables assigned
2.      PRINT Value of each Variable
3.      EXIT or RETURN                                # EXIT for only one solution
                                                    # RETURN for more solutions

4.  V := PickUnassignedVariable()
5.  Assigned[V] := TRUE
6.  for d := each member of CurDom(V)
7.      Value[V] := d
8.      DWOccured:= False
9.      for each constraint C over V such that C has only one
        unassigned variable X in its scope:
10.         if FCCheck(C,X) == DW0:  # X domain becomes empty
11.             DWOccured:= True
12.             BREAK                # stop checking constraints
13.         if NOT DWOccured:        # all constraints were ok
14.             FC(Level+1)
15.             RestoreAllValuesPrunedByFCCheck()
16.  Assigned[V] := FALSE          # UNDO as we have tried all of V's values
17.  RETURN
```

- After we backtrack from the current assignment the values that were pruned (as a result of that assignment) must be restored.
- Some bookkeeping needs to be done to remember which values were pruned by which assignment.

Forward Checking Heuristics: Human Analogy

What variables would you try first?

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

Forward Checking: Variable and Value Ordering Heuristics

- Heuristics can be used to determine
 - the **order** in which **variables** are assigned:
`PickUnassignedVariable()`
 - the **order** of **values** tried for each variable.
- The choice of the next variable can vary from branch to branch.
Example: Under the assignment $V_1 = a$ we might choose to assign V_4 next, while under $V_1 = b$ we might choose to assign V_5 next.
- This **dynamically** chosen variable ordering has a tremendous impact on performance.

Degree Heuristic: Select the variable that is part of the **most remaining unsatisfied constraints**.

Minimum Remaining Values Heuristic (MRV):

- Always branch on a variable with the **smallest remaining values** (smallest CurDom).
Intuition: If a variable has only one value left, that value is forced, so we should propagate its consequences immediately.
- This heuristic tends to produce skinny trees at the top.
More variables can be instantiated with fewer nodes searched.
- More constraint propagation/DWO failures occur when the tree starts to branch out.
Hence, inconsistencies can be found much faster.

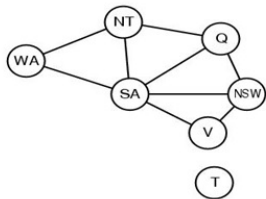
Example: Map Colouring

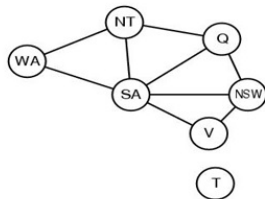
Problem Statement: Color the following map using **red**, **green**, and **blue** such that **adjacent** regions have different colors.



Problem formulation:

- **Variables:**
- **Domains:**
- **Constraints:**





WA

NT

Q

NSW

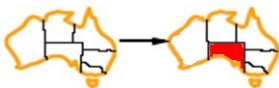
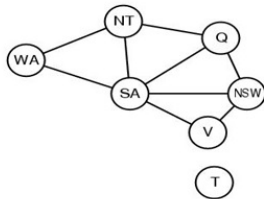
V

SA

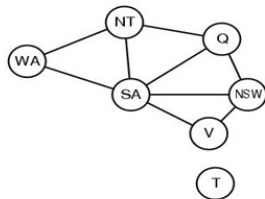
T



. $\{SA = red\}$ (using Degree Heuristic)

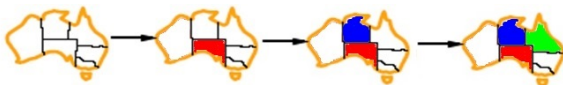
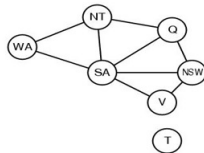


. $\{SA = \text{red}, NT = \text{blue}\}$ (using Degree Heuristic)



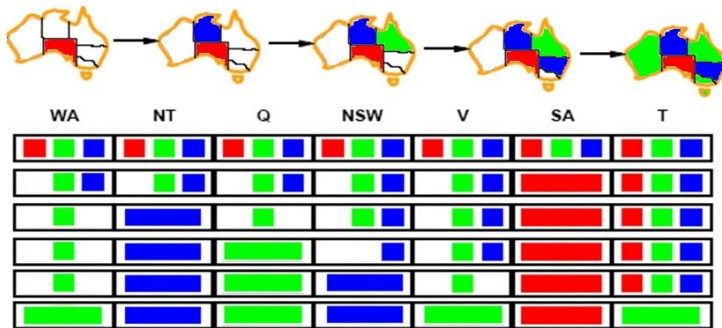
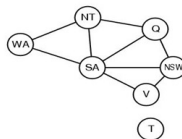
WA	NT	Q	NSW	V	SA	T
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>
<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>

- $\{SA = \text{red}, NT = \text{blue}, Q = \text{green}\}$ (using Degree Heuristic and MRV)



WA	NT	Q	NSW	V	SA	T
<div><div>red</div><div>green</div><div>blue</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>
<div><div></div><div>green</div><div>blue</div></div>	<div><div></div><div>green</div><div>blue</div></div>	<div><div></div><div>green</div><div>blue</div></div>	<div><div></div><div>green</div><div>blue</div></div>	<div><div></div><div>green</div><div>blue</div></div>	<div><div>red</div><div>red</div><div>red</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>
<div><div></div><div>green</div><div></div></div>	<div><div>blue</div><div>blue</div><div>blue</div></div>	<div><div></div><div>green</div><div></div></div>	<div><div></div><div>green</div><div>blue</div></div>	<div><div></div><div>green</div><div>blue</div></div>	<div><div>red</div><div>red</div><div>red</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>
<div><div></div><div>green</div><div></div></div>	<div><div>blue</div><div>blue</div><div>blue</div></div>	<div><div>green</div><div>green</div><div>green</div></div>	<div><div></div><div></div><div>blue</div></div>	<div><div></div><div>green</div><div>blue</div></div>	<div><div>red</div><div>red</div><div>red</div></div>	<div><div>red</div><div>green</div><div>blue</div></div>

. $\{SA = \text{red}, NT = \text{blue}, Q = \text{green}, NSW = \text{blue}, V = \text{green}, WA = \text{green}, T = \text{green}\}$



Example: Map Colouring

Try the map coloring example without MRV and Degree heuristics.

Forward Checking: Empirically

- FC often is about 100 times faster than BT.
- FC with MRV (minimal remaining values) often 10000 times faster.
- On some problems the speed up can be much greater.
Converts problems that are not solvable to problems that are solvable.
- Still FC is not that powerful.
Other more powerful forms of constraint propagation are used in practice.