SWINBURNE UNIVERSITY OF TECHNOLOGY

COS30018: Intelligent Systems – W2

Problem-Solving Agents:

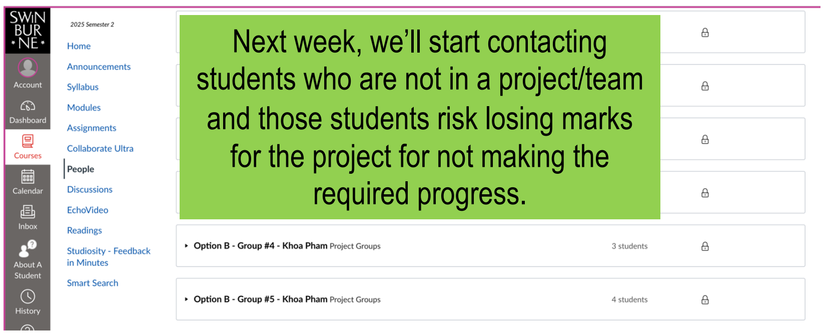
Search &

Constraint Satisfaction Problem

1

Assignment check-in

- Have you found a project/team?
  - ☐ Yes: Great, please discuss with your team on a plan to tackle the project (and use a tool such as Jira for project management and Github for version control)
  - ☐ No: Make sure that you attend this week tutorial to finalise with your tutor your project/team. By Friday, you should be in one of the groups on Canvas:



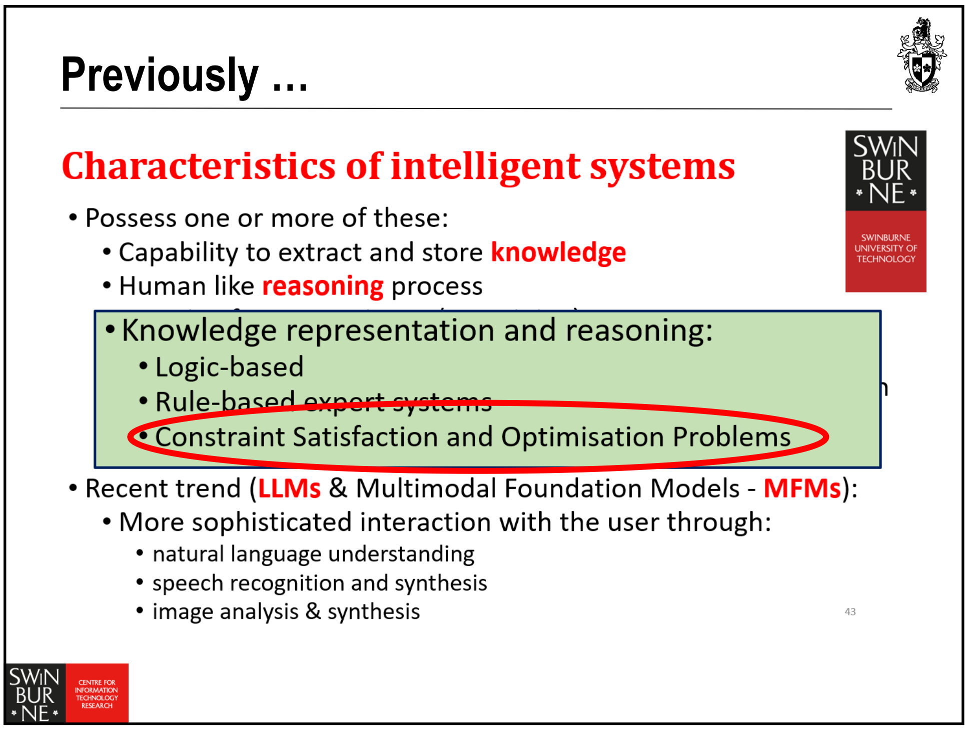
Option B - Group #4 - Khoa Pham Project Groups 3 students

Option B - Group #5 - Khoa Pham Project Groups 4 students

Next week, we'll start contacting students who are not in a project/team and those students risk losing marks for the project for not making the required progress.

CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

2



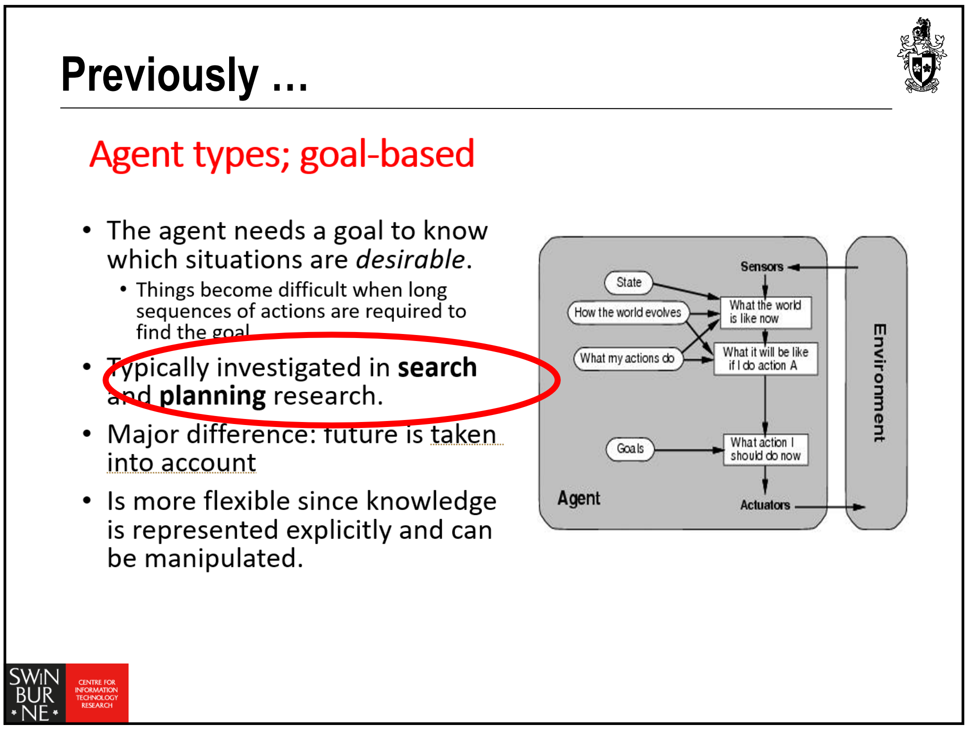
Previously …

Characteristics of intelligent systems

- Possess one or more of these:
  - Capability to extract and store **knowledge**
  - Human like **reasoning** process
  - Knowledge representation and reasoning:
    - Logic-based
    - Rule-based expert systems
    - Constraint Satisfaction and Optimisation Problems
- Recent trend (**LLMs** & Multimodal Foundation Models - **MFMs**):
  - More sophisticated interaction with the user through:
    - natural language understanding
    - speech recognition and synthesis
    - image analysis & synthesis

43

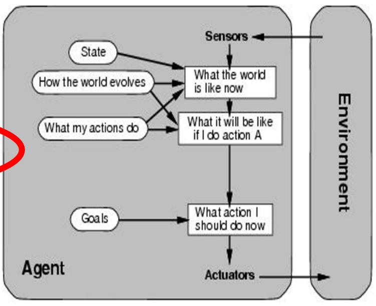
3



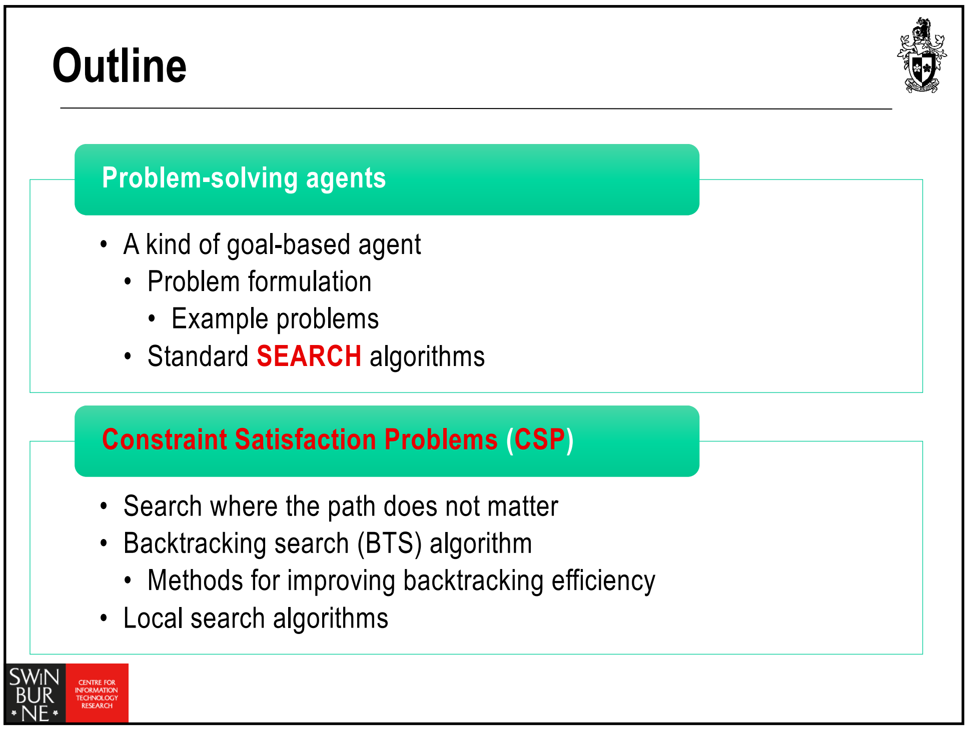
Previously …

Agent types; goal-based

- The agent needs a goal to know which situations are *desirable*.
  - Things become difficult when long sequences of actions are required to find the goal.
- Typically investigated in **search** and **planning** research.
- Major difference: future is taken into account
- Is more flexible since knowledge is represented explicitly and can be manipulated.


Sensors
State
How the world evolves
What my actions do
What the world is like now
What it will be like if I do action A
Goals
What action I should do now
Agent
Actuators
Environment

4



Outline

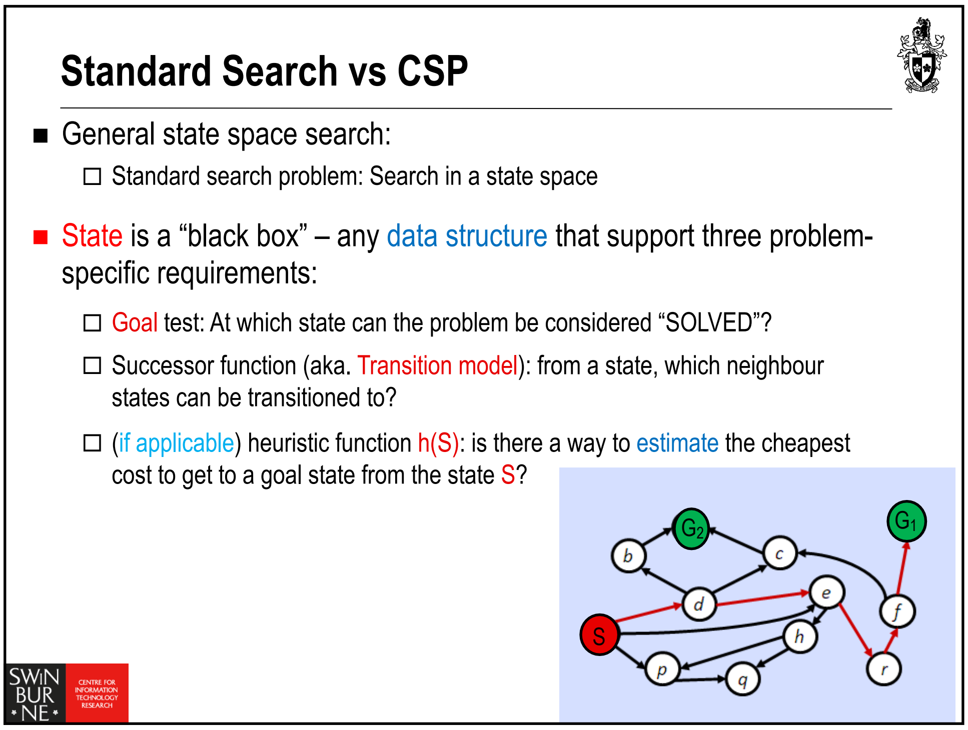
Problem-solving agents

- A kind of goal-based agent
  - Problem formulation
    - Example problems
  - Standard **SEARCH** algorithms

Constraint Satisfaction Problems (CSP)

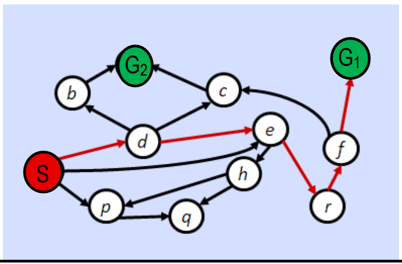
- Search where the path does not matter
- Backtracking search (BTS) algorithm
  - Methods for improving backtracking efficiency
- Local search algorithms

5



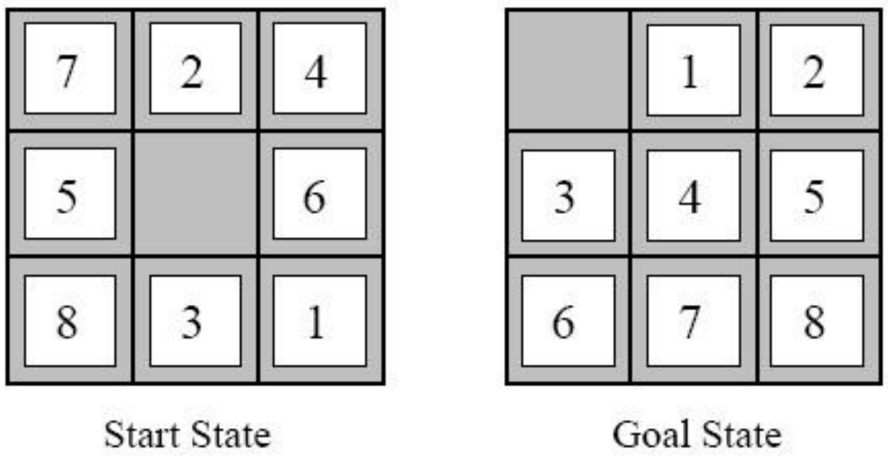
Standard Search vs CSP

- General state space search:
  - Standard search problem: Search in a state space
- State is a "black box" – any data structure that support three problem-specific requirements:
  - Goal test: At which state can the problem be considered "SOLVED"?
  - Successor function (aka. Transition model): from a state, which neighbour states can be transitioned to?
  - (if applicable) heuristic function h(S): is there a way to estimate the cheapest cost to get to a goal state from the state S?

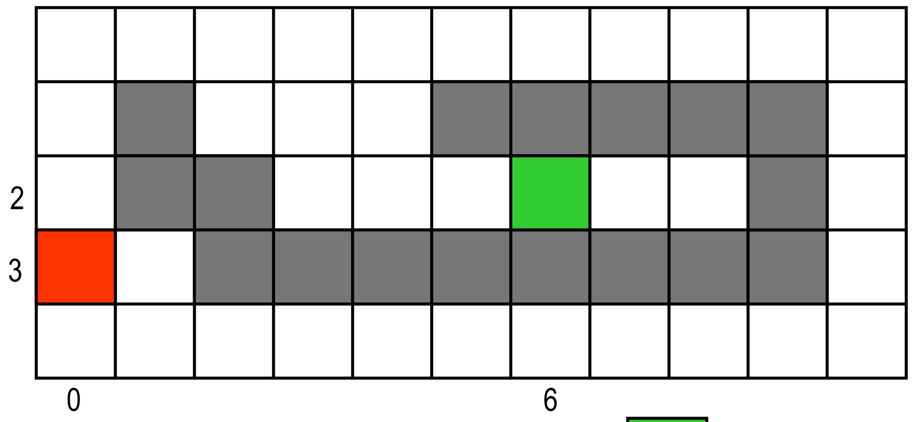


6

Example: 8-puzzle



Example: Robot Navigation



🟥 Start state [0,3]

🟩 Goal state [6,2]

Single-state problem formulation

A problem is defined by four items:

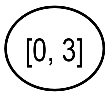
1. initial state e.g., [0, 3]
2. Transition model or successor function *S(x)* = set of action–state pairs
   - ☐ e.g., *S([0,3])* = {<*UP*, [0, 2]>, <*DOWN*, [0, 4]>, <*RIGHT*, [1, 3]>}
3. goal test, can be
   - ☐ explicit, e.g., *x* = [6, 2]
   - ☐ implicit, e.g., *Checkmate(x)*
4. path cost (additive)
   - ☐ e.g., sum of distances, number of actions executed, etc.
   - ☐ *c(x,a,y)* is the step cost, assumed to be ≥ 0

■ A solution is a sequence of actions leading from the initial state to a goal state

SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

9

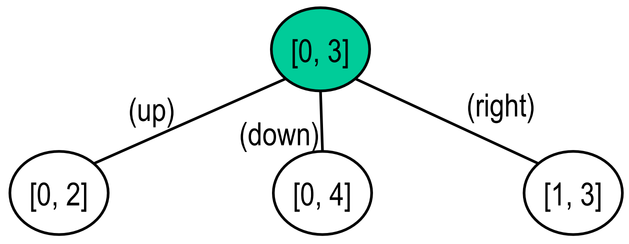
Simple tree search example



SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

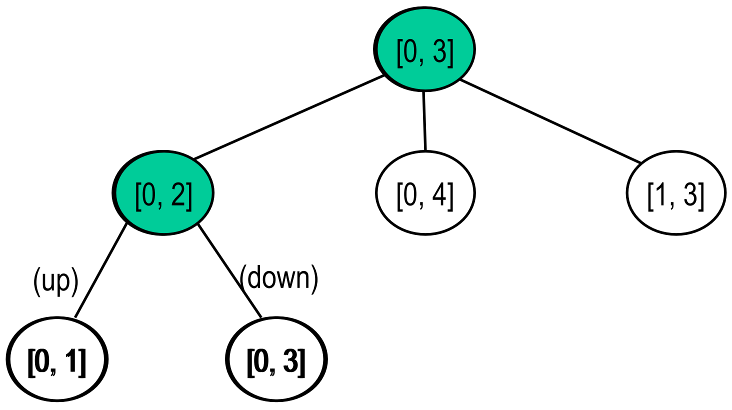
10

Simple tree search example



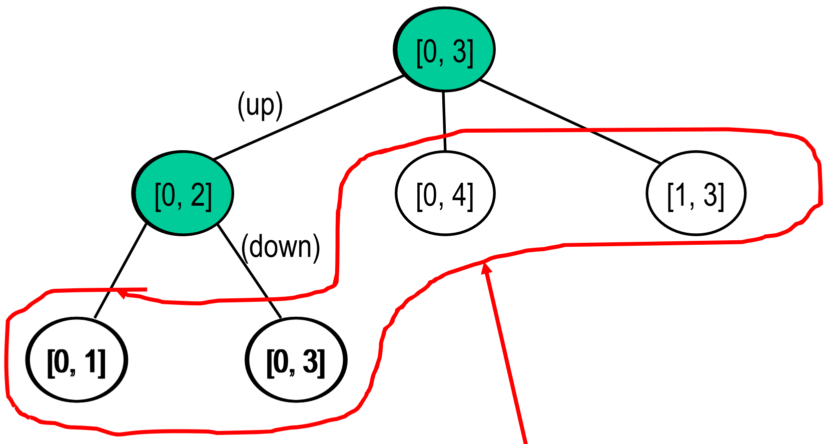
11

Simple tree search example



12

What is the *FRONTIER*?



FRONTIER= contains generated nodes which are not yet expanded.

13

Search where the path doesn't matter

- So far, we looked at problems where the path was the solution
  - ☐ Traveling on a graph
  - ☐ Eights puzzle
- However, in many problems, we just want to find a goal state
  - ☐ Doesn't matter how we get there

14

Queens puzzle

- Place eight queens on a chessboard so that no two attack each other

				Q			
		Q					
Q							
						Q	
	Q						
							Q
					Q		
			Q				

SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

15

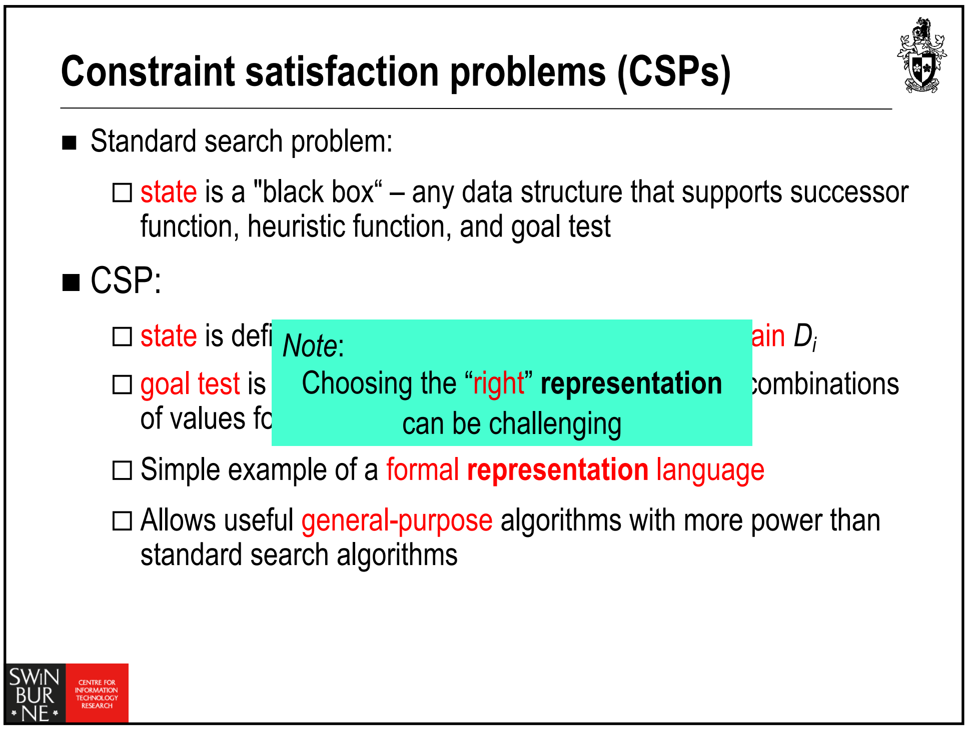
Sudoku

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

- Each row, column and “major block” must be all different
- “Well posed” if it has unique solution: 27 constraints

SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

16



Constraint satisfaction problems (CSPs)

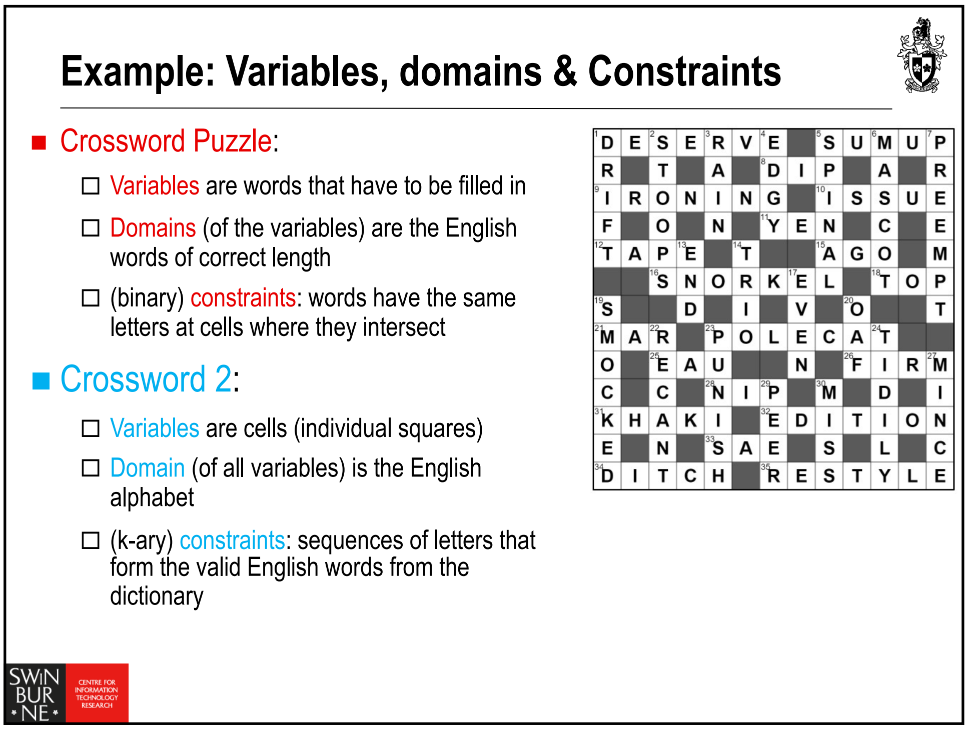
- Standard search problem:
  - state is a "black box" – any data structure that supports successor function, heuristic function, and goal test
- CSP:
  - state is defi ain D_i
  - goal test is combinations of values fo

Note:
Choosing the "right" **representation** can be challenging

  - Simple example of a formal **representation** language
  - Allows useful general-purpose algorithms with more power than standard search algorithms

SWIN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

17



Example: Variables, domains & Constraints

- Crossword Puzzle:
  - Variables are words that have to be filled in
  - Domains (of the variables) are the English words of correct length
  - (binary) constraints: words have the same letters at cells where they intersect
- Crossword 2:
  - Variables are cells (individual squares)
  - Domain (of all variables) is the English alphabet
  - (k-ary) constraints: sequences of letters that form the valid English words from the dictionary

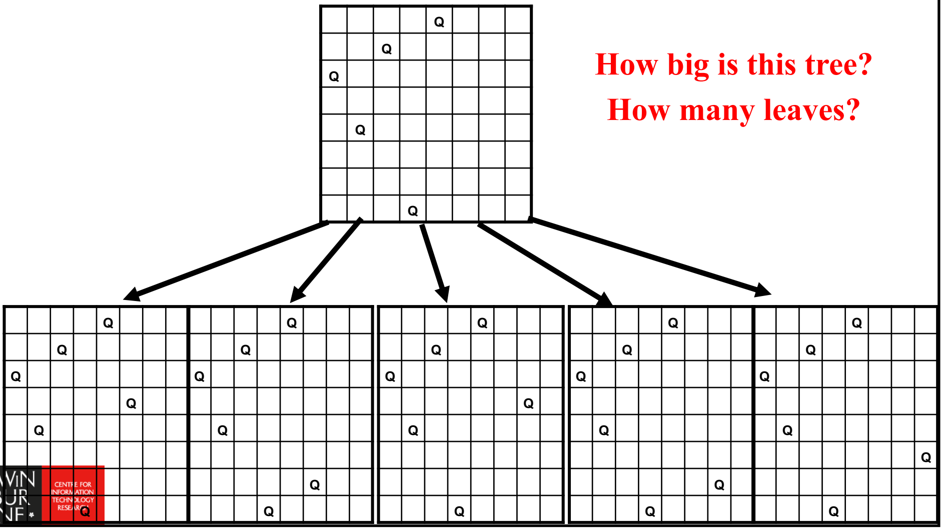
1 D	E	2 S	E	3 R	V	4 E	■	5 S	U	6 M	U	7 P
R	■	T	■	A	■	8 D	I	P	■	A	■	R
9 I	R	O	N	I	N	G	■	10 I	S	S	U	E
F	■	O	■	N	■	11 Y	E	N	■	C	■	E
12 T	A	P	13 E	■	14 T	■	■	15 A	G	O	■	M
■	■	16 S	N	O	R	K	17 E	L	■	18 T	O	P
19 S	■	■	D	■	I	■	V	■	20 O	■	■	T
21 M	A	22 R	■	23 P	O	L	E	C	A	24 T	■	■
O	■	25 E	A	U	■	■	N	■	26 F	I	R	27 M
C	■	C	■	28 N	I	29 P	■	30 M	■	D	■	I
31 K	H	A	K	I	■	32 E	D	I	T	I	O	N
E	■	N	■	33 S	A	E	■	S	■	L	■	C
34 D	I	T	C	H	■	35 R	E	S	T	Y	L	E

SWIN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

18

Search formulation of the queens puzzle

- **Successors**: all valid ways of placing additional queen on the board; **goal**: eight queens placed



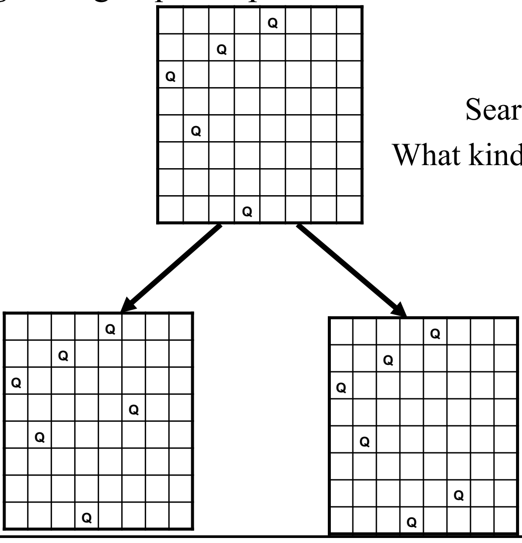
How big is this tree?

How many leaves?

19

Search formulation of the queens puzzle

- **Successors**: all valid ways of placing a queen in the next column; **goal**: eight queens placed

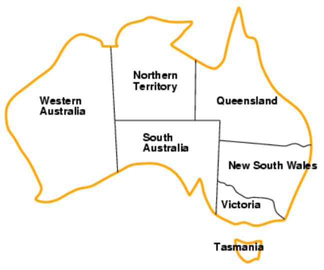


Search tree size?

What kind of search is best?

20

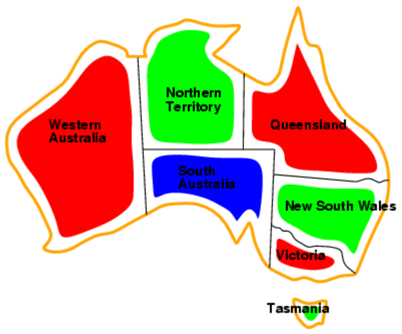
Example: Map-Coloring



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

21

Example: Map-Coloring

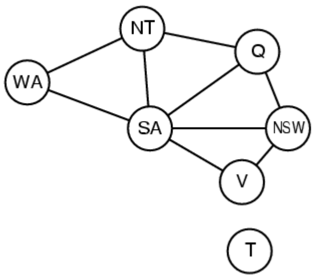


- **Solutions** are **complete** and **consistent** assignments, e.g., WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green

22

Constraint graph

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



23

Varieties of CSPs

- Discrete variables
  - finite domains:
    - *n* variables, domain size *d* → $O(d^n)$ complete assignments
    - e.g., Boolean CSPs, incl.~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

24

Real-world CSPs

- Staff assignment problems
  - e.g., who teaches what unit, which software developer does what tasks
- Timetabling problems
  - e.g., which class is offered when and where?
- Transportation scheduling
- Factory scheduling
- Notice that many real-world problems involve real-valued variables
-

25

Standard search formulation (quick analysis)

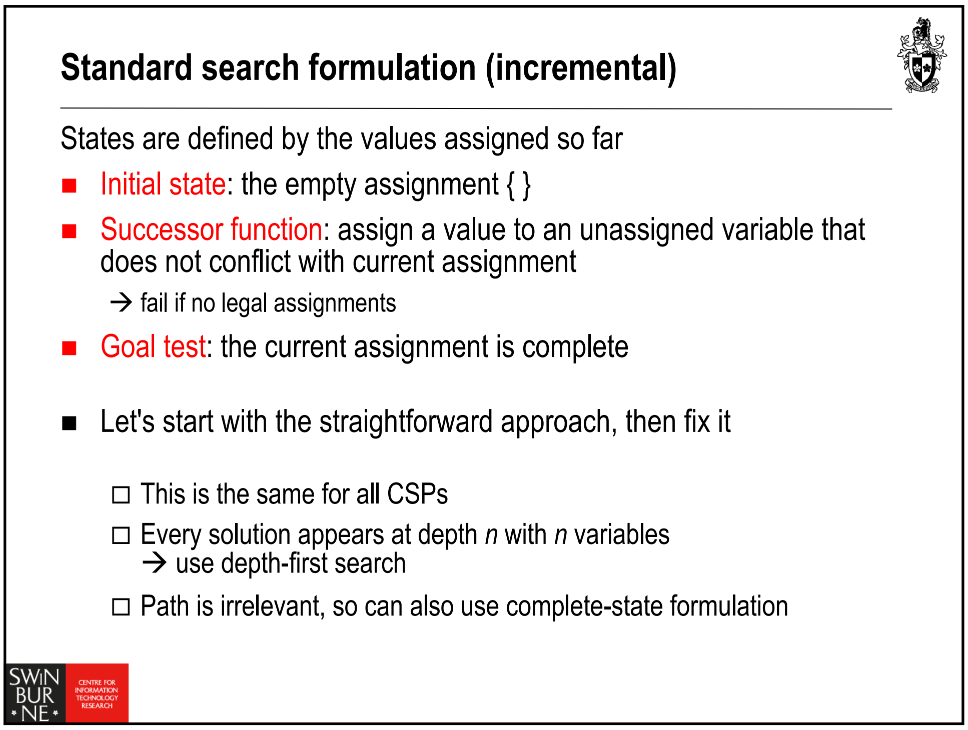
States are defined by the values assigned so far

- 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

- Would search methods based on BFS work well?
  - **Short answer is NO.**
  - **Large branching factor** -> Memory consumption alone is a barrier
  - Most of the time, we only need one solution!

26



Standard search formulation (incremental)

States are defined by the values assigned so far

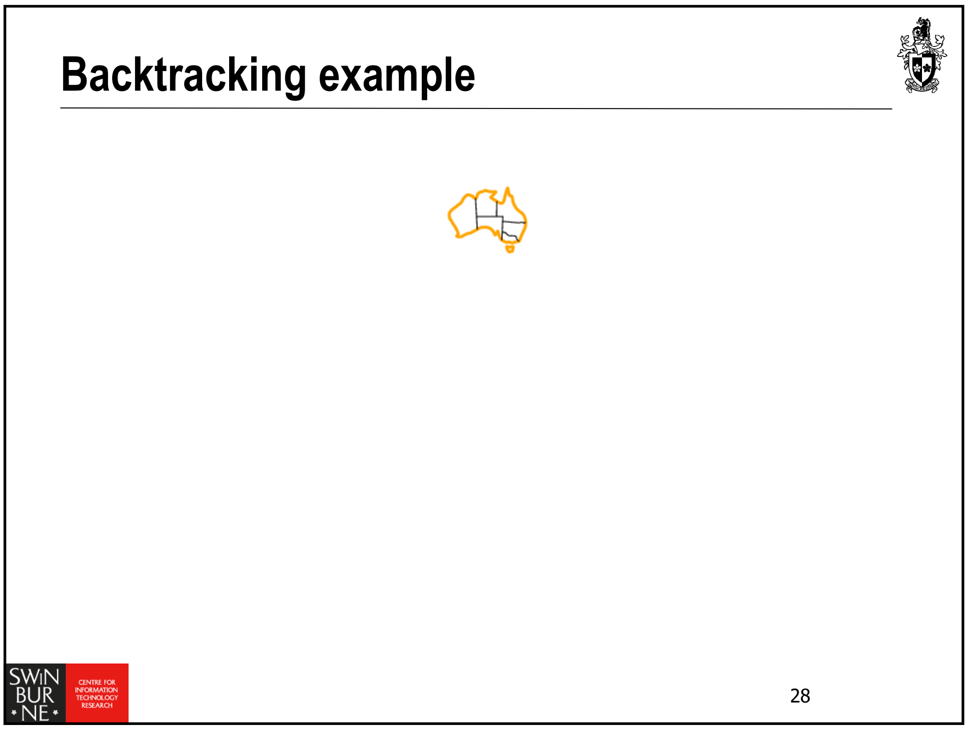
- Initial state: the empty assignment { }
- Successor function: assign a value to an unassigned variable that does not conflict with current assignment

  → fail if no legal assignments
- Goal test: the current assignment is complete

- Let's start with the straightforward approach, then fix it

  - This is the same for all CSPs
  - Every solution appears at depth *n* with *n* variables
    → use depth-first search
  - Path is irrelevant, so can also use complete-state formulation

27

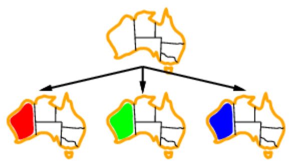


Backtracking example

28

28

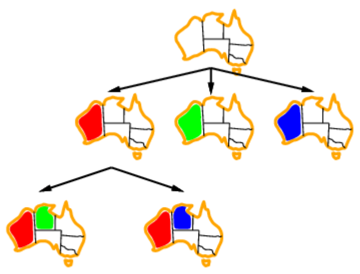
Backtracking example



29

29

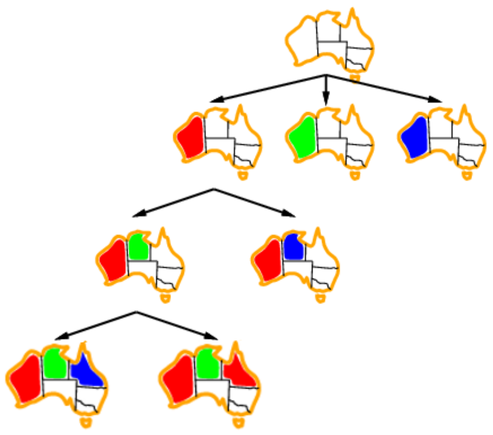
Backtracking example



30

30

Backtracking example



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 search

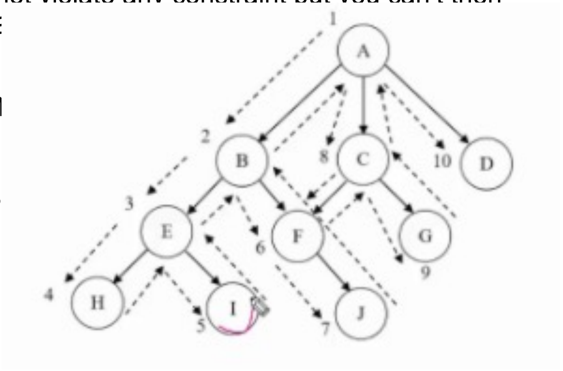
- Depth-first search for CSPs with single-variable assignments is called backtracking search
  - ☐ Backtracking is the idea that, as you go depth-first down a branch of the search tree and you try to find an assignment for the next variable that does not violate any constraint but you can't then you have to backtrack (from that current branch) to get to another branch
- Backtracking search is the basic **uninformed** algorithm for CSPs
- Can solve *n*-queens for $n \approx 25$
-

SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

33

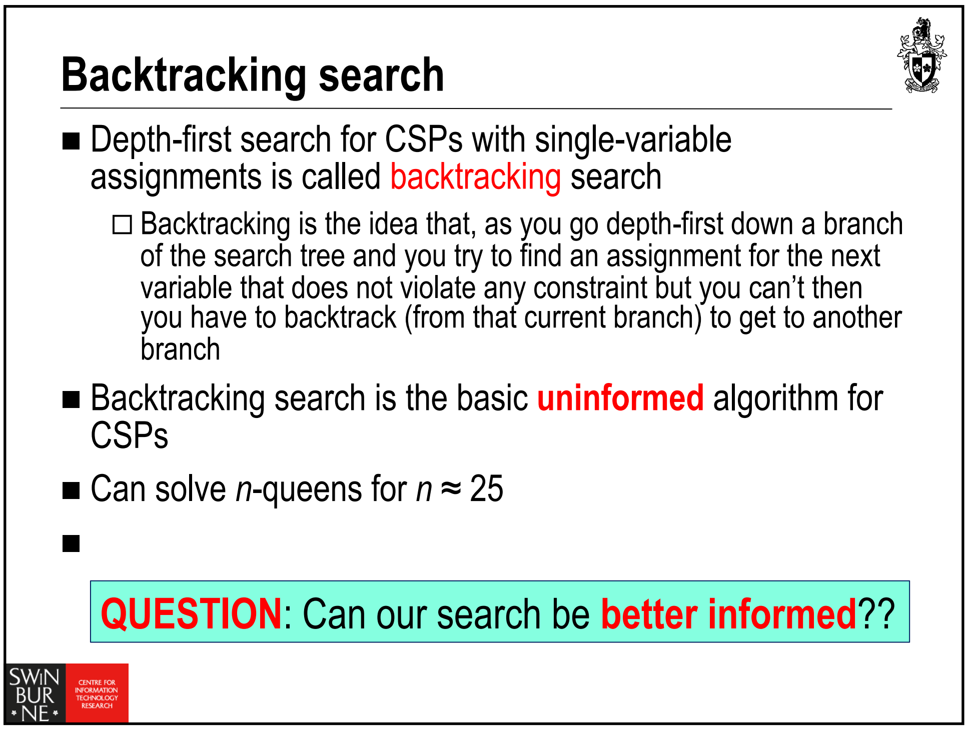
Backtracking search

- Depth-first search for CSPs with single-variable assignments is called backtracking search
  - ☐ Backtracking is the idea that, as you go depth-first down a branch of the search tree and you try to find an assignment for the next variable that does not violate any constraint but you can't then you have to backtra branch
- Backtracking search CSPs
- Can solve *n*-queens
-



SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

34

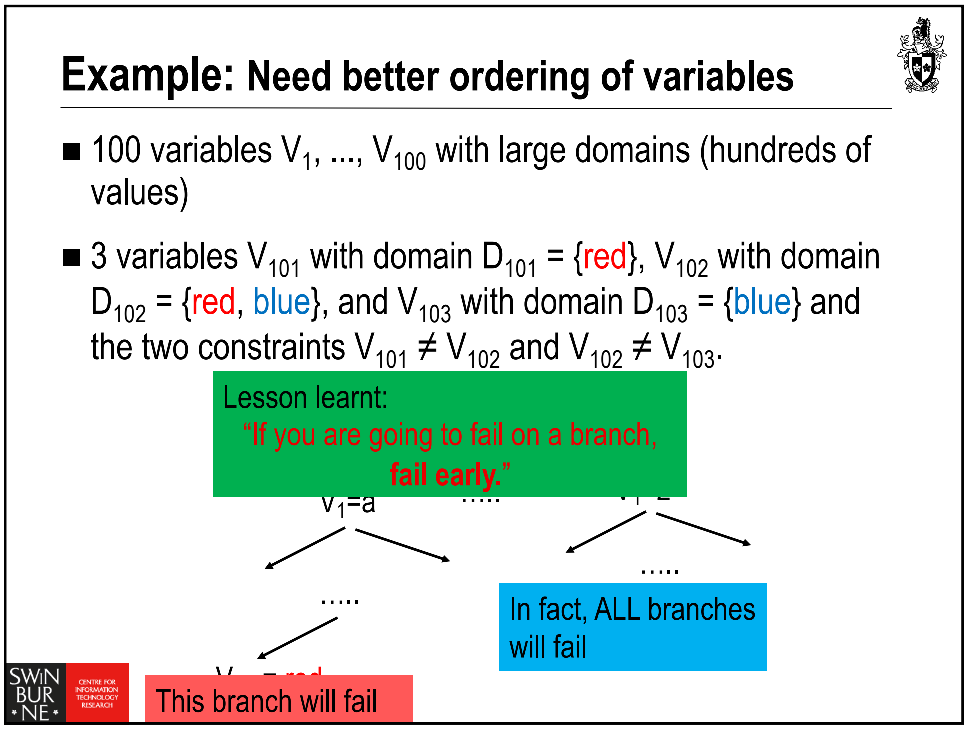


Backtracking search

- Depth-first search for CSPs with single-variable assignments is called backtracking search
  - Backtracking is the idea that, as you go depth-first down a branch of the search tree and you try to find an assignment for the next variable that does not violate any constraint but you can't then you have to backtrack (from that current branch) to get to another branch
- Backtracking search is the basic **uninformed** algorithm for CSPs
- Can solve *n*-queens for $n \approx 25$
-

QUESTION: Can our search be **better informed**??

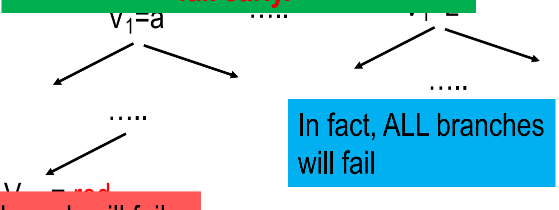
35



Example: Need better ordering of variables

- 100 variables $V_1, \ldots, V_{100}$ with large domains (hundreds of values)
- 3 variables V_{101} with domain D_{101} = {red}, V_{102} with domain D_{102} = {red, blue}, and V_{103} with domain D_{103} = {blue} and the two constraints $V_{101} \neq V_{102}$ and $V_{102} \neq V_{103}$.

Lesson learnt:
"If you are going to fail on a branch, **fail early.**"



In fact, ALL branches will fail

This branch will fail

36

Improving backtracking efficiency

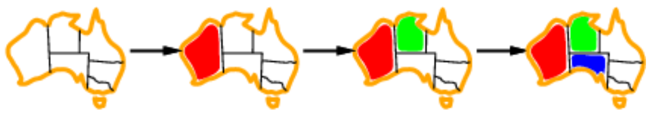
- General-purpose methods can give huge gains in speed (i.e. search in a smart way, **not in an uninformed manner**):
  - ☐ Which variable should be assigned next?
  - ☐
  - ☐ In what order should its values be tried?
  - ☐
  - ☐ Can we detect inevitable failure early?
  - ☐

37

Most constrained variable

- Most constrained variable:

  choose the variable with the fewest legal values

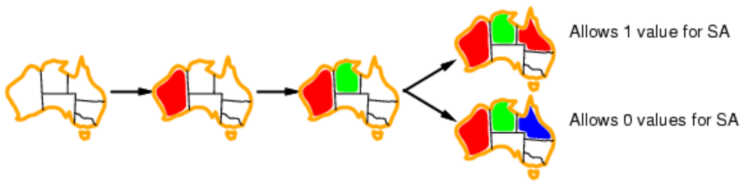


- a.k.a. minimum remaining values (MRV) heuristic
-

38

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
-

39

Forward checking

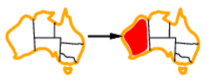
- Idea:
  - ☐ Keep track of remaining legal values for unassigned variables
  - ☐ Terminate search when any variable has no legal values
  - ☐

WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦

40

Forward checking

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



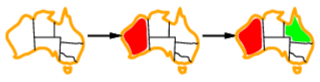
WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦

SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

41

Forward checking

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



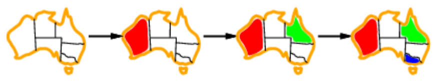
WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦

SWiN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

42

Forward checking

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



WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦
🟥	🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟩🟦	🟥🟩🟦
🟥	🟦	🟩	🟥🟦	🟥🟩🟦	🟦	🟥🟩🟦
🟥	🟦	🟩	🟥	🟦		🟥🟩🟦

43

Constraint propagation

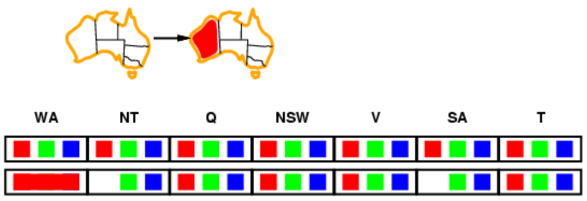
- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:
-

WA	NT	Q	NSW	V	SA	T
🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦	🟥🟩🟦

44

Constraint propagation

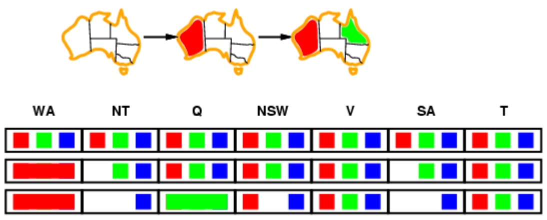
- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:
-



45

Constraint propagation

- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:
-

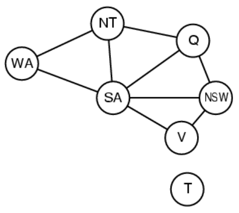


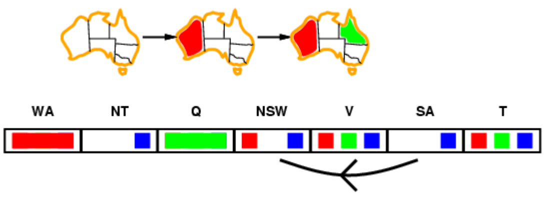
- NT and SA cannot both be blue!
- Constraint propagation repeatedly enforces constraints locally
-

46

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

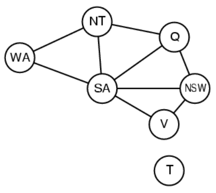


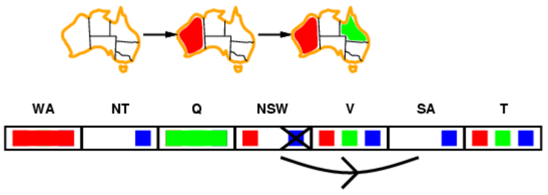


47

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





48

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
  -

SWIN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

49

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 is usually effective in practice

SWIN BUR NE CENTRE FOR INFORMATION TECHNOLOGY RESEARCH

50