Backtracking Search for CSPs

School of Electronic and Computer Engineering
Peking University

Wang Wenmin

Contents

Overview of Backtracking Search 回溯搜索概述

- It is a general algorithm on depth-first search, used for finding solutions to some computational problems, notably CSPs.

  是一种深度优先搜索的通用算法，用于查找某些计算问题的答案，尤其是CSPs。

- Backtracking search incrementally builds candidates to the solutions, and abandons each *partial candidate* c (backtracks), as soon as it determines that c cannot possibly be completed to a valid solution.

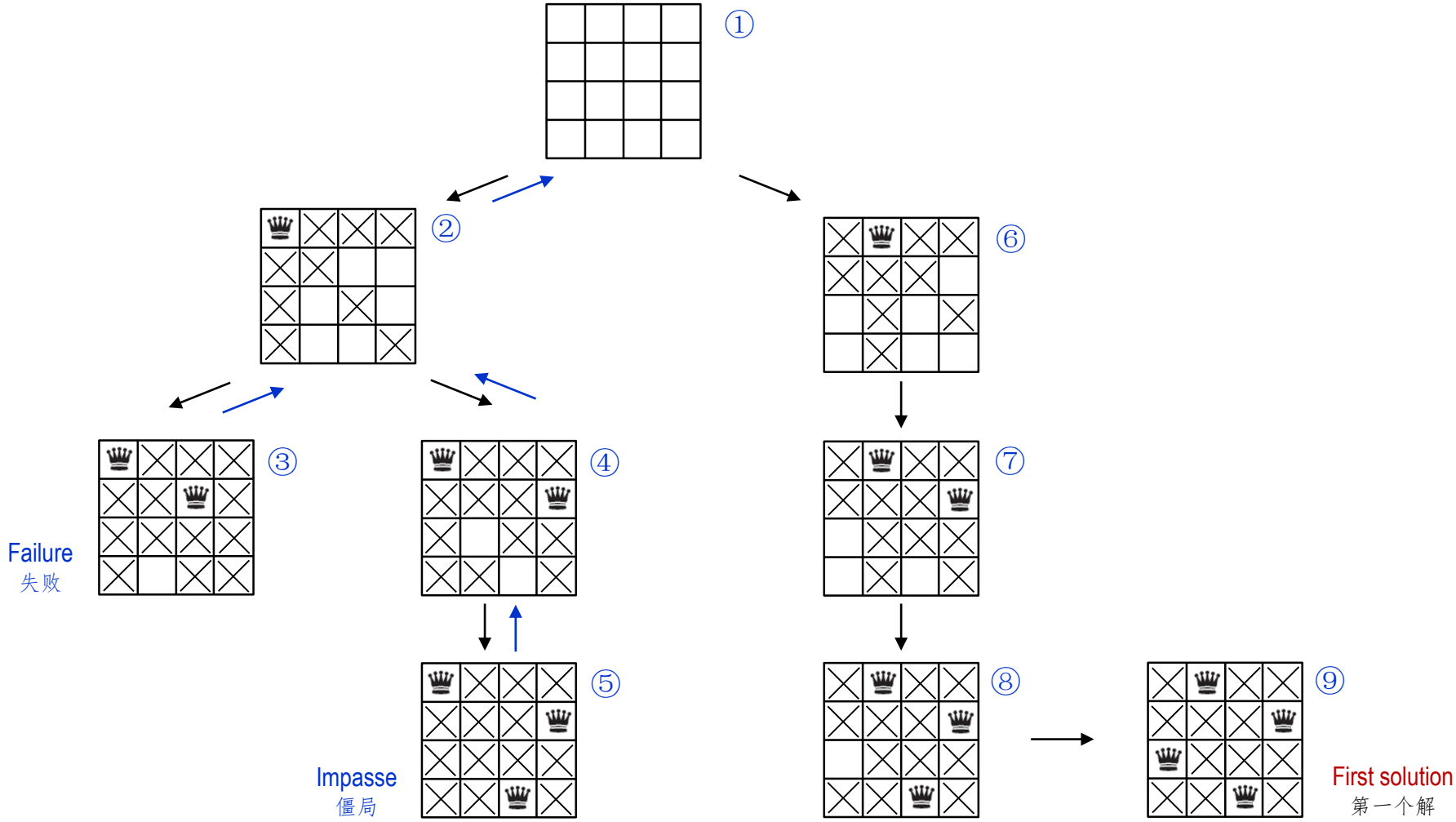
  回溯搜索递增地构建解的候选，而且一旦确定部分候选c不能成为一个合法的解，就将c抛弃（回溯）。

Example: 8-queens puzzle 8皇后难题

- In the common backtracking approach, the partial candidates are arrangements of k queens in the first k rows of the board, all in different rows and columns.

  常见的回溯方法，部分候选是在棋盘的在前k行上布局k个皇后，所有这些要在不同的行与列上。

Example: 4-queen problem 4皇后问题



Overview of Backtracking Search 回溯搜索概述

- It is the basic uninformed algorithm, a depth-first search with two improvements, for solving CSPs.
  是基本的无信息算法，一种具有两种改进的深度优先搜索，用于求解CSPs问题。
- *Idea* 1: One variable at a time 构思1：每次一个变量
  - Variable assignments are commutative, i.e.,
    变量赋值是可交换的，例如
    $[WA = red$ then $NT = green]$, same as $[NT = green$ then $WA = red]$
- *Idea* 2: Check constraints as you go 构思2：检查所需约束
  - I.e., consider only values which do not conflict previous assignments
    即，仅需考虑与前面赋值不发生冲突的值
  - Might have to check the constraints. “Incremental goal test”
    也许需要检查该约束。递增式目标检测

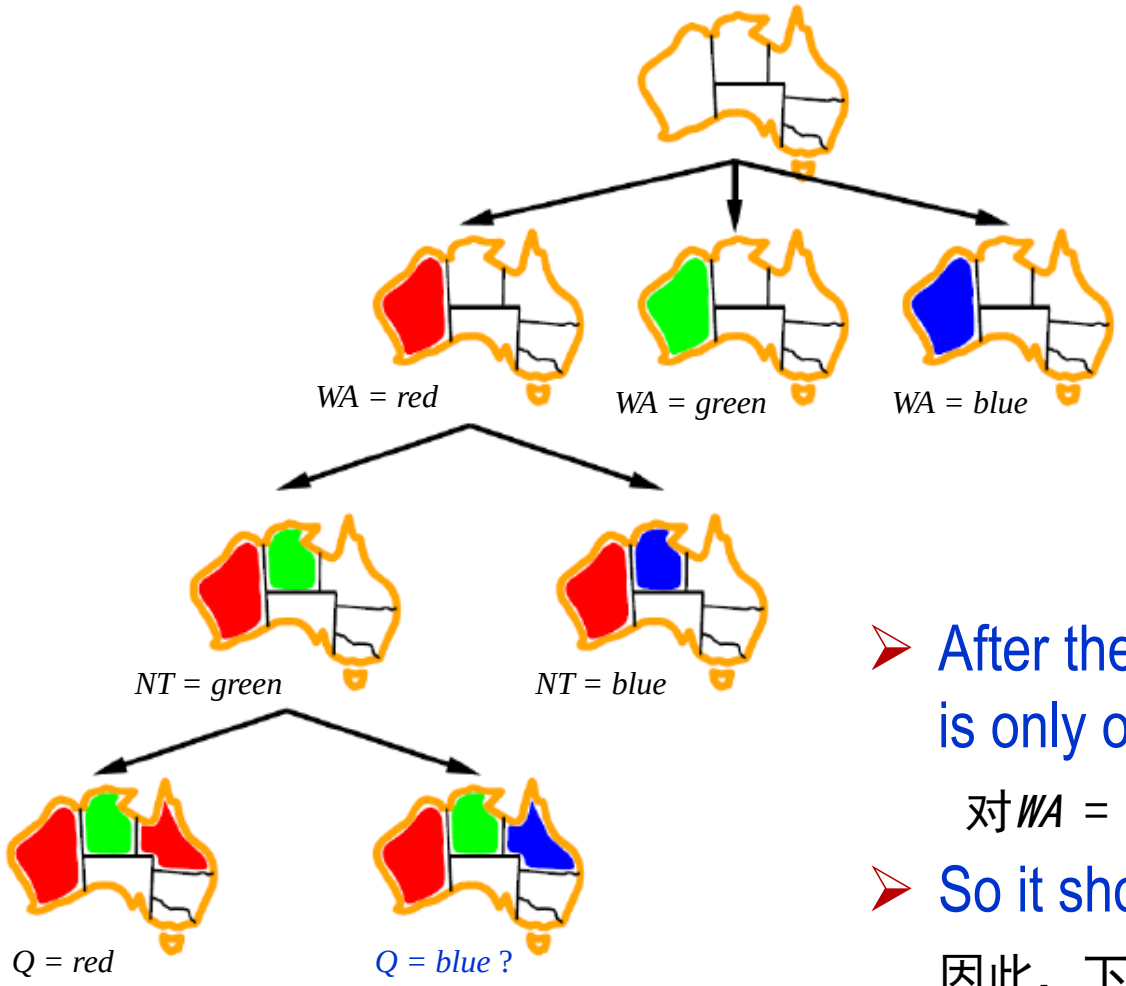
A Simple Backtracking Algorithm for CSPs 一个简单的CSPs回溯算法

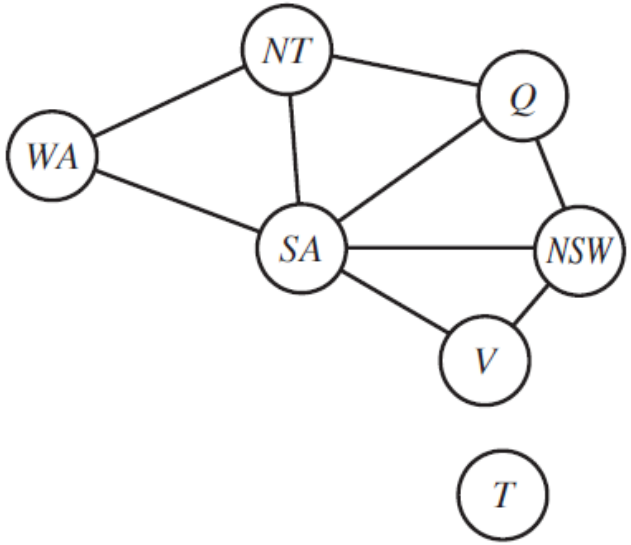
```
function BACKTRACK-SEARCH(csp) returns a solution, or failure
    return BACKTRACK({ }, csp)
function BACKTRACK(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment then
            add {var = value} to assignment
            inferences ← INFERENCE(csp, var, value)
            if inferences ≠ failure then
                add inferences to assignment
                result ← BACKTRACK(assignment, csp)
                if result = failure then return result
        remove {var = value} and inferences from assignment
    return failure
```

Questions to Improve Backtracking 改善回溯的若干问题

- Question 1: 问题1：
  - Which variable should be assigned next? 下一步哪个变量应该被赋值?
    (SELECT-UNASSIGNED-VARIABLE)
  - And, in what order should its values be tried? 并且，尝试值应该以什么顺序?
    (ORDER-DOMAIN-VALUES)
- Question 2: 问题2：
  - What inferences should be performed at each step? 每一步应该完成什么推理?
    (INFERENCE)
- Question 3: 问题3：
  - When the search arrives at an assignment that violates a constraint, can the search avoid repeating this failure? 当搜索到一个违反约束的赋值时，该搜索是否能避免重复这个失败?

Discussion 1: Variable and Value Ordering 变量与值的排序





- After the assignments for $WA = red$ and $NT = green$, there is only one possible value for SA.

  对*WA = red*和*NT = green*赋值之后，对*SA*来说仅剩一个可能的值。

- So it should assign $SA = blue$ next rather than assigning Q.

  因此，下一步应该赋值*SA = blue* 而不是赋值*Q*。

Discussion 1: Variable and Value Ordering 变量与值的排序

The heuristic strategies of ordering 排序的启发式策略

- Minimum-remaining-values (MRV) 最小剩余值

  to choose the variable with the fewest "legal" values, also has been called the "most constrained variable".

  选择具有最少“合法”值的变量，也被称为“最受约束变量”。

- Degree heuristic 程度启发式

  to reduce the branching factor on future choices by selecting the variable that is involved in the largest number of constraints on other unassigned variables.

  通过选择参与其它未分配变量的最大约束数，来减少未来选择的分支因子。

- Least-constraining-value heuristic 最少约束值启发式

  to prefer the value that rules out the fewest choices for the neighboring variables in the constraint graph.

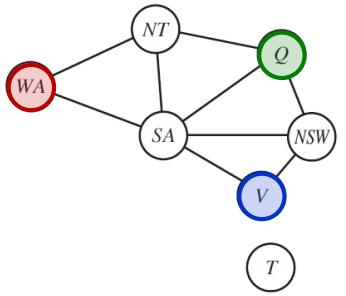
  尽量选取能排除约束图中相邻变量最少选择的值。

Discussion 2: Inference in Search 搜索中的推理

- The inference forward checking can be powerful in a search.
  前向检查推理在搜索中会很有用。

Example: Backtracking search with forward checking 具有前向检查的回溯搜索

	WA	*NT*	*Q*	*NSW*	*V*	*SA*	*T*
Initial domains	R G B	R G B	R G B	R G B	R G B	R G B	R G B
After *WA=red*	(R)	G B	R G B	R G B	R G B	G B	R G B
After *Q=green*	(R)	B	(G)	R B	R G B	B	R G B
After *V=blue*	(R)	B	(G)	R	(B)		R G B



- *WA=red* is assigned first; then it deletes "R" from the domains of the neighboring variables *NT* and *SA*.
  WA=red先被赋值；然后从相邻变量NT和SA的范畴中删除“R”。
- After *Q=green* is assigned, "G" is deleted from the domains of *NT*, *SA*, and *NSW*.
  Q=green被赋值后，从NSW、SA以及NSW的范畴中删除“G”。
- After *V=blue* is assigned, "B" is deleted from the domains of *NSW* and *SA*, leaving *SA* with no legal values.
  V=blue被赋值后，从NSW和SA的范畴中删除“B”，剩下SA没有合法值。

Discussion 3: Intelligent Backtracking 智能回溯

- BACKTRACKING-SEARCH algorithm has a policy when a search fails: back up to the preceding variable and try a different value.

  BACKTRACKING-SEARCH算法搜索失败时有一个策略：回到先前的变量并尝试不同的值。

  - This is called chronological backtracking.

    这被称为按时间回溯。

- E.g., for a variable ordering 例如，对于一个变量排序

  $\{Q, NSW, V, T, SA, WA, NT\}$,

  the partial assignment 该部分赋值

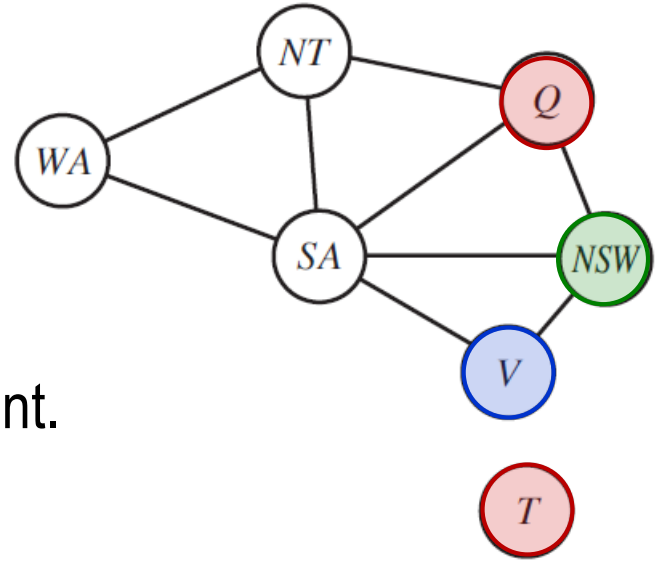
  $\{Q=red, NSW=green, V=blue, T=red\}$.

  - Try next variable SA, we see that every value violates a constraint.

    试图对下一个变量*SA*赋值时，我们看到每个值都违反约束。

  - Backjumping would jump over T and try a new value for V.

    后退跳跃法将跳过*T*并尝试将一个新的值赋给*V*.



Thank you for your attention!