Constraint Satisfaction Problem

UI 4

Markošová

Summary of the last lecture

1. Local search, properties.
2. Local search: hill climbing, genetic algorithm
3. Simulated annealing

Outline

1. CSP problems: definition, examples
2. Node and edge consistency, binarization of constraits.
3. Backtracking for CSP.
4. Heuristics for CSP.
5. Games – introduction.

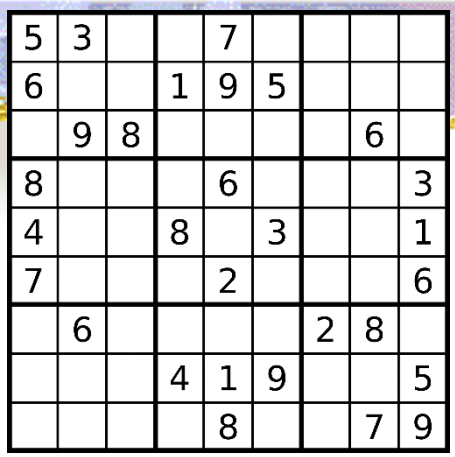
CSP problem

CSP Example

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

Sudoku game:
Fill all the squares in a correct way, respecting several constraints.

Constraints: On each square there is one number 1, 2….9. There are different numbers in all squares of each row, each columns and in each bigger square.



CSP definition:

1. ***Set of n variables*** $X_1, X_2, \ldots X_n$ (small squares in Sudoku)
2. ***Set of constraints*** $C_1, C_2, \ldots C_m$
3. Each variable has a ***nonempty set*** D_i of its possible values (set {1,2…,9}in Sudoku)
4. ***State*** : defined by assigning a value to the variable.
5. ***Consistent state***: assignment which does not violate the constraints
6. ***Complete state***: state where all variables has values assigned
7. ***Solution of CSP***: complete and consistent state

Another example

Variables: x, y, z

Domains: Each variable can have a value from this set $\{1,2,3,4,5,6\}$

Constraints: $x + y = 6,\ y.z \leq 20$

Some asignments:

x=1, y=5, z=4

x=1, y=5, z=3

x=1, y=5, z=2

x=2, y=4, z=5

x=2, y=4, z=2

Some consistent and complete value assignment

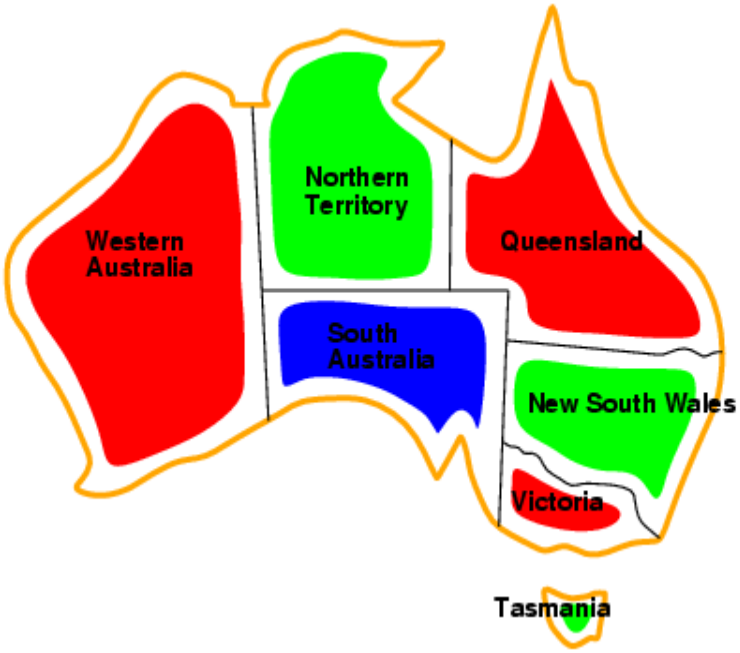
Example: Map-Coloring (Russel, Norwig)

- Variables *WA*, *NT*, *Q*, *NSW*, *V*, *SA*, *T*
- Domains D_i = {red, green, blue}
- Constraints: neigbor regions have different colours
  e.g., WA ≠ NT, or (WA,NT) v {(red,green),(red,blue),(green,red), (green,blue),(blue,red),(blue,green)}

Example: Map-Coloring

How many solutions we have for this problem?

For each coloring of WA, Q and V we have two possibilities to color other regions. There are 6 possibilities.



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

Example:

Problem of 8 queens as CSP:

Variables: $Q_1, Q_2, \ldots Q_8$ position on the rows $\{1, 2, 3, \ldots, 8\}$

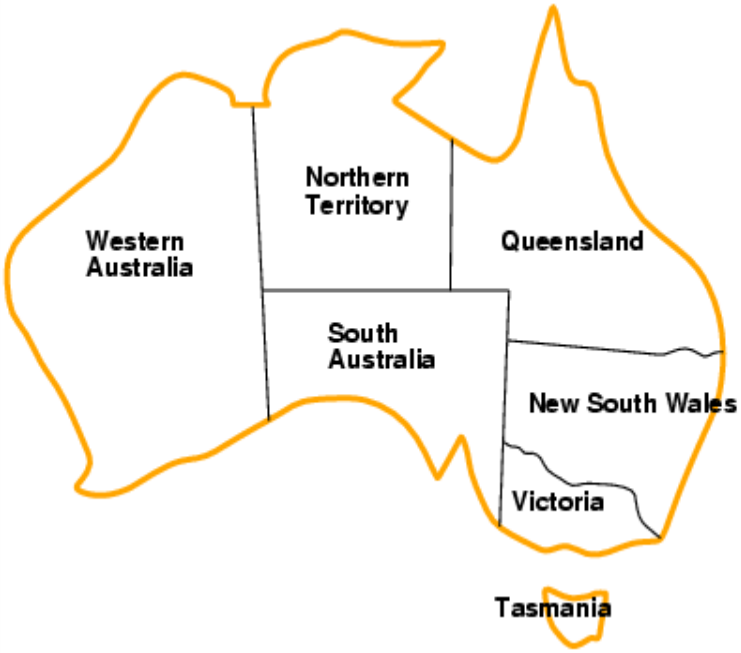
Domains: each variable has a domain {1,2,3,4,5,6,7,8}

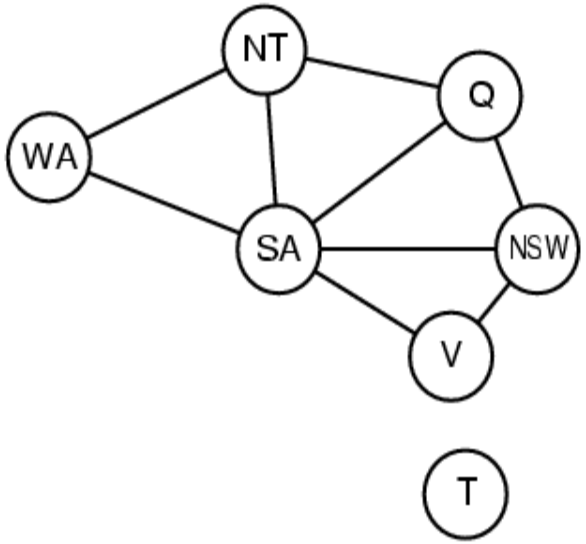
CSP with final discrete domain

Constraints: queens cannot attack each other

Constraint graph (graph vizualizing constraints)

- Binary CSP : each constrain is a relation of two variables.
  Constraint graph: nodes are variables, edges are constraints.





Variants of CSP problems

- Problem in discrete variables
  - With finite domains
    - n variables, number of complete assigments $d \rightarrow O(d^n)$
  - With infinite domains:
    - integers, strings, etc.
- Problem in finite continuous variables
  - e.g., start/end times of observations in experiment
  - linear constraints are solvable in the polynomial time with a help of the linear programming

Variants of constraints

- Unary constraint only one variable is affected,
  - E.g., SA ≠ green
- Binary constraints pair of variables is affected,
  - napr., SA ≠ WA (the colour of neighbor regions is different)
- Higher-order or n –ary constraint more variables are involved
  - cryptarithmetic constraint

Cryptoaritmetic problem:

Replacing letters by the numbers gives a correct result.

```
  T W O
+ T W O
-------
F O U R
```

Domains? Numbers 0 - 9

Constraints? Each letter has different value.

$F \neq 0$

$O \neq 0$

$2T \geq 10$

…

Solution? E.g.. O=5, R=0, T=7, W=6, U=3, F=1

Binarization

CSP algorithm: Work with unary or binary constraints → **Problem of the constraint binarization**

Binary CSP: can be visualized as a constraint graph

Example:

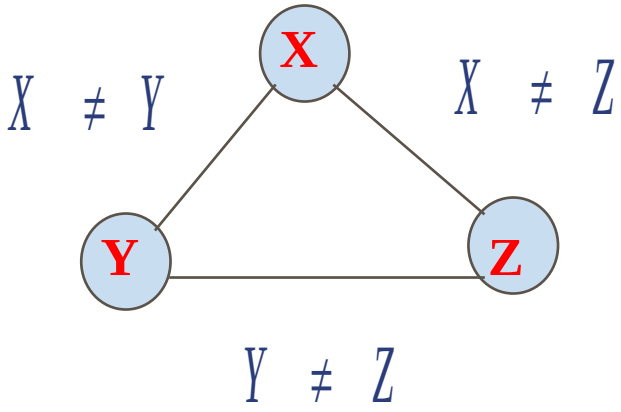
Constraint systém:

$$X \neq Y$$

$$Y \neq Z$$

$$X \neq Z$$

Constraint graph



Good news: each CSP can be turned to a binary CSP.

Node and arc consistency

Knowing the consistency techniques, it is possible to reduce the problem complexity

Node consistency

In the constraint graph the node represents certain variable.

Definition of the node consistency:

Node is node consistent if for each value x in the domain of the variable, unary constraints are fullfilled. .

How to reach the node consistency:

Simply remove those values from the domain which causes the problem.

Example:

Variables : X, Y

Domain for X: {0,10} **Domain for Y**: {5, 20} integers

Constraints: binary constraint Y > X, unary constrait X>5

Node consistent variant of the problem

Variables : X, Y

Domain for X: {10}, **Domain for Y**: (5, 20)

Constraint: binary constraint Y > X.

Arc consistency:

Binary constraints are represented by the arcs of the constraint graph.

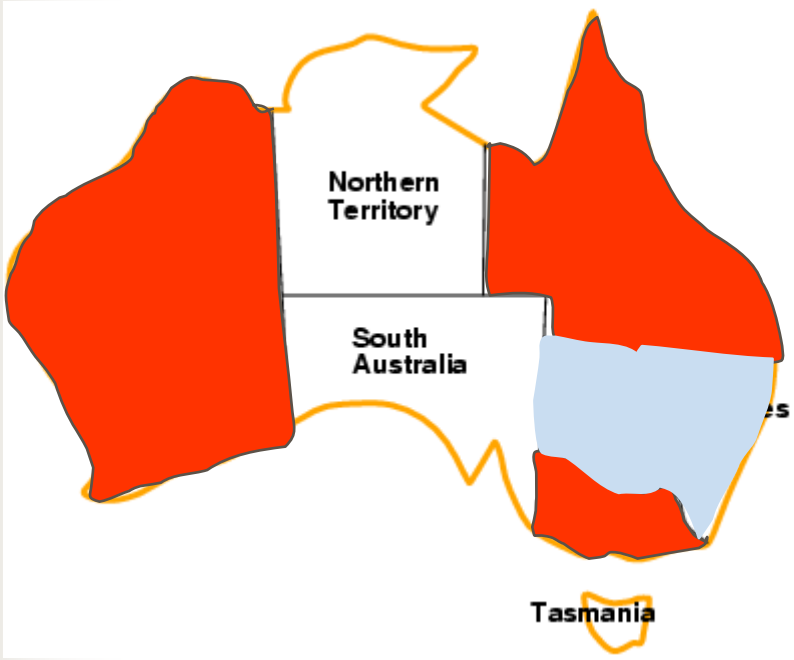
Definition of the arc consistency:

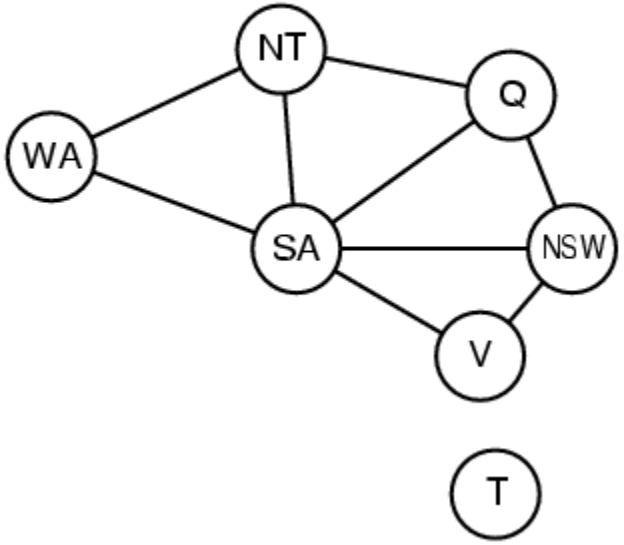
Arc (V_i, V_j) is arc consistent if for each value x from the domain of V_i variable there exists some value y from the domain of V_j such that $V_i = x$ and $V_j = y$ is allowed by the binary constraint.

How to reach arc consistency:

Remove such values from the domain of V_i to which there is no corresponding value in the domain of V_j.

Example:

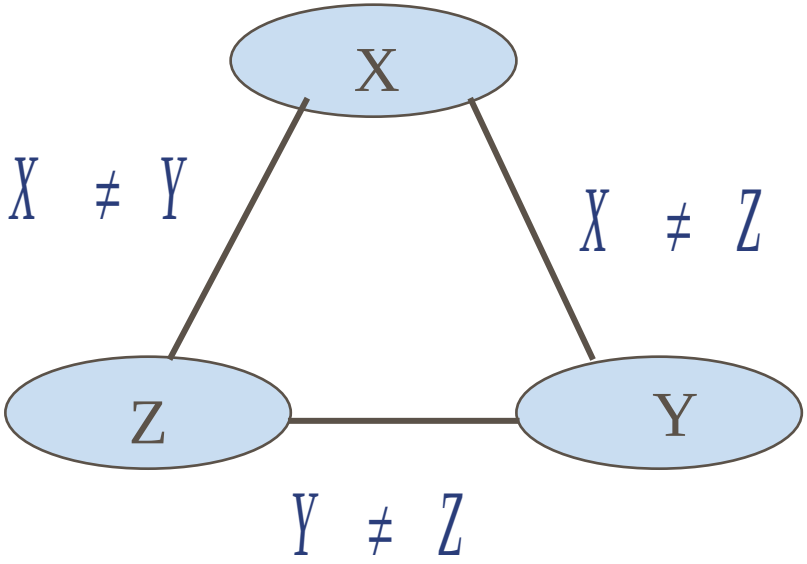




From the domain of NT remove red and from the domain of SA remove red and blue.

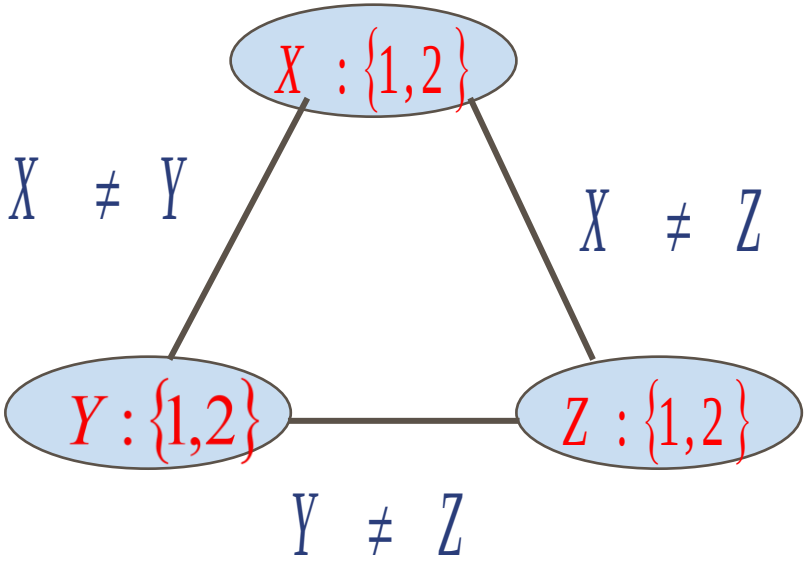
Question: Is it possible, that the problem is arc consistent, and in spite of this CSP has no solution?

Example:



Question: Is it possible, that the problem is arc consistent, and in spite of this CSP has no solution?

Example:



Algorithms of the systematic searching

Generate and test algoritmus (GT)

Generate systematically possible assigments to the variables and the complete assigment is tested whether the constraints are fullfiled . Solution is the first appropriate combination.

Algorithm is: -complete for infinite time

- unoptimal

- exponential time and memory complexity

CSP as a searching problem

Initial state: empty assignment, no variable has a value

Successor function : operators assigning values, avoiding conflicts with those previously assigned

Goal test: positive for complete and consistent assignment

Step cost: for example constant for each step

CSP as a searching problem for the state colouring

CSP can be formulated as a searching problem (n – number of variables (states) , d – number of values (colours)).

Bred first search:

1. level branching - nd

2. level branching- $(n-1)d$

.

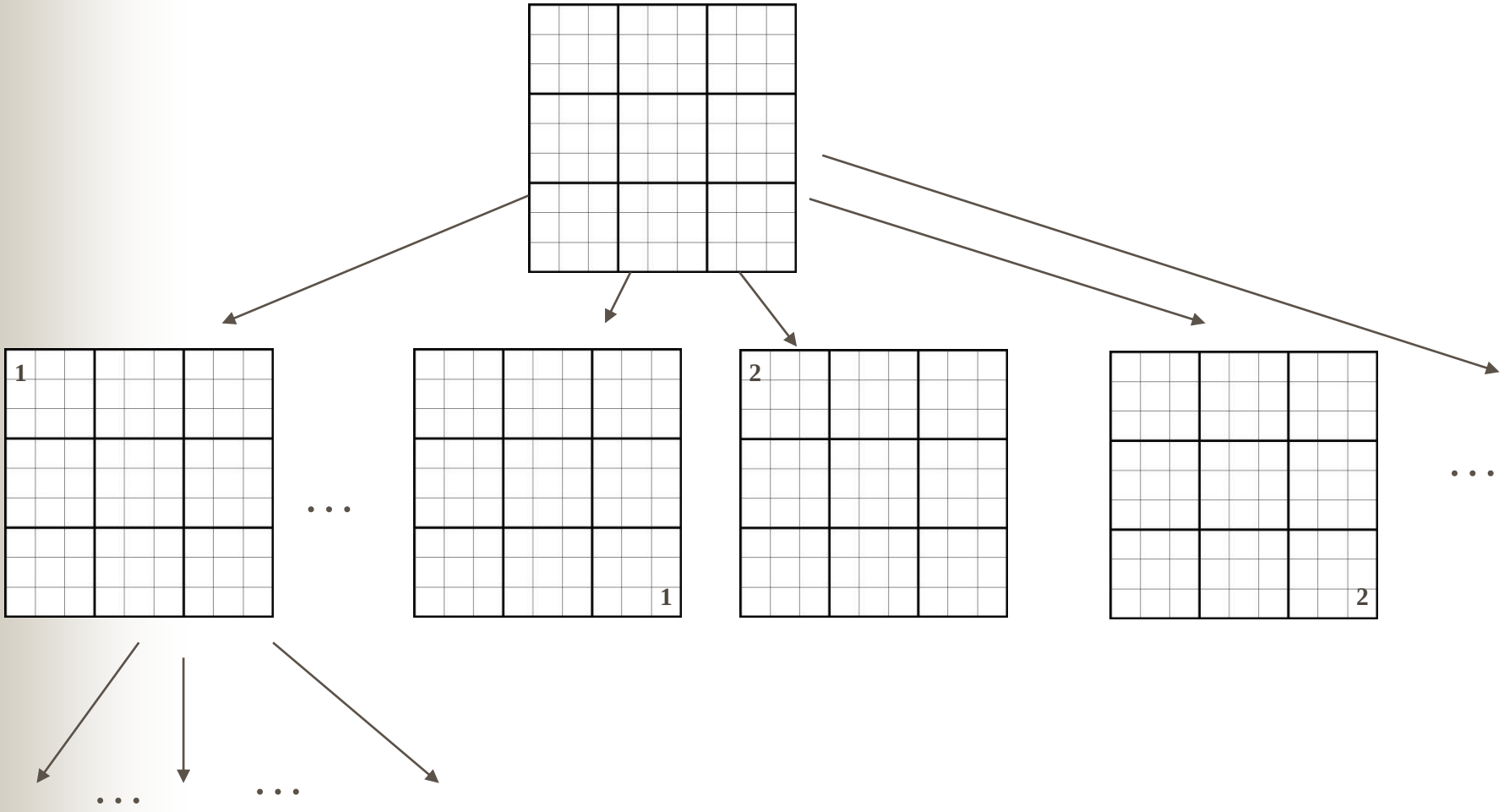
.

.

n. level - $n!\, d^n$ leaves

There are only : d^n complete configurations

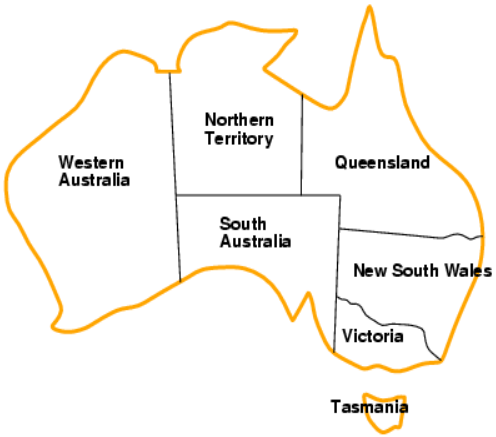
Example Sudoku:



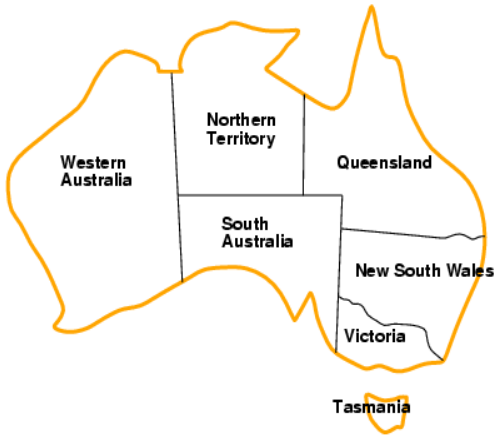
Backtracking search

Backtracking is in fact depth first search, which in each step assignes the value to only one variable. Algorithm back tracks when there is no variable to assign.

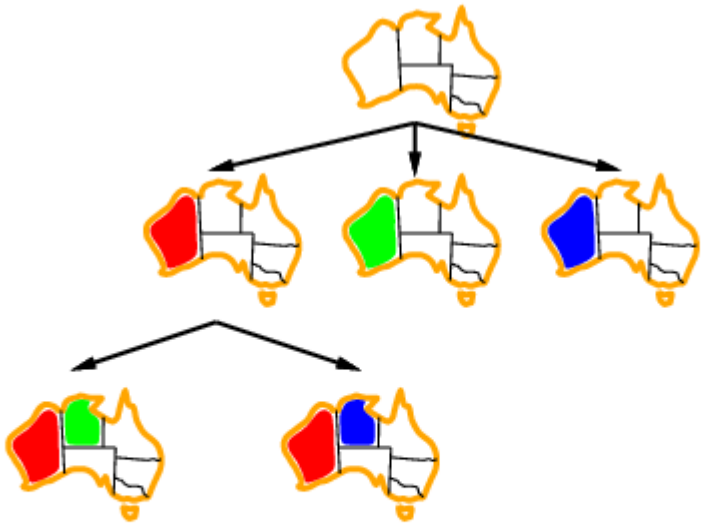
Backtracking example

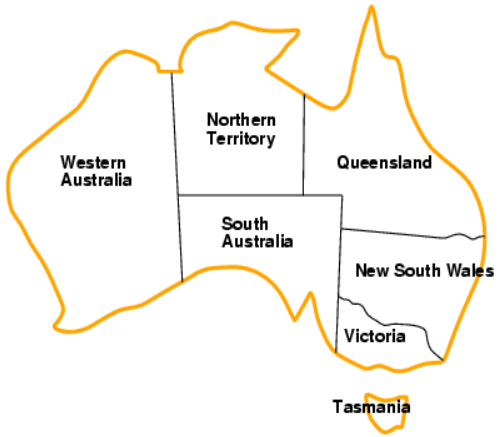


Backtracking example

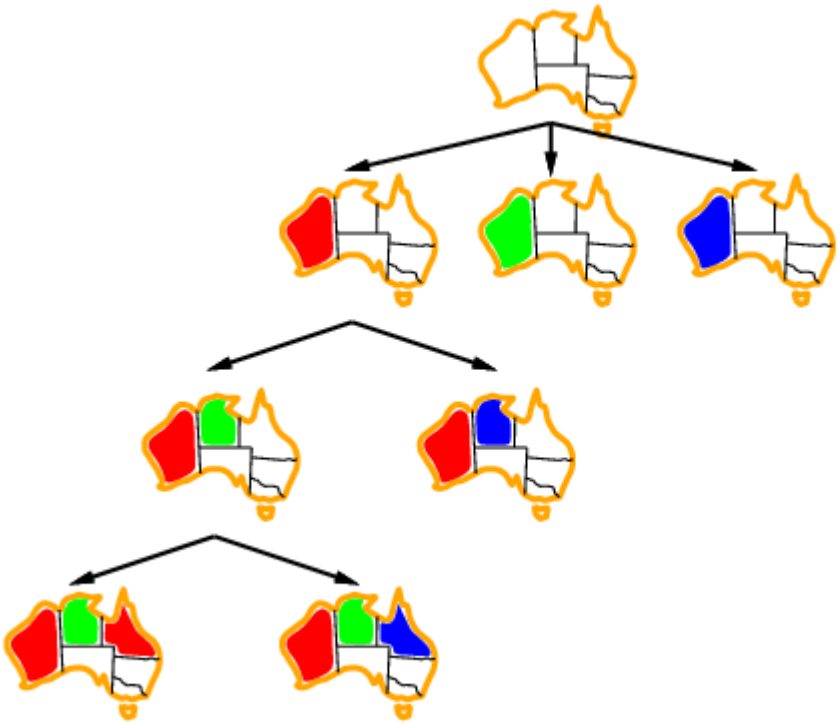


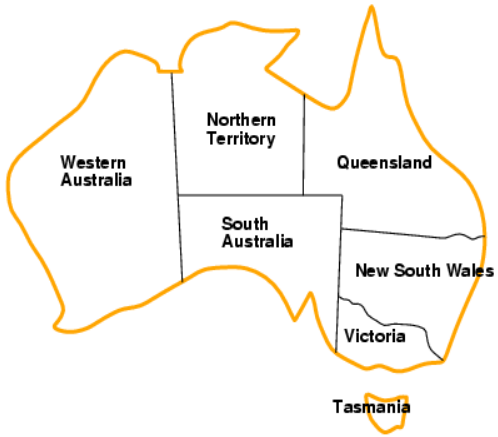
Backtracking example





Backtracking example





Backtracking search

- Assignment of the variables is commutative, e.g.,

[WA = red then NT = green] is the same as [NT = green then WA = red]

- In each step only one variable is assigned.

- **Backtracking search is the basic algorithm for the CSP problem**

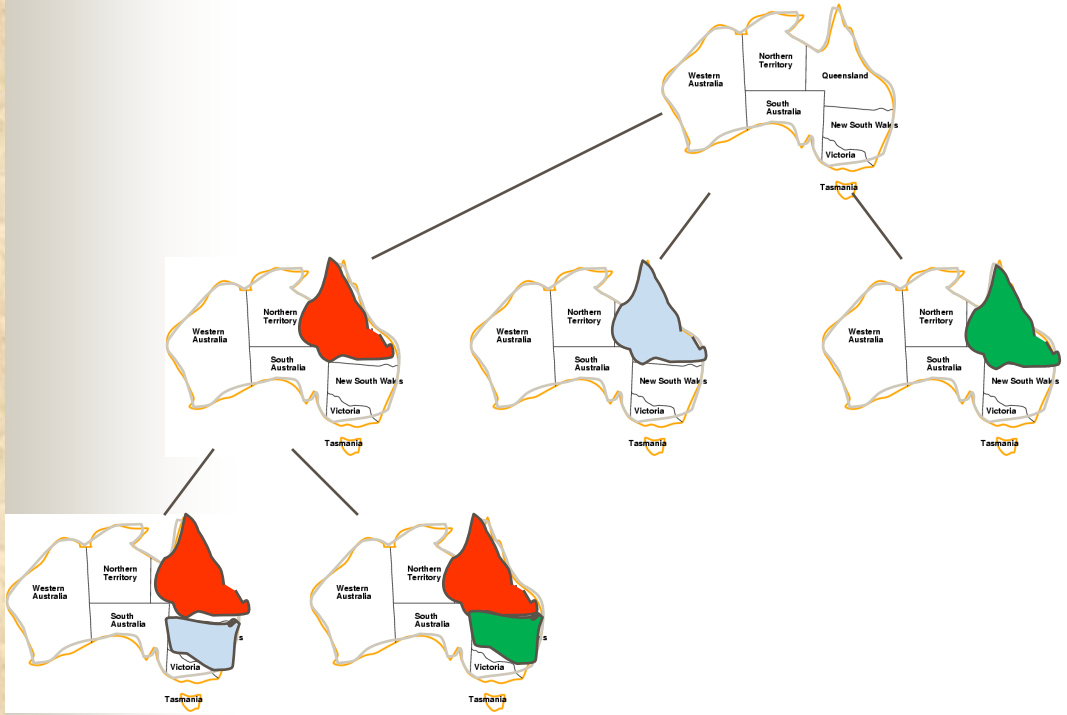
- Can solve n-queens for $n \approx 25$

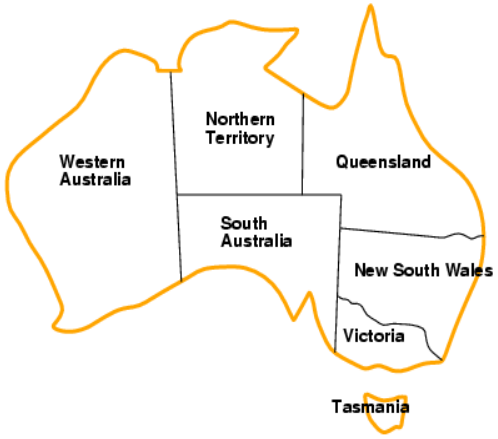
Backtracking, map colouring example

1. First the map of Australia is empty. Variables are ordered in a que.
2. Assign all possible values to the first variable in a que, for example WA teritory.
3. Take the first unexpanded node as in the depth first and assign all rest colours to the second variable in a que, for example NT.
4. Repeat 3 until we get a correct assignment or the constraint violation.
5. If the partial solution violates some constrain, BT algorithm back tracks to the last assigned value, which has an alternative.

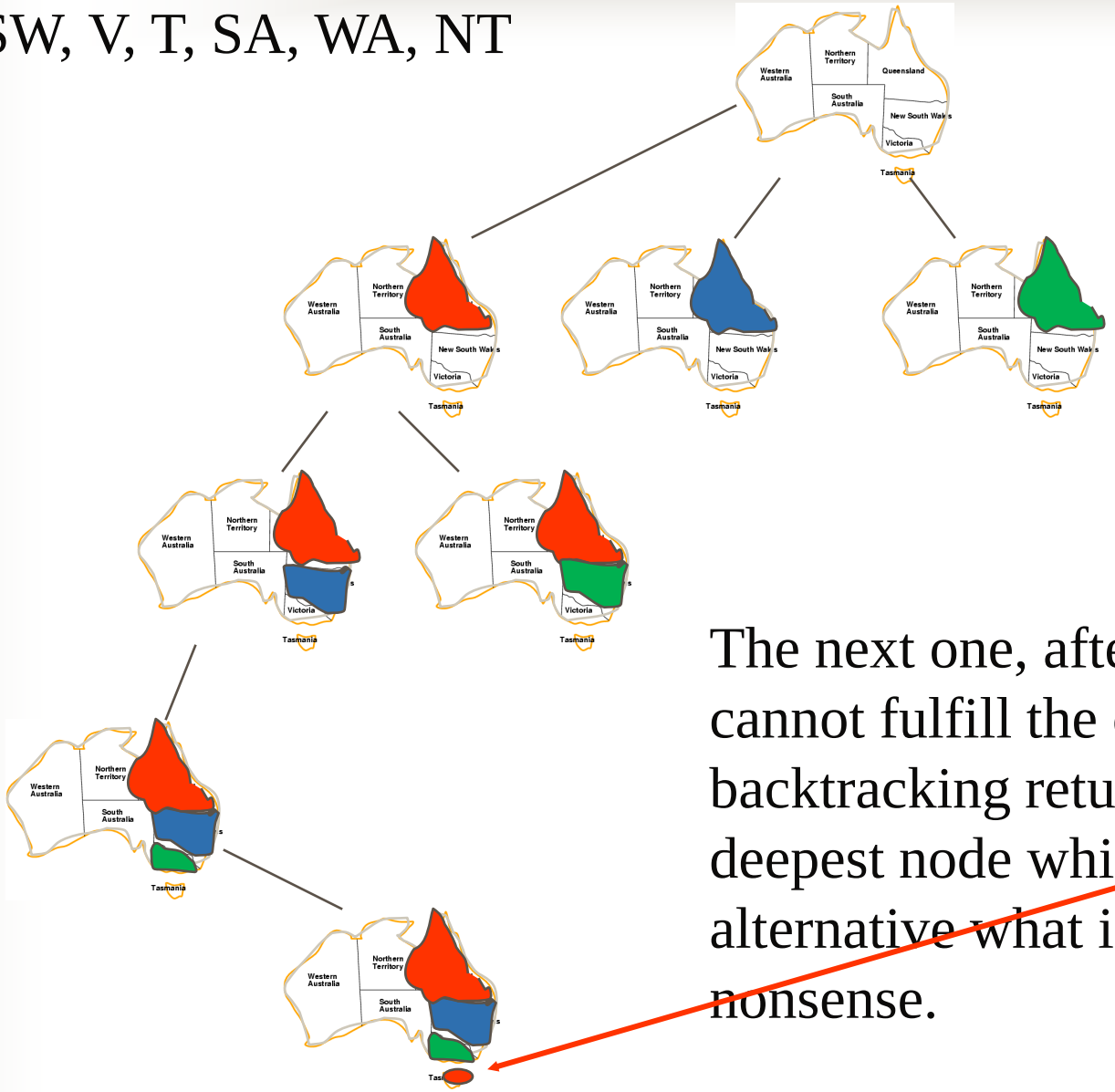
More intelligent backtracking:

Let the ordering of the variables in the simple backtracking is: Q, NSW, V, T, SA, WA, NT





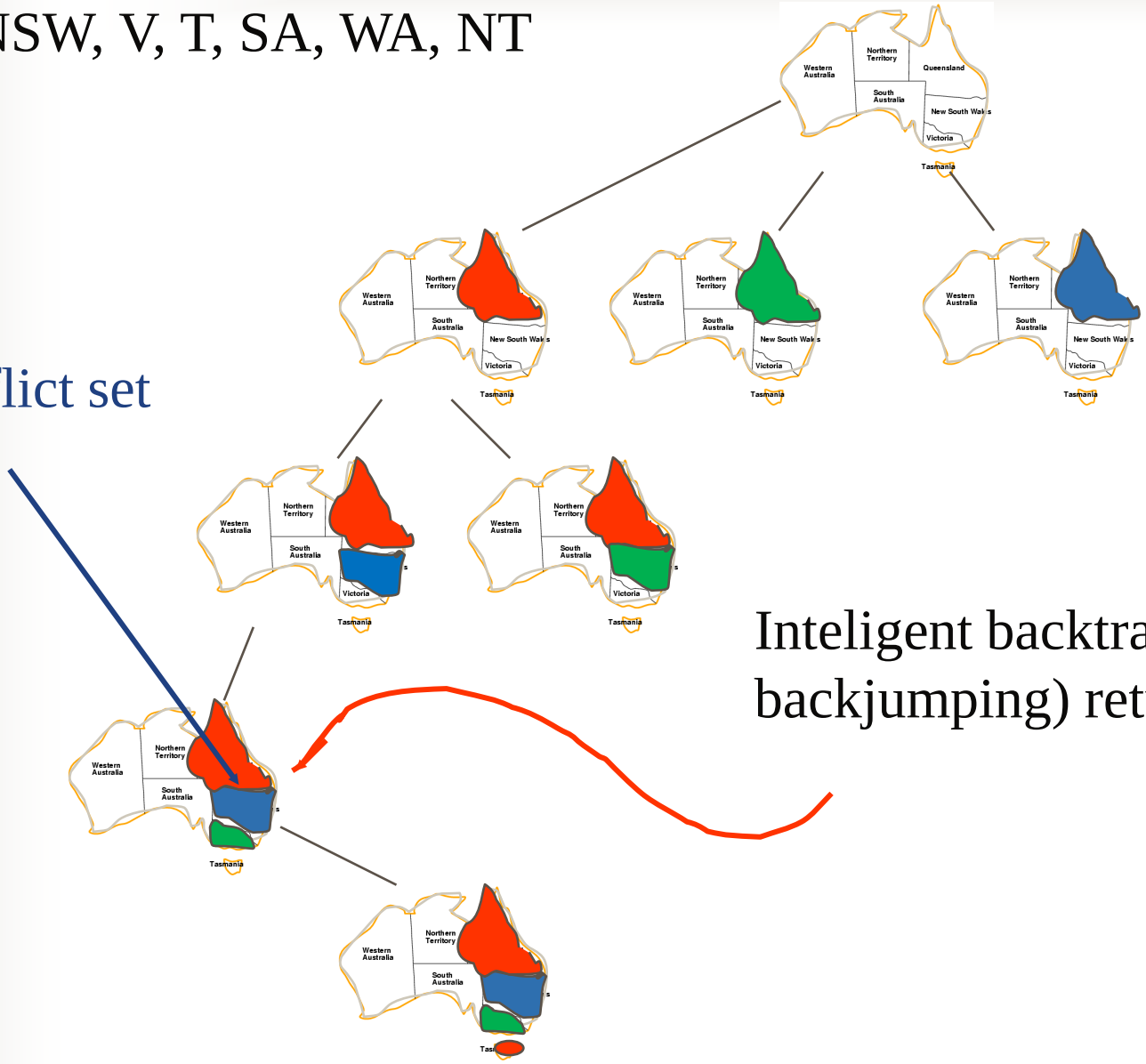
Q, NSW, V, T, SA, WA, NT



The next one, after T, is SA, and cannot fulfill the constraint, but backtracking returns to the deepest node which has an alternative what is T. This is nonsense.

Q, NSW, V, T, SA, WA, NT

conflict set



Inteligent backtracking (called backjumping) returns here.

Conflict set :

Conflict set for the X variable is a set of values assigned before, which are connected with X by constraint.

Simple backtracking and its improvements :

Possibilities of the improvement:

1. Effectivity can be improved by **the aprropriate ordering of the variables** and appropriate **ordering of the value assignemnt**
2. Is it possible to estimate the possible conflict set?

Heuristics for the CSP

Heuristics in CSP concerns state, not the path to the goal and help us to find the violation of the constraints as soon as possible

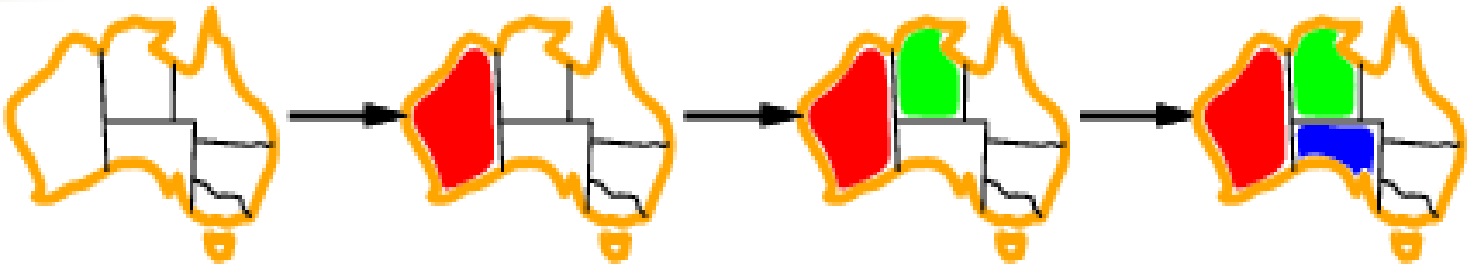
1. **Most constraint variable**, or minimum remaining value (MRV) heuristics.
2. **Most constraining variable**, or the degree heuristika.
3. **Least constraining value** heuristics.

Minimum remaining value (most constrained variable)

- **Minimum remaining value**

  Choose a variable with the less possible number of values.

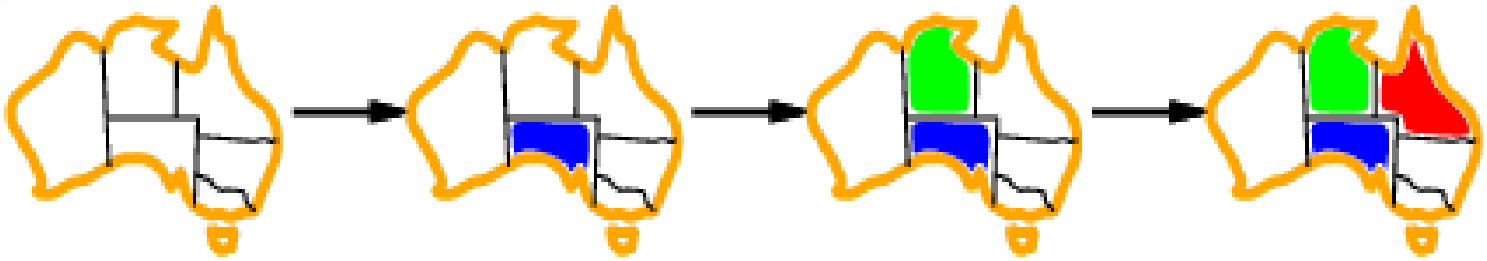
  (because this variable leads quickly to the constraint violation)



Degree heuristics

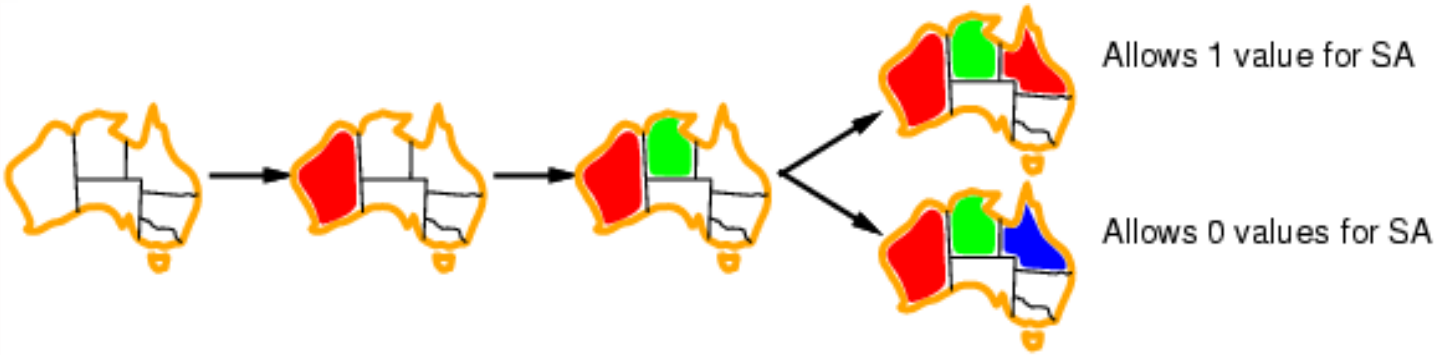
- **Most constraining variable**:

  Degree heuristics reduces the branching factor for the future assignments, because the chosen variable constraints most of the other variables,



Least constraining variable

- For certain variable choose the least constraining value:
  - This value removes the smallest number of the other variables. (This heuristics is good for estimating how to order the values from the domain at the end of the run)



Methods for quick CSP solution

1. **Forward checking**.
2. **Arc consistency control** (MAC, maintaining arc consistency method)
3. **Special type constraints controll** (**alldiff** a **atmost** constrain)

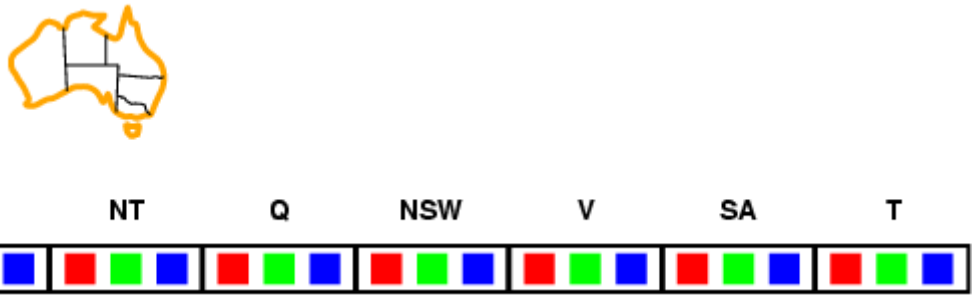
Forwad checking:

After giving value to the variable *X*, find a variable *Y* which is in the relation with *X*. From the domain of *Y* remove all values which are not consistent with the value of *X*.

Combine with the „most constrained variable“ heuristics.

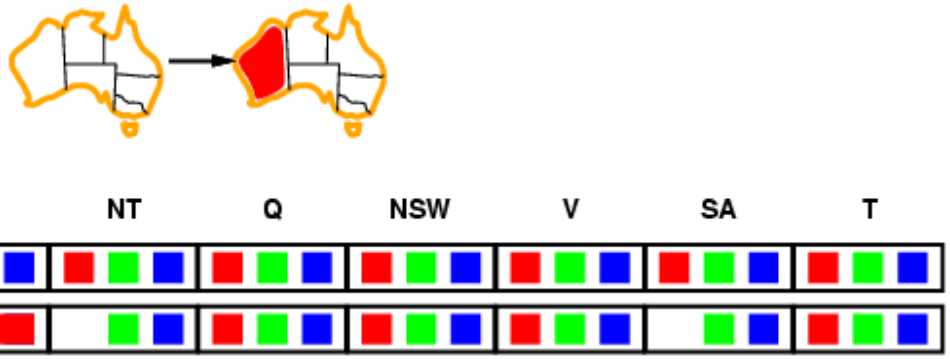
Forward checking

- Idea:
  - Keep possible rest values for unassigned variables
  - Stop if there is no value to assigne for certain variable.



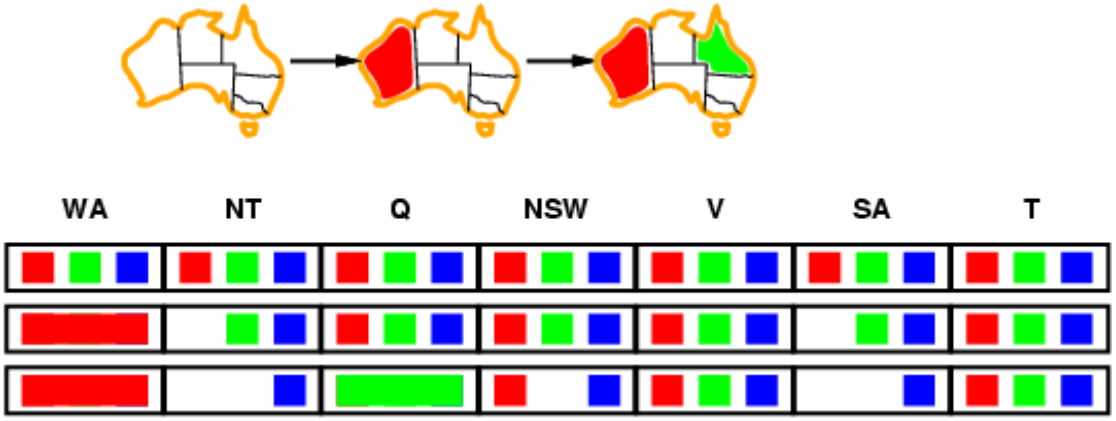
Forward checking

- Idea:
  - Keep possible rest values for unassigned variables
  - Stop if there is no value to assigne for certain variable



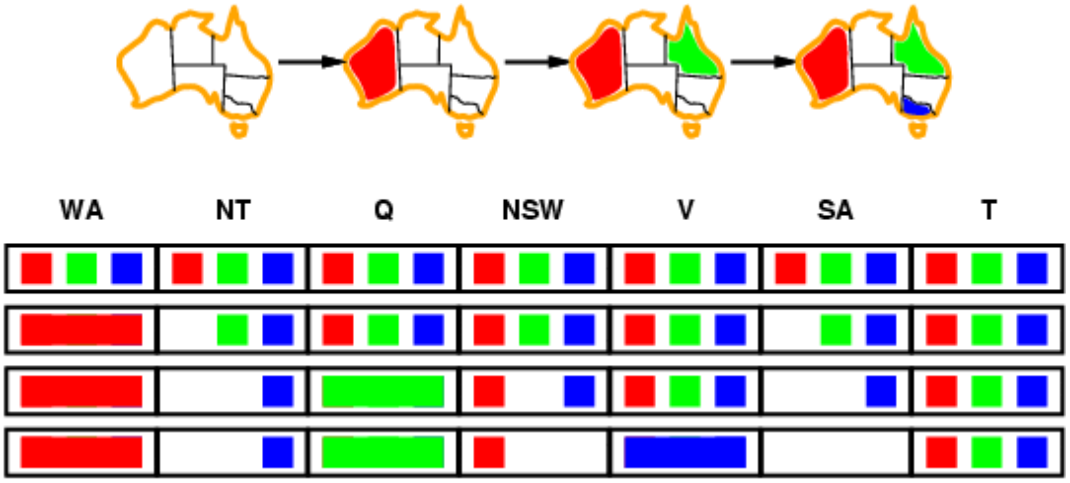
Forward checking

- Idea:
  - Keep possible rest values for unassigned variables
  - Stop if there is no value to assigne for certain variable

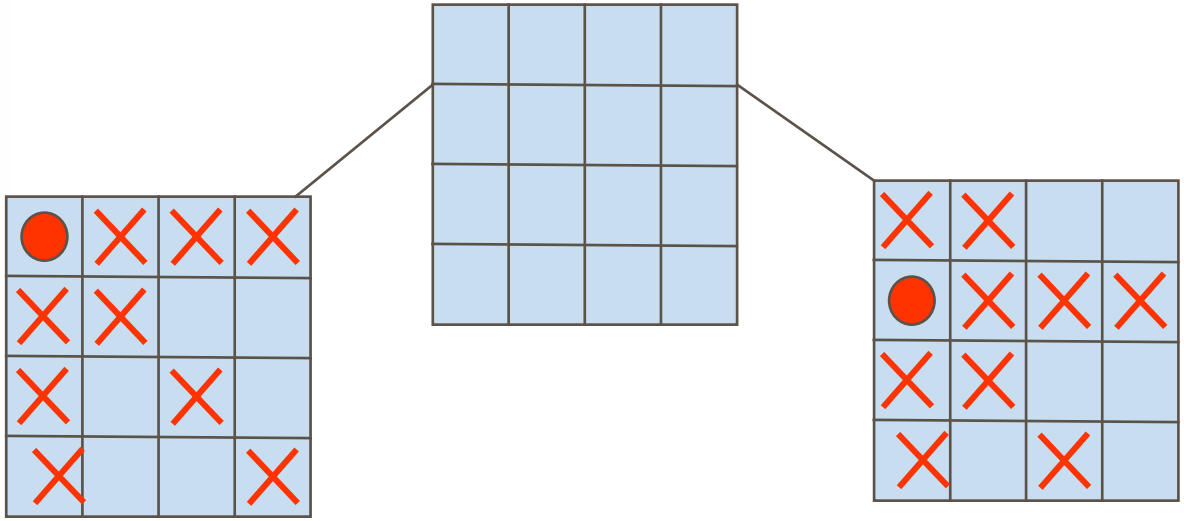


Forward checking

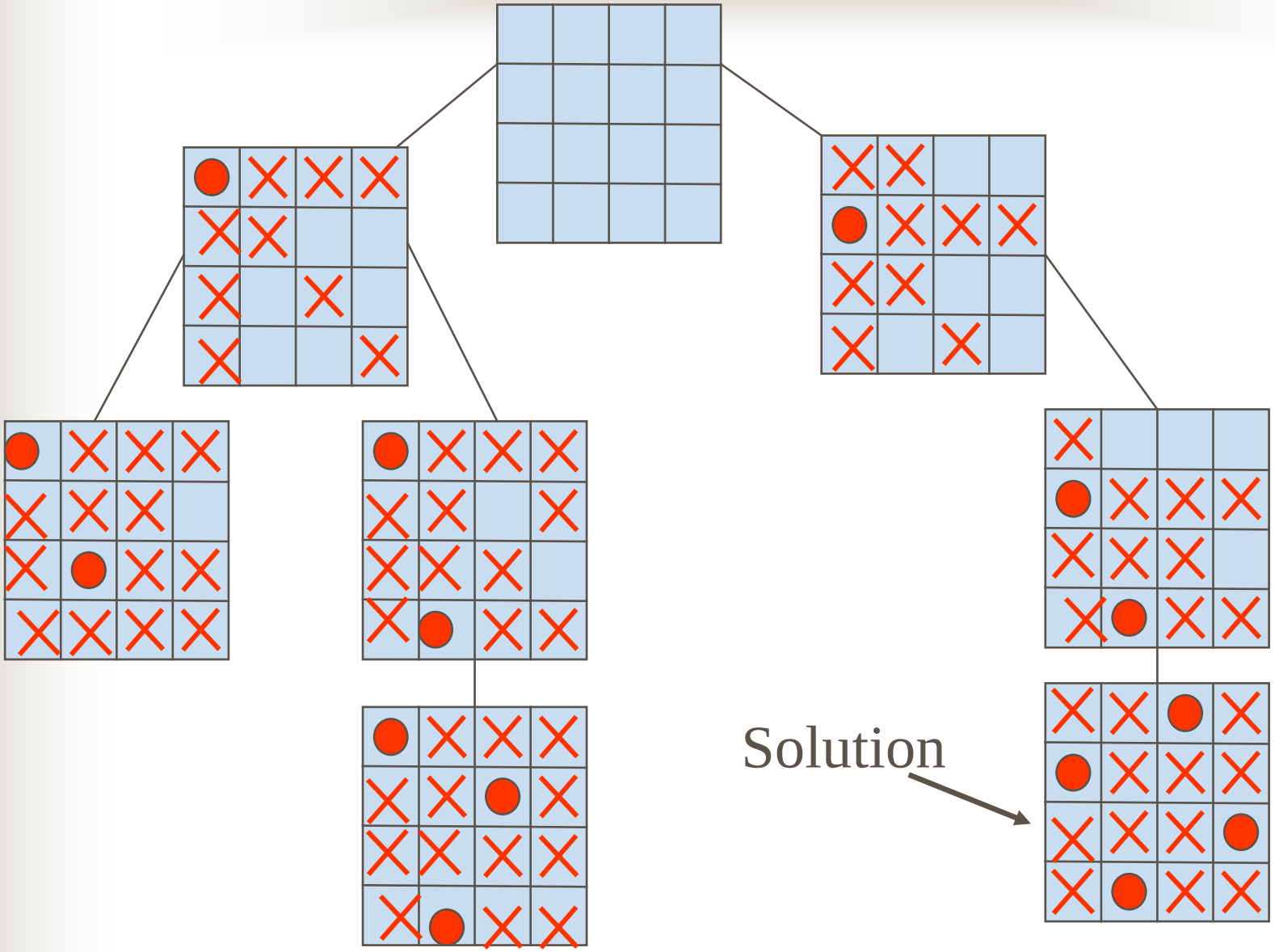
- Idea:
  - Keep possible rest values for unassigned variables
  - Stop if there is no value to assigne for certain variable



Four queen problem with a help of forward checking.



n-queen problem: Place n queens on a board n * n so, that they do not attack each other.

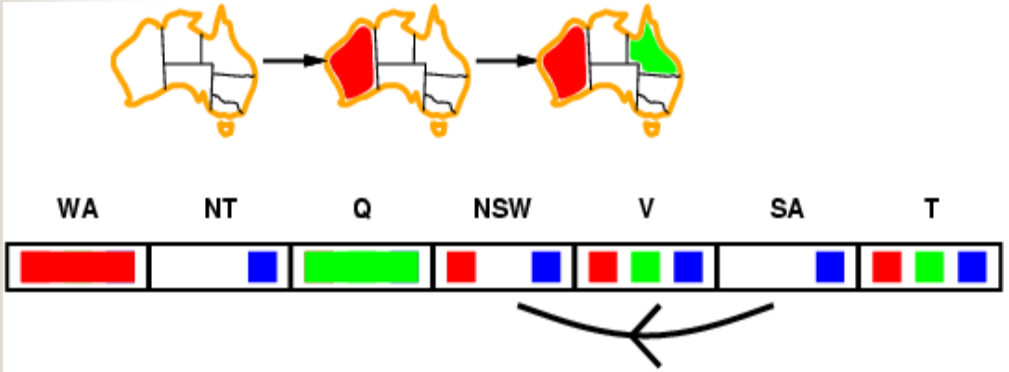

Solution

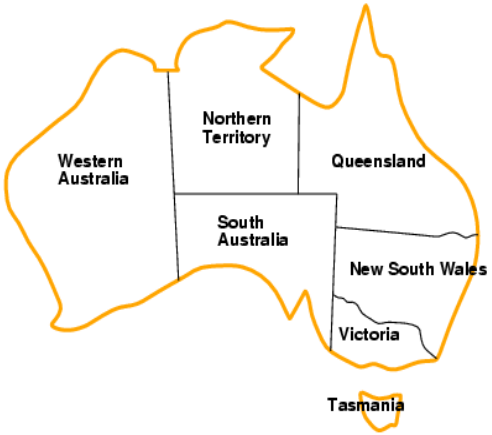
ARC consistency
(MAC method)

-Reveals conflict set sooner then forward checking, but needs more controls in each step.

MAC- maitaining arc consistency

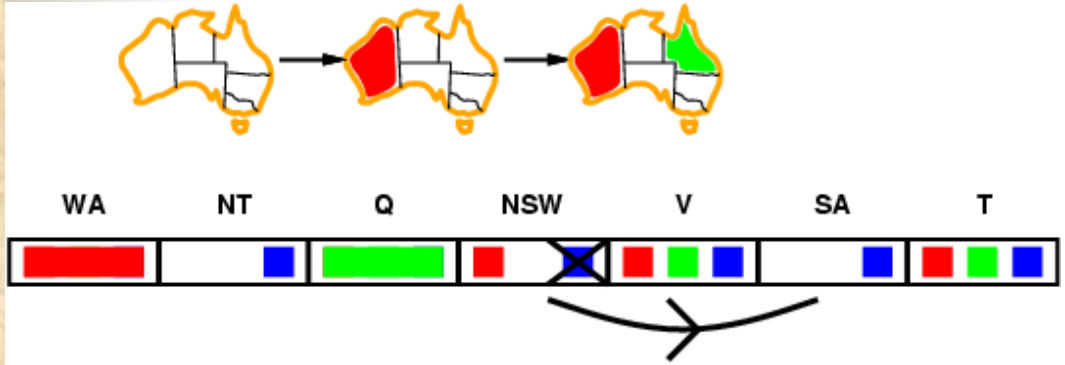
- Principle: For this state NSW and SA have only one option, blue. That means before we assign blue to SA, we remove blue from all variables in constrain with SA. The same with NT.
- $X \rightarrow Y$ is consistent if
  - For all values x from the domain of X exists possible value y in the domain of Y.

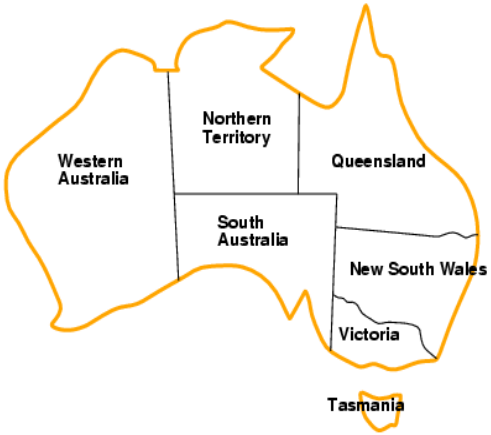




Arc consistency (uzlová konzistencia)

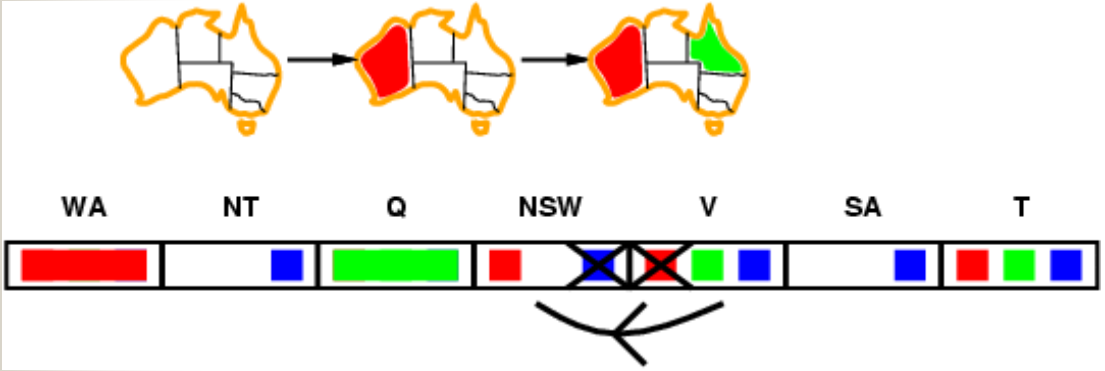
- Principle: For this state NSW and SA have only one option, blue. That means before we assign blue to SA, we remove blue from all variables in constrain with SA. The same with NT.





Arc consistency

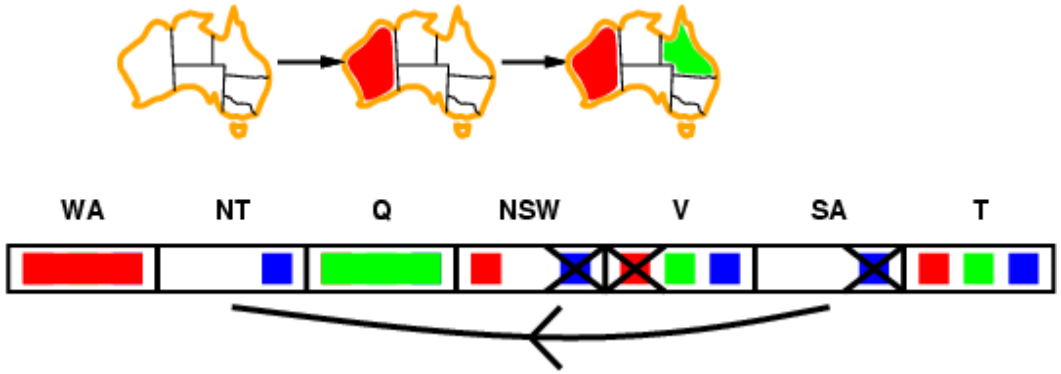
- Principle: For this state NSW and SA have only one option, blue. That means before we assign blue to SA, we remove blue from all variables in constrain with SA. The same with NT.



- If the variable X looses a value, variables in constrain with X are to be controlled.

Arc consistency

- Principle: For this state NSW and SA have only one option, blue. That means before we assign blue to SA, we remove blue from all variables in constrain with SA. The same with NT.



- If *X* looses value all variables in constrain with *X* are to be controlled
- Do this after each assignment.

Alldiff constraint

Alldiff constraint: All variables connected by constraints should have different values (example – Australia problem).

How to detect it?

1. Let us have *m* variables connected by alldiff constraint. These m variables have all together *n* possible different values. If $m > n$ alldif constraint is violated.
2. If domains of certain variables have only one value left in the alldif constrain, we remove this from the domains of other variables. We again controll numbers *m*, *n* as in 1.
3. Repeat from 2 until there are domains with one possibility only.

Example:

NT,SA,Q create alldiff

constraint, because thei are neighbors

inconsistency, three variables, states

have only two different values to use (green, blue)

Atmost constrain

Some set of values should have sum w **at most (**for example, not necessarily sum**).** For example in Sudoku the sum of values at row , column and in the middle sized square is at most 1+2+3+…+8+9=w.

Summary

1. Definition of CSP.
2. Arity of the problem, binary CSP
3. Backtracking
4. Heuristics for CSP
5. Forward checking, alldif constraint

Example to think about.

Consider the following logic puzzle: In five houses, each with a different color, live 5 persons of different nationalities, each of whom prefer a different brand of cigarette, a different drink, and a different pet. Given the following facts, the question to answer is
"Where does the zebra live, and in which house do they drink water?"

The Englishman lives in the red house.
The Spaniard owns the dog.
The Norwegian lives in the first house on the left.
Kools are smoked in the yellow house.
The man who smokes Chesterfields lives in the house next to the man with the fox.
The Norwegian lives next to the blue house.
The Winston smoker owns snails.
The Lucky Strike smoker drinks orange juice.
The Ukrainian drinks tea.
The Japanese smokes Parliaments.
Kools are smoked in the house next to the house where the horse is kept.
Coffee is drunk in the green house.
The Green house is immediately to the right (your right) of the ivory house.
Milk is drunk in the middle house.

Discuss different representations of this problem as a CSP. Why would one prefer one representation over another?

Games

1. One player game
2. Many players game.

One player (agent) game

- It is basically a searching problem
  - Find a sequence of moves to get a required state
  - Example: Rubik cube
  - Solution by searching with an appropriate heuristics

Rubik cube

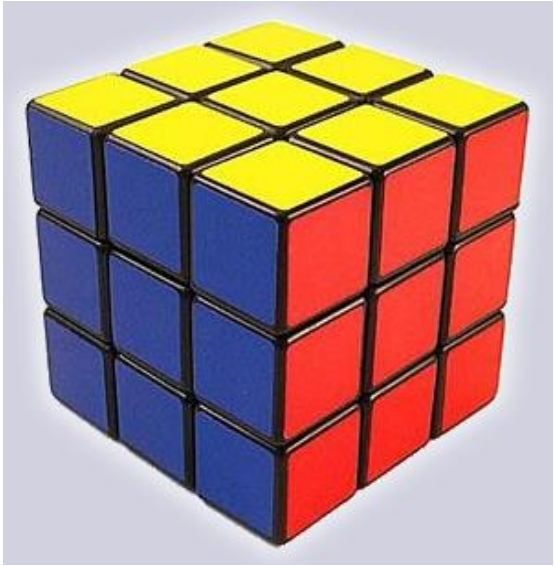
Problem: Find a desirable state, on the picture from an arbitrary initial state.

Partial problems:

-Find a move sequence which exchanges position of two small cubes not violating others.

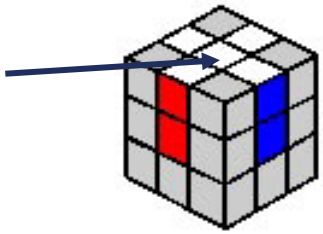
-example: there is known 14 ply sequence how to exchange a position of 2 corner cubes

-It is a difficult searching problem

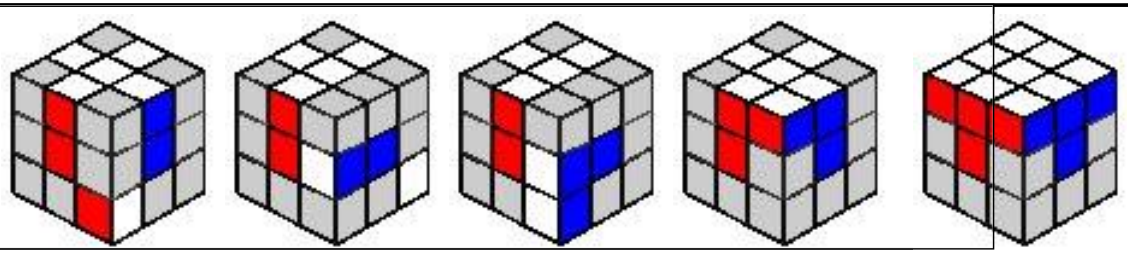


Heuristic steps:

1. Find a cross:



2. Place first corner cubes on the first level, then complete the first level:

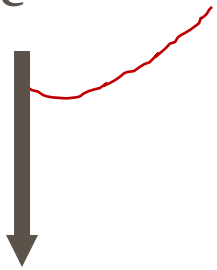


3. Complete T shapes :

Multiplayer game : multiagent environment

Multiagent environment types

1. Agents are cooperative
2. Agents are competitive



game

Game : consists of competitive agents in the multiagent environment.

Simultaneous games

Players make decisions at the same time.

Example: Prisoner's dilemma – described by the payoff matrix

Both prisoners A and B are interrogated at the same time and they are both told:
If you do not defect your partner, and you cooperate with him, the number of years in prison are: if your opponent cooperates too you both get 2 years in prison. If you cooperate, but your opponent defect's you, you get 10 years and he zero. If you both defect you both get five years.

A \ B		Prisoner B: Cooperation	Prisoner B: Defection
Prisoner A	Cooperation	A: 2 years B: 2 years	A: 10 years B: 0 year
Prisoner A	Defection	A: 0 year B: 10 years	A: 5 years B: 5 years

Prisoner's dilemma strategies

1. There is only one interrogation. What is the best strategy?

   To defect.

2. There are more interrogation and you are informed what your opponent did previously, the strategy changes.

   Tit for tat is the best strategy.

Sequential games

The players do not decide simultaneously, but sequentially. First player makes a move and the second player answers the move.

If there are more then two players, they make alternate moves.

Example: chess

These games use evaluation function.

The rest of the lecture is devoted to the sequential games.

Game vs. searching problems

- „Unpredictable" opponent → player has to find an answer to each opponent move
- Time limit → makes a perfect solution difficult, often improbable. Player has to approximate.

What is a game? Summary

Game for us will be: idealized world, in which states are well defined, rules are well defined. In this world two (most simple case) competitive agents fight to gain as much utility (grades) as possible on the cost of the partner.

a) Environment is fully observable, deterministic, multiagent. In our case we have first two (later more) agents.

Episodic ?

b) Actions of two competitive agents alternate and are done with respect to the game rules.

a) Utility values at the end of a game can be equal but with opposite sign. (e.g win +1, loss –1, no loss no win 0 = zero sum games).

Example: chess, tic tac toe, checkers