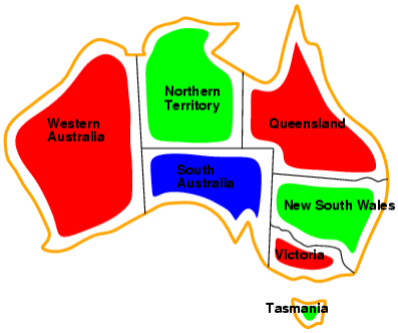
约束满足问题 CSP

- 标准搜索问题：
  - 状态是一个“黑匣子”——任何支持目标测试、评估、后续的旧数据结构
- CSP:
  - 状态由变量`Xi`和(值域)`Di`域中的值定义
  - 目标测试是一组约束条件，每个约束包括一些变量的子集，并指定这些子集的值之间允许进行的合并

示例：地图着色

- 变量WA、NT、Q、NSW、V、SA、T
  域Di={红、绿、蓝}
  限制：相邻区域必须有不同的颜色
- 原始图：

- 一种解决方案：

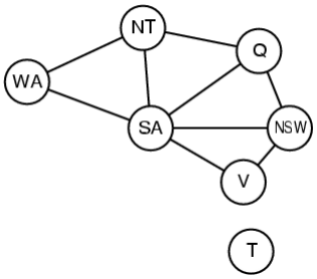


解决方案是满足所有约束的分配，例如

{WA=red, NT =green, Q=red, NSW =green, V =red, SA=blue, T =green}

约束图

- 二进制CSP：每个约束与两个变量相关
- 约束图：节点是变量，圆弧是约束



- 通用CSP算法使用图结构来加快搜索速度。

CSP的种类

- **离散变量：**
  - finite domains 有限值域:
    - n个变量，域大小$d \rightarrow O(d^n)$
    - 布尔CSPs，包括布尔可满足性（NP完全)如3-SAT
  - infinite domains 无限值域 (integers, strings, etc.)
    - 例如，作业调度，变量是每个作业的开始/结束日期
    - need a constraint language（约束语言）
    - 线性约束可解，非线性不可解
- **连续变量：**
  - 例如，哈勃太空望远镜观测的开始/结束时间
  - 用线性规划（LP）方法在多项式时间内可解的线性约束

约束类型

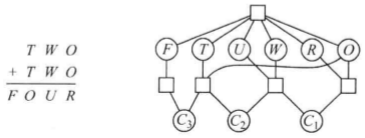
- **Unary（一元**）约束包含单个变量
  - 例如，SA≠green
- **Binary（二元）** 约束包含成对的变量
  - 例如，SA≠WA
- **Higher-order（更高阶**）约束包含3个或更多变量
  - 例如，密码运算(密码算数） 列约束
- **偏好（软约束）**，例如，红色比绿色好
- 通常用每个个体变量赋值的耗散来表示→**约束优化问题**

举例:密码算法

变量个数任意的约束称为**全局约束**（名称很传统但容易引起混淆，因为它并不一定涉及问题中的所有变量）。最常见的全局约束是 *Alldiff*，指的是约束中的所有变量必须取不同的值。在数独游戏中（见第 6.2.6 节），一行或一列中的所有变量必须满足 *Alldiff* 约束。另一个例子是**密码算术**谜题（见图 6.2（a））。密码算术中每个字母都表示不同的数字。在图 6.2（a）所示的情况中，表示为六变量约束 *Alldiff*(F, T, U, W, R, O)。此谜题的四列算式可表示为如下涉及多个变量的 n 元约束：

$$O + O = R + 10 \cdot C_{10}$$
$$C_{10} + W + W = U + 10 \cdot C_{100}$$
$$C_{100} + T + T = O + 10 \cdot C_{1000}$$
$$C_{1000} = F$$

其中 C_{10}，C_{100}，C_{1000} 表示十位、百位、千位上的进位变量。这些约束可以用**约束超图**表示，如图 6.2（b）所示。超图中包含普通结点（图中的圆圈结点）和表示 n 元约束的超结点（用方块表示）。



(a) (b)

图 6.2

现实世界的CSP

- 分配作业问题，例如，谁教什么课，谁复习哪些论文
- 时间表问题，例如，何时何地提供哪门课
  - 硬件配置
  - 运输调度
  - 工厂调度
  - 平面布置
- 请注意，许多现实世界中的问题都涉及实值变量
- **逐个枚举**：
  - 低效
  - 需要指数时间d^n
  - 但完备
    - 如何令指数时间算法高效一点

标准搜索公式

- **incremental增量形式化**
  - 让我们从简单的方法开始，然后完善它
  - 状态由迄今为止分配的值定义
    - **初始状态**：空赋值， {}
    - **后继函数**：
      - 为未赋值的变量赋值，该变量与当前赋值不冲突
      - 如果没有合法分配，则失败
    - **目标测试**：当前任务已完成

1. **优点**： 对所有CSP，形式化方法都是一样的
2. **每个解都出现在深度n处**，有n个变量
   - 所以可以**用深度优先搜索**
3. 路径之间是不相关的，所以可以使用完全状态形式化

4. **缺点**：CSP有n个值域大小为d的变量，顶层的分支因子为nd，因为有n个变量，每个变量的取值可以是d个值中的任何一个。在下一层，分支因子是(n-1)d，依此类推n层。生成了一棵有n!·d^n个叶子的搜索树，尽管可能的完整赋值只有d^n个！

5. 看来合理但是弱智的问题形式化忽略了所有 CSP 的一个共同的至关紧要的性质：**可交换性**。如果行动的先后顺序对结果没有影响，那么问题就是可交换的。CSP 是可交换的，因为给变量赋值的时候不需考虑赋值的顺序。因此，只需考虑搜索树一个结点的单个变量。例如，在澳大利亚地图着色的搜索树的根结点，要在 $SA = red$，$SA = green$ 和 $SA = blue$ 之间选择，但永远不需要在 $SA = red$ 和 $WA = blue$ 之间选择。有了这个限制，叶结点的个数如我们所希望的减少到了 d^n 个。

回溯搜索

- 变量赋值是可交换的
  - `[ WA = red then NT = green ]` 和 `[ NT = green then WA = red ]` 是相同的
- 具有单变量赋值的CSP的深度优先搜索称为回溯搜索
- 回溯搜索是CSP的基本无信息算法。可以解决n≈25的n皇后问题

术语**回溯搜索**用于深度优先搜索中，它每次为一变量选择一个赋值，当没有合法的值可以赋给某变量时就回溯。算法见图 6.5。它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。一旦检测到不相容，BACKTRACK 失败，返回上一次调用尝试另一个值。图 6.6 显示了澳大利亚问题的部分搜索树，其中按照 *WA*、*NT*、*Q*……的顺序来给变量赋值，因为 CSP 表示是标准化的，不需要给 BACKTRACKING-SEARCH 提供领域专用的初始状态、行动函数、转移模型或目标测试。

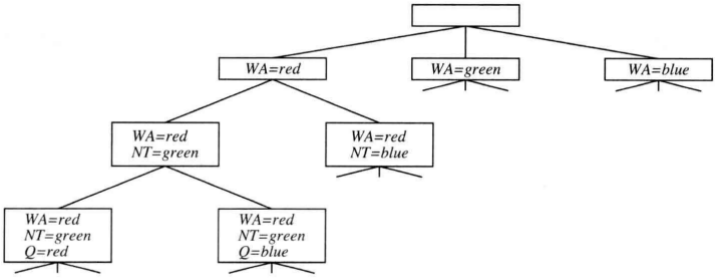


图 6.6　图 6.1 中地图着色问题的部分搜索树

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
  return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/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 given CONSTRAINTS[csp] then
      add {var = value} to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove {var = value} from assignment
  return failure
```

改进回溯搜索的效率

- 通用方法可以在速度上带来巨大的收益：
  1. 接下来应该分配哪个变量？
  2. 应该按照什么顺序尝试它的值？
  3. 我们能发现当前的方案是行不通的吗？提前失败？
  4. 我们能利用问题结构吗？

最少剩余值启发式

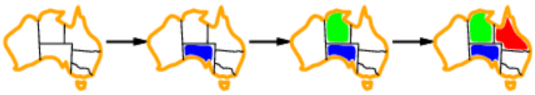
解决：下一步应该分配哪个变量

- Minimum remaining values 最少剩余值(MRV):选择合法值最少的变量，但是在这里对选择第一个着色区域没有帮助，因为初始时每个区域都有合法的三种颜色可选

 - 相比于随机或静态的排序，MRV启发式通常性能更好-可以选择出最可能很快导致失败的变量

度启发式

解决：下一步应该分配哪个变量

- 度启发式：通过选择与其他未赋值变量约束最多的变量来试图降低未来的分支因子
- 最少剩余值启发式通常是一个强有力的导引，而度启发式对打破僵局非常有用。
- 在这里，度启发式可以决定选择第一个着色区域 

最少约束值启发式

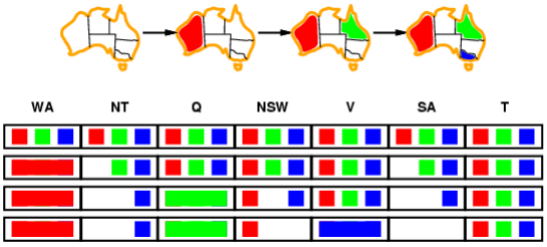
解决：应该按照什么顺序尝试它的值

- 一旦一个变量被选定，算法需要决定检验它的取值顺序。有时最少约束值启发式很有效
- 它优先选择的值是给邻居变量留下更多的选择。
- 排除剩余变量中最小值的一个；请注意，这可能需要一些计算才能确定！

- 将这些启发式方法结合起来，那么1000皇后问题也是可行的

Forward checking—前向检验

- 思想：跟踪未分配变量的剩余合法值，当任何变量没有合法值时终止搜索



Constraint propagation — 约束传播

- 前向检查将信息从已分配的变量传播到未分配的变量，但不能为所有故障提供早期检测：
- 在第三个状态出现问题：NT和SA不能都是蓝色的！说明后续的状态是无意义的。