华东师范大学计算机科学与技术学院
School of Computer Science and Technology

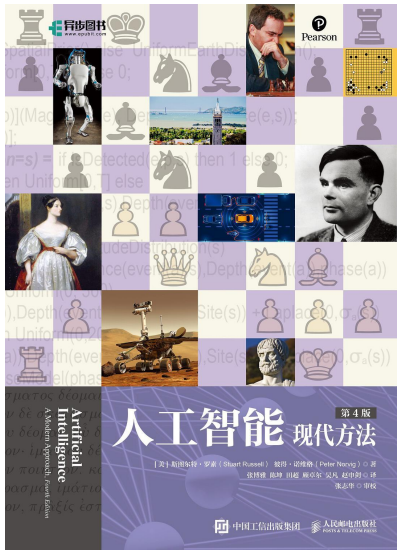
2024年春季学期

《人工智能》

第6讲：约束满足

第6章 约束满足

- **主要内容**
  - 6.1 约束满足问题定义
  - 6.2 求解CSP的一般搜索算法
  - 6.3 求解CSP的启发式算法
  - 6.4 本章小结
- **参考书目**
  - 《人工智能：现代方法（第4版）》（美）罗素，（美）诺维格，人民邮电出版社，2022。
    Ch5: pp.142-162.

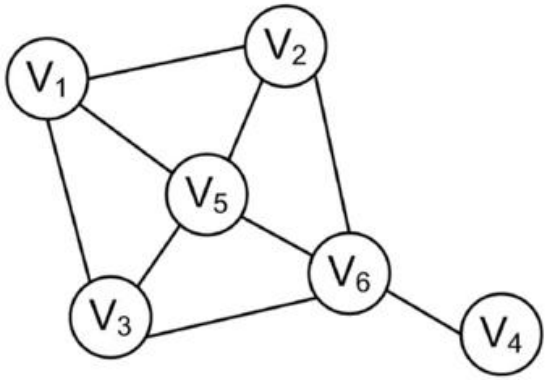


6.1 约束满足问题定义

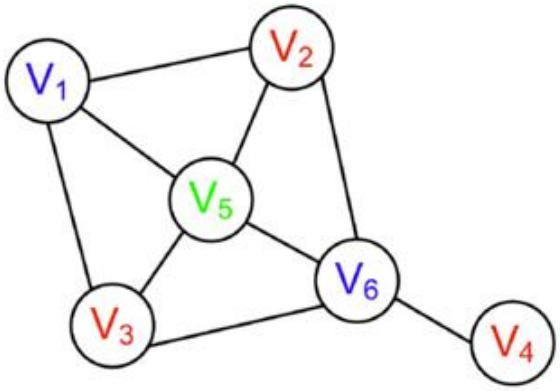
- **地图着色**是指分配地图的每一个面一种颜色，使得相邻的面(指有公共边界边)具有不同的颜色。

6.1 约束满足问题定义

- **图形着色定义**
  - 考虑一个图形中的N个节点
  - 把变量$V_1,\ldots,V_N$的值赋给N个节点
  - 变量取值范围{R, G, B}
  - 约束：如果节点i和j之间有边，则V_i不同于V_j



问题模型



一个解

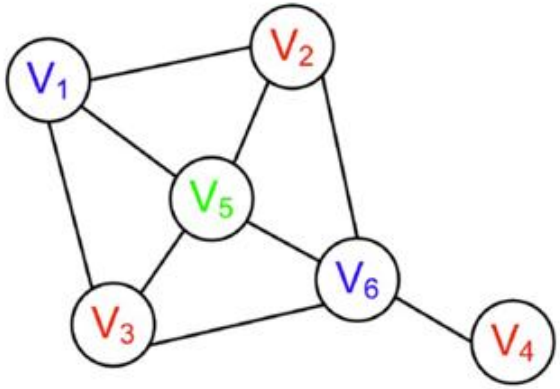
6.1 约束满足问题定义

- **约束满足问题 (Constraint Satisfaction Problems, CSP)**
  - CSP={V, D, C}
  - 变量：$V=\{V_1,\ldots,V_N\}$，如图中节点
  - 取值域：每个变量的取值范围，如D={R, G, B}
  - 约束集：$C = \{C_1,\ldots,C_K\}$
    - 每个约束有一组变量与一系列该组变量的允许取值组成，如

      $[(V_2, V_3), \{R,G\},\{R,B\},\{G,B\}, \{G, R\},\{B, R\},\{B, G\}]$
    - 通常隐式定义约束，如对每条边(i,j)，要求$V_i \neq V_j$
- **CSP问题的解**
  - 变量满足所有约束要求的赋值：$\{V_1=v_1,\ldots,V_N=v_n\}$
- **CSP问题特点**
  - 没有优化目标，与一般优化问题不同

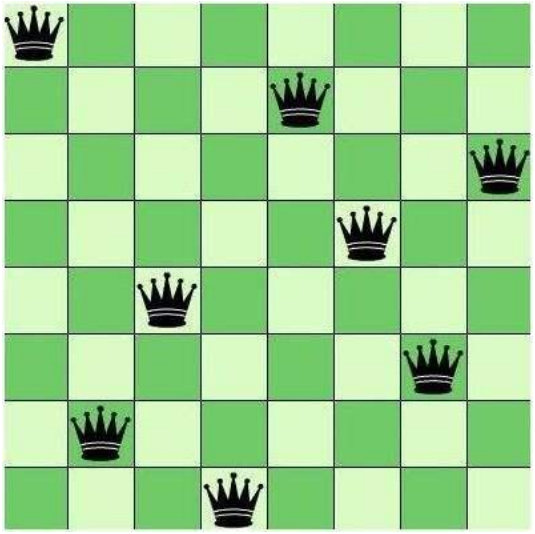
6.1 约束满足问题定义

- **二元CSP问题的约束图表示**
  - 约束与两个变量有关
  - 节点代表变量
  - 连线代表约束
  - 类似着色问题

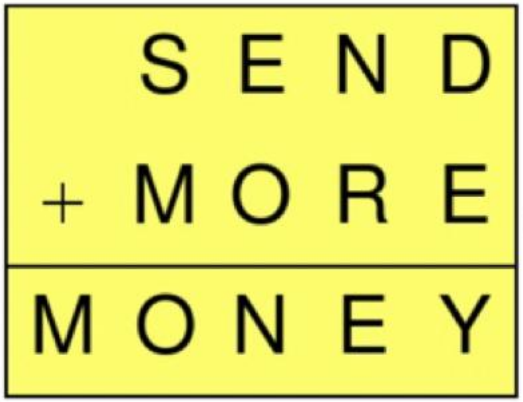


6.1 约束满足问题定义

- **例子：N皇后**
  - 变量：$Q_1, Q_2, \ldots, Q_N$
  - 域：$D_i = \{1, 2, \ldots, N\}$
  - 约束
    - $Q_i \neq Q_j$, 不在同一列
    - $|Q_i - Q_j| \neq |i - j|$，不在同一对角线



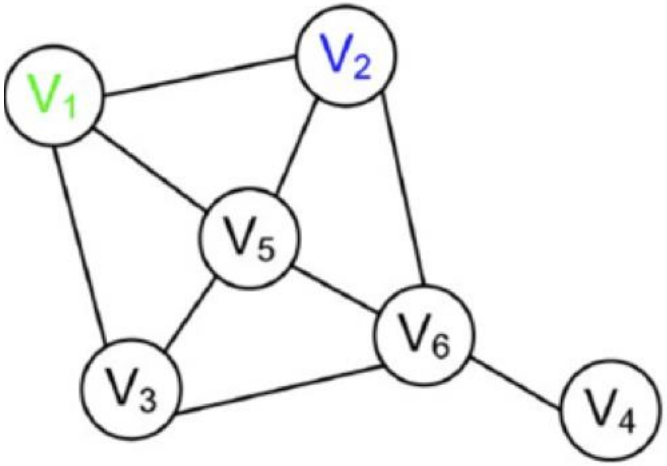
- **例子：加密问题**
  - 变量：D,E,M,N,O,R,S,Y
  - 域：$D_i = \{0,1, 2, \ldots, 9\}$
  - 约束
    - $M \neq 0, S \neq 0$
    - Y=D+E或Y=D+E-10
    - $D \neq E$， $D \neq M$， $D \neq N$等



- **其它例子：数独、调度、设计、分配、规划等**

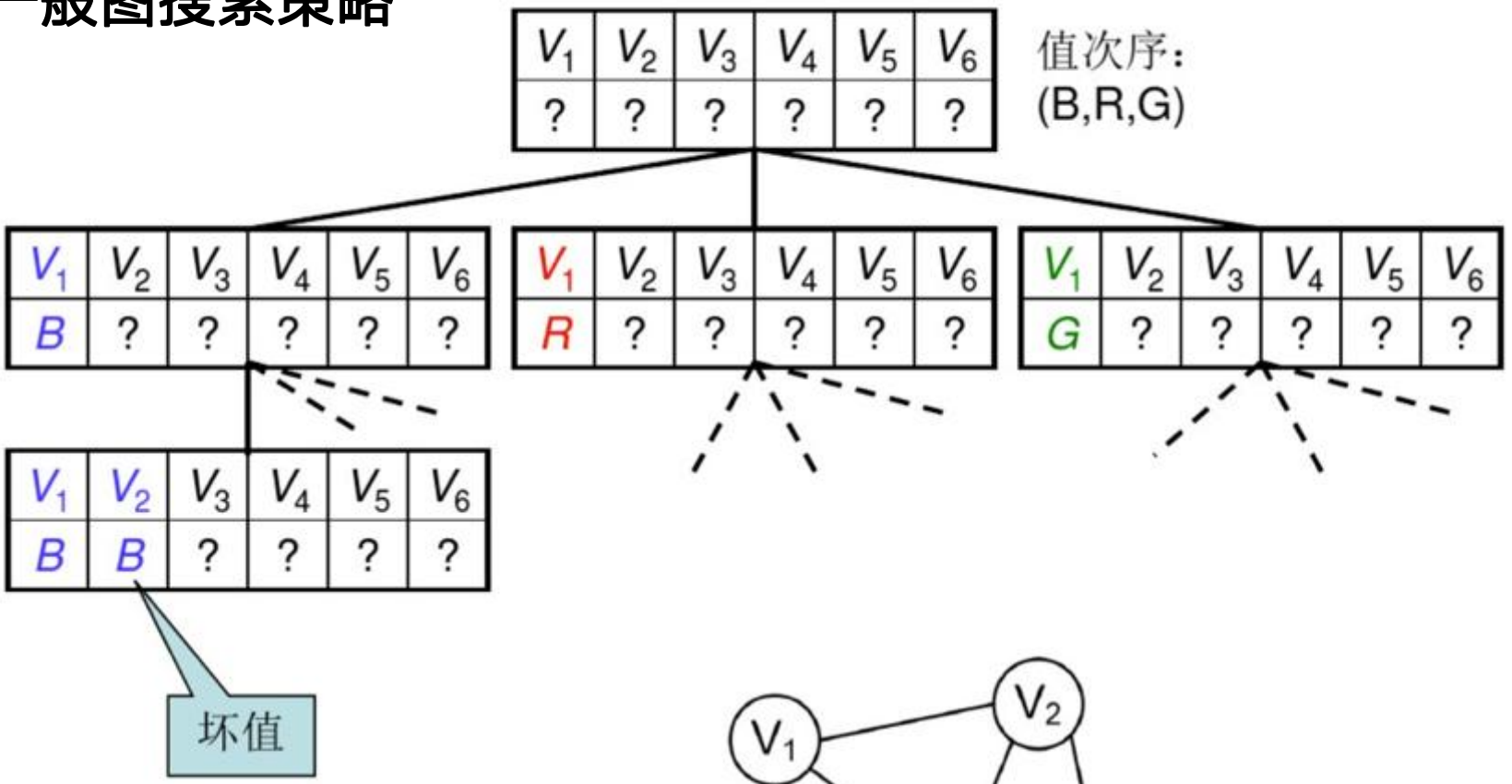
6.2 CSP一般搜索算法

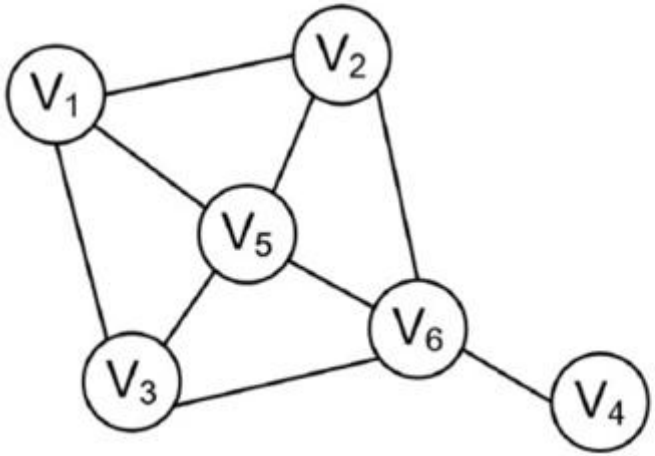
- **一般图搜索策略**
  - 状态：给定K个取值，K+1,…,N个变量未取值
  - 后继状态：给第K+1个变量赋值
  - 初始状态：V_1=?, V_2=?, …, V_N=?
  - 目标状态：所有变量均赋值，并且满足所有约束要求
  - 采用一般图搜索算法即可求解



6.2 CSP一般搜索算法

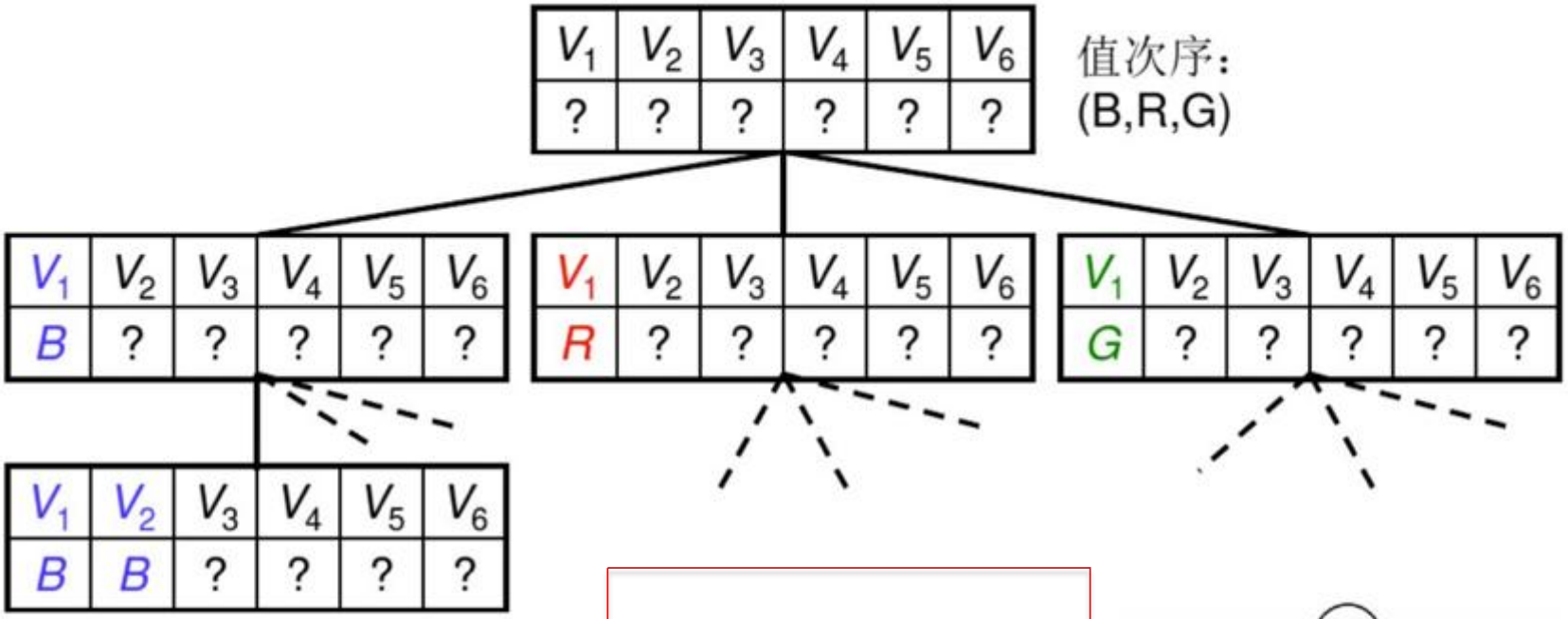
- 一般图搜索策略



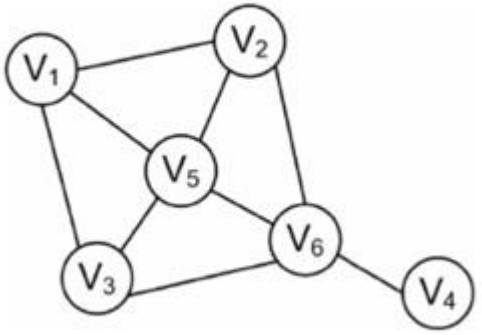


6.2 CSP一般搜索算法

- 一般图搜索策略（DFS）



效率低下



- 采用递归方式：

  对*D*中每个可能值：

  为后续态中的下个未赋值变量赋该值

  赋值后，评估当前态的后续态

  一旦找到解，就停止

6.2 CSP一般搜索算法

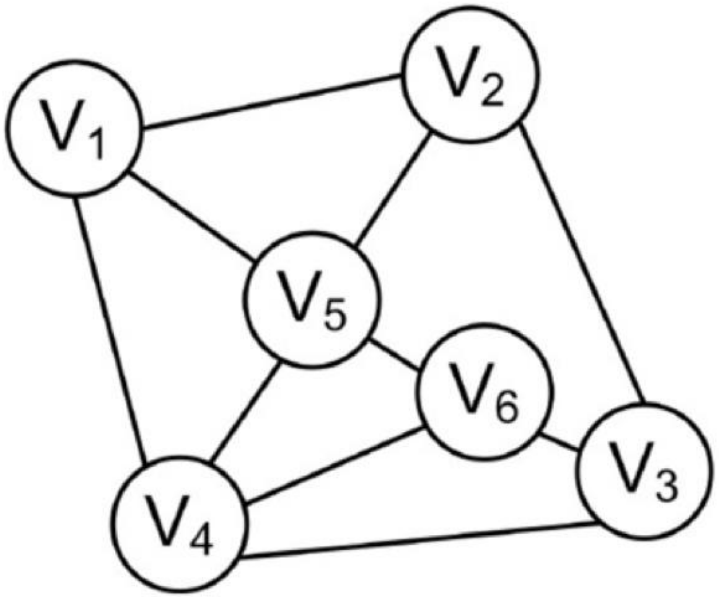
- **DFS改进策略**
  - 基本思想：尽可能剪枝
  - 只评估那些赋值，它们不违反任何与目前赋值相关的约束
  - 不搜索那些明显不可能通往解的分支
  - 预测合法的赋值
  - 控制变量与值的排序

6.2 CSP一般搜索算法

- **向前查看策略**
  - 对于未赋值的变量，跟踪余下的合法值
  - 当变量无合法值时，回溯

值次序：(R,B,G)

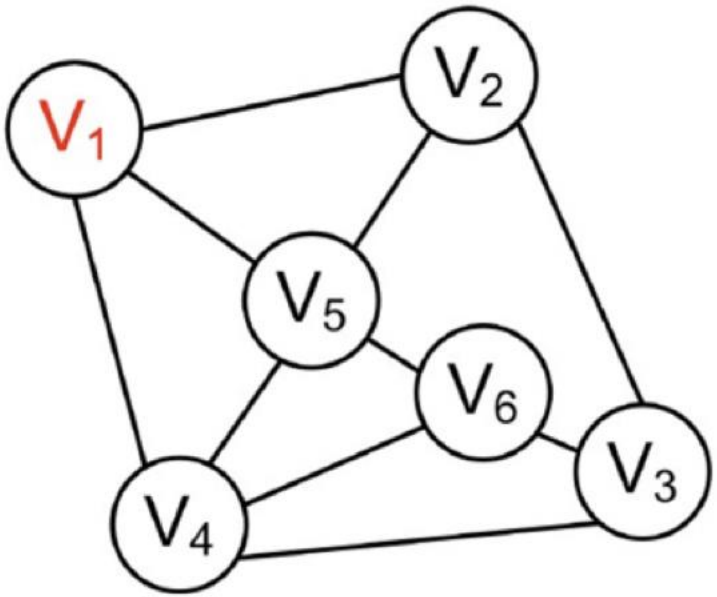
	V_1	V_2	V_3	V_4	V_5	V_6
R	?	?	?	?	?	?
B	?	?	?	?	?	?
G	?	?	?	?	?	?



6.2 CSP一般搜索算法

- **向前查看策略**
  - 对于未赋值的变量，跟踪余下的合法值
  - 当变量无合法值时，回溯

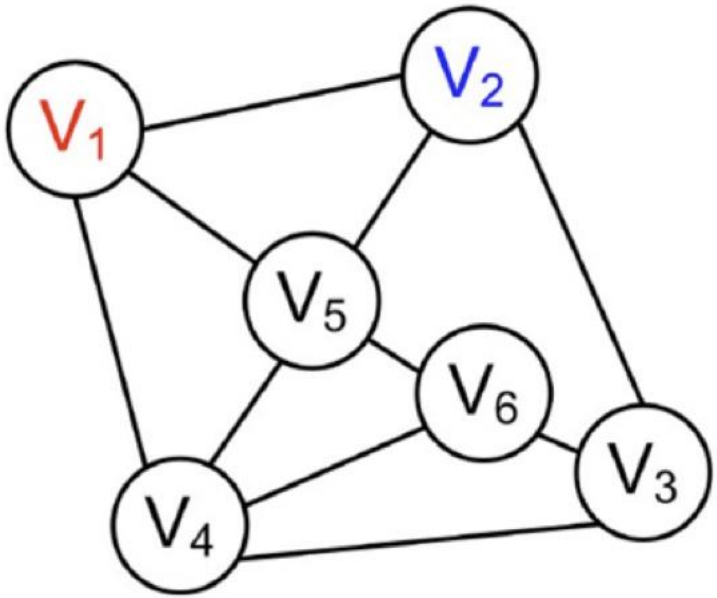
	V_1	V_2	V_3	V_4	V_5	V_6
R	O	X	?	X	X	?
B		?	?	?	?	?
G		?	?	?	?	?



6.2 CSP一般搜索算法

- **向前查看策略**
  - 对于未赋值的变量，跟踪余下的合法值
  - 当变量无合法值时，回溯

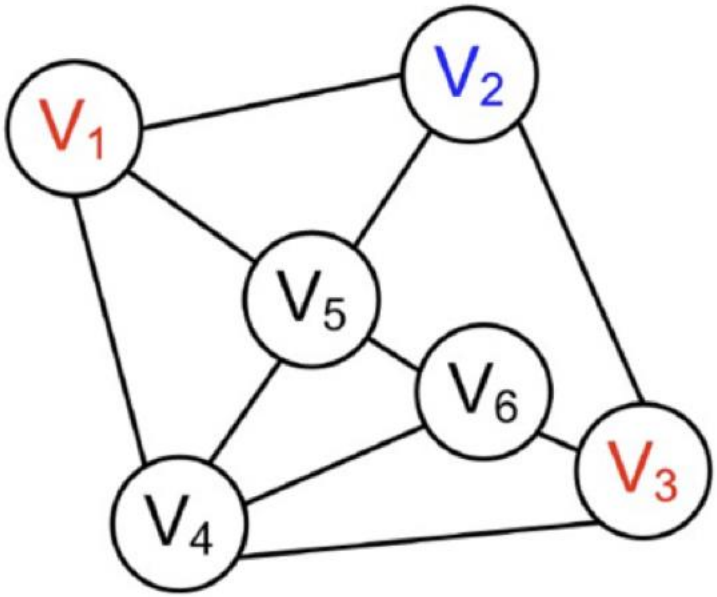
	V_1	V_2	V_3	V_4	V_5	V_6
R	O		?	X	X	?
B		O	X	?	X	?
G			?	?	?	?



6.2 CSP一般搜索算法

- **向前查看策略**
  - 对于未赋值的变量，跟踪余下的合法值
  - 当变量无合法值时，回溯

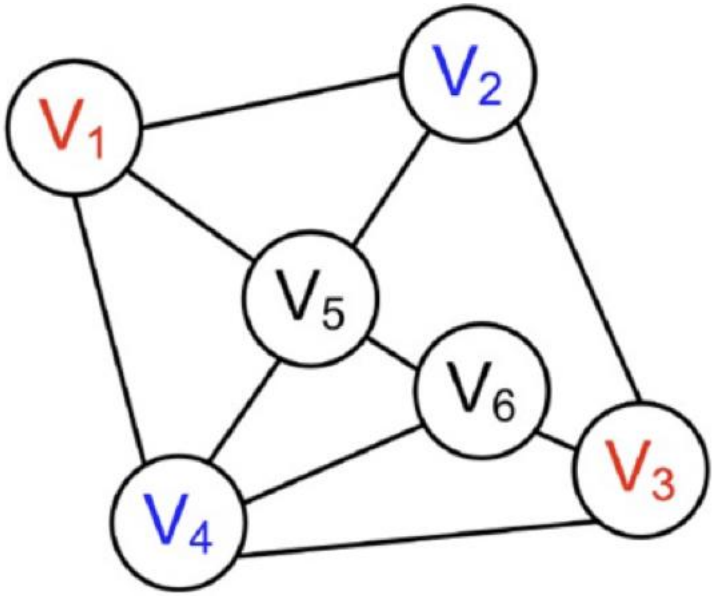
	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O	X	X	X
B		O		?	X	?
G				?	?	?



6.2 CSP一般搜索算法

- **向前查看策略**
  - 对于未赋值的变量，跟踪余下的合法值
  - 当变量无合法值时，回溯

	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O		X	X
B		O		O	X	X
G					?	?

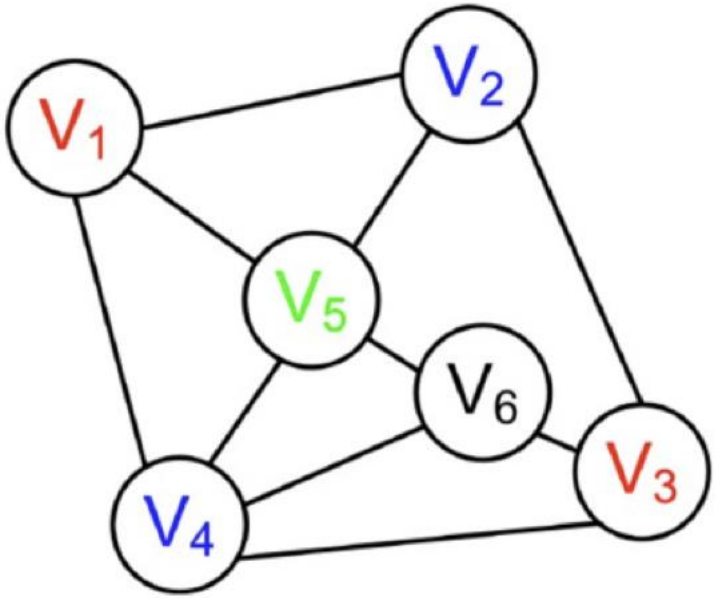


6.2 CSP一般搜索算法

- **向前查看策略**
  - 对于未赋值的变量，跟踪余下的合法值
  - 当变量无合法值时，回溯

	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O			X
B		O		O		X
G					O	X

无合法值能赋给V_6，
因此需要回溯

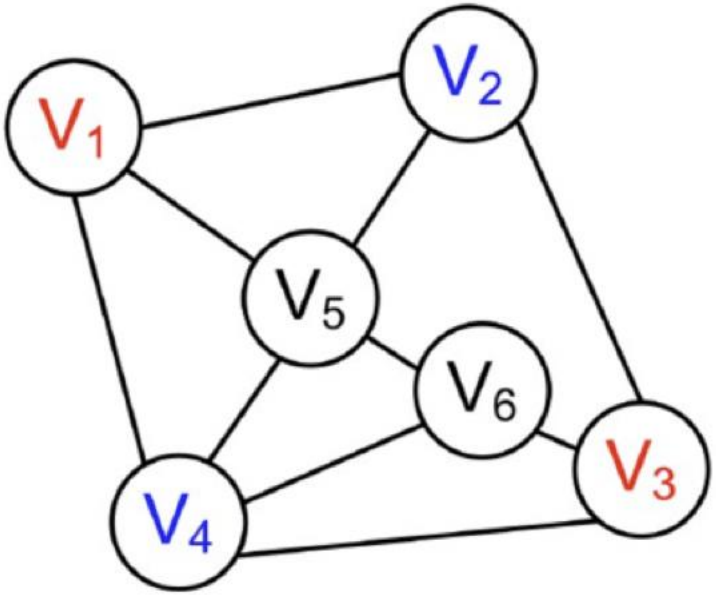


6.2 CSP一般搜索算法

- **向前查看策略**
  - 向前查看不检查所有的不一致性，因为它只检查与当前赋值相关的约束
  - 向前查看已经剪掉很多分支，能否看得更远？

	V_1	V_2	V_3	V_4	V_5	V_6
R	O		O		X	X
B		O		O	X	X
G					?	?

在该处已可看出，此路径不通向解，因为域中剩余值在赋给V_5与V_6后不能保持一致性。



6.2 CSP一般搜索算法

- **约束传播（Constraint Propagation, CP）策略**
  - V=在搜索的当前层次，需要赋值的变量
  - 将D(V)中的一个值赋给V
  - 对与V相连的每个变量V’：
    - 去掉D(V’)中与已赋值变量不一致的值
    - 对与V’相连的每个变量V”：
      - ✓ 去掉D(V”)中与已赋值变量不一致的值
      - ✓ 对与V”相连的每个变量重复执行直到不再有能被去掉的值为止
  - 注：
    - 清理D(V’)属于已有的向前查看
    - 清理D(V”)…属于新的约束传播

6.2 CSP一般搜索算法

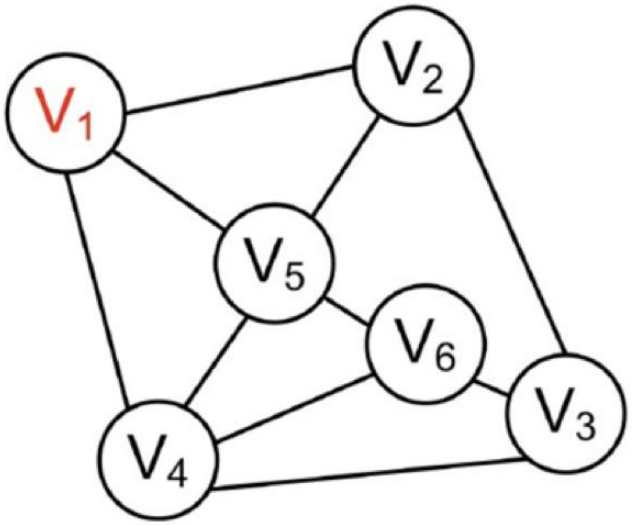
- **求解图形着色问题的约束传播算法**

```
Propagate(node, color)
1.从node的所有近邻的值域中去掉color
2.对每个近邻n
    if 第1步后，D(n)中只剩一种颜色，即D(n)={c}
        Propagate(n, c)
```

- **例子**

在传播(V_1,R)后：

	V_1	V_2	V_3	V_4	V_5	V_6
R	O	X	?	X	X	?
B		?	?	?	?	?
G		?	?	?	?	?

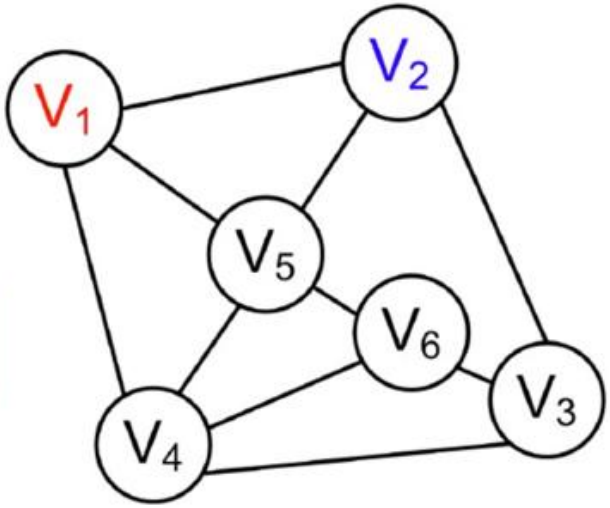


6.2 CSP一般搜索算法

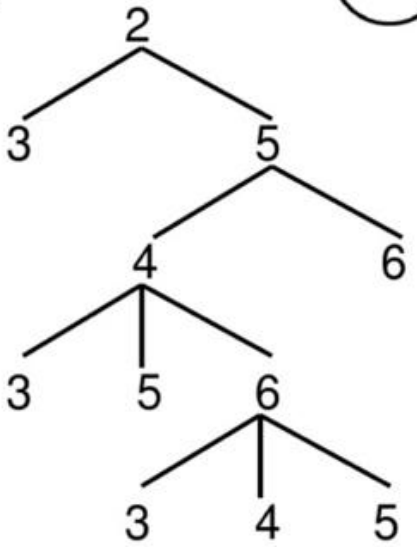
- **约束传播策略**

在传播(V_2,B)后：

	V_1	V_2	V_3	V_4	V_5	V_6
R	O		X	X	X	?
B		O	X	?	X	X
G			?	X	?	X



传播次序：

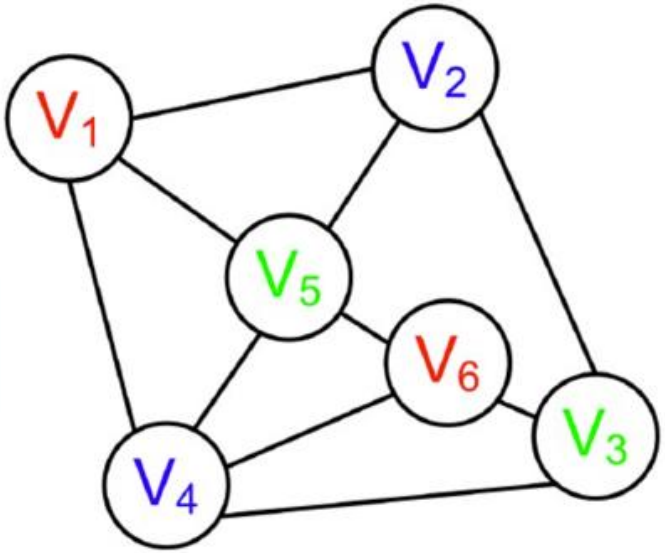


6.2 CSP一般搜索算法

- **约束传播策略**

在传播(V_2,B)后：

	V_1	V_2	V_3	V_4	V_5	V_6
R	O		X	X	X	?
B		O	X	?	X	X
G			?	X	?	X



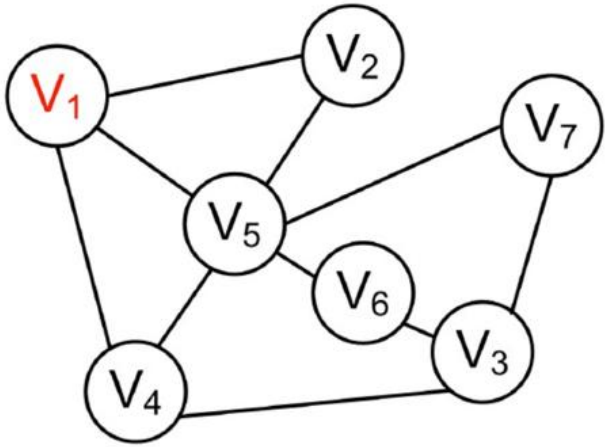
注：

- 在设置V_2后，无需更多搜索，只需一步CP就直接得到一个解。
- 一些问题甚至可无需任何搜索，直接由CP来解。

6.3 CSP启发式搜索算法

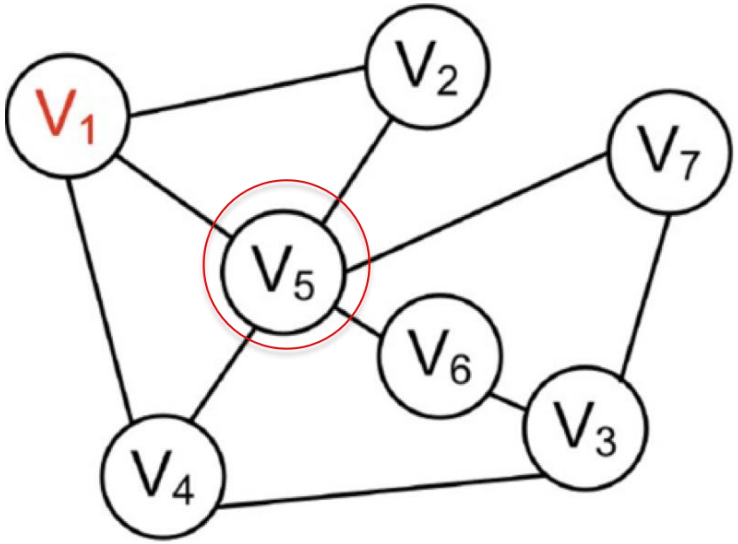
- **变量与值的启发式算法**
  - 一般搜索算法中，是以一个固定的次序来选择下一个变量和下一个值
  - 问题：
    - 有更好的方法来选择下一个变量吗？
    - 有更好的方法来选择下一个赋值给当前变量吗？

V_1	V_2	V_3	V_4	V_5	V_6	V_7
R	?	?	?	?	?	?



6.3 CSP启发式搜索算法

- **变量排序**
  - 最多约束变量(MCV)
  - 选择一个贡献最多约束数的变量，会对其它变量有极大的影响，因此有希望裁剪掉大部分搜索
  - 要求：在约束图中找到与最多变量相连的变量



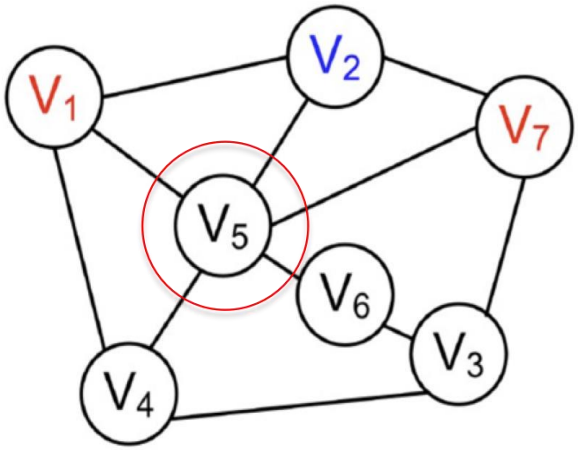
V_1	V_2	V_3	V_4	V_5	V_6	V_7
R	?	?	?	?	?	?

变量V_5影响5个变量。

变量V_2、V_3或V_4只影响较少的变量。

6.3 CSP启发式搜索算法

- **变量排序**
  - 最少余下值(MRV)
  - 选择一个侯选值最少的变量，由此极可能导致一个早期的失败(失败优先启发式策略)



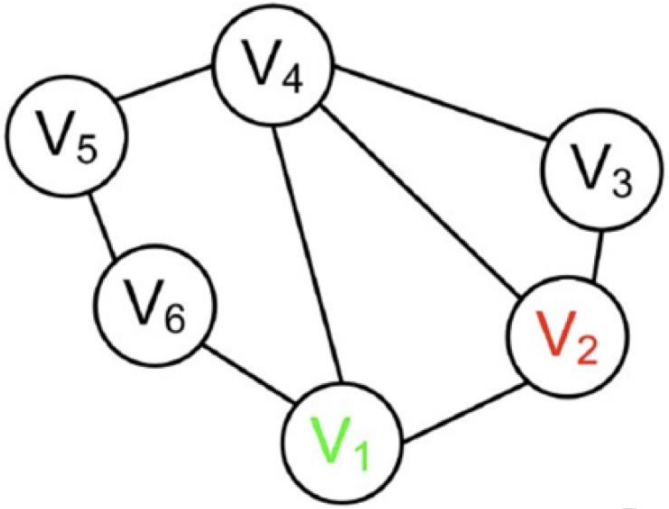
	V_1	V_2	V_3	V_4	V_5	V_6	V_7
R	O		X	X	X	?	O
B		O	?	?	X	?	
G			?	?	?	?	

变量V5是最受约束的变量，

且最有可能用来剪枝搜索树

6.3 CSP启发式搜索算法

- **值排序**
  - 最少值约束(LCV)
  - 选择使相邻变量可用值减少最少的值（即对邻居可用值影响最小）
  - 优先选用最有可能的值（也即为随后的变量赋值提供最大的灵活性）来获得一个解



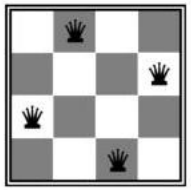
V_1	V_2	V_3	V_4	V_5	V_6
G	R	?	?	?	?

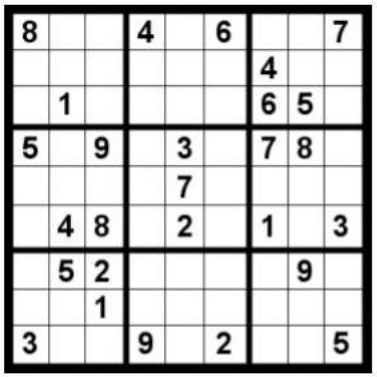
四种颜色：D={R,G,B,Y}

要给V_3赋哪个值？

6.4 本章小结

- **约束满足是一类特殊的搜索问题**
  - 只要求满足所有约束，不要求目标最优
- **采用各种搜索策略求解**
- **特有的问题信息**
  - 向前查看
  - 约束传播
  - 变量排序
  - 值排序





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

- **练习**
  - 澳大利亚地图着色

澳大利亚地图

图搜索算法

向前查看

变量排序

值排序

搜索算法总结

- **一般图搜索算法**
  - 最基础的搜索算法，与问题无关，效率低下
- **启发式搜索算法**
  - 如何有效利用问题信息，提高算法搜索效率
- **不确定性搜索算法**
  - 初始化完整解，并逐步改进，引入随机因素，求近似最优解

偏算法

- **博弈搜索**
  - 针对二人博弈问题，剪枝和评估函数值估计是关键
- **约束满足**
  - 无特定目标，寻找满足所有约束条件的解，剪枝是关键

偏问题

搜索算法总结

- **一般图搜索算法**
  - BFS、DFS
- **启发式搜索算法**
  - A f(n)=g(n) + h(n)
  - A* h(n)<=h*(n)
  - 改进：g(n) == g*(n)
- **不确定搜索算法**
  - 不确定搜索算法（邻域构建，优化：变步长、初始点选择）
  - 模拟退火算法（非最有解也有被选择的可能）
- **博弈搜索算法**
  - f(p)评估值，正方的得分评估
  - alpha-beta剪枝法
- **约束搜索算法**
  - 向前查看
  - 约束传播
  - 变量排序
  - 值排序

搜索算法总结

- **两大类基本搜索算法**
  - 构造法：从零开始逐渐构造一个最优解
  - 逼近法：从一个差的解开始逐渐逼近最优解
- **搜索算法设计的本质**
  - 在给定的硬件条件下，更快更好地解决问题
  - 更快：降低搜索空间大小 + 问题相关启发式信息指导+硬件加速
  - 更好：有限时间内获得更好的近似最优解或局部最优解
- **问题+算法+硬件**
  - 算法与问题无法分割，脱离问题的算法效率过低（穷举法等）
  - 算法在具体的硬件上执行，需考虑执行环境（研究相对较少）
  - 问题分类->》寻找同类问题共有的启发式信息->》设计或改进求解算法
  - 将人的认识融入到计算机自动执行的算法之中

谢谢