

# 高级算法设计与分析



Advanced Computer Algorithm Design &  
Analysis

---

2022.10

吕志鹏

[zhipeng.lv@hust.edu.cn](mailto:zhipeng.lv@hust.edu.cn)



# 回顾

---

1. 什么是局部搜索
2. 邻域的概念
3. 邻域结构如何确定
4. 局部搜索算法的框架

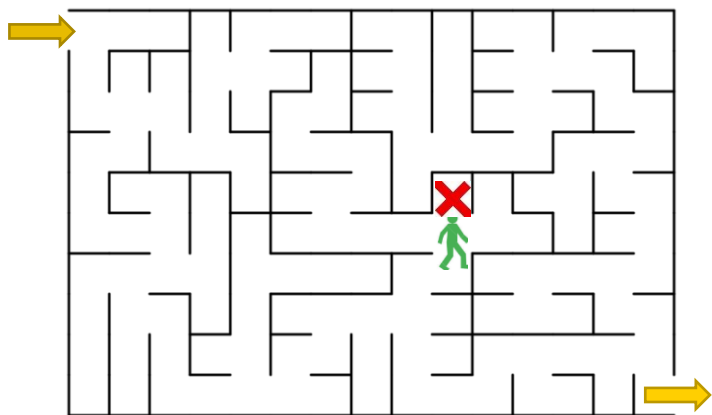




# **1. 禁忌搜索算法的基本原理**

# 背景

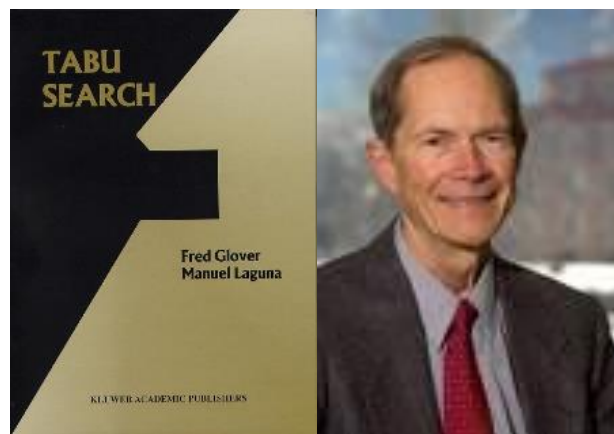
- 对于简单的局部搜索算法，简单易行，但很容易陷入局部最优解，且无法跳出；
- 人类在做选择时具有记忆能力，如走迷宫；
- 借鉴人类的智能思考特性，在局部搜索的基础上采用禁忌策略尽量避免迂回搜索就构成了禁忌搜索算法。



✗ 表示之前执行过，再次选择动作时将尽量避免，这样可以避开迂回搜索

# 算法的提出

- 早在1977年,Fred Glover就提出了禁忌搜索算法,并用来求解整数规划问题,在1986年被正式提出。
- 1989--1990年Glover在《Operation Research of America》上系统地介绍了禁忌搜索算法及一些成功的应用,自此禁忌搜索算法引起了广泛的关注。





# 算法基本思想

---

- **禁忌搜索(TabuSearch, TS)算法**是继遗传算法后出现的又一种元启发式算法，是局部搜索算法的推广。
- **算法基本思想**：标记并禁忌近期的搜索过程，阻止算法重复进入。



# 算法基本思想

---

## ■ 具体实现思路:

1. 采用邻域选优, 但为了能逃离局部最优解, 算法能够接受劣解。
2. 为了避免循环, 记录最近接受的一些移动, 在以后的迭代中加以禁止。
3. 当达到一定的迭代次数后, 被禁忌的动作不再禁忌。



# 算法的特点

---

1. 禁忌——在一定时间内禁止重复前面做过的动作。
  - 目的：跳出局部最优解。
  - 例：一日三餐：
    - a) 第一天中午：鱼+米饭
    - b) 第二天中午：为了营养最大化均衡，第一天中午的选择将被禁忌一段时间，直到一定的天数后被解禁。



# 算法的特点

2. 移动动作不是随机的：

- 选择最好的可移动动作；
- 不像爬山算法只能上坡，禁忌搜索算法即可以上坡也可以下坡。

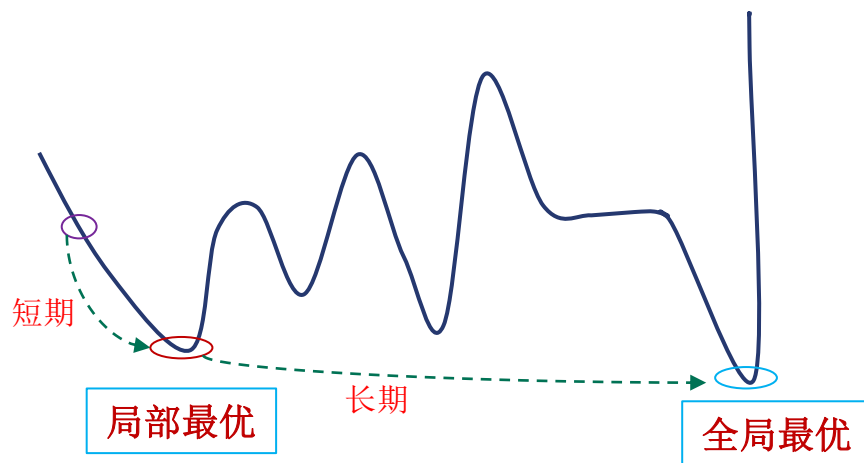
3. 禁忌约束并非在任何情况下都是不可违反的。

4. 短期

- 得到局部最优

5. 长期

- 集中性和疏散性
- 跳出局部最优获取全局最优





# 算法的组成要素

---

禁忌搜索算法中很多构成要素对搜索的速度与质量至关重要,构成要素包括:

- ✓ .编码方式(Encode)
- ✓ .适值函数 (一般直接用目标函数)
- ✓ .解的初始化
- ✓ .移动(Moving)与邻域(Neighborhood)
- ✓ .禁忌表(TabuList)
- ✓ .选择策略(SelectionStrategy)
- ✓ .特赦准则(AspirationRule)
- ✓ .停止条件(Stoppingcondition)



# 算法的组成要素

---

## ✓ 编码方式(Encode)

编码就是将实际问题的解用一种便于算法操作的形式来描述，一般用数学形式编码

根据问题具体情况,可以灵活地选择编码方式,例如:

1. 对于背包问题,可以采用0-1编码,0表示不选择这件物品,1表示选择这件物品.
2. 对于图着色问题使用整数表示颜色, 如1-红色, 2-黄色
3. 对于八皇后问题, 可以使用坐标表示皇后



# 算法的组成要素

---

## ✓ 初始解

1. 禁忌搜索算法可以随机给出初始解，也可以事先使用其他启发式等算法给出一个较好的初始解
2. 由于禁忌搜索算法主要是基于邻域搜索，初始解的好坏对搜索的性能影响很大.
3. 例：对于八皇后问题的初始解，使用均匀分布的随机构造法（即每行每列只有一个皇后）



# 算法的组成要素

---

## ✓ 禁忌表 (Tabulist)

1. 在禁忌搜索算法中，禁忌表是用来防止搜索过程中出现循环，避免陷入局部最优的。它通常记录最近接受的若干次移动，在一定次数内禁止再次被访问；
2. 禁忌表是禁忌搜索算法中的核心，它的功能和人类的短期记忆功能十分相似，因此又称为“短期表”；
3. 禁忌表两要素：
  - 禁忌对象:即放入禁忌表中的元素。
  - 禁忌长度 (*tabutenure*) :即禁忌表的大小。



# 算法的组成要素

---

## ■ 禁忌对象

- 通常禁忌对象选择的是解决方案的**属性**（禁忌对象的选择十分灵活, 可以是最近访问过的点、状态、状态的变化以及目标值等），而不是解本身（代价太昂贵）
- 例如对于八皇后问题，将两两交换的对象（横坐标）作为禁忌对象，也即禁止了状态的变化。

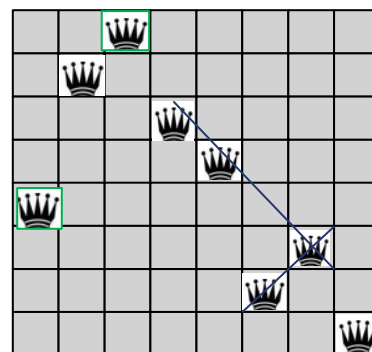
# 算法的组成要素

## ■ 禁忌对象

禁忌对象主要有以下三种方式给出:

- ① 以状态本身或者状态的变化作为禁忌对象;
- ② 以状态分量或者状态分量的变化作为禁忌对象;
- ③ 将目标值作为禁忌对象

- 如八皇后问题: 邻域动作  
→ 交换皇后的横坐标



移动  
<1,5>

禁忌方式	禁忌对象
①	动作<1,5>
②	1和5（分开禁忌）



# 算法的组成要素

---

## ■ 禁忌对象

- 以上三种做法:
  1. 第一种做法的禁忌范围较小;
  2. 第二种做法的禁忌范围适中;
  3. 第三种做法的禁忌范围较大.
- 如果禁忌范围比较大, 则可能陷入局部最优解; 反之, 则容易陷入循环。
- 实际应用时, 要根据问题的规模、禁忌表的长度等具体情况来确定禁忌对象。





# 算法的组成要素

---

- **禁忌长度:**

- 禁忌对象进入禁忌表后, 经过确定的迭代次数, 才能从禁忌表中退出。
- 禁忌长度影响搜索时间。若禁忌表长, 广域搜索性能好; 若禁忌表短, 则其局域搜索性能好。
- 经验表明, 有效的禁忌长度取决于问题实例的规模。



# 算法的组成要素

---

- **禁忌长度:**

- 禁忌表长度的设定方法:

1. **禁忌长度 $t$ 固定不变**。第一种：取与问题无关的常数，如 $t=5, 7, 11$ 等；第二种：禁忌长度应该与问题的规模 $n$ 相关，如 $t=n$ ，此法方便简单易实现。
2. **禁忌长度 $t$ 随迭代的进行而改变**。根据迭代的具体情况，按照某种规则，禁忌长度在区间内变化。

禁忌长度的区间可与问题无关，如 $[1, 10]$ ；或者与要求解问题的规模有关，如 $[0.9n, 1.1n]$ 。该区间的两个端点也可随着迭代的进行而改变。



# 算法的组成要素

---

## ■ 选择策略

- 选择策略就是从邻域中选择一个比较好的解作为下一次迭代初始解的方法，可以表示为：

$$x' = \underset{s(x) \in V}{\text{opt}} \ s(x) = \arg \left[ \max / \min_{s(x) \in V} c'(s(x)) \right]$$

$x$ : 当前解;

$x'$ : 选出的邻域最好解;

$s(x) \in V$ : 邻域解;

$c'(s(x))$ : 候选解 $s(x)$ 的适值函数;

$V \subseteq S(x)$ : 候选解集，它是邻域的一个子集。



# 算法的组成要素

---

## ■ 特赦准则

- 在某些特定的条件下，即使某个动作禁忌，也接受。该动作满足的这个特定条件，称为特赦准则(Aspiration criterion),也称为渴望水平、蔑视准则等。



# 算法的组成要素

---

## ■ 终止条件

停止禁忌搜索的条件通常有以下几种：

1. 达到最大迭代次数
2. 达到目标函数值未改进的最大时长
3. 找到理想解（如对于八皇后问题：冲突数为0时）

**禁忌策略和特赦准则是禁忌搜索的两大核心，用来平衡集中性和疏散性。**



### 3. 禁忌搜索算法流程及算例



# 禁忌搜索的基本执行步骤

---

- 第1步:初始化。给出初始解,禁忌表设为空;
- 第2步:判断是否满足停止条件。若满足,输出结果,算法停止;否则继续以下步骤;
- 第3步:对于候选解集中的最好解,若禁忌,判断其是否满足特赦准则。如果满足更新当前解,转第5步;否则继续以下步骤;
- 第4步:选择候选解集中不被禁忌(不在禁忌表中)的最好解作为当前解;
- 第5步:更新禁忌表,转第2步。



# 初始化

- 构造初始解

- 初始解的构造根据具体问题而定
- 这里，使用均匀分布确保每行每列只有一个元素的前提下随机构造八皇后问题的初始解

- 冲突函数：

- 在八皇后问题中，因为每次都确保每行每列有且只有一个皇后，所以冲突只可能发生在斜对角线上，如果两个节点冲突，则有：

$$\boxed{i - q_i = j - q_j \text{ 或 } i + q_i = j + q_j}$$

————→ 左对角线冲突

————→ 右对角线冲突

$i$ :第 $i$ 行的皇后;

$q_i$ :第 $i$ 行的皇后在第 $q_i$ 列;

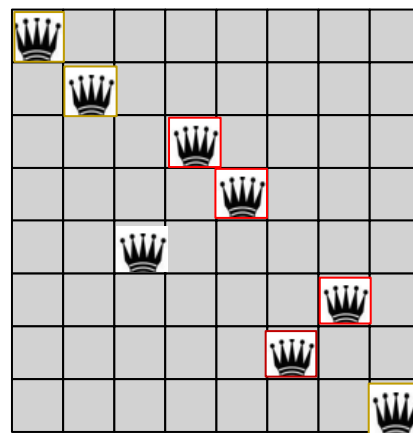


# 初始化

## ■ 构造禁忌表

	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2	-	-	0	0	0	0	0	0
3	-	-	-	0	0	0	0	0
4	-	-	-	-	0	0	0	0
5	-	-	-	-	-	0	0	0
6	-	-	-	-	-	-	0	0
7	-	-	-	-	-	-	-	0
8	-	-	-	-	-	-	-	-

初始解 $s=\{1,2,4,5,3,7,6,8\}$



初始解

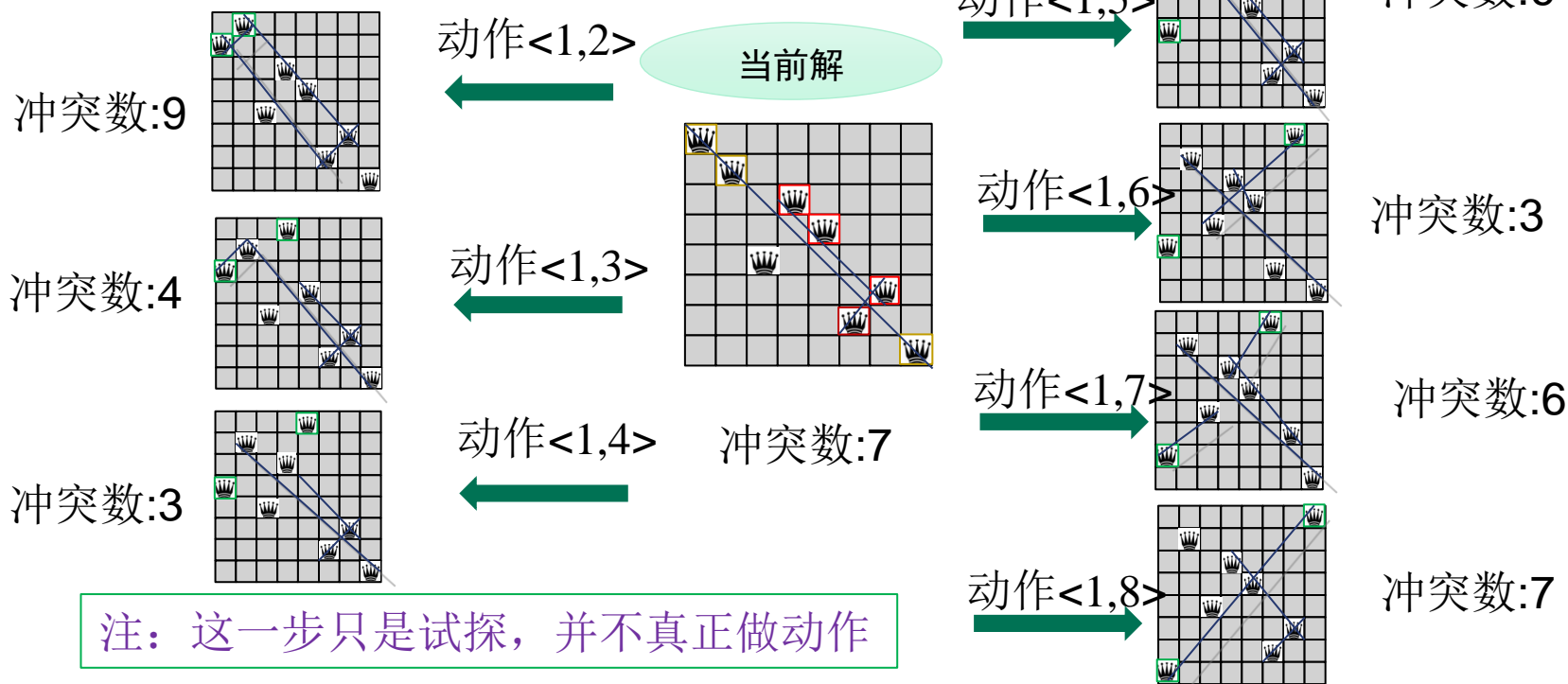
冲突数 $f:7$

禁忌表的横纵都表示棋盘行数（皇后的横坐标），即皇后两两之间交换横坐标后，一定迭代长度内不能再交换回去

# 寻找最好的邻域动作

- 邻域动作：交换两皇后之间的横坐标

依次试探寻找最好的动作（冲突数为0的皇后两两之间不做试探-非关键动作）





# 寻找最好的邻域动作

- 邻域动作：交换两皇后之间的横坐标

依次试探寻找最好的动作（冲突数为0的皇后两两之间不做试探-非关键动作）

依次试探：<2,3>、<2,4>、<2,5>、<2,6>、<2,7>、<2,8>

<3,4>、<3,5>、<3,6>、<3,7>、<3,8>

<4,5>、<4,6>、<4,7>、<4,8>

<5,6>、<5,7>、<5,8>

<6,7>、<6,8>

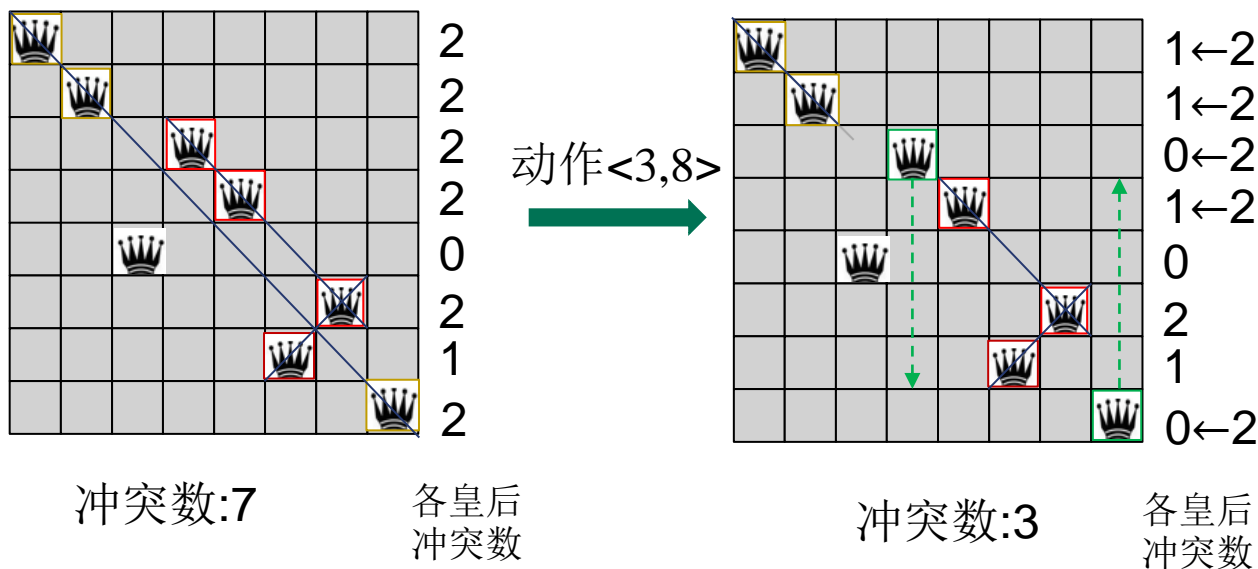
<7,8>

并找到最好的邻域动作（**禁忌或非禁忌**），这里最好的动作有多个（冲突数为3），随机选择一个，如<3,8>

# 寻找最好的邻域动作

- 邻域动作：交换两皇后之间的横坐标

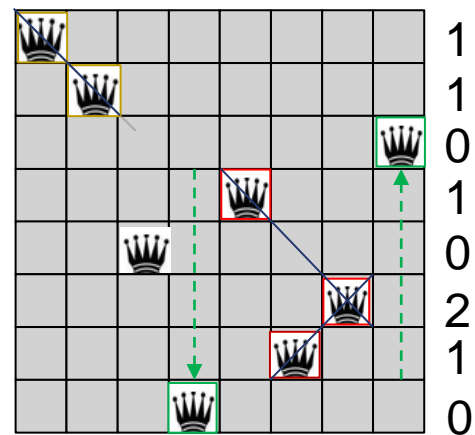
依次试探寻找最好的动作（冲突数为0的皇后两两之间不做试探-非关键动作）



# 执行更新

- 更新禁忌表、冲突数、当前最好解：

	1	2	3	4	5	6	7	8
1	-	0	0	0	0	0	0	0
2	-	-	0	0	0	0	0	0
3	-	-	-	0	0	0	0	12
4	-	-	-	-	0	0	0	0
5	-	-	-	-	-	0	0	0
6	-	-	-	-	-	-	0	0
7	-	-	-	-	-	-	-	0
8	-	-	-	-	-	-	-	-

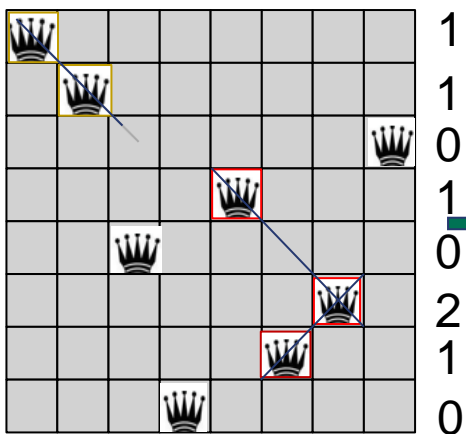


执行动作<3,8>后 各皇后  
冲突数

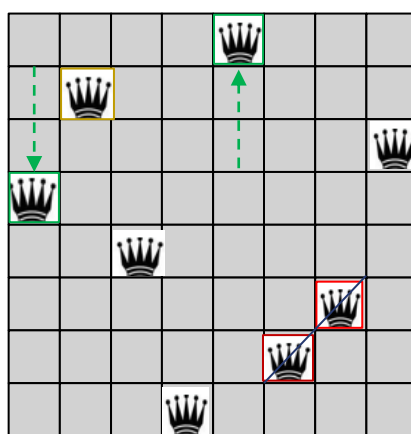
冲突数：3

# 重复禁忌搜索过程

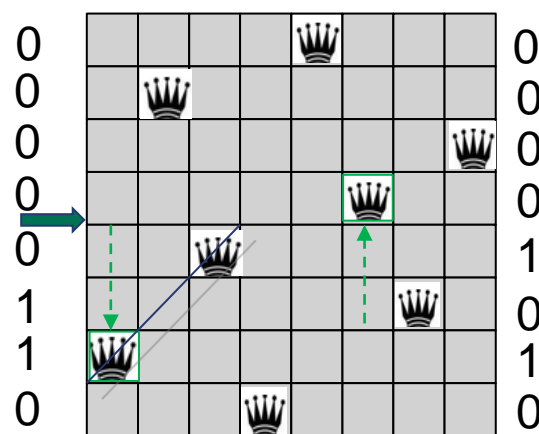
- 从当前解出发，再次寻找最好的邻域动作，并执行该动作；
- 直到冲突数为0或者迭代数大于最大迭代次数



冲突数：3



执行动作<1,4>后  
冲突数：1

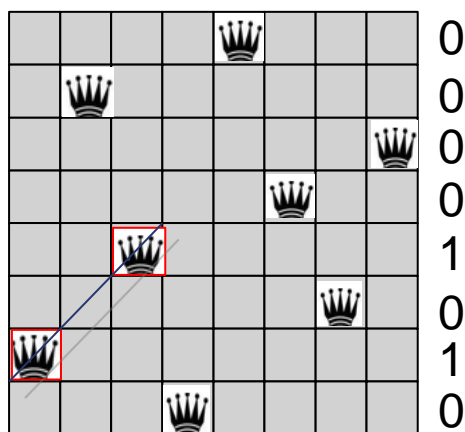


执行动作<4,7>后  
冲突数：1

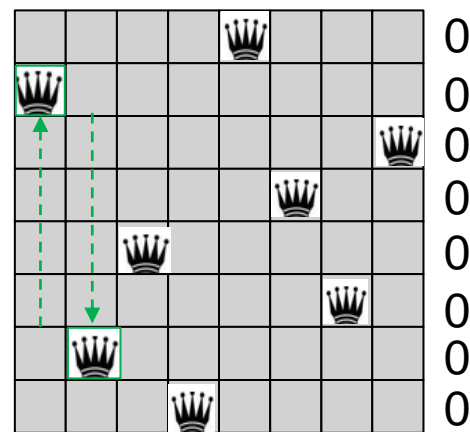
存在多个执行动作，随机选择一个

# 重复禁忌搜索过程

- 直到冲突数为0或者迭代数大于最大迭代次数



冲突数：1

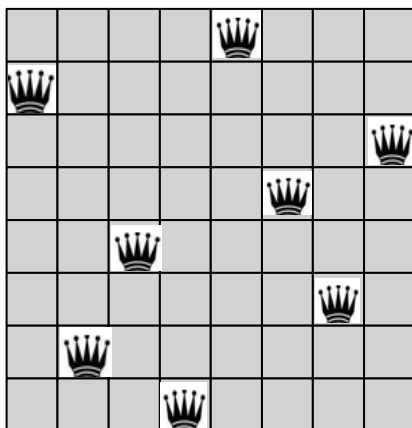


执行动作<2,7>后

冲突数：0

# 结束禁忌搜索过程

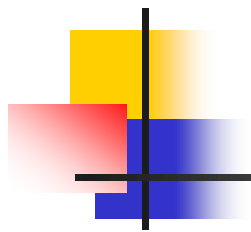
- 找到一个可行解 $S=\{5,1,8,6,3,7,2,4\}$ ，结束



迭代次数	交换动作	冲突对数	当前最好解
1		7	{1,2,4,5,3,7,6,8}
2	<3,8>	3	{1,2,8,5,3,7,6,4}
3	<1,4>	1	{5,2,8,1,3,7,6,4}
4	<4,7>	1	{5,2,8,6,3,7,1,4}
5	<2,7>	0	{5,1,8,6,3,7,2,4}

过程



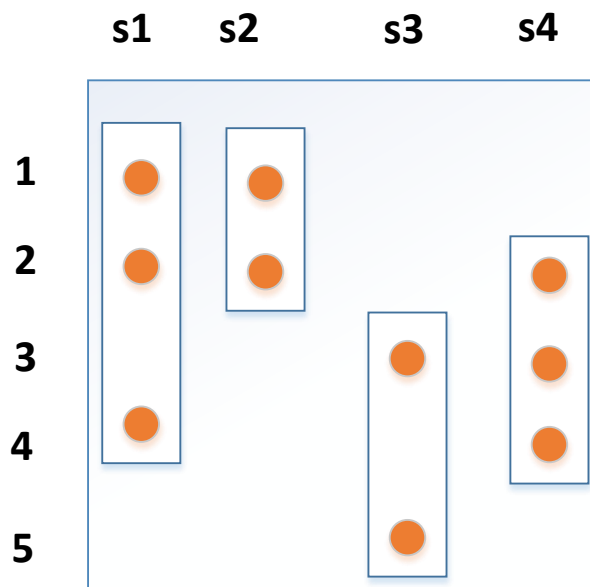


## 4. 禁忌搜索算法应用案例

--图着色、集合覆盖

# 集合覆盖——问题描述

- 已知一个由一系列元素构成的集合 $X = \{x_1, x_2, \dots, x_n\}$ , 集合集 $S = \{s \mid s \subseteq X\}$ 并且有 $\bigcup_{s \in S} s = X$ , 目标是发现一个子集合集 $F \subseteq S$ 能够包含 $X$ 中的所有元素且要使得 $|F|$ 个数最少。





# 数学模型

---

**min**

$$\sum_{j=1}^m x_j$$

**s.t**

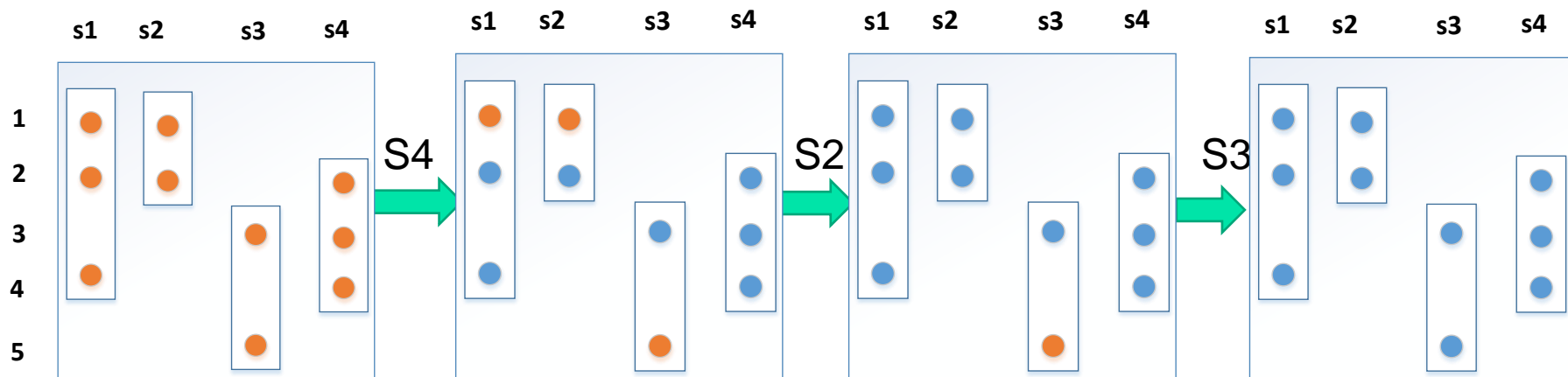
$$\sum_{j: e_i \in S_j} x_j \geq 1, i = 1, 2, \dots, n$$

$$x_j \in \{0, 1\}, j = 1, 2, \dots, m$$

1. 约束条件确保所有元素 $e_i$ 至少被一个集合覆盖;
2.  $x_j = 1$ 表示第 $j$ 列集合被选中;

# 初始解

- 对于集合覆盖问题的初始解可以使用贪心构造的方法：
  1. 每次从 $S$ 中挑出覆盖当前（未被覆盖）元素最多的集合；
  2. 直到所有元素都被覆盖。



初始解:  $\{s_2, s_3, s_4\}$



# 邻域动作

---

- 集合覆盖问题的邻域动作分解为两步：
  1. 删除一个集合；
  2. 添加一个集合。
    - 若当前选择的子集集合  $F$  能覆盖所有元素，则删除一个集合；
    - 若当前选择的子集集合  $F$  无法覆盖所有元素，则交换（即删除一个已选集合并添加一个未选集合）一对集合。
- 邻域评估：未被覆盖的元素数最小



## 禁忌对象和禁忌长度

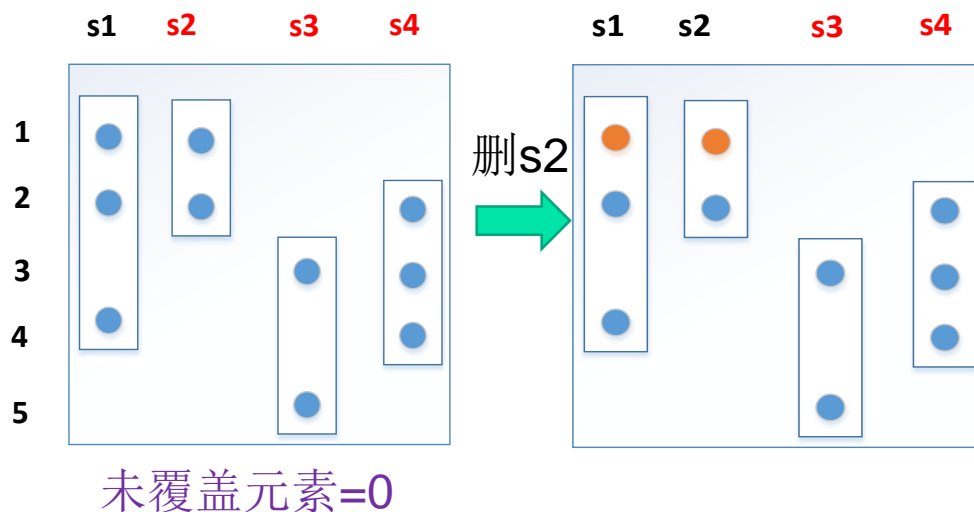
- 因为这里的添加和删除动作是单独执行，所以选择分量变化作为禁忌对象：

Remove $s_j$	
$s_1$	0
$s_2$	0
$s_3$	0
$s_4$	0

Add $s_j$	
$s_1$	0
$s_2$	0
$s_3$	0
$s_4$	0

- 禁忌长度具体的选择根据算例的规模而定，并且remove对象的禁忌长度还要根据当前已选集合的个数确定。

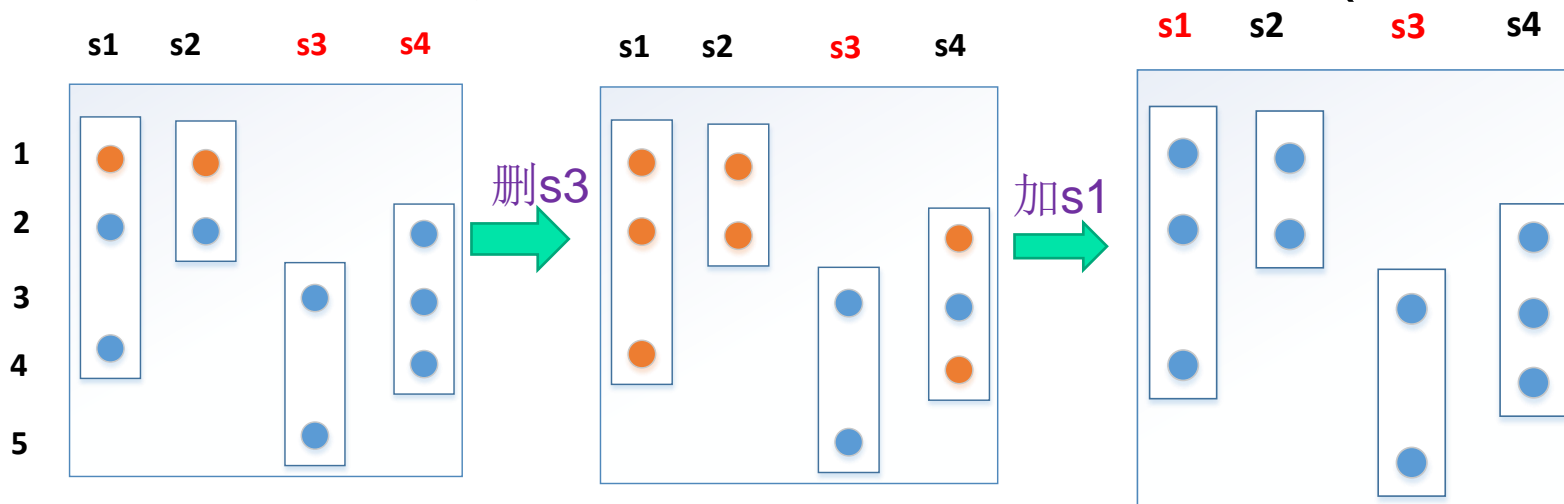
- 例：初始化后未覆盖元素为0，邻域动作：删除一个集合



Add $s_j$	
$s_1$	0
$s_2$	1
$s_3$	0
$s_4$	0

依次试探 $s_2, s_3, s_4$ ，最终选择删除后未覆盖数最小的 $s_2$   
(元素5只能被 $s_3$ 覆盖，所以 $s_3$ 是已经确定的集合)

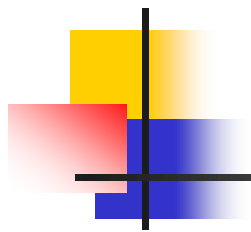
- 当前元素未覆盖数=1, 邻域动作: 交换集合对 (先删后加)



删除的集合只能选s4

添加: 依次试探s1,s2 (s2当前是禁忌的), 最终选择删除后未覆盖数最小的s1

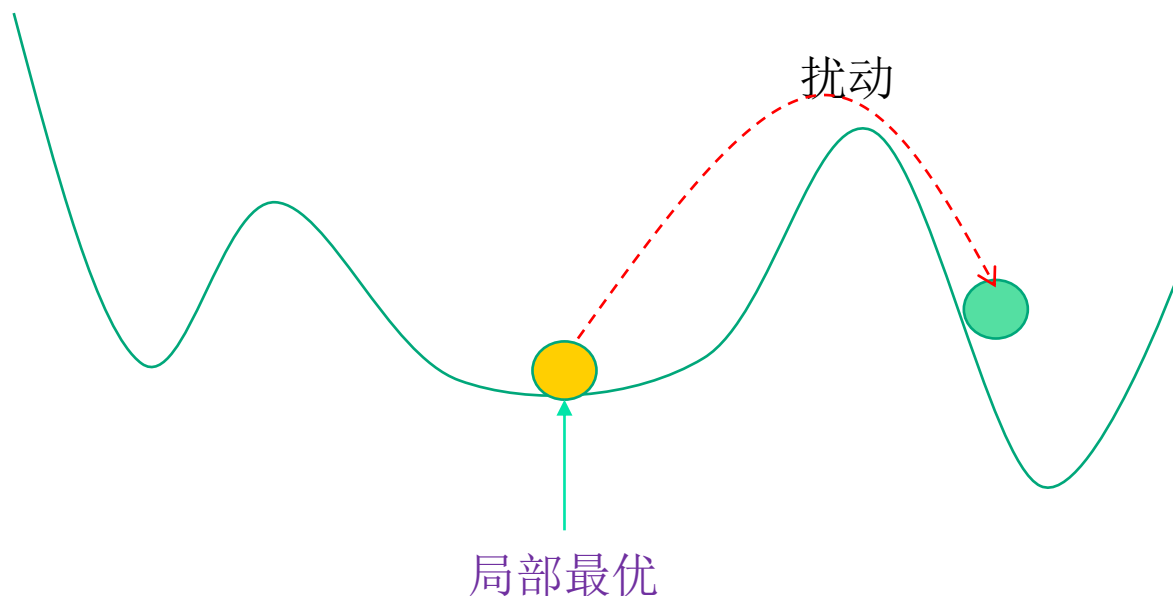





## 5. 禁忌搜索算法的改进及变形

## 加入扰动策略

- **引入原因：**虽然禁忌搜索有一定的跳出局部最优能力，但有时也可能无法跳出，为了解决这一问题，引入了扰动策略。
- **添加策略：**当算法在一定的迭代次数内不能改进时，使用扰动策略。





# 加入扰动策略

---

- **常用的扰动方法：**
  - 随机扰动（即随机的选择一些邻域动作并执行）：分为完全随机和针对关键元素的随机。
- **优点：**可以增强算法的疏散性，从而使得算法跳出局部最优；引领搜索过程到更有前途的新邻域。
- **缺点：**扰动幅度太小无法跳出局部最优；扰动幅度太大可能跳过全局最优。



# 主动式禁忌搜索 (RTS)

---

## ■ 历史

- 1994年被提出
- 提出者：Robert Battiti和Giampietro Tecchioli

## ■ 思想

- 保留了原禁忌搜索的思想，增加了如下思想：
  - 一种完全自动化的方法，可以根据问题和当前搜索的演变来调整禁忌表的长度；
  - 一种逃逸策略：当第一个机制不足时，用于使搜索多样化。



# 主动式禁忌搜索 (RTS)

---

## ■ 特征

### 1. 动态地自调整禁忌列表长度

当有迹象表明需要多样化时，禁忌表长度会增大，当迹象消失时，禁忌表的长度相应会减小。

### 2. 逃逸机制

当过多地重复已执行过的解时，将触发逃逸阶段。逃逸机制避免了长期循环和混乱陷阱。

### 3. 使用搜索历史的快速算法

过去事件的存储和访问是通过哈希方案执行的。



# 主动式禁忌搜索 (RTS)

---

## ■ 优点

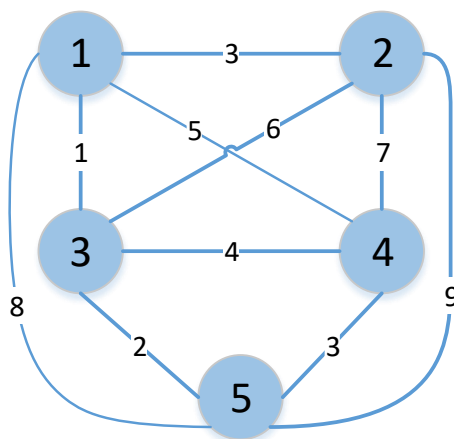
1. 动态地自调整禁忌列表长度
2. 避免了循环
3. 有效的探索和利用了解空间
4. 充分利用了搜索历史

## ■ 缺点

1. 引入的参数太多

# 单元测试

1. 如果八皇后问题的初始解是完全随机的（横、纵、对角线都有可能存在冲突），请给出一种邻域动作和禁忌表的设计。
2. 请仿照第三节中八皇后的例子，设计并写出一种如下图TSP问题的禁忌搜索过程。



# 总结

