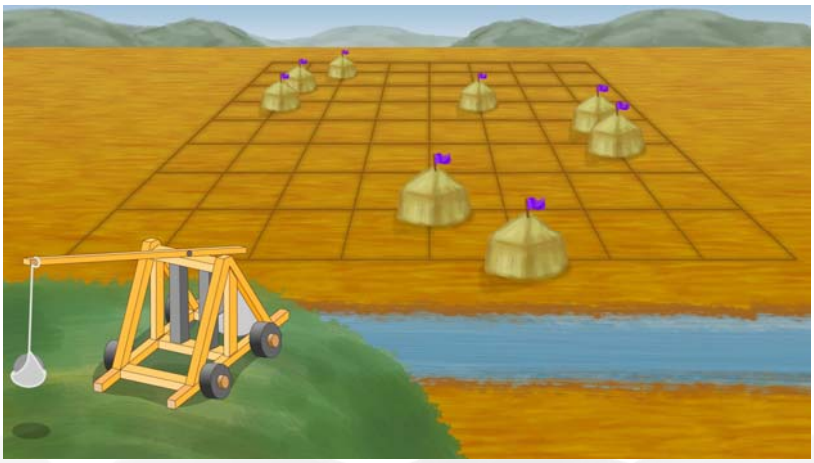
纯划分

李浩文、彼得·斯塔基

香港中文大學 The Chinese University of Hong Kong THE UNIVERSITY OF MELBOURNE

投石车问题

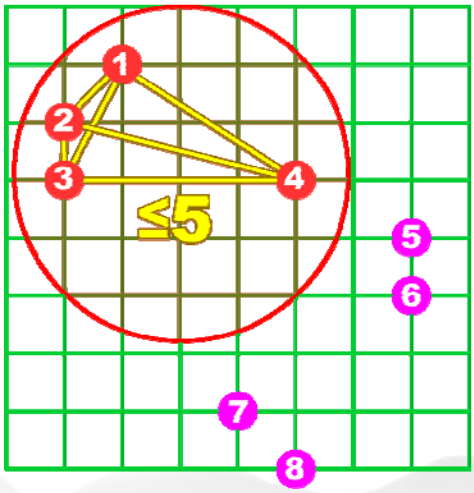


2

投石车问题

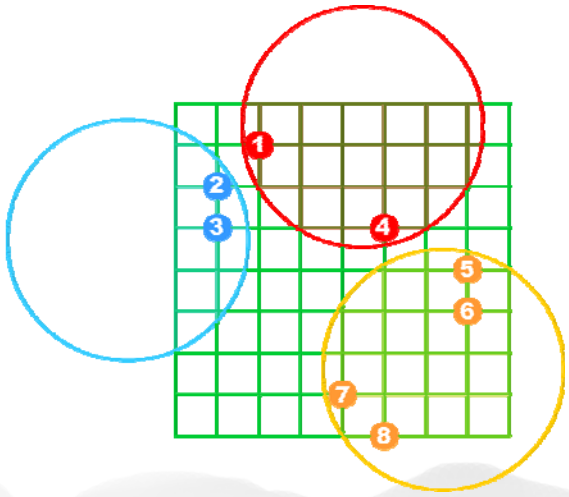
3

投石车问题



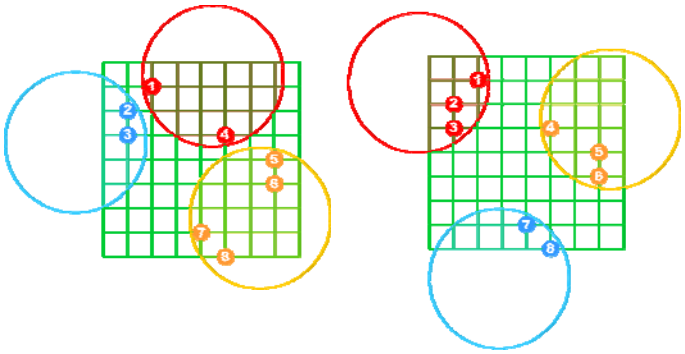
4

投石车问题



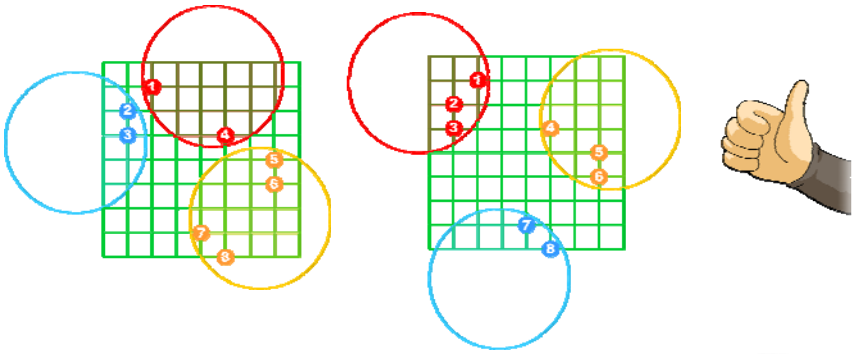
5

投石车问题



6

投石车问题



7

一个聚类问题

- 给定nSpots个点, 把它们划分到最多k个类中使得
  - 同一类中任何两点最多相距maxSep
  - 最大化不同类的两点之间的最短距离

8

纯划分问题

- 确定一个函数 f: DOM ➔ COD
- 可以看成是一个划分问题

- 但如果我们只希望划分DOM
  - DOM = 点 and COD = 类
  - COD无指明意义且未给定

- 容易的情况: 划分类个数不超过k
  - COD = 1..k;
- 更难的情况: 我们不知道有多少个类？

9

纯划分

- 把DOM划分为最多k部分
- 对DOM的每一个元素, 用一个变量来指明被分配到了哪个划分类
- 简单的数组表示

```
array[DOM] of var 1..k: x
```

- 注意会出现多重表示，因为用哪个数字表示类别是无关紧要的
- 例如， DOM = 1..4, k = 3
  - x = [1,2,1,3] ➔ {1,3} {2} {4}
  - x = [3,1,3,2] ➔ {2} {4} {1,3}
  - x = [3,2,3,1] ➔ {4} {2} {1,3}

10

多重表示

- 添加约束使每种划分只有一个表示
- 值对称
  - 1..k中的数字是互相对称的
  - 我们不关心类的名称（数字）
- 这个特点出现在很多离散优化问题中

11

去除值对称

- 我们如何促使只有一个表示?
- 以各自的最小元素对划分集合进行排序
  - 例如，{2}，{4}，{1,3} 被排序为 {1,3}，{2}，{4}
  - 唯一地表示成 x = [1,2,1,3]
- 我们如何用约束来限制?
  - 第i个划分类的最小元素比第i+1个划分类的最小元素小

```
forall(i in 1..k-1)
   (min([j | j in DOM where x[j] = i])
    < min([j | j in DOM where x[j] = i+1]));
```

12

投石车问题数据 (catapult.mzn)

- 定义聚类实例的数据

```
int: nSpot; % points to be clustered
set of int: SPOT = 1..nSpot;

array[SPOT,SPOT] of float: dist;
   % distance between two points

int: k; % number of clusters
set of int: CLUSTER = 1..k;

float: maxSep;
```

13

投石车问题 (catapult.mzn)

- 决策变量

```
array[SPOT] of var CLUSTER: shot;
```

- 值对称

```
forall(i in 1..k-1)
   (min([j | j in SPOT where shot[j] = i])
    < min([j | j in SPOT where shot[j] = i+1]));
```

- 但MiniZinc为去对称提供了更好的工具

14

value_precede_chain

- MiniZinc包含了一个用于去值对称的全局约束

```
value_precede_chain(array[int] of int: c,
                    array[int] of var int: x)
```

- 强制c[i]在x中的第一次出现先于c[i+1]在x中的第一次出现

  例如, `value_precede_chain([1,2,3], x)`

  `x = [1,1,2,3]` ✓, `[1,3,1,2]` ✗,
  `[1,2,1,2]` ✓

- 我们可以用以下的约束来替换我们的去对称约束

```
value_precede_chain(
   [i | i in CLUSTER], shot)
```

15

投石车问题 (catapult.mzn)

- 约束: 类中点之间的最大距离

```
forall(i,j in SPOT where i < j
   /\ shot[i] = shot[j])(dist[i,j] <= maxSep);
```

- 目标

```
float: maxdist = max([dist[i,j] | i,j in SPOT]);
var float: obj = min(i,j in SPOT where i < j)
   (if shot[i]=shot[j] then
      maxdist else dist[i,j] endif);
solve maximize obj;
```

16

求解全局约束模型

- 基于已有数据，用修改后的模型求解

```
[1, 1, 1, 1, 2, 2, 2, 2], obj = 2.236
----------
[1, 1, 1, 2, 2, 2, 2, 2], obj = 3.606
----------
==========
```

17

投石车问题 (catapultExact.mzn)

- 这个模型并不要求**正好**而只是**最多**k个类
  - 可以更少

- 我们可以很容易地添加一条约束来强制每个类都有成员
  - 下界 = 1
  - 上界 = nSpot - k + 1

```
global_cardinality_low_up_closed(shot,
  [i | i in CLUSTER], [1 | i in CLUSTER],
  [nSpot-k+1 | i in CLUSTER]);
```

18

求解确切聚类模型

- 基于已有数据，用确切聚类模型求解

```
[1, 1, 1, 1, 2, 3, 2, 2], obj = 1.0
----------
[1, 1, 1, 1, 2, 2, 3, 2], obj = 1.414
----------
[1, 1, 1, 1, 2, 2, 3, 3], obj = 2.236
----------
[1, 1, 1, 2, 2, 2, 3, 3], obj = 3.606
----------
==========
```

19

纯划分

- 如果我们不知道类的数量?
- 简单: 不会多于nSpots个划分
  - int: k = nSpots;

20

小结

- 纯划分问题伴随（其他条件）
  - **不多于k个**类
  - **已知k个**类
  - 类的数量未知
- **表示**类的数字是无关紧要的
  - 导致值对称
- **去除**值对称

  `value_precede_chain`

21

小结

- **聚**类问题是一个被充分研究的问题
  - 对于纯聚类问题应选用专门设计的聚类方法
- **半**监督聚类法会向问题添加约束，例如
  - **某些点必**须在相同或不同的类中
    - **用相等**（=）约束或者非等（≠）约束表达
  - 类大小的界限
    - **用全局**势约束表达
  - **其他种**类的约束
- **离散**优化适用于各种半监督聚类问题

22

图像引用

所有图像由Marti Wong设计提供, © 香港中文大学与墨尔本大学 2016

23