约束满足问题

约束满足问题 CSP

- 状态元素化表示：一组变量，每个变量有赋值。
- 当每个变量均有赋值，且同时满足所有关于变量的约束时，问题就得到了解决。
- 找到所有变量的一个（或多个）赋值，使所有约束都得到满足。

Constraint Satisfaction Problems

<scope, rel>
scope: 约束中的变量组
rel：变量取值应满足的关系

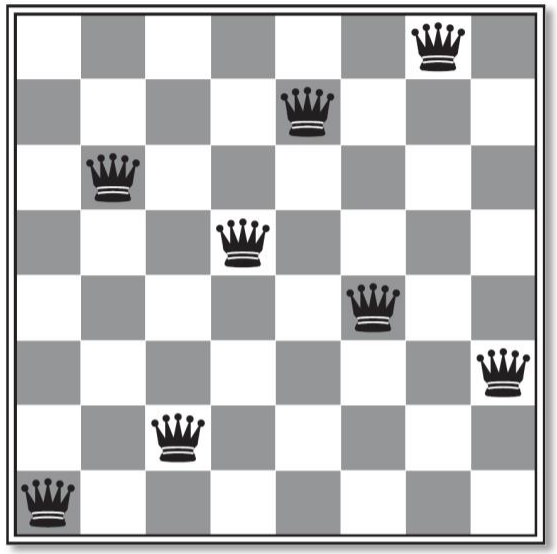
- A set of **variables**: $X = \{x_1, \ldots, x_n\}$
- A set of **domains** for the variables: $\mathcal{D} = \{D_1, \ldots, D_n\}$
- A set of **constraints** over the variables: $\mathcal{C} = \{c_1, \ldots, c_m\}$
- **State**: an **assignment** of values to a subset of X
- A **consistent assignment** is a state that satisfies all constraints
- A **complete assignment**: an assignment of values to all of X
- **Solution**: a **consistent** and **complete** assignment

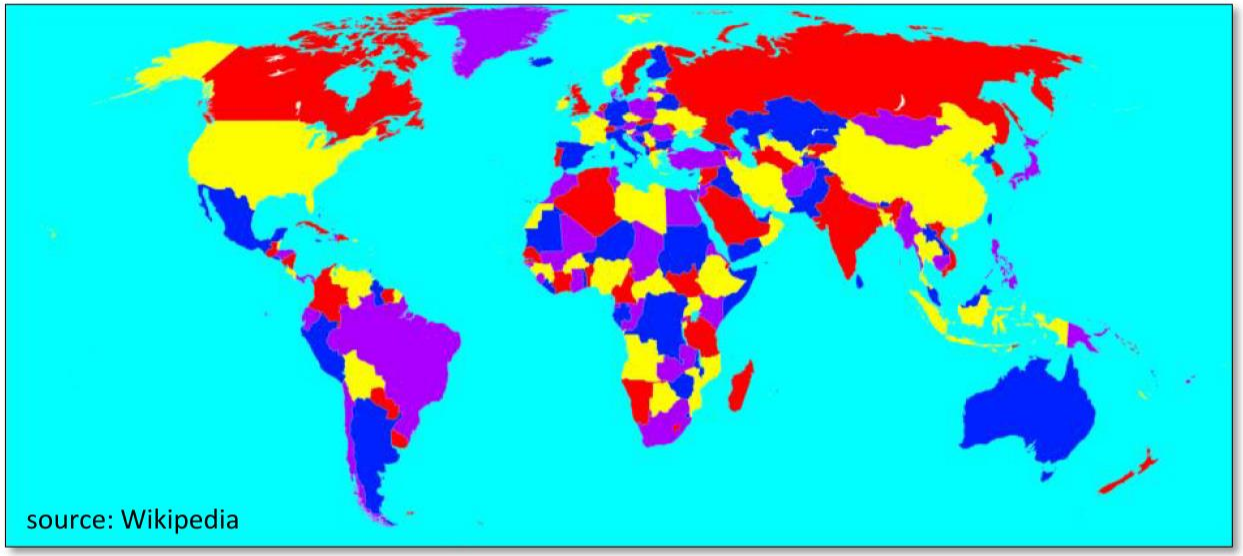
约束满足问题：包含一组变量和一组变量间的约束。
变量集合：X1, X2,…,Xn（变量Xi的值域Di）
约束集合：C1, C2,…,Cn

- 约束：用于描述对象的性质、相互关系、任务要求、目标…
  - ✓一元谓词
  - ✓序关系语言
  - ✓形如x−y>c的方程
  - ✓线性方程和不等式
  - ✓布尔组合
  - ✓代数和三角方程

- 约束表示易于理解、编码和实现

Constraint Satisfaction Problems: Examples





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

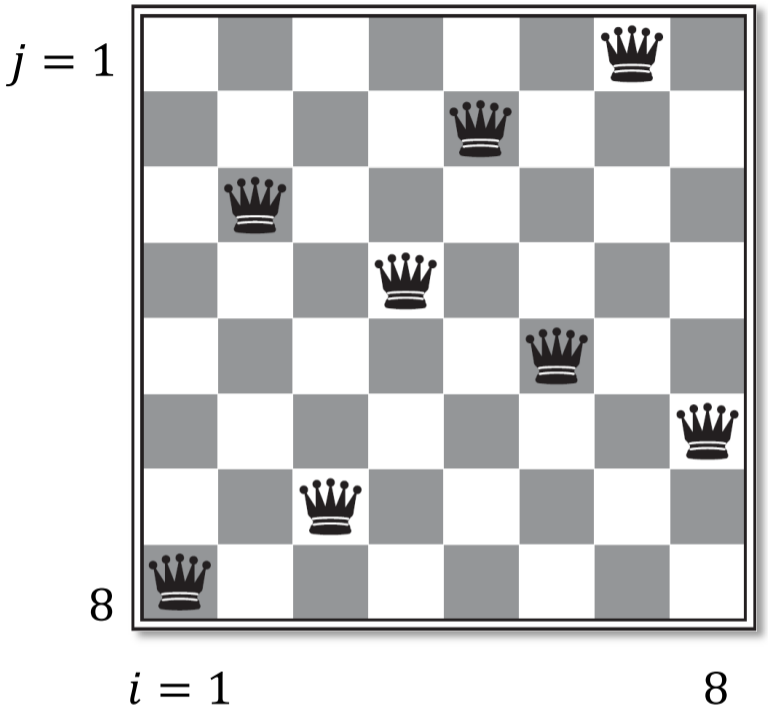
Cryptarithmetic

- Variables: $x_T, x_W, x_O, x_F, x_U, x_R, c_1, c_2$
- Domains: $x_i \in \{0, \dots, 9\}, c_j \in \{0,1\}$
- Constraints
  - $x_O + x_O = x_R + 10c_1$
  - $x_W + x_W + c_1 = x_U + 10c_2$
  - $x_T + x_T + c_2 = x_O + 10x_F$
  - $x_i \neq x_j$
  - $x_T \neq 0, x_F \neq 0$

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$

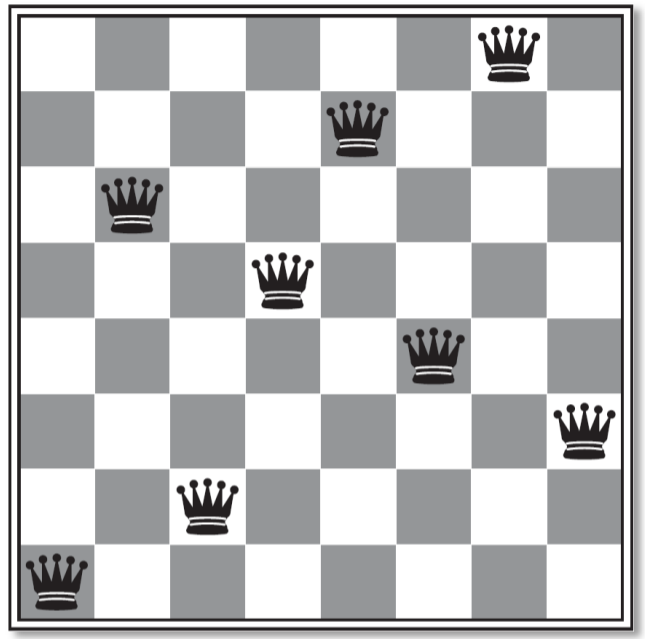
Eight-Queens

- ⇨Variables: $X = \{x_{ij}\}, 1 \leq i, j \leq 8$
- ⇨Domains: $\mathcal{D} = \{D_{ij}\}, D_{ij} = \{0,1\}$
- ⇨Constraints
  - ⇨Total number of queens: $\sum x_{ij} = 8$
  - ⇨No two queens should attack
    - ⇨ Rows and columns: $\sum_i x_{ij} \leq 1, \sum_j x_{ij} \leq 1$
    - ⇨ Diagonals: $\sum_{k=1}^{8} x_{k,k} \leq 1, \sum_{k=2}^{8} x_{k+1,k} \leq 1, \ldots$
- ⇨Is this **unique**?



Another CSP Formulation for Eight-Queens

- Variables: $X = \{x_1, \ldots, x_8\}$
- Domains: $\mathcal{D} = \{D_1, \ldots, D_8\}, D_i = \{1, \ldots, 8\}$
- Constraints
  - No two queens on same row: $x_i \neq x_j$
  - No diagonal attack: $|x_i - x_j| \neq |i - j|$
- One can be very creative in formulating CSPs for the same problem
- Can you provide another CSP formulation for 8-queens?
- Some additional CSP examples an be found at http://www.csplib.org



$i = 1$ $\qquad\qquad$ 8

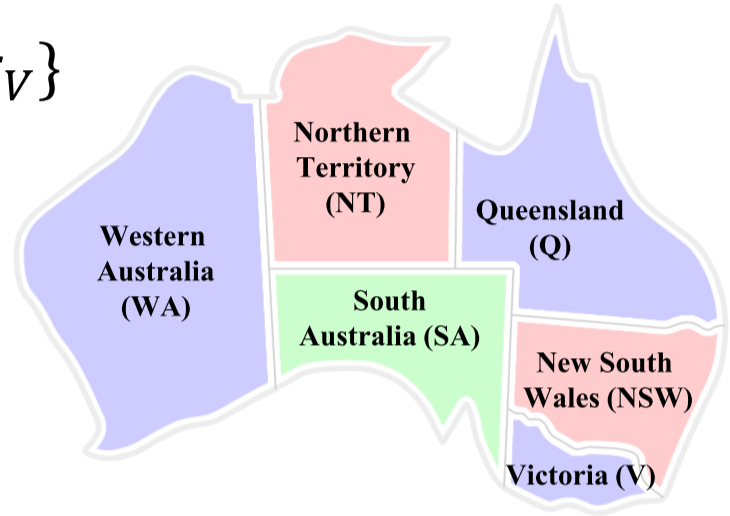
The Map Coloring Problem

⇨Task: color a map (say, the contiguous part of Australia)

⇨Variables: $X = \{x_{WA}, x_{NT}, x_{SA}, x_Q, x_{NSW}, x_V\}$

⇨Domains: $D_i = \{red, green, blue\}$

⇨Constraints: $x_i \neq x_j$ if i, j are adjacent



⇨Easy if we use many colors, but

- ⇨Too many colors will be difficult to tell apart
- ⇨Printing many colors is more costly

⇨Interesting fact about map coloring

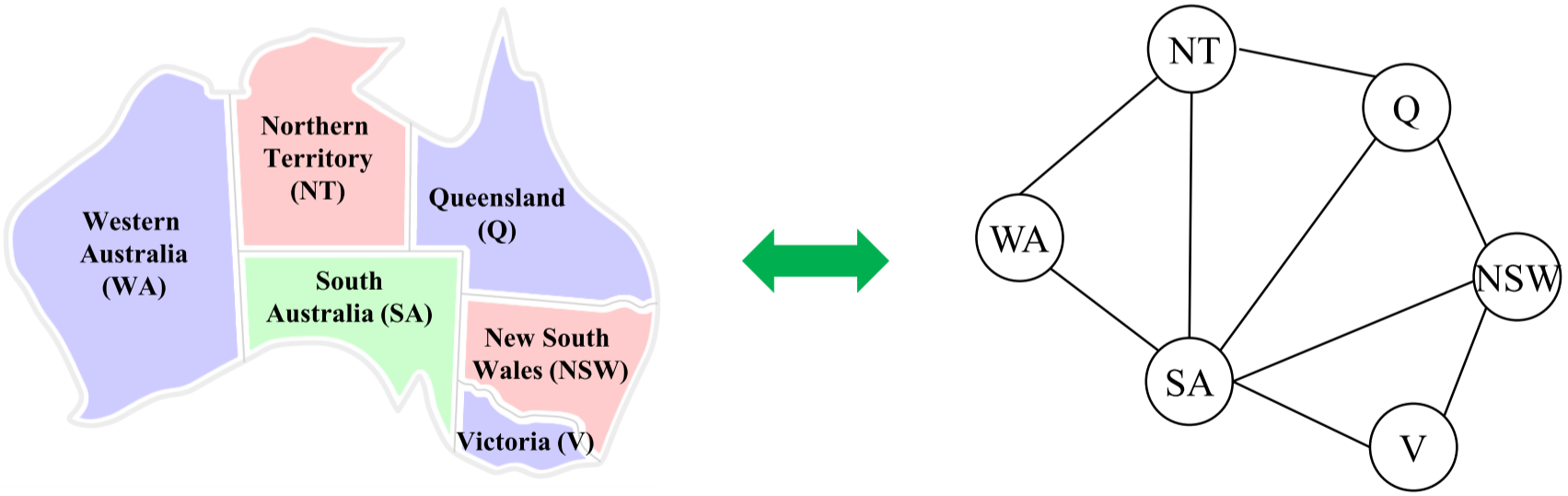
Four-Color Theorem. Given a separation of the plane into contiguous regions, no more than four colors are need to color the regions so that no two adjacent regions share the same color.

Appel & Haken, 1976

- ⇨The proof is done by computer

Representing Adjacency Using Planar Graph

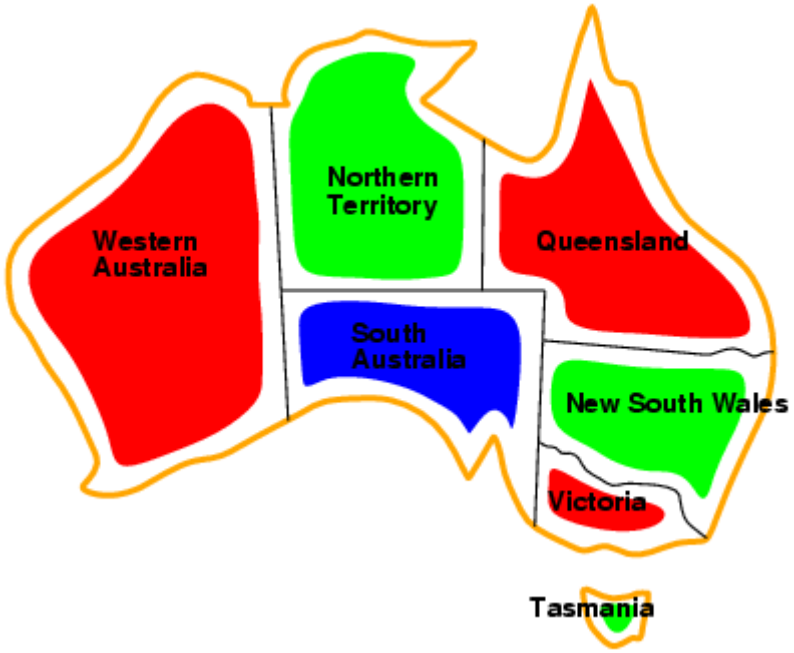
⇨The **adjacency relationship** of regions in map coloring may be represented using a **planar graph** – a graph that can be drawn in the plane without any two edges crossing each other



地图着色问题

变量集合： *WA, NT, Q, NSW, V, SA, T*
值域: *Di* = {red, green, blue}
约束：相邻区域颜色不同
例如, WA ≠ NT, 或 (WA, NT) 组合
{(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)}

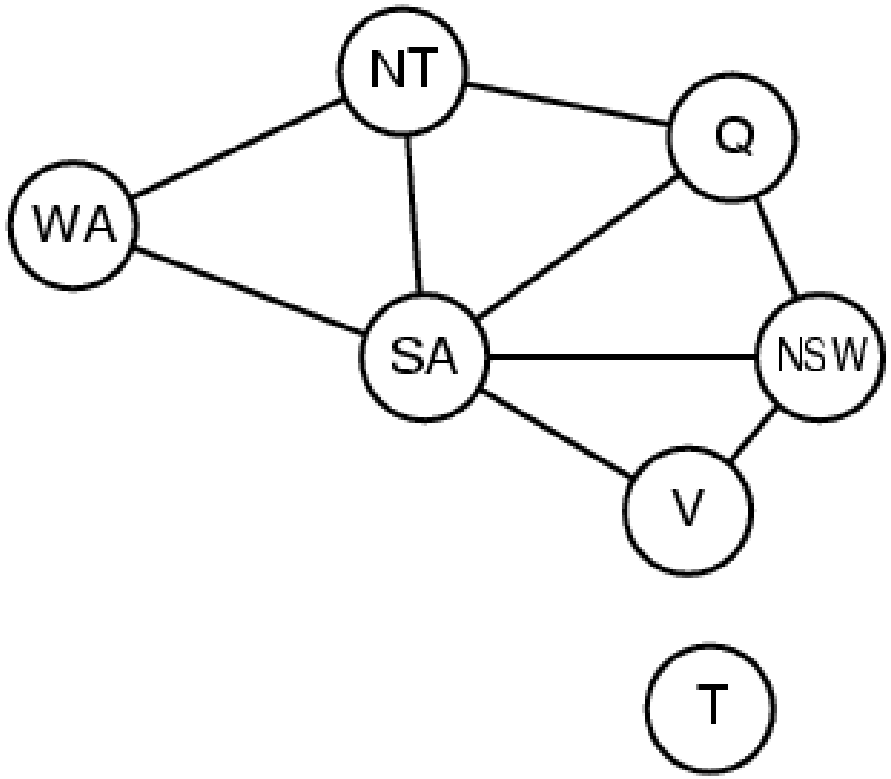
地图着色问题



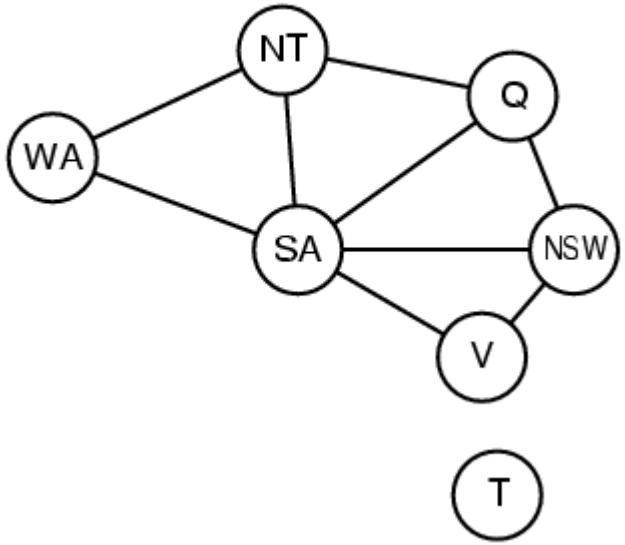
有很多可能的解，例如, WA = red, NT = green,Q = red,NSW = green,V = red,
SA = blue,T = green

约束图

- 节点：变量；弧：约束



CSP问题的特点



CSP问题的增量形式化：

初始状态：空的赋值{}；

后继函数：可以给任何未赋值的变量赋一值，且和先前赋值的变量不冲突

目标测试：检验当前赋值是否完全

路径耗散：每一步耗散为常数

完全状态形式化：

每个状态是一个满足或不满足约束条件的完全赋值。

CSP问题的特点

变量类型：

离散型

连续型：线性规划问题

值域：

有限值域

无限值域：例如工作安排问题

使用约束语言描述约束

例如：$StartJob1 + 5 \leq StartJob3$

约束类型：

一元约束：只限制单个变量的取值，例如：$SA \neq green$

二元约束：和两个变量有关，例如：$SA \neq WA$

高阶约束：涉及三个或更多变量

可表示为约束图。

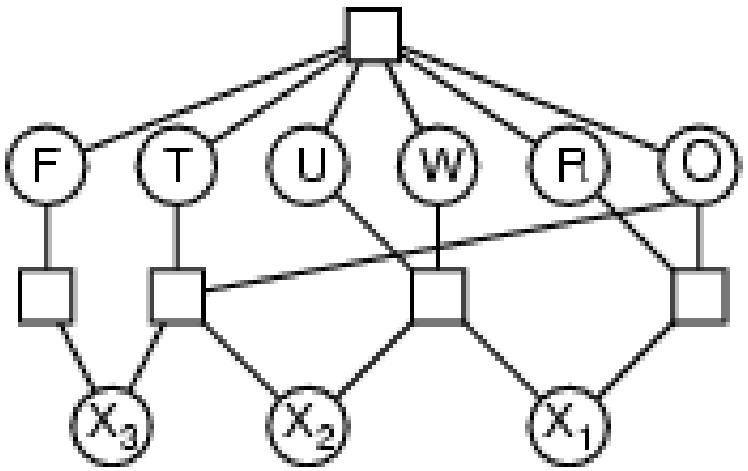
- 例：密码算术
- 可用超约束图表示。

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

变量：F T U W R O X1 X2 X3

值域：{0,1,2,3,4,5,6,7,8,9}

约束：*Alldiff (F,T,U,W,R,O)*



- *O + O = R + 10 ·X1*
- *X1 + W + W = U + 10 ·X2*
- *X2 + T + T = O + 10 ·X3*
- *X3 = F*

高阶的、有限值域的约束可简化为一个二元约束集合：F ≠T， F≠U…

CSP问题的特点

- 绝对约束: 任何违反规则的都排除在解之外
- 偏好约束：指出哪些解是更偏好的

部分赋值CSP问题的回溯搜索

CSP问题的每一个解必须有一个完全赋值，如有n个变量，解的深度为n搜索树则扩展到n.

回溯搜索：深度优先搜索，一次为一个变量赋值，当没有合法的值赋给某变量时就回溯。

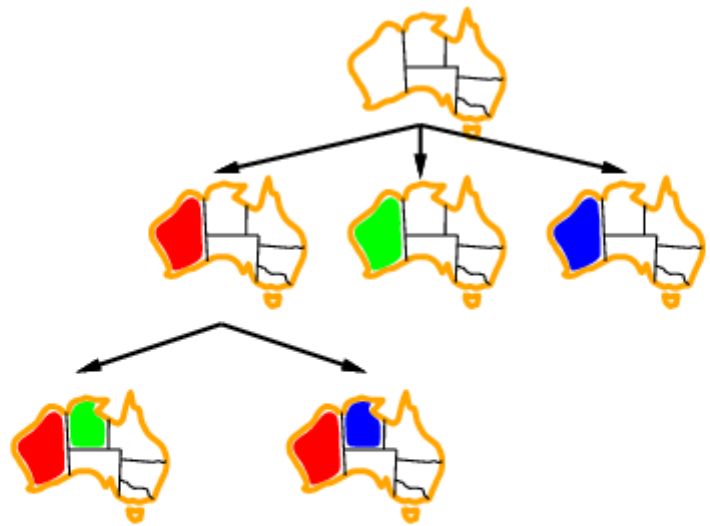
1、为一个新生成的变量选择赋值；

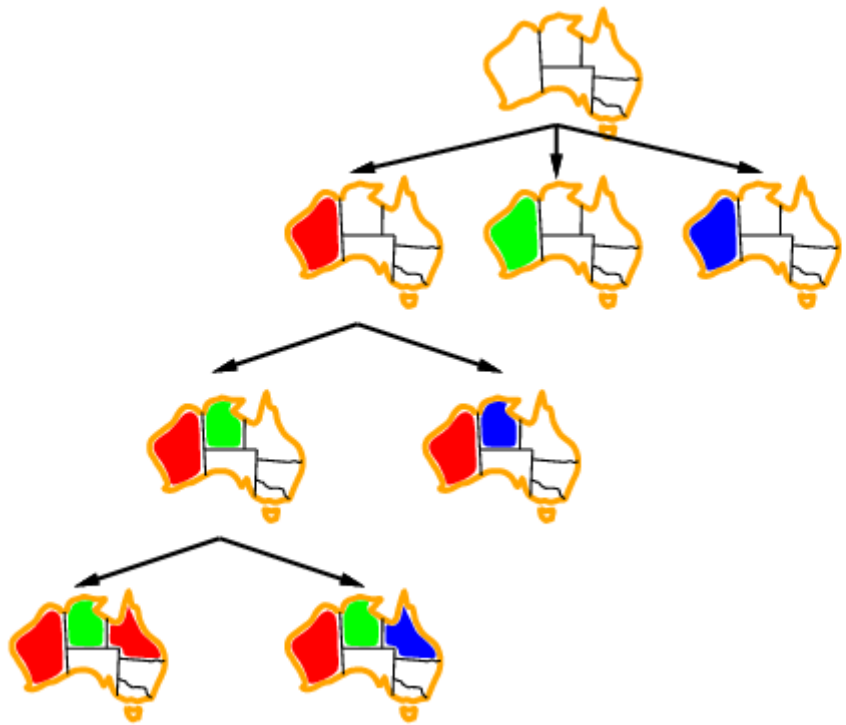
2、比较，合理吗？

3、不合理，回溯

不断选择未赋值变量，轮流尝试变量值域中的每一个值，试图找到一个解。

```
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
```





CSP问题的回溯搜索

1、下一步选哪个变量，按什么顺序尝试它的值；

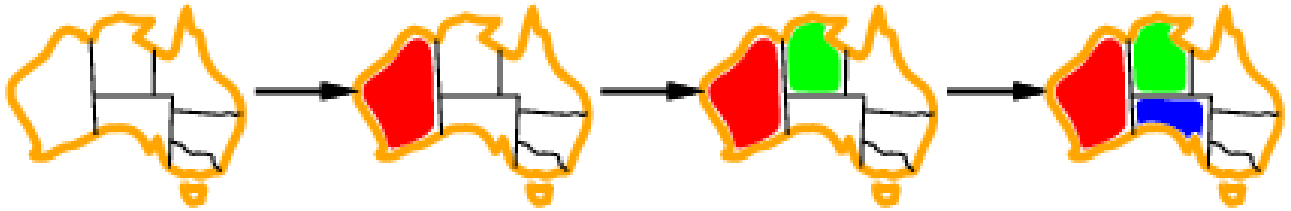
2、当前变量与未赋值变量的关系；

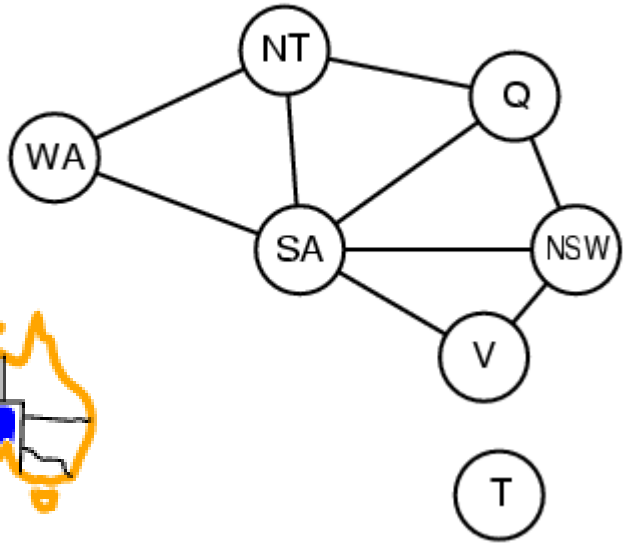
3、如何避免失败，即当一条路径失败后，搜索后面的路径如何避免这种失败。避免重复同样的失败（弧相容）

变量选择和取值顺序

选择合法取值最少的变量
最少剩余值(minimum remaining values,MRV)启发式
最受约束变量(Most constrained variable)启发式
失败优先启发式

有效剪枝

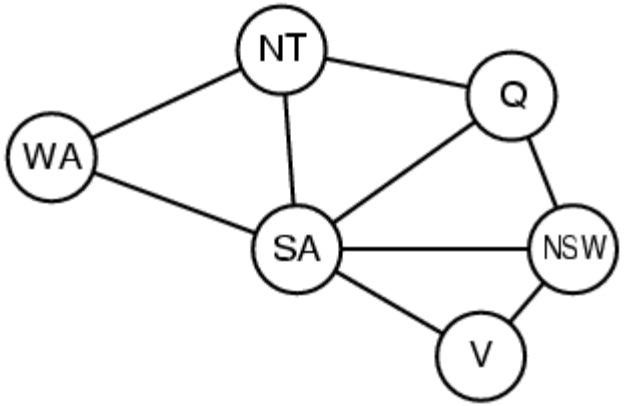


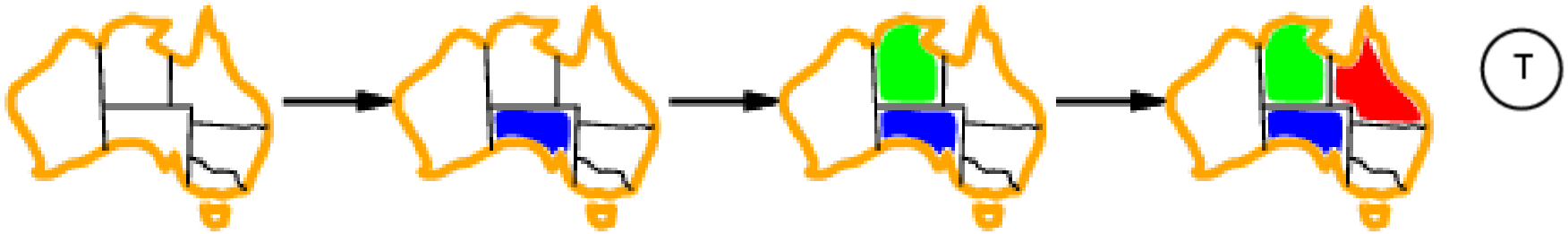


变量选择和取值顺序

- 度启发式：选择涉及对其他未赋值变量的约束数最大的变量来降低未来选择的分支因子。

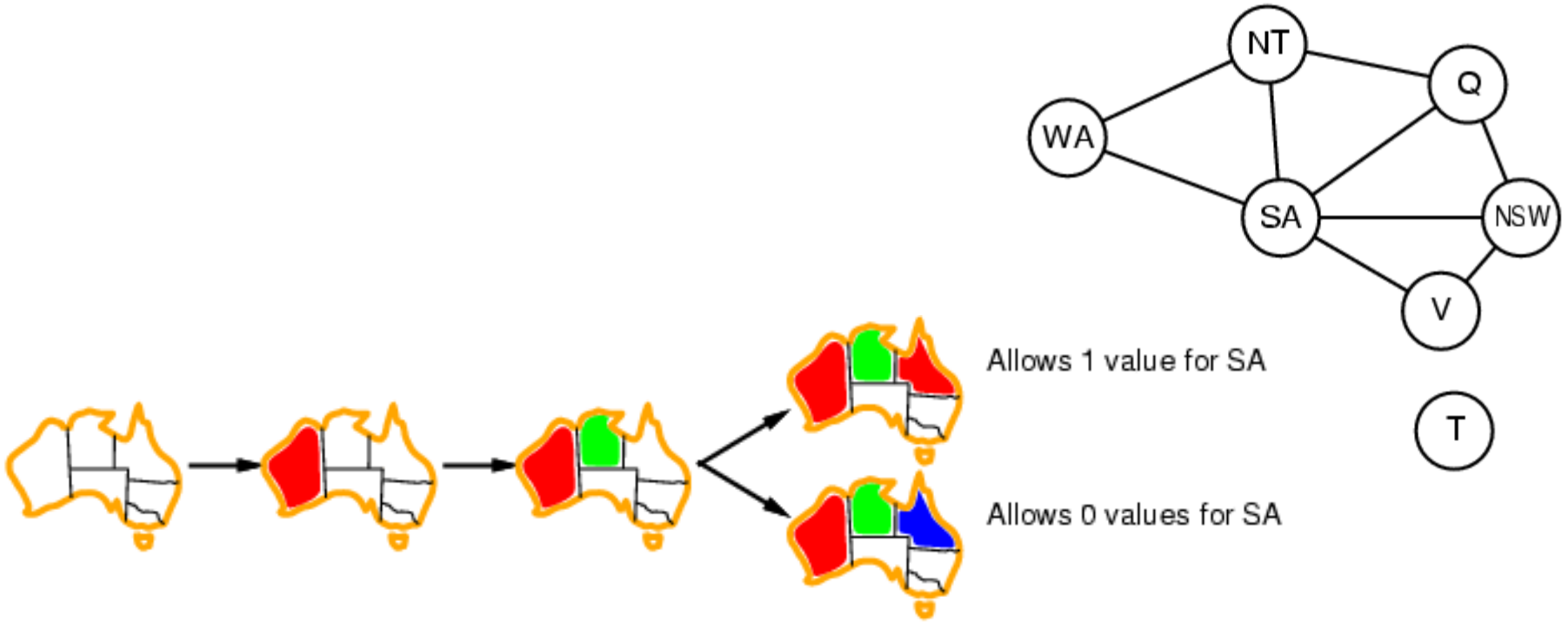
SA的度为5， T的度为0





变量选择和取值顺序

- 最少约束值启发式：优先选择在约束图中排除邻居变量的可选值最少的。这样能给剩下的变量赋值留下最大的灵活性。

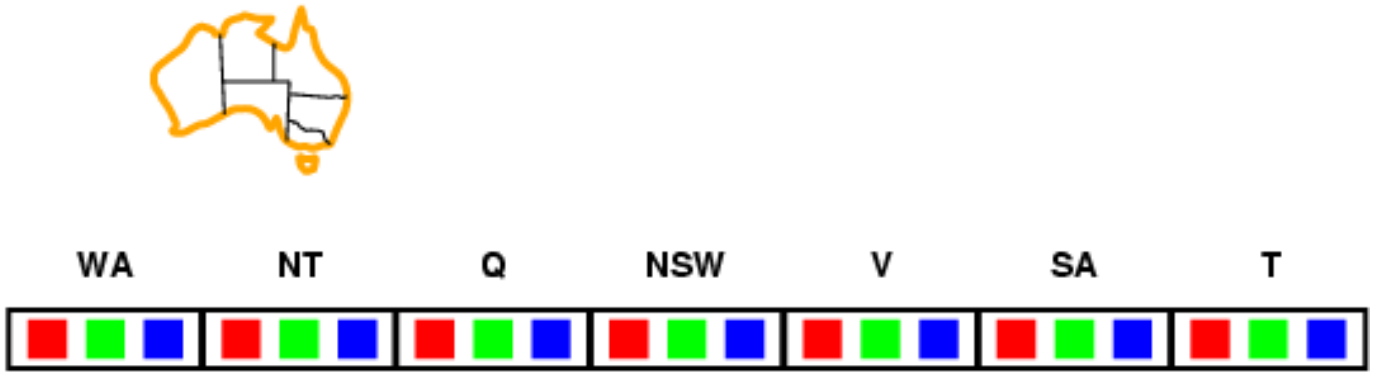


减小搜索空间：

- 1、前向检验
- 2、约束传播
- 3、处理特殊约束
- 4、智能回溯

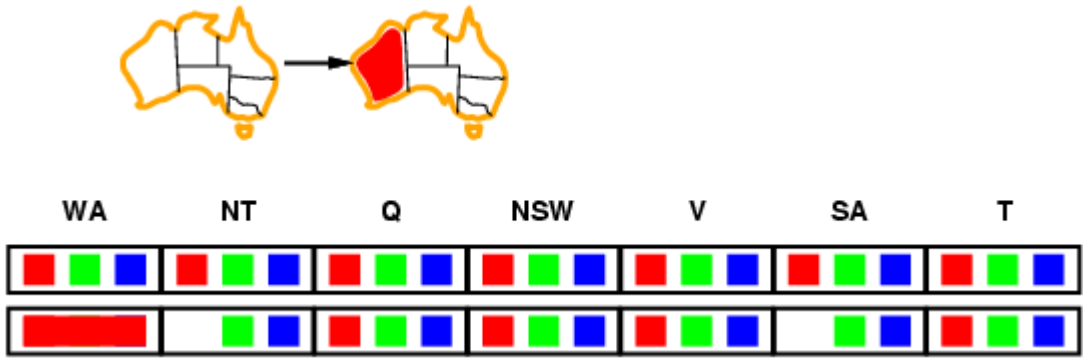
前向检验

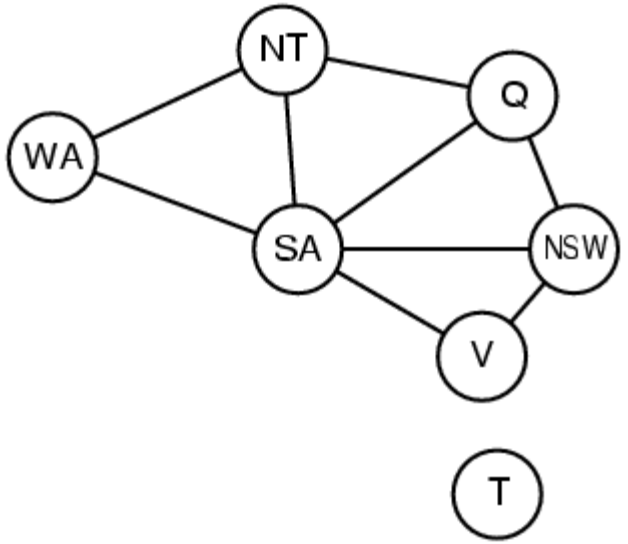
- 当变量X被赋值，则对每个与X相连的未赋值变量Y进行考察，从Y的值域中删去与X矛盾的值。



前向检验

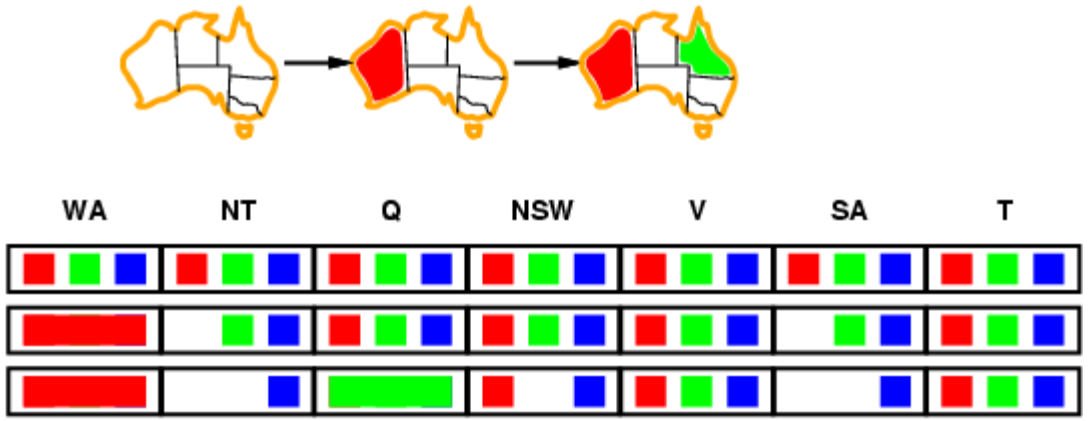
- 当变量X被赋值，则对每个与X相连的未赋值变量Y进行考察，从Y的值域中删去与X矛盾的值。

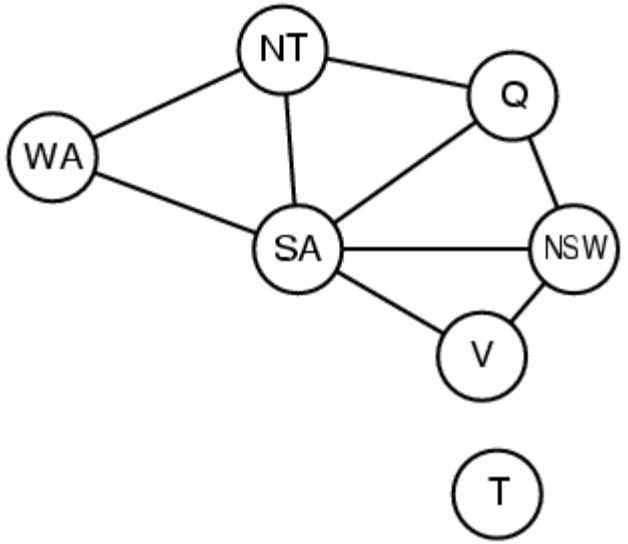




前向检验

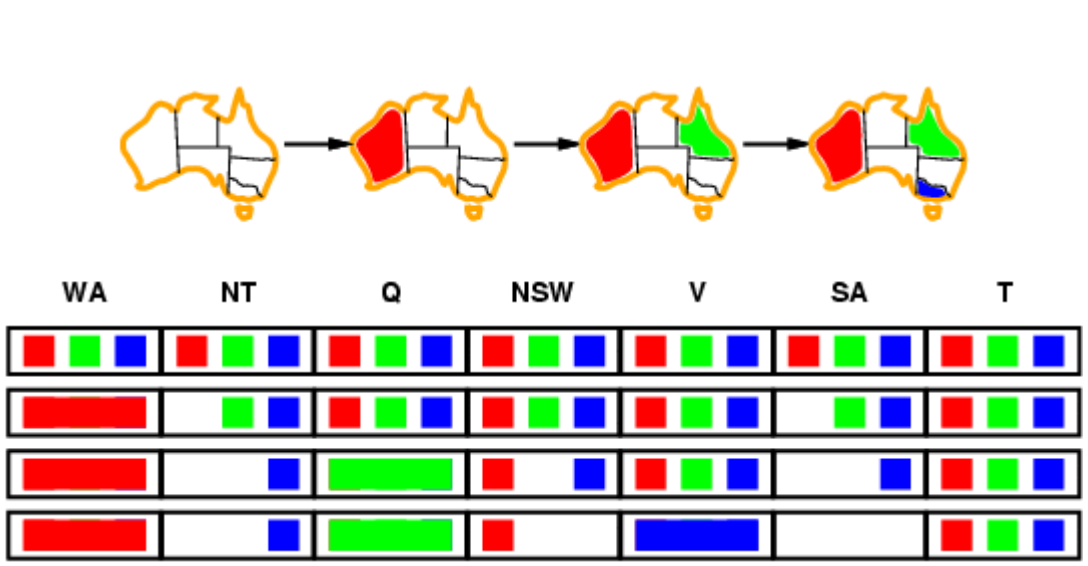
- 当变量X被赋值，则对每个与X相连的未赋值变量Y进行考察，从Y的值域中删去与X矛盾的值。

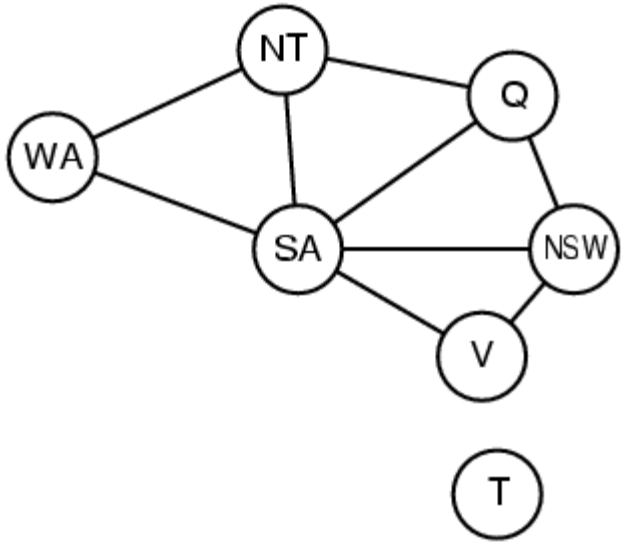




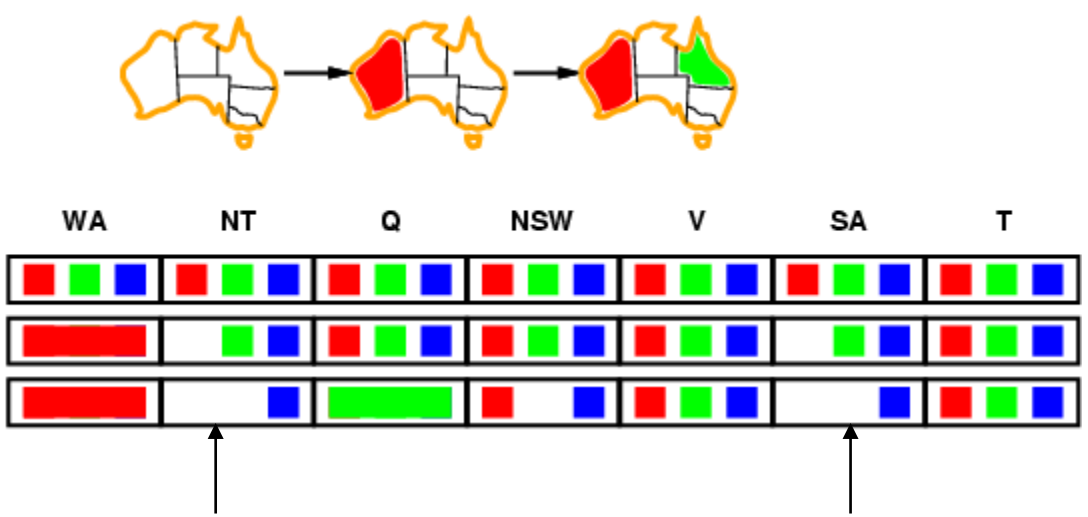
前向检验

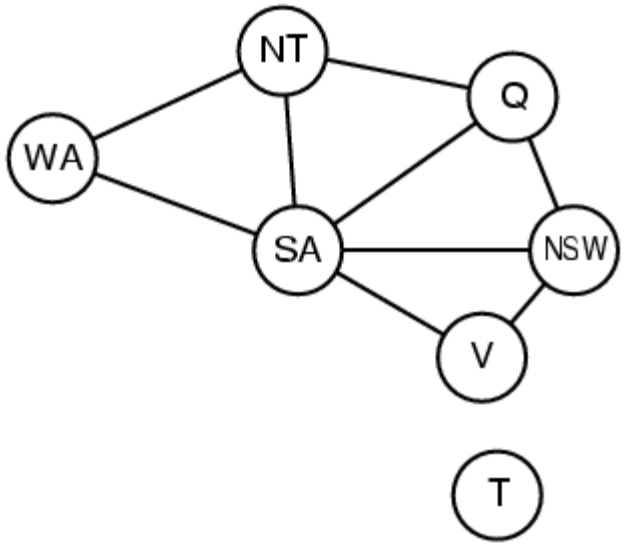
- 当变量X被赋值，则对每个与X相连的未赋值变量Y进行考察，从Y的值域中删去与X矛盾的值。





约束传播

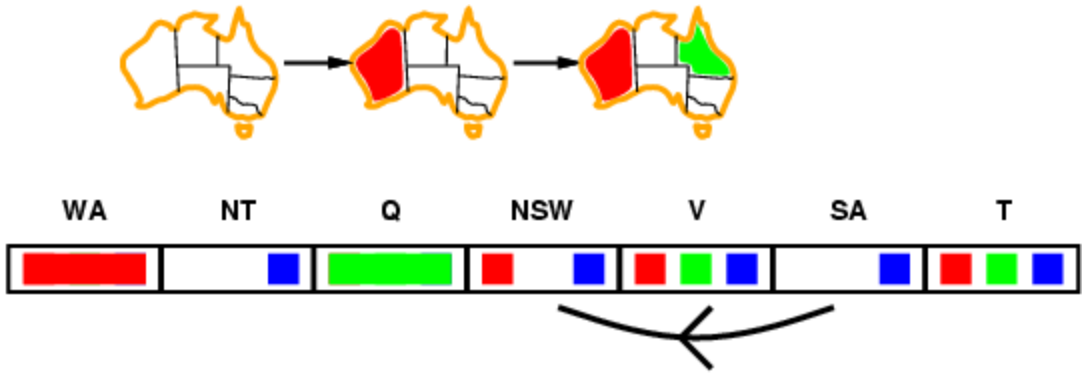


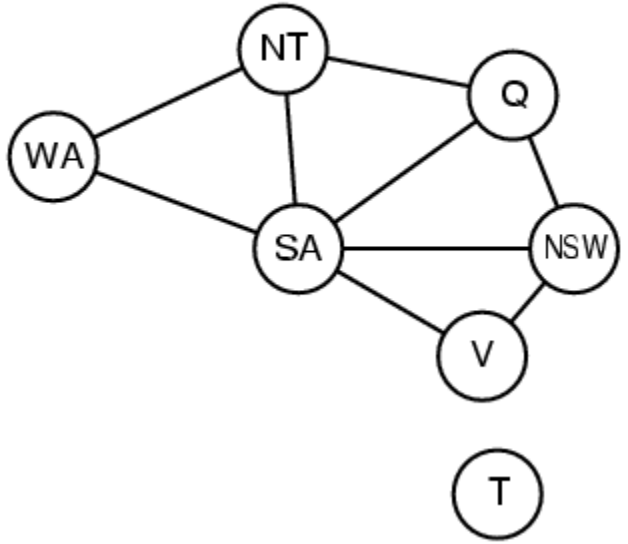


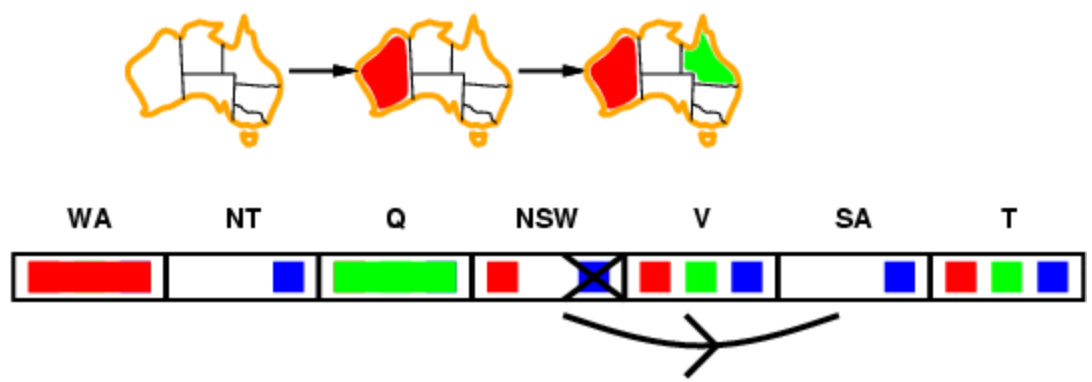
约束传播：将一个变量的约束内容传播到其他变量。

弧相容

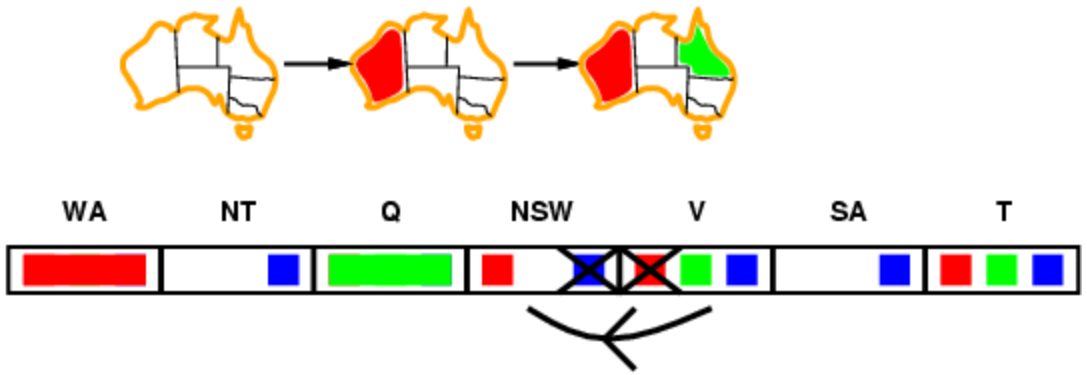
- 有向弧：约束图中连接两个变量
- 弧相容：如果对于变量X的每个取值x，变量Y都有某个取值能和x保持相容，则连接X->Y的弧是相容的。

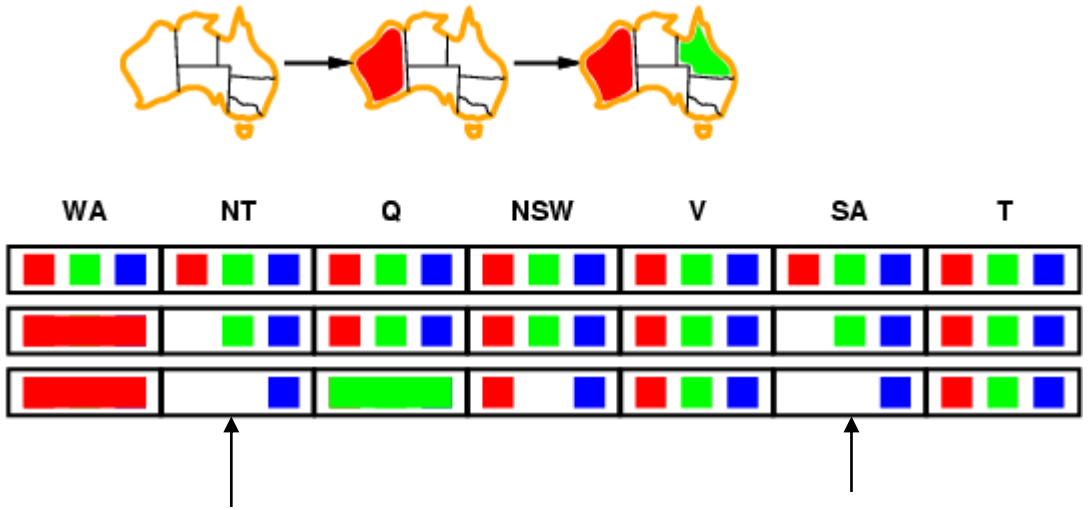


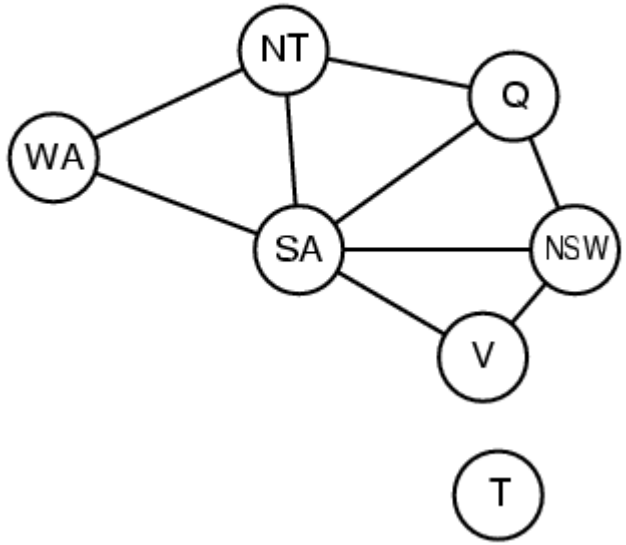



WA
NT
Q
NSW
V
SA
T

- 当X的值有删除，X的邻居需重新检验相容性。







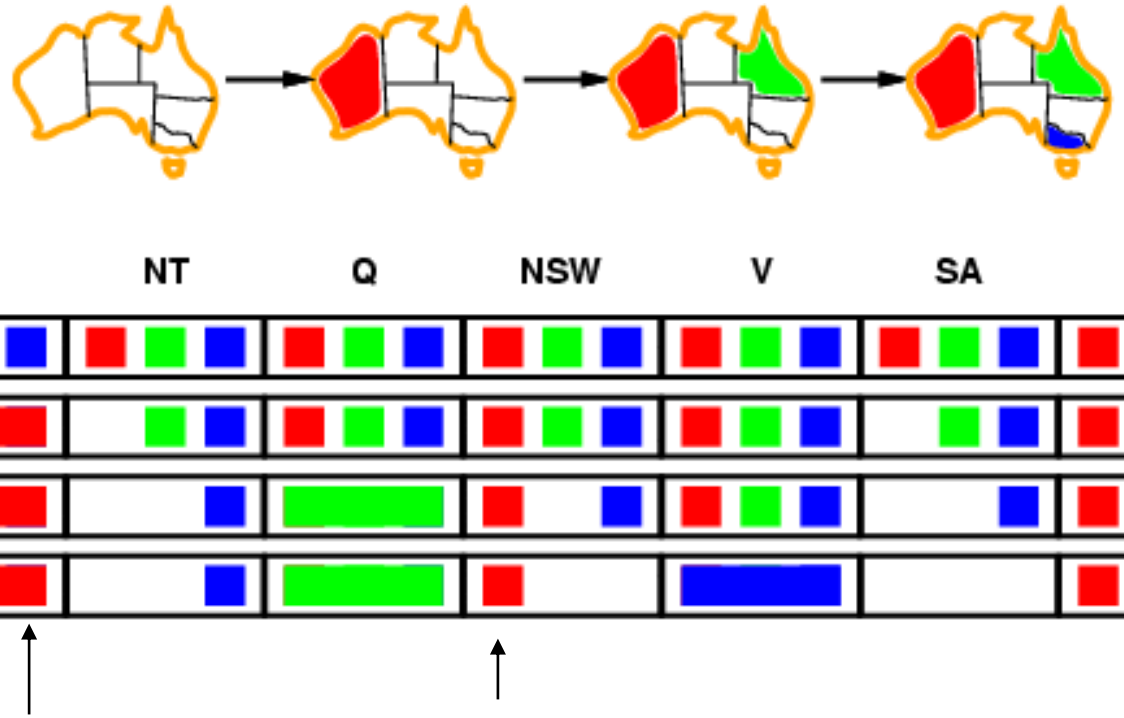
应用弧相容能够更早检测到前向检验不能发现的矛盾。
可在搜索过程中每次赋值后作传播约束，保持弧相容，
即从变量值域中删除某值以消除弧不相容。

```
function AC-3( csp) returns the CSP, possibly with reduced domains
  inputs: csp, a binary CSP with variables {X_1, X_2, ..., X_n}
  local variables: queue, a queue of arcs, initially all the arcs in csp

  while queue is not empty do
    (X_i, X_j) ← REMOVE-FIRST(queue)
    if RM-INCONSISTENT-VALUES(X_i, X_j) then
      for each X_k in NEIGHBORS[X_i] do
        add (X_k, X_i) to queue

function RM-INCONSISTENT-VALUES( X_i, X_j) returns true iff remove a value
  removed ← false
  for each x in DOMAIN[X_i] do
    if no value y in DOMAIN[X_j] allows (x,y) to satisfy constraint(X_i, X_j)
      then delete x from DOMAIN[X_i]; removed ← true
  return removed
```

Mackworth 1977

AC-3：用一队列记录需检验相容性的弧(Xi,Xj)

若Xi的值要删除，则每个指向Xi的弧必须重新插入队列检验。

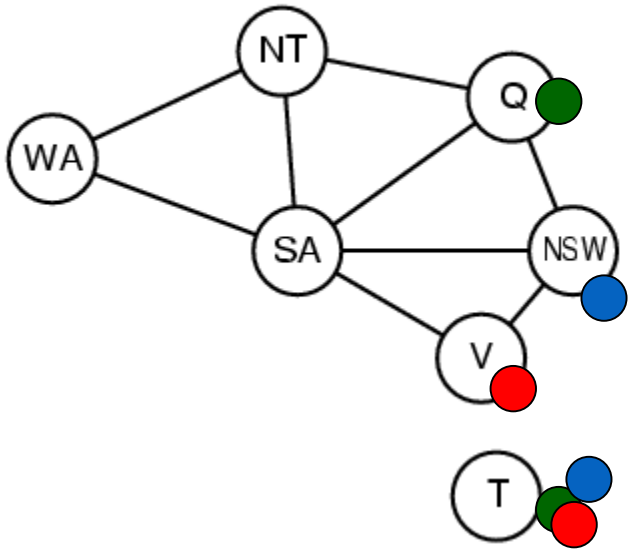


K相容：如果对于任何k-1个变量的相容赋值，第k个变量总能被赋予一个和前k-1个变量相容的值，则该CSP问题是k相容的。

强K相容：k相容，k-1相容，…，1相容。

N个节点若是强N相容的，则不需要回溯就能求解该CSP问题。

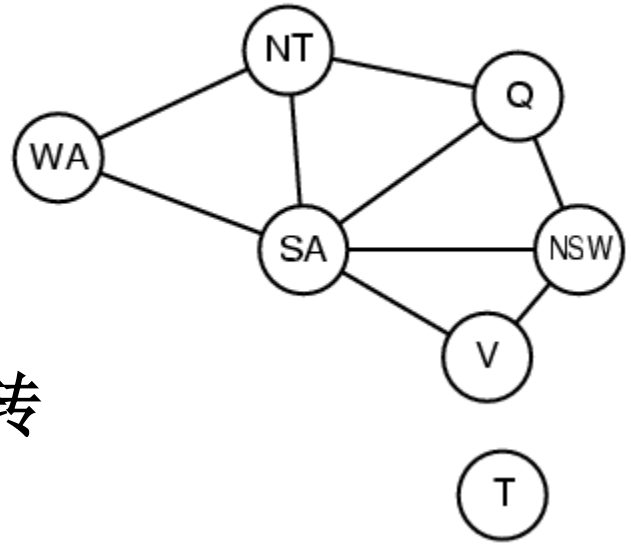
- 时序回溯：当一个分支搜索失败就倒退回前一个变量并尝试另一个值。重新访问的是时间最近的决策点。



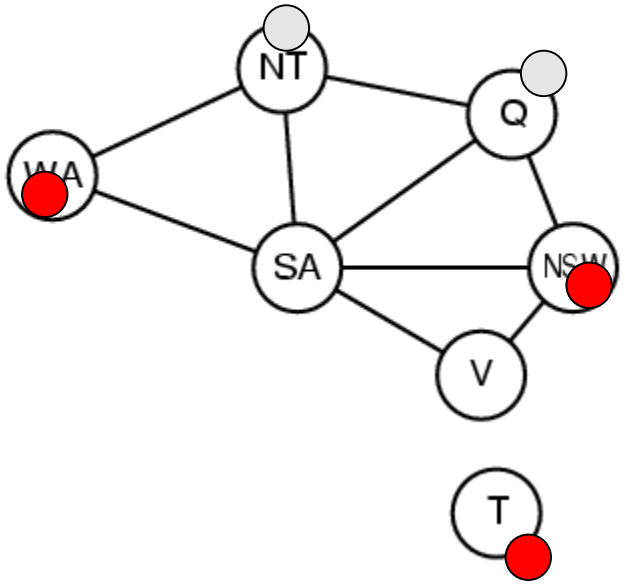
智能回溯：向后看

- 倒退回导致失败的变量集合（冲突集）中的一个变量。

变量X的冲突集是通过约束和X相连接的先前已赋值变量的集合。



冲突指导的后向跳转：
设Xj是当前变量，其冲突集conf(Xj)
当Xj的每个可能取值都失败，后向跳转到conf(Xj)中最近一个变量Xi，并置
conf(Xi) ← conf(Xi) ∪ conf(Xj)-{Xi}

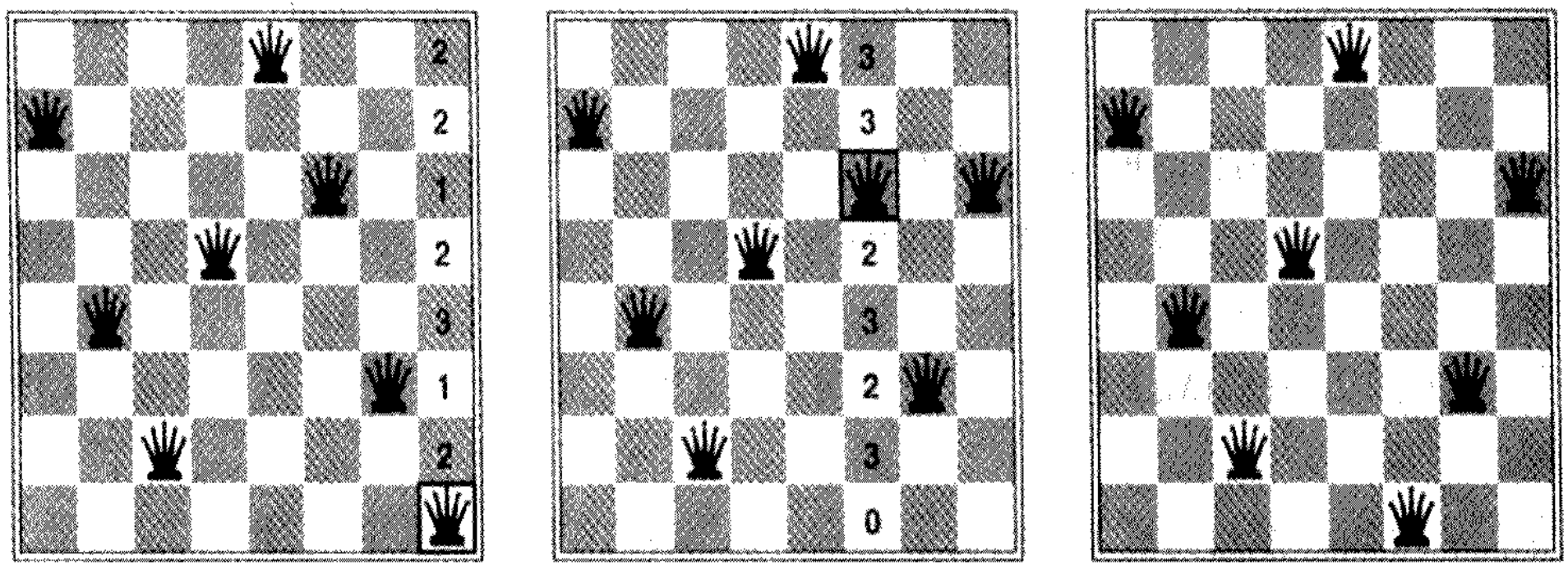


SA失败回溯到Q，Q的冲突集此时为{NT,NSW,WA}
再回溯到NT，NT的冲突集此时为{NSW,WA}
回溯到NSW

完全赋值CSP问题的局部搜索

- 局部搜索算法可求解CSP问题，其中CSP问题使用完全状态形式化：初始状态每个变量都赋值，后续函数一次改变一个变量取值。

最小冲突启发式:选择会造成和其他变量的冲突最小的值。



CSP 的实现

基本过程：

1、为问题选择适当的状态及操作形式化描述方法;

2、从某个初始状态出发每次使用一个操作;

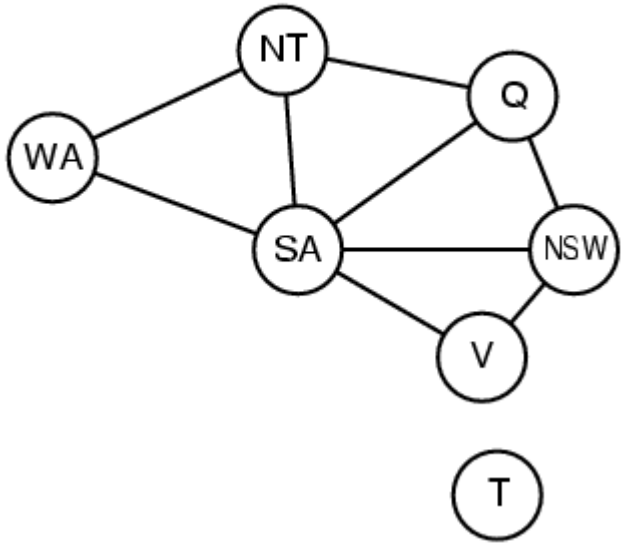
3、递增建立操作序列;

4、一直到达到目标。

问题的结构

将CSP问题分解成独立的

约束图中寻找连通域。



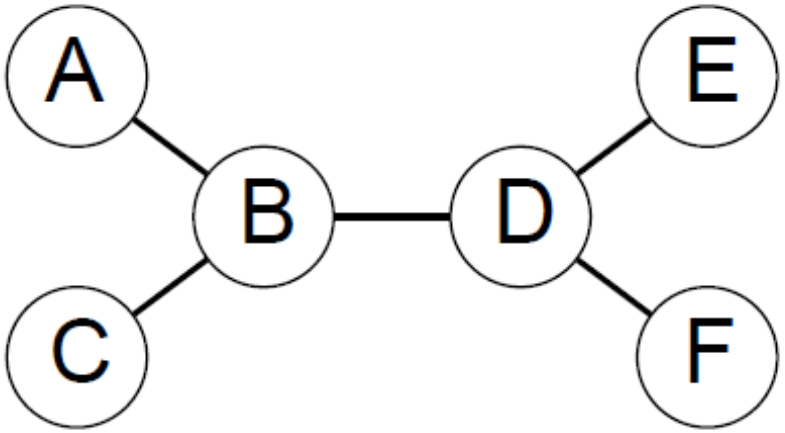
E.g. n=80, d=2

2^{80}节点, 10百万节点/s，需4十亿年

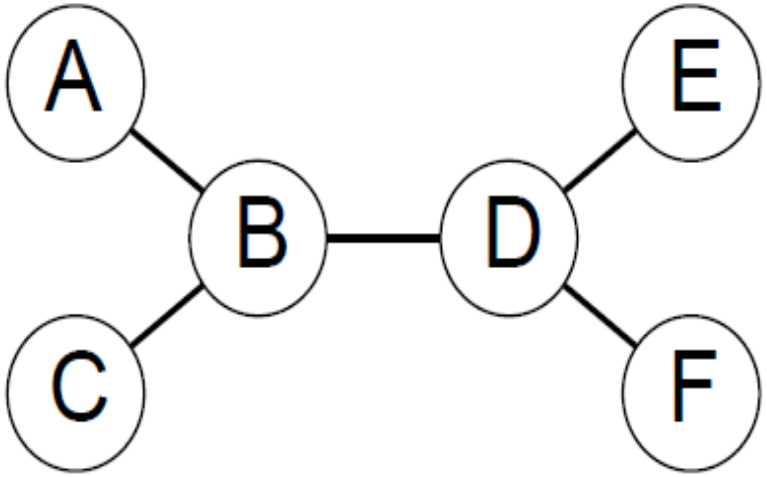
$4*2^{20}$节点，需0.4s

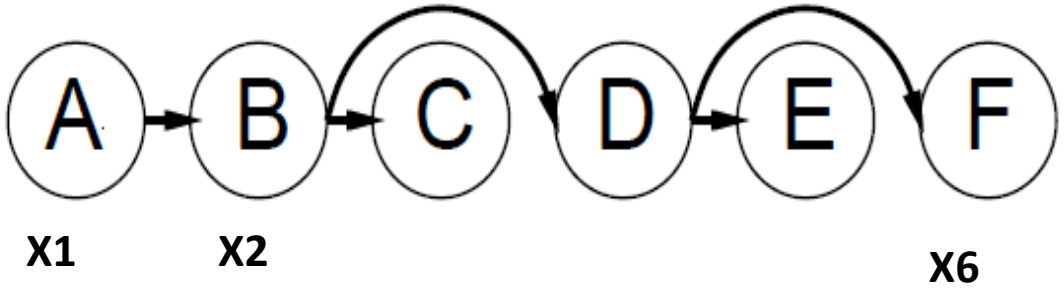
树状结构的CSP问题：

任何两个变量最多通过一条路径连通。



树状结构的CSP问题可以在变量个数的线性时间内求解.





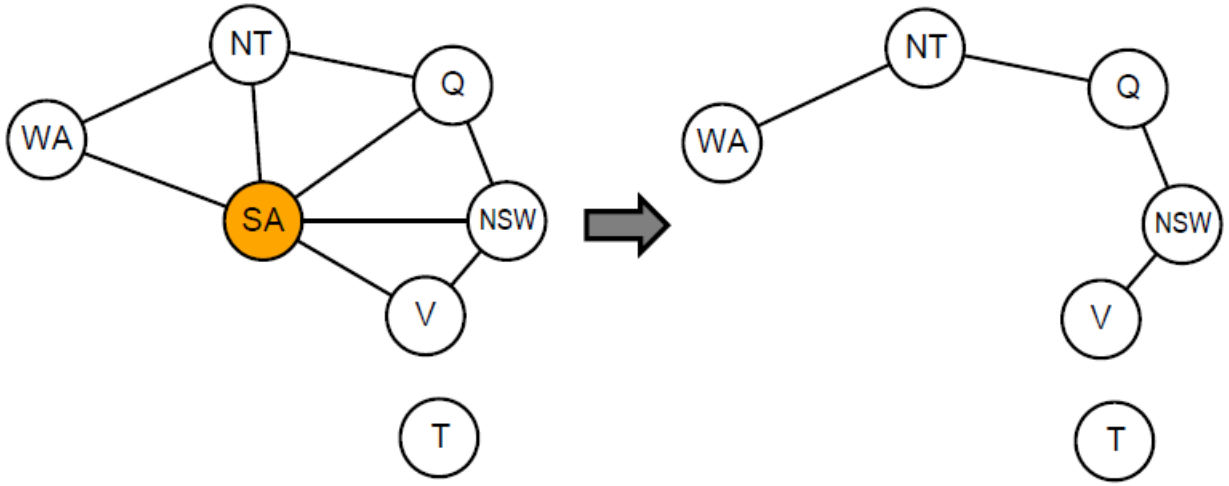
- 树状结构的CSP问题可以在变量个数的线性时间内求解：

1. 任选一节点作为树的根节点，从根节点到叶节点顺序排列：每个节点的父节点在它前面。

2. j = n to 2： 弧(Xi, Xj)上应用弧相容算法，删除Xj冲突值

3. j = 1 to n：赋给Xj跟Xi的值相容的任何值。

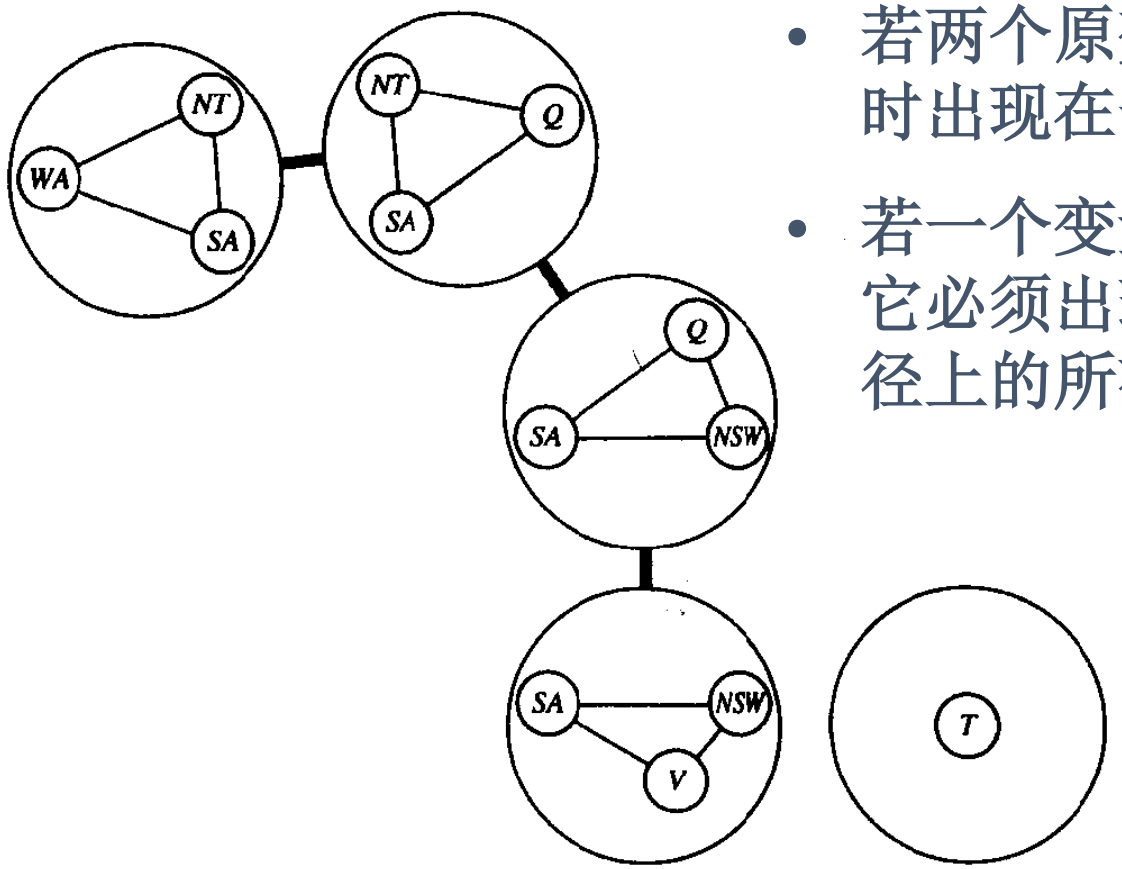
约束图→树



割集调整：对其中某些变量赋值，并从其他变量的值域中删除不相容值，使剩下的变量能形成树。

约束图→树

树分解：



- 原变量至少在一个子问题中出现
- 若两个原变量有约束，则应至少同时出现在一个子问题。
- 若一个变量出现在两个子问题中，它必须出现在连接这些子问题的路径上的所有子问题中。

- **约束满足问题**（CSP）的状态是变量/值对的集合，解条件通过一组变量上的约束表示。许多重要的现实世界问题都可以描述为 CSP。
- 很多推理技术使用约束来推导变量/值对是相容的，哪些不是。这些可以是结点相容、弧相容、路径相容和 *k*-相容。
- **回溯搜索**，深度优先搜索的一种形式，经常用于求解 CSP。推理和搜索可以交织进行。
- **最少剩余值**和**度**启发式是在回溯搜索中决定下一步选择哪个变量的方法，都独立于领域。**最少约束值**启发式帮助变量选取适当的值。回溯发生在变量找不到合法取值的时候。**冲突引导的回跳**直接回溯到导致问题的根源。
- 使用**最小冲突**启发式的局部搜索在求解约束满足问题方面也取得了成功。

课后习题

1. 试述遗传算法的基本原理，说明遗传算法的求解步骤