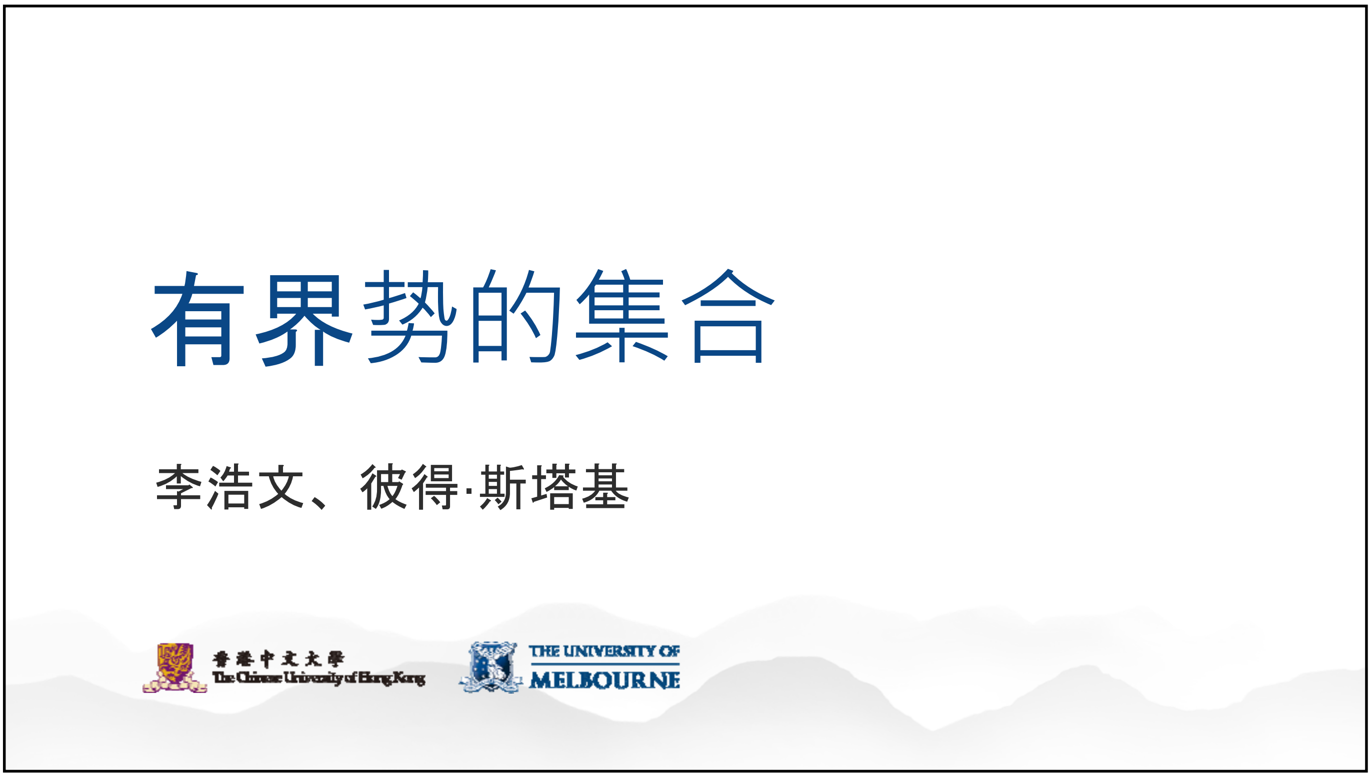
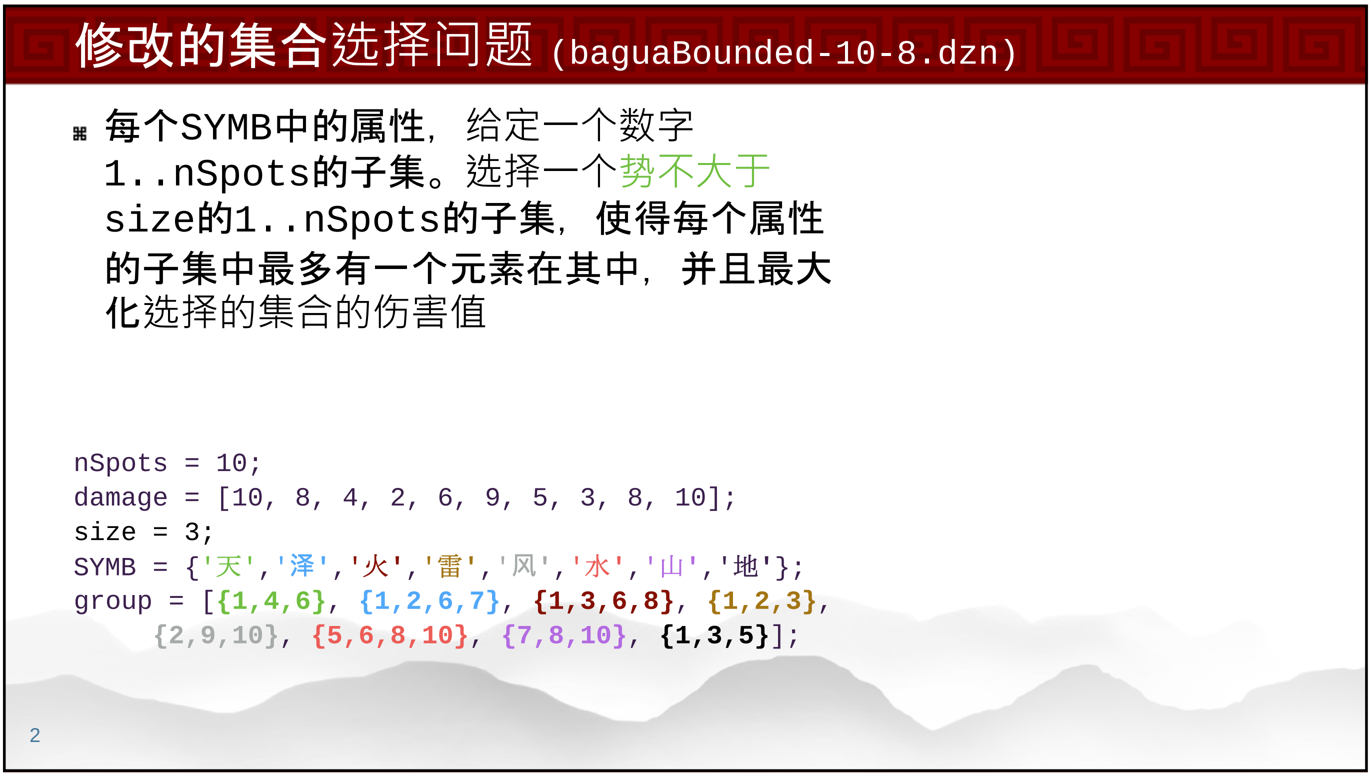


有界势的集合

李浩文、彼得·斯塔基

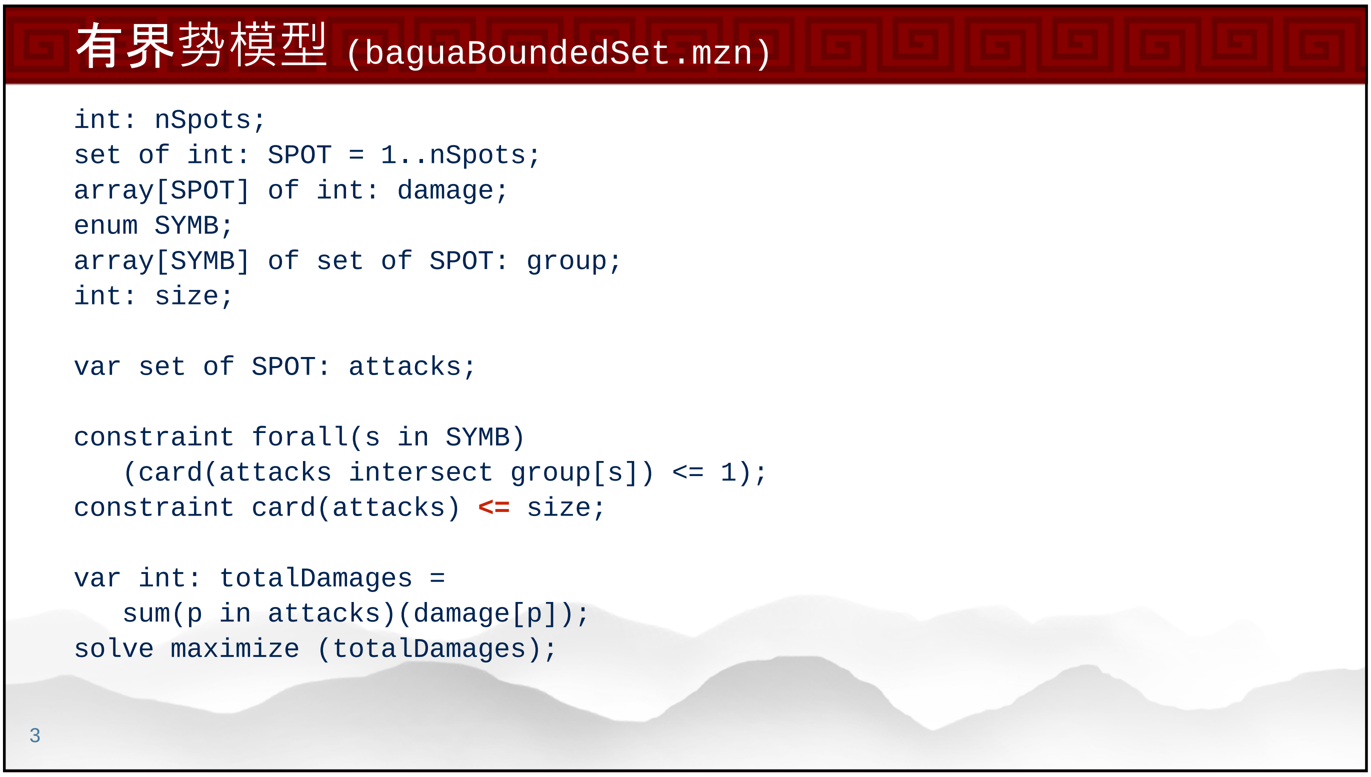


修改的集合选择问题 (baguaBounded-10-8.dzn)

- 每个SYMB中的属性，给定一个数字1..nSpots的子集。选择一个势不大于size的1..nSpots的子集，使得每个属性的子集中最多有一个元素在其中，并且最大化选择的集合的伤害值

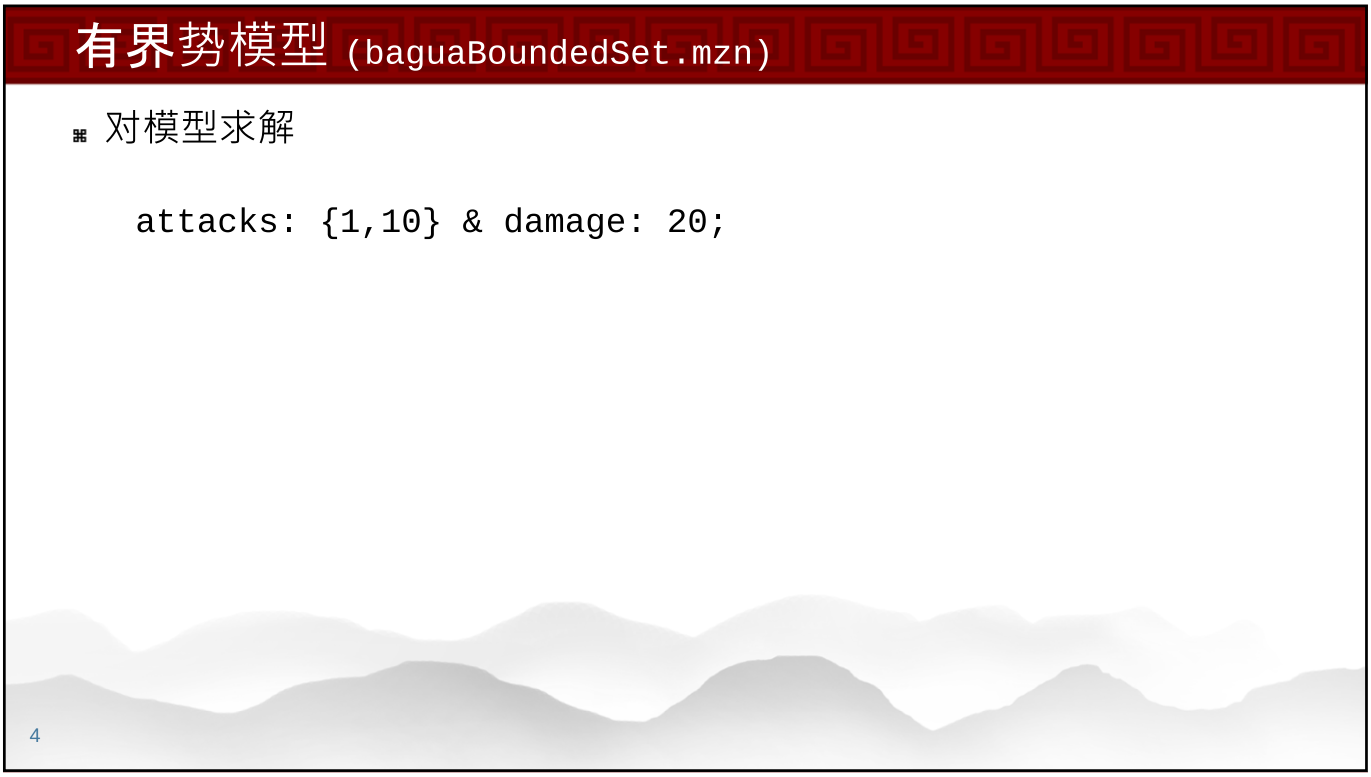
```
nSpots = 10;
damage = [10, 8, 4, 2, 6, 9, 5, 3, 8, 10];
size = 3;
SYMB = {'天','泽','火','雷','风','水','山','地'};
group = [{1,4,6}, {1,2,6,7}, {1,3,6,8}, {1,2,3},
     {2,9,10}, {5,6,8,10}, {7,8,10}, {1,3,5}];
```

2



有界势模型 (baguaBoundedSet.mzn)

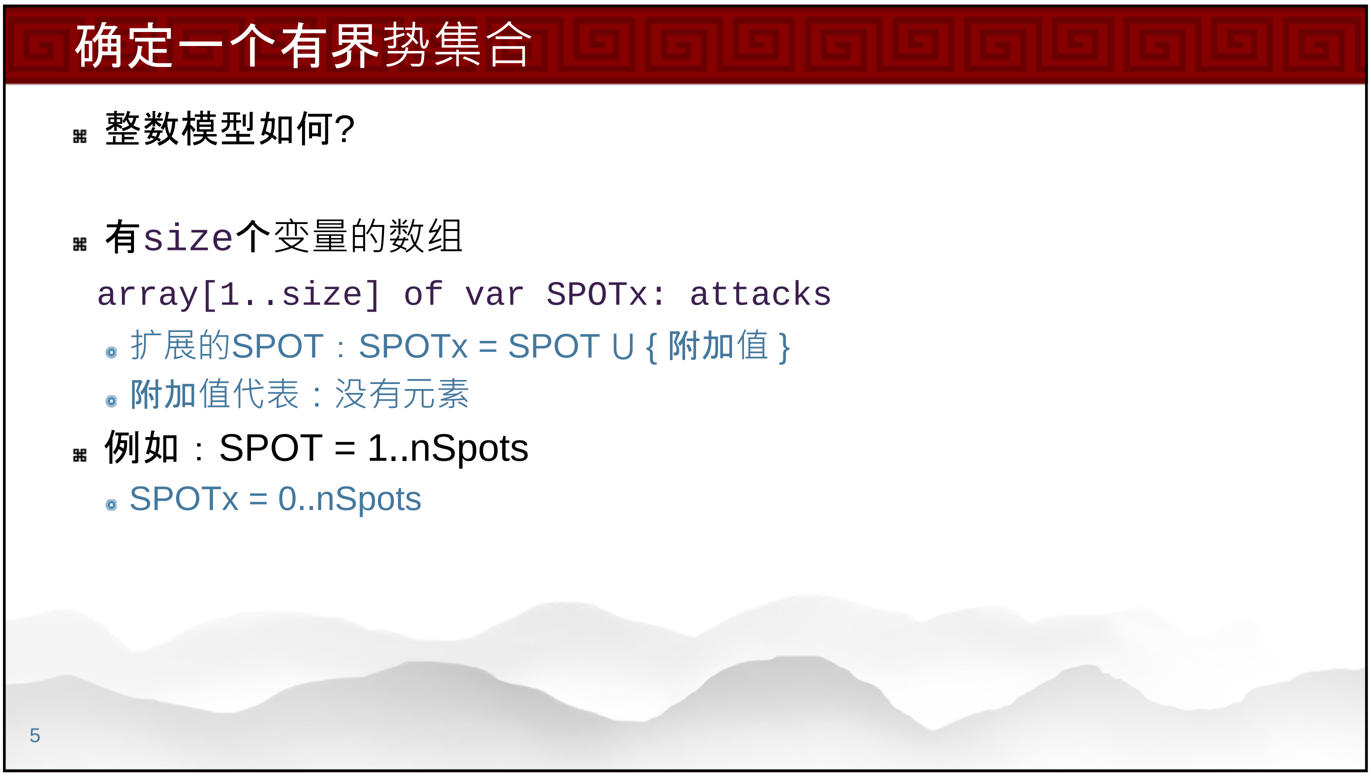
```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

var set of SPOT: attacks;

constraint forall(s in SYMB)
   (card(attacks intersect group[s]) <= 1);
constraint card(attacks) <= size;

var int: totalDamages =
   sum(p in attacks)(damage[p]);
solve maximize (totalDamages);
```



有界势模型 (baguaBoundedSet.mzn)

- 对模型求解

```
attacks: {1,10} & damage: 20;
```

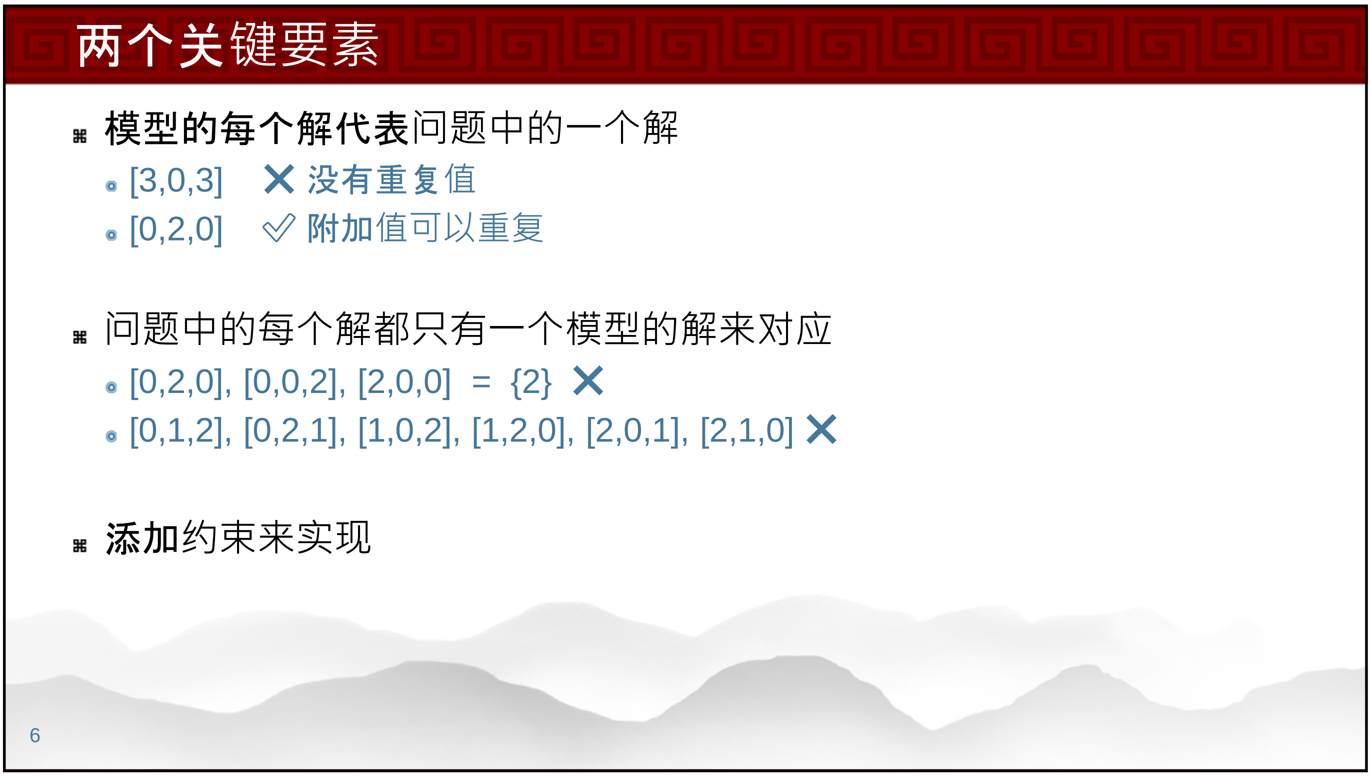


确定一个有界势集合

- 整数模型如何?
- 有size个变量的数组

  `array[1..size] of var SPOTx: attacks`
  - 扩展的SPOT：SPOTx = SPOT U { 附加值 }
  - 附加值代表：没有元素
- 例如：SPOT = 1..nSpots
  - SPOTx = 0..nSpots

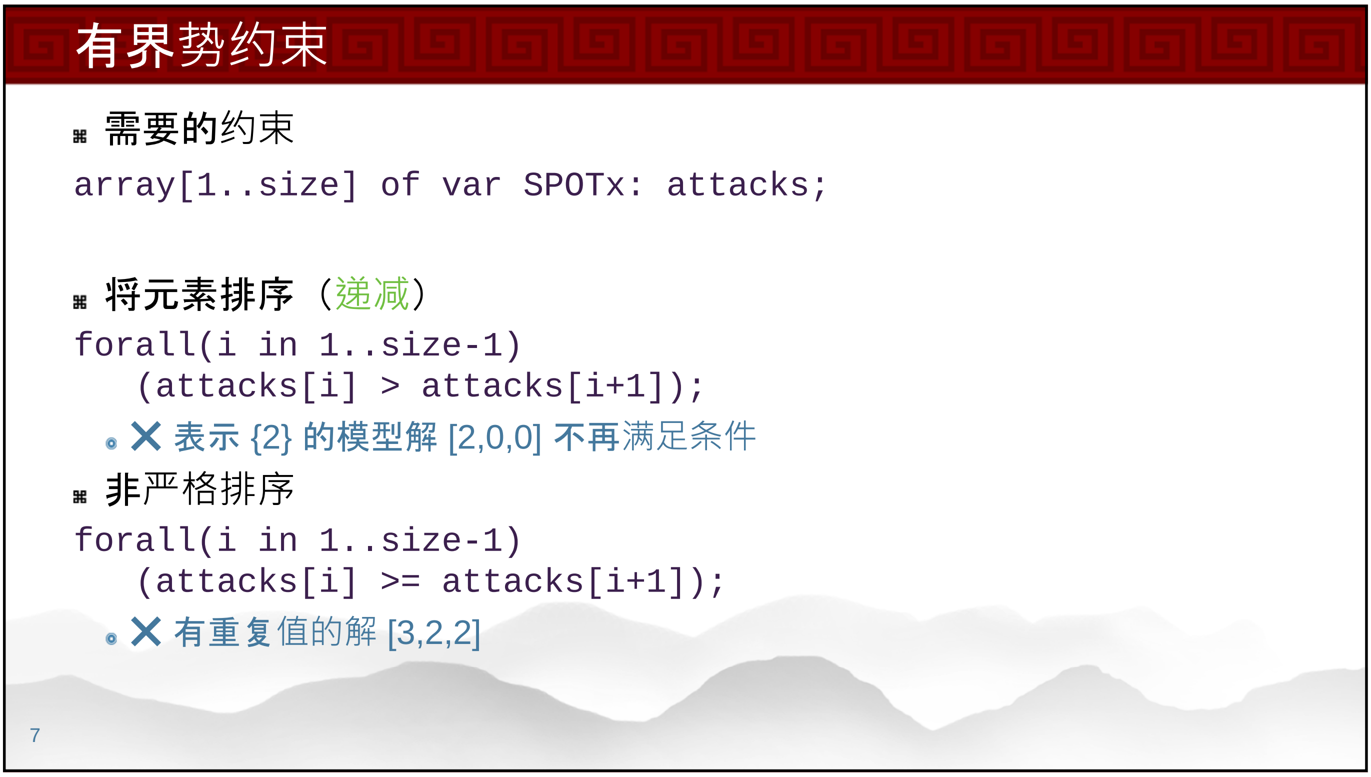
5



两个关键要素

- **模型的每个解代表**问题中的一个解
  - [3,0,3] ✗ **没有重复值**
  - [0,2,0] ✓ **附加值**可以重复
- 问题中的每个解都只有一个模型的解来对应
  - [0,2,0], [0,0,2], [2,0,0] = {2} ✗
  - [0,1,2], [0,2,1], [1,0,2], [1,2,0], [2,0,1], [2,1,0] ✗
- **添加**约束来实现

6



有界势约束

- 需要的约束

```
array[1..size] of var SPOTx: attacks;
```

- 将元素排序（递减）

```
forall(i in 1..size-1)
   (attacks[i] > attacks[i+1]);
```

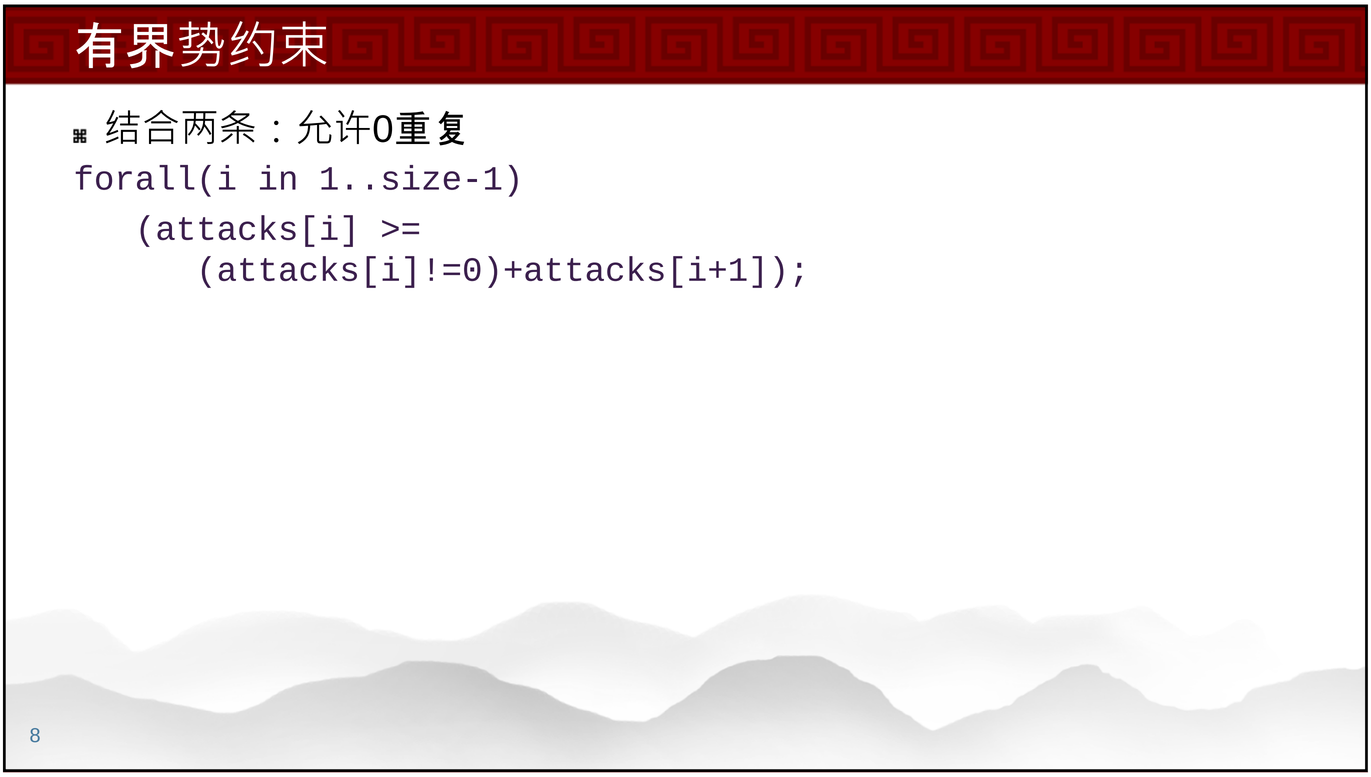
  - ✕ 表示 {2} 的模型解 [2,0,0] 不再满足条件

- 非严格排序

```
forall(i in 1..size-1)
   (attacks[i] >= attacks[i+1]);
```

  - ✕ 有重复值的解 [3,2,2]

7

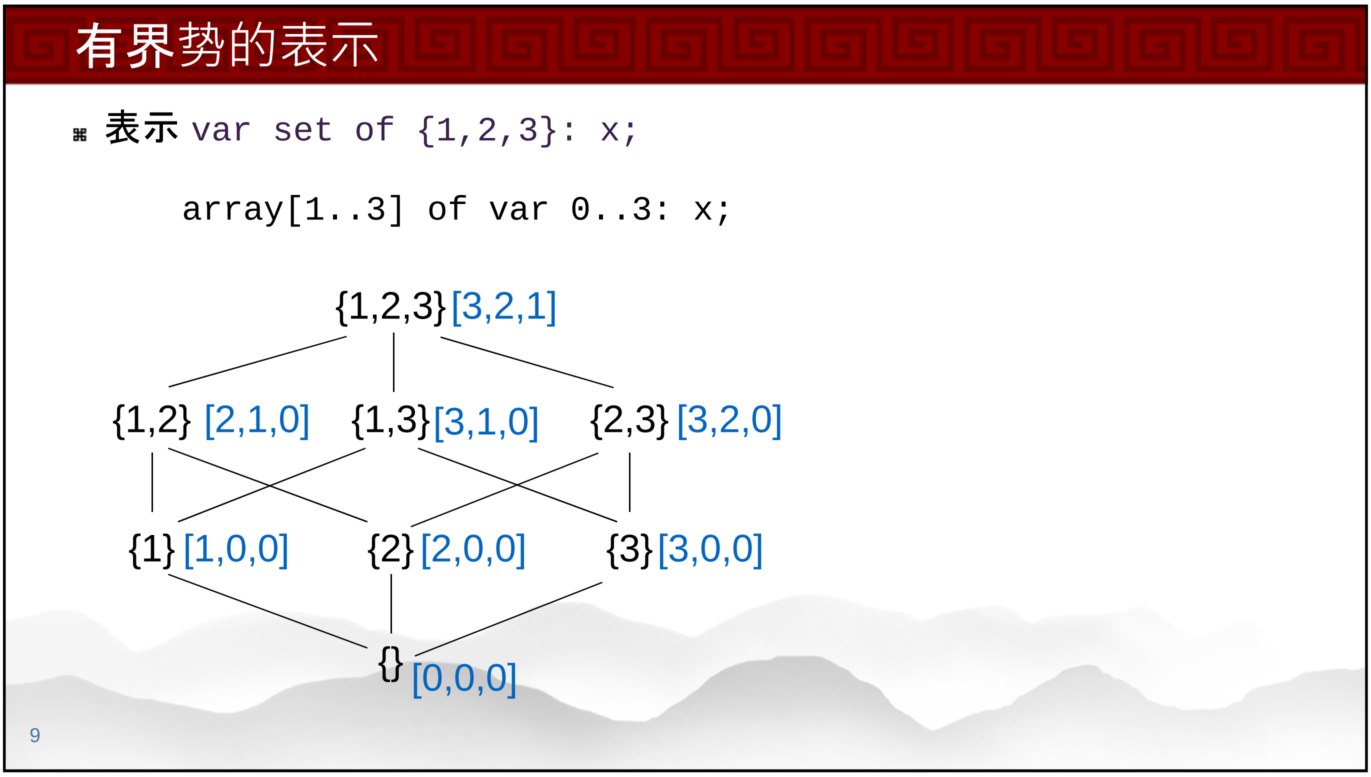


有界势约束

- 结合两条：允许0重复

```
forall(i in 1..size-1)
   (attacks[i] >=
      (attacks[i]!=0)+attacks[i+1]);
```

8



有界势的表示

- 表示 `var set of {1,2,3}: x;`

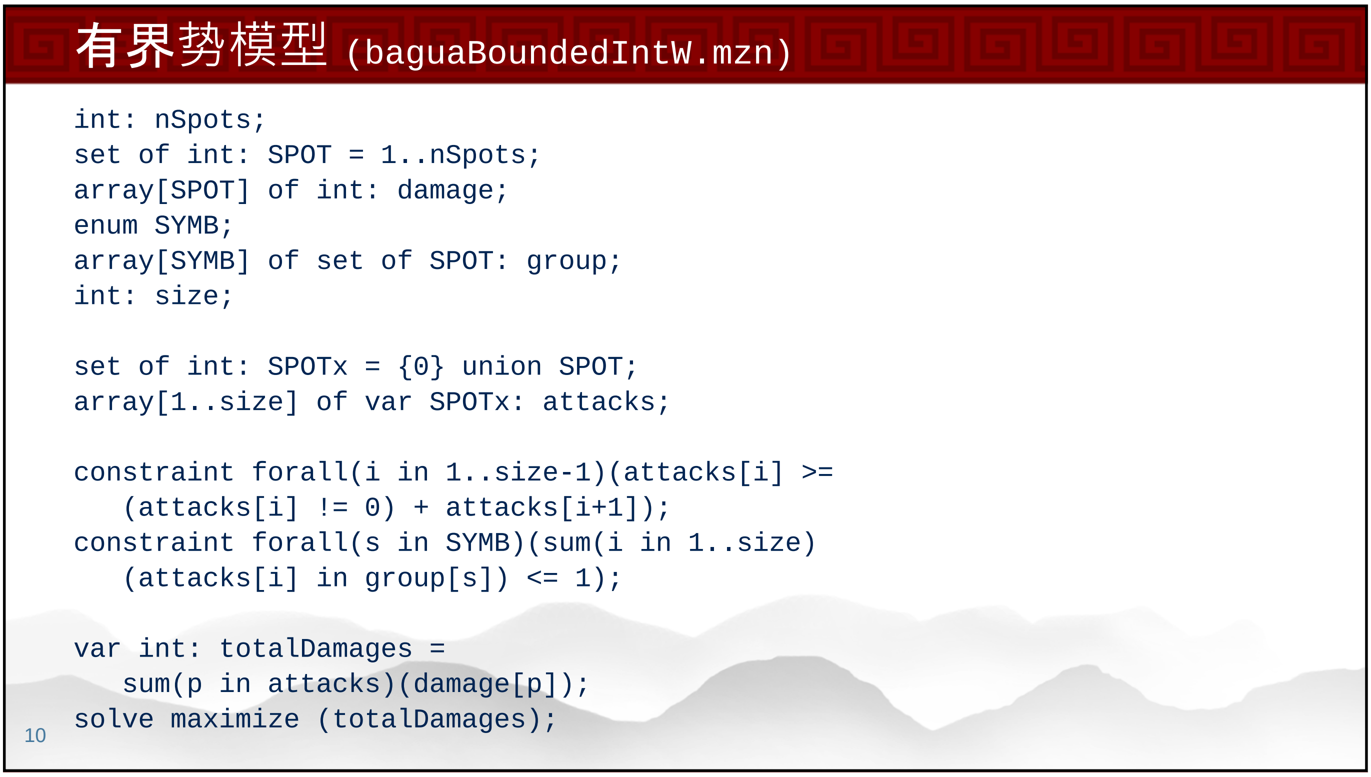
```
array[1..3] of var 0..3: x;
```

{1,2,3} [3,2,1]

{1,2} [2,1,0] {1,3}[3,1,0] {2,3} [3,2,0]

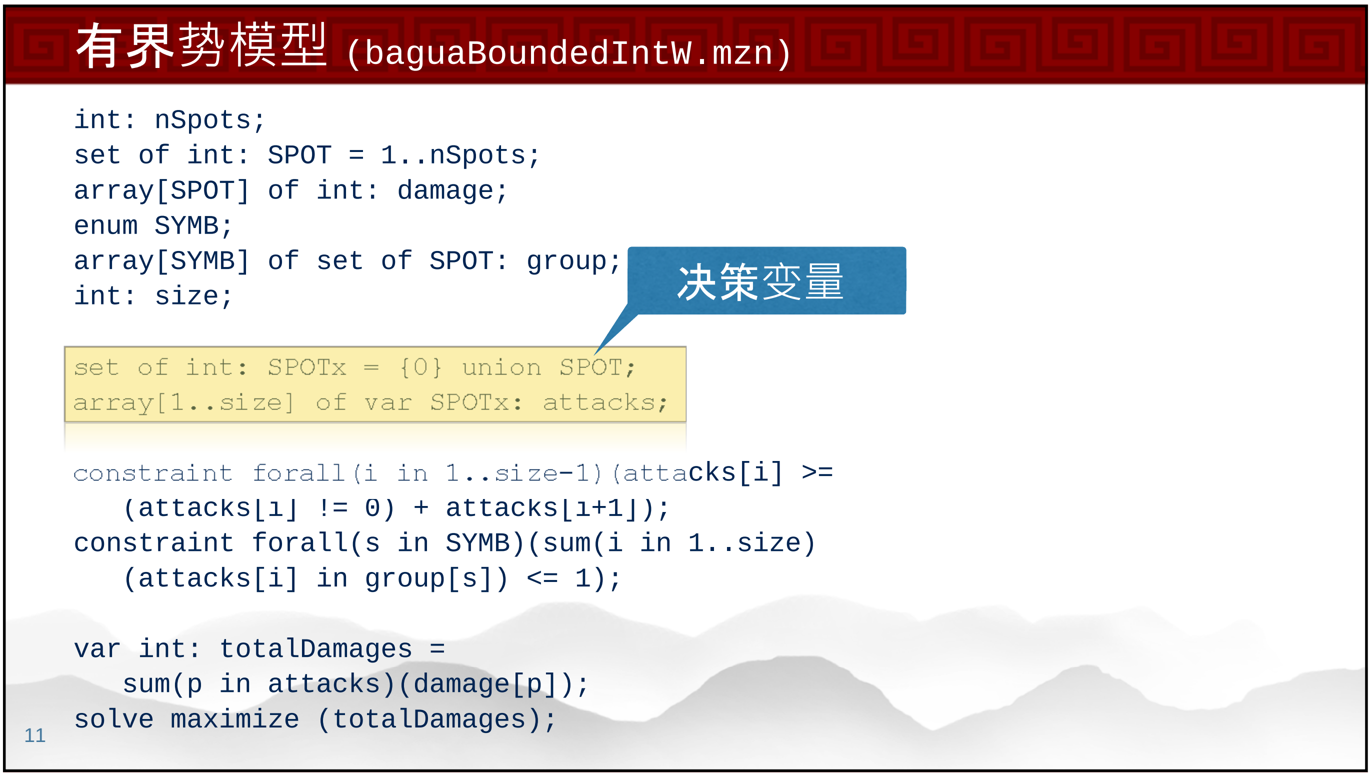
{1} [1,0,0] {2} [2,0,0] {3} [3,0,0]

{} [0,0,0]



有界势模型 (baguaBoundedIntW.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

set of int: SPOTx = {0} union SPOT;
array[1..size] of var SPOTx: attacks;

constraint forall(i in 1..size-1)(attacks[i] >=
   (attacks[i] != 0) + attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(p in attacks)(damage[p]);
solve maximize (totalDamages);
```

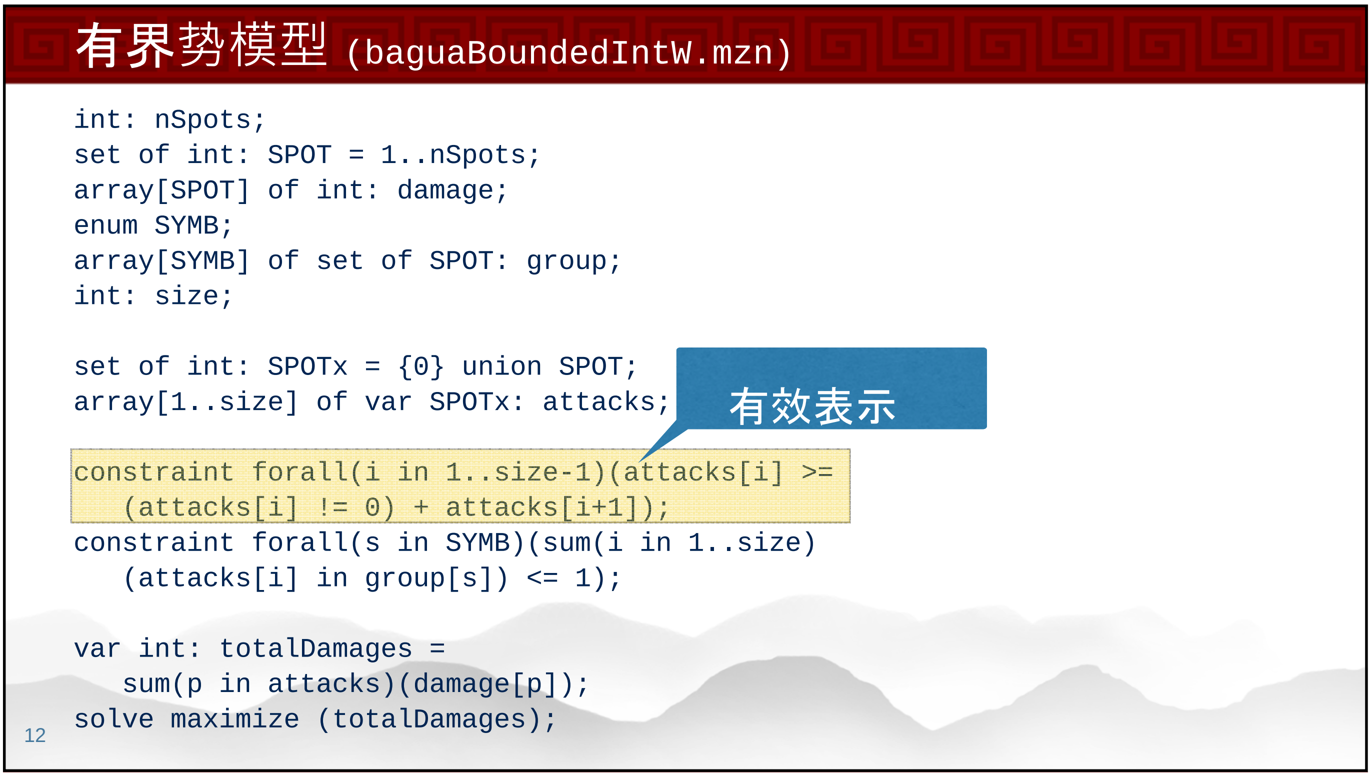


有界势模型 (baguaBoundedIntW.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

set of int: SPOTx = {0} union SPOT;
array[1..size] of var SPOTx: attacks;

constraint forall(i in 1..size-1)(attacks[i] >=
   (attacks[i] != 0) + attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(p in attacks)(damage[p]);
solve maximize (totalDamages);
```

决策变量

11

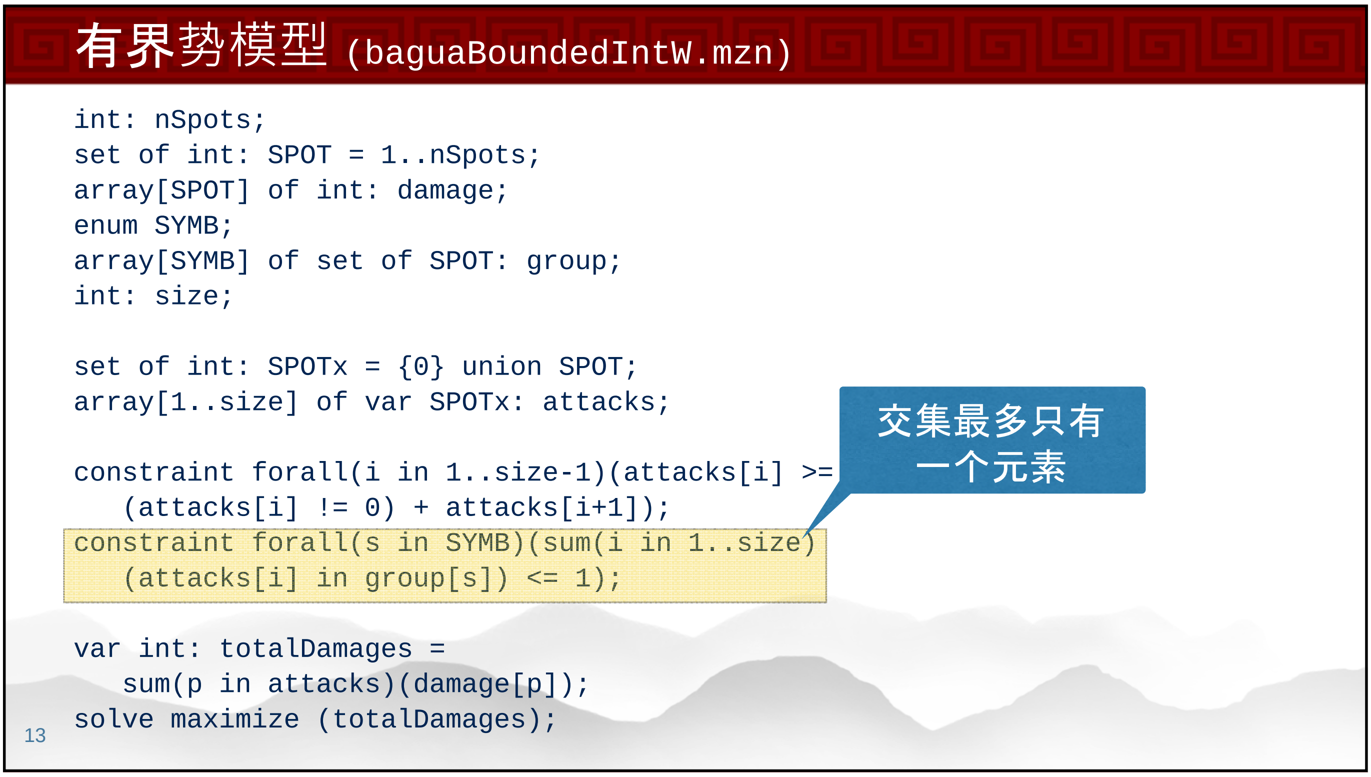


有界势模型 (baguaBoundedIntW.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

set of int: SPOTx = {0} union SPOT;
array[1..size] of var SPOTx: attacks;

constraint forall(i in 1..size-1)(attacks[i] >=
   (attacks[i] != 0) + attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(p in attacks)(damage[p]);
solve maximize (totalDamages);
```

有效表示

12

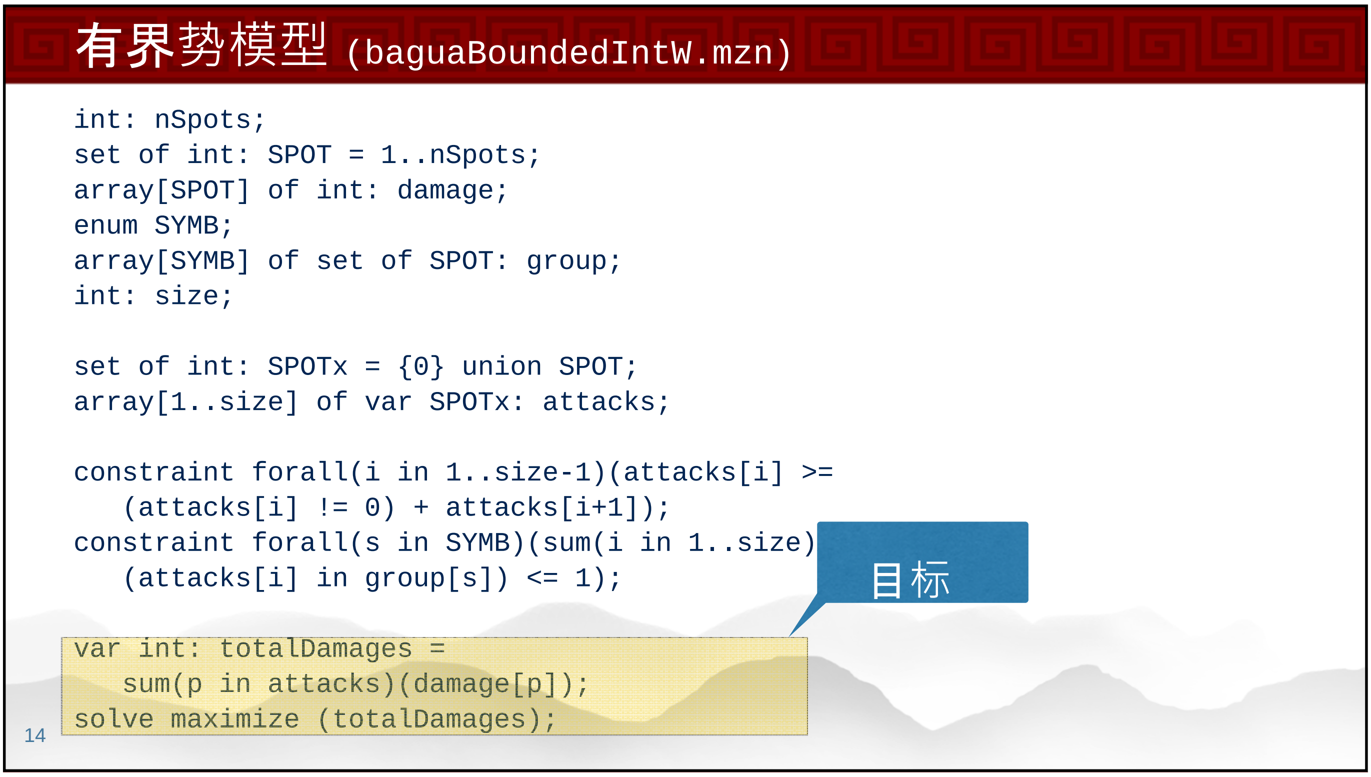


有界势模型 (baguaBoundedIntW.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

set of int: SPOTx = {0} union SPOT;
array[1..size] of var SPOTx: attacks;

constraint forall(i in 1..size-1)(attacks[i] >=
   (attacks[i] != 0) + attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(p in attacks)(damage[p]);
solve maximize (totalDamages);
```

交集最多只有一个元素

13

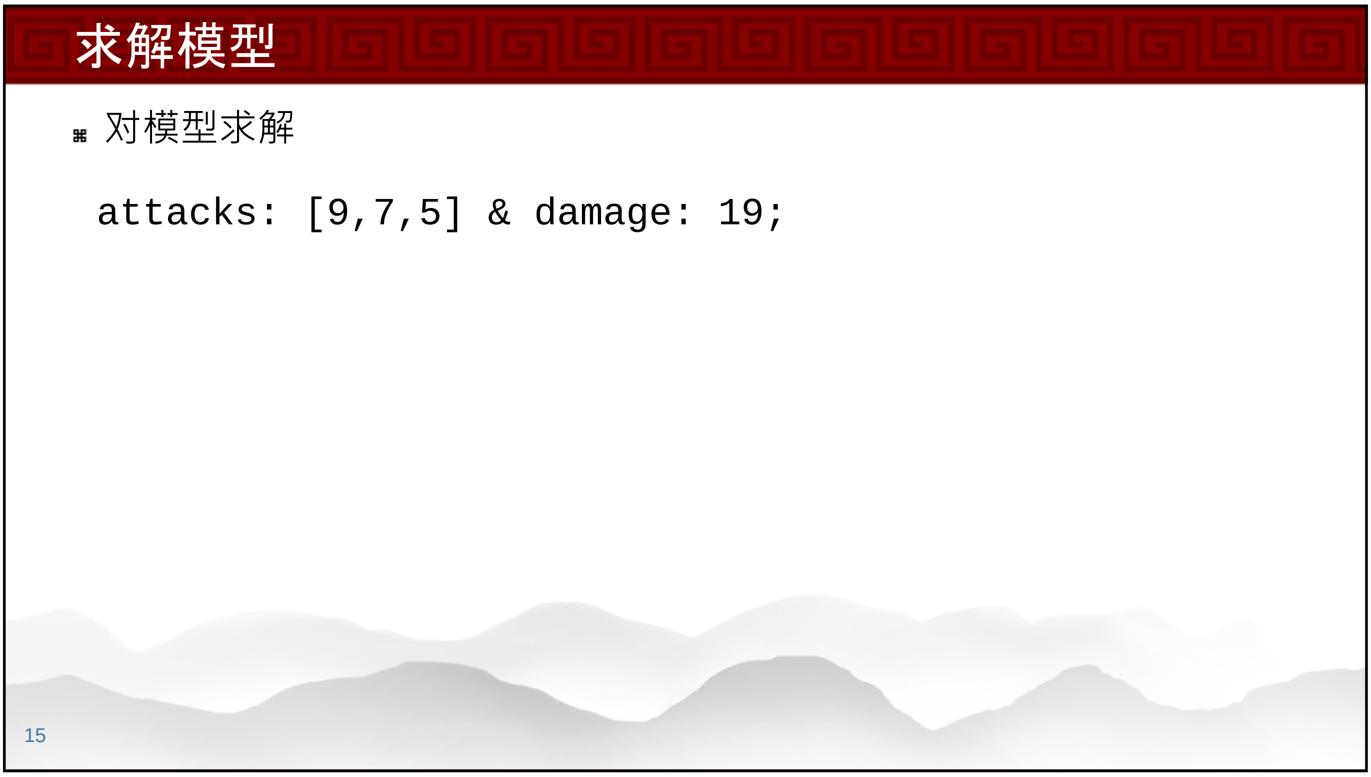


有界势模型 (baguaBoundedIntW.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

set of int: SPOTx = {0} union SPOT;
array[1..size] of var SPOTx: attacks;

constraint forall(i in 1..size-1)(attacks[i] >=
   (attacks[i] != 0) + attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(p in attacks)(damage[p]);
solve maximize (totalDamages);
```

目标

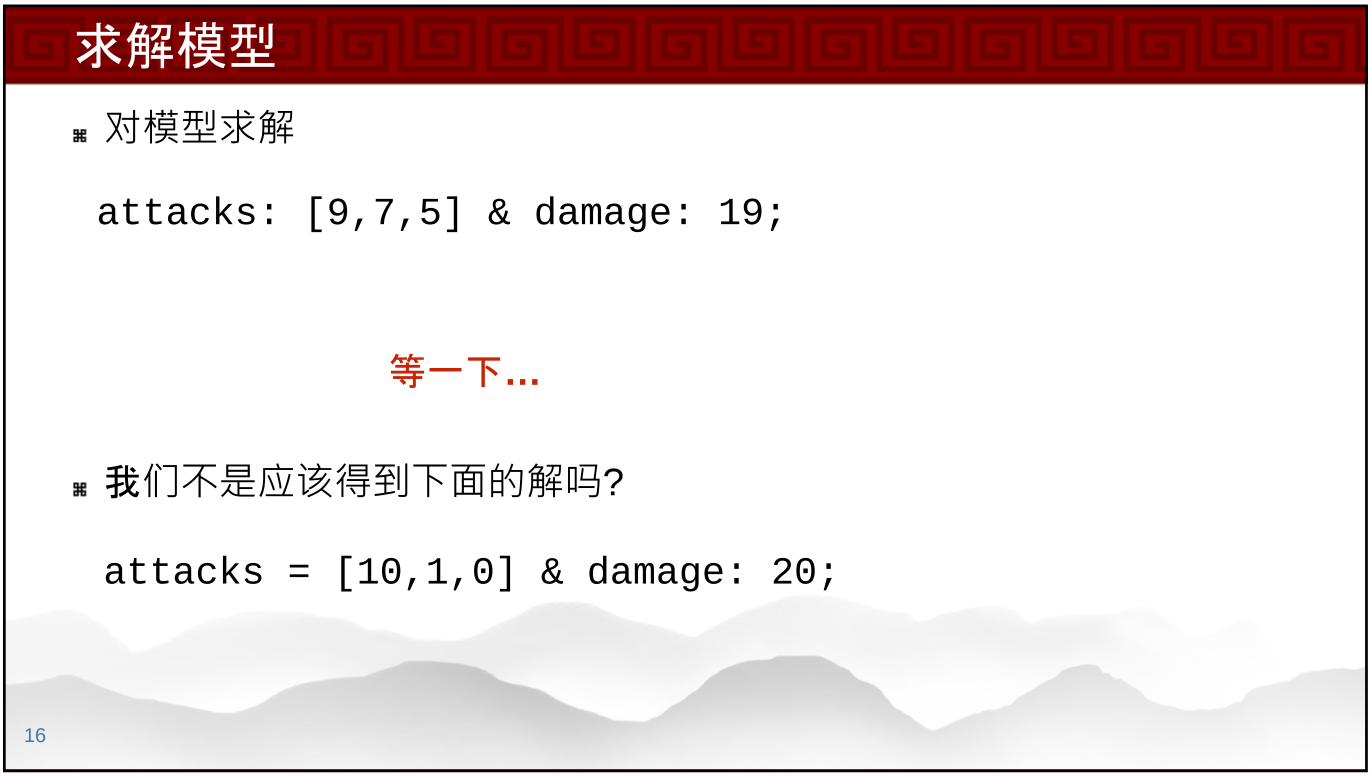
14



求解模型

- 对模型求解

```
attacks: [9,7,5] & damage: 19;
```



求解模型

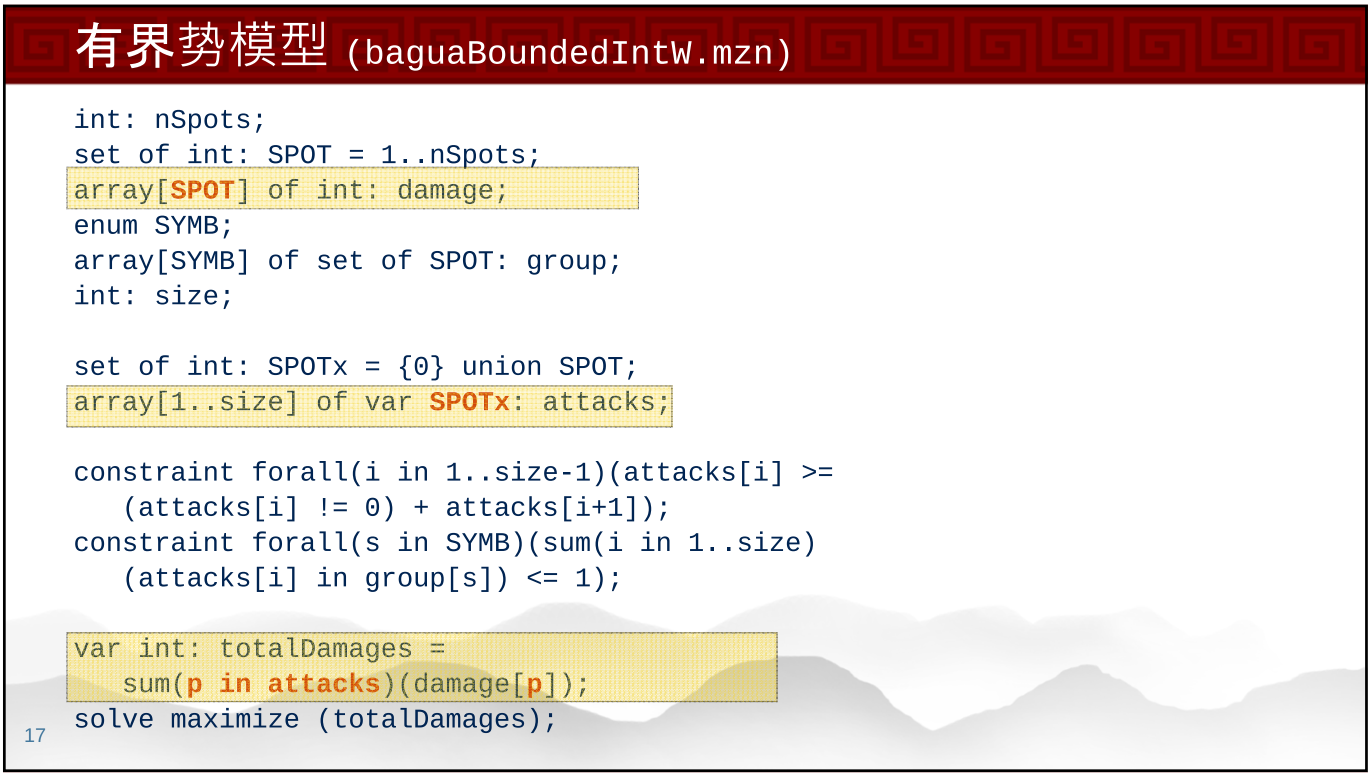
- 对模型求解

```
attacks: [9,7,5] & damage: 19;
```

等一下...

- 我们不是应该得到下面的解吗?

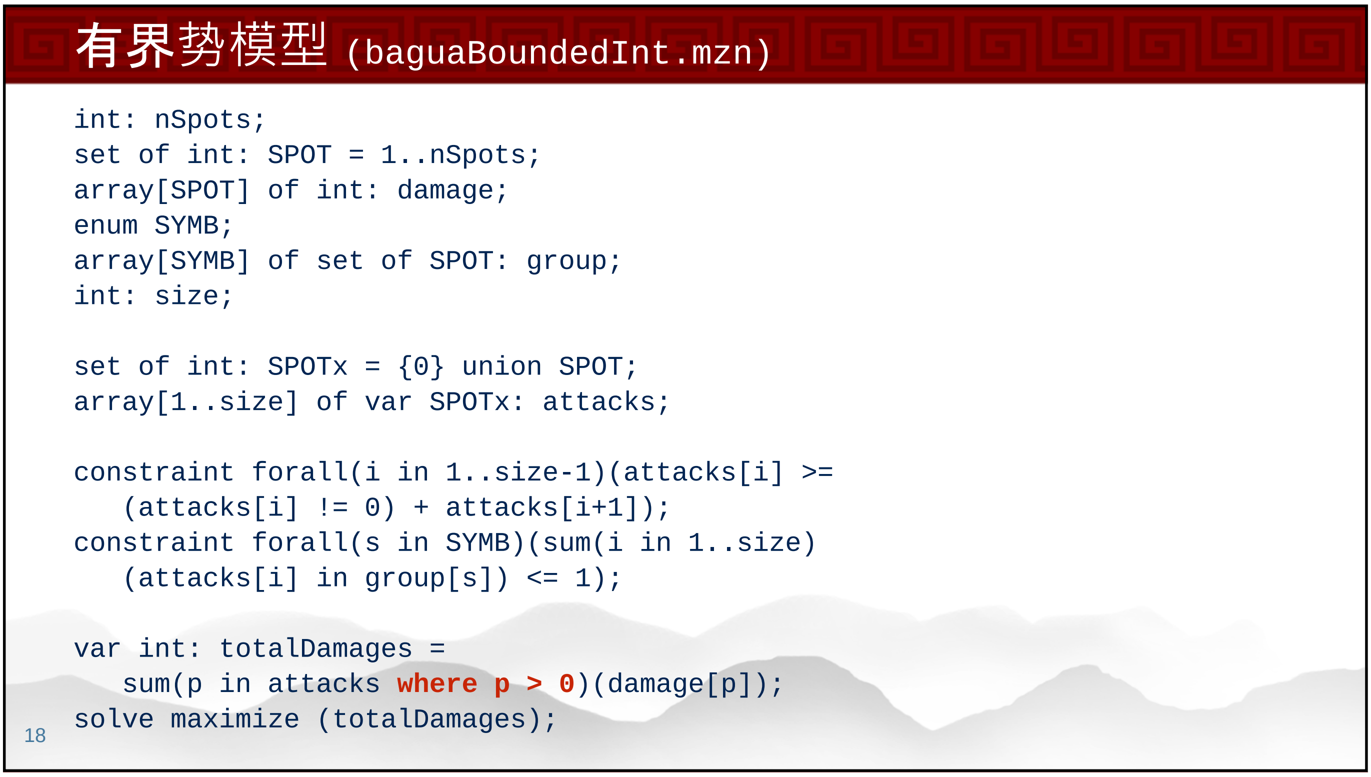
```
attacks = [10,1,0] & damage: 20;
```



有界势模型 (baguaBoundedIntW.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

set of int: SPOTx = {0} union SPOT;
array[1..size] of var SPOTx: attacks;

constraint forall(i in 1..size-1)(attacks[i] >=
   (attacks[i] != 0) + attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(p in attacks)(damage[p]);
solve maximize (totalDamages);
```

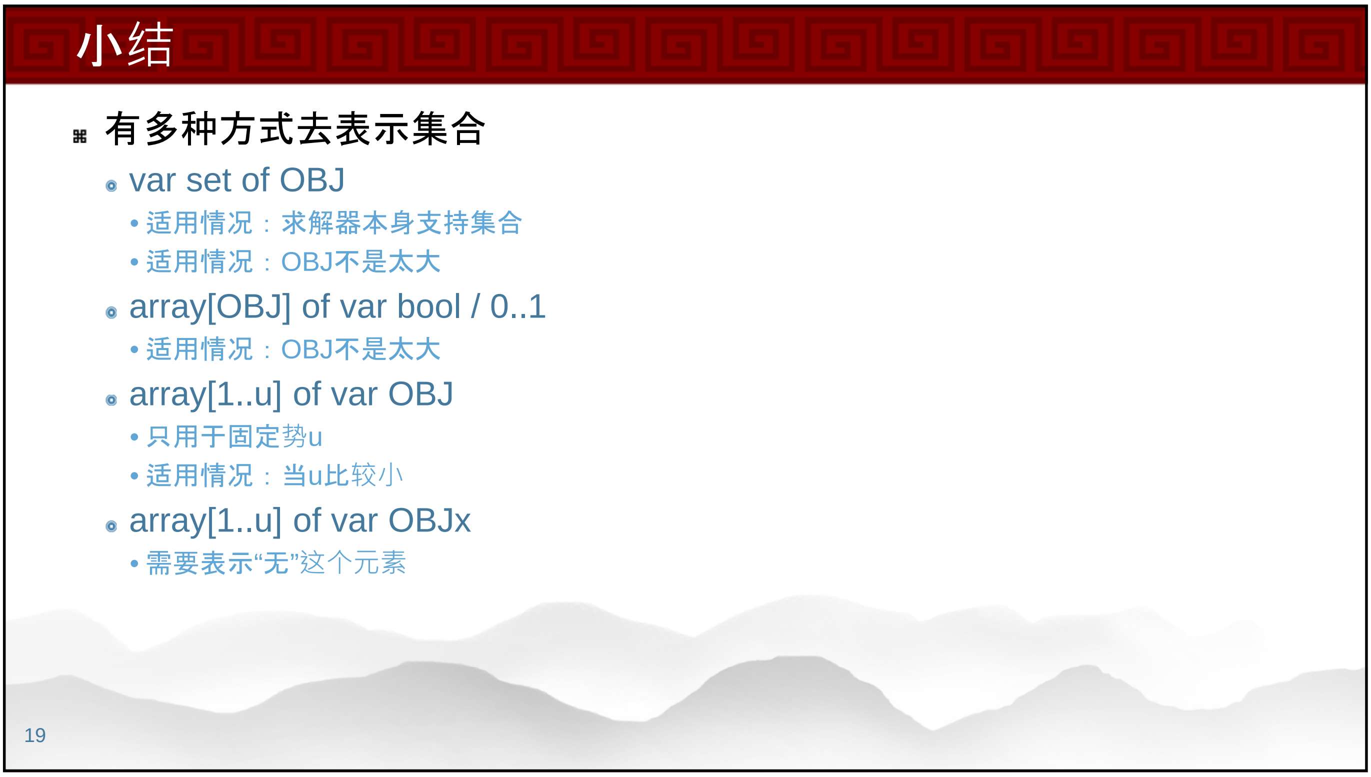
17



有界势模型 (baguaBoundedInt.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

set of int: SPOTx = {0} union SPOT;
array[1..size] of var SPOTx: attacks;

constraint forall(i in 1..size-1)(attacks[i] >=
   (attacks[i] != 0) + attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(p in attacks where p > 0)(damage[p]);
solve maximize (totalDamages);
```

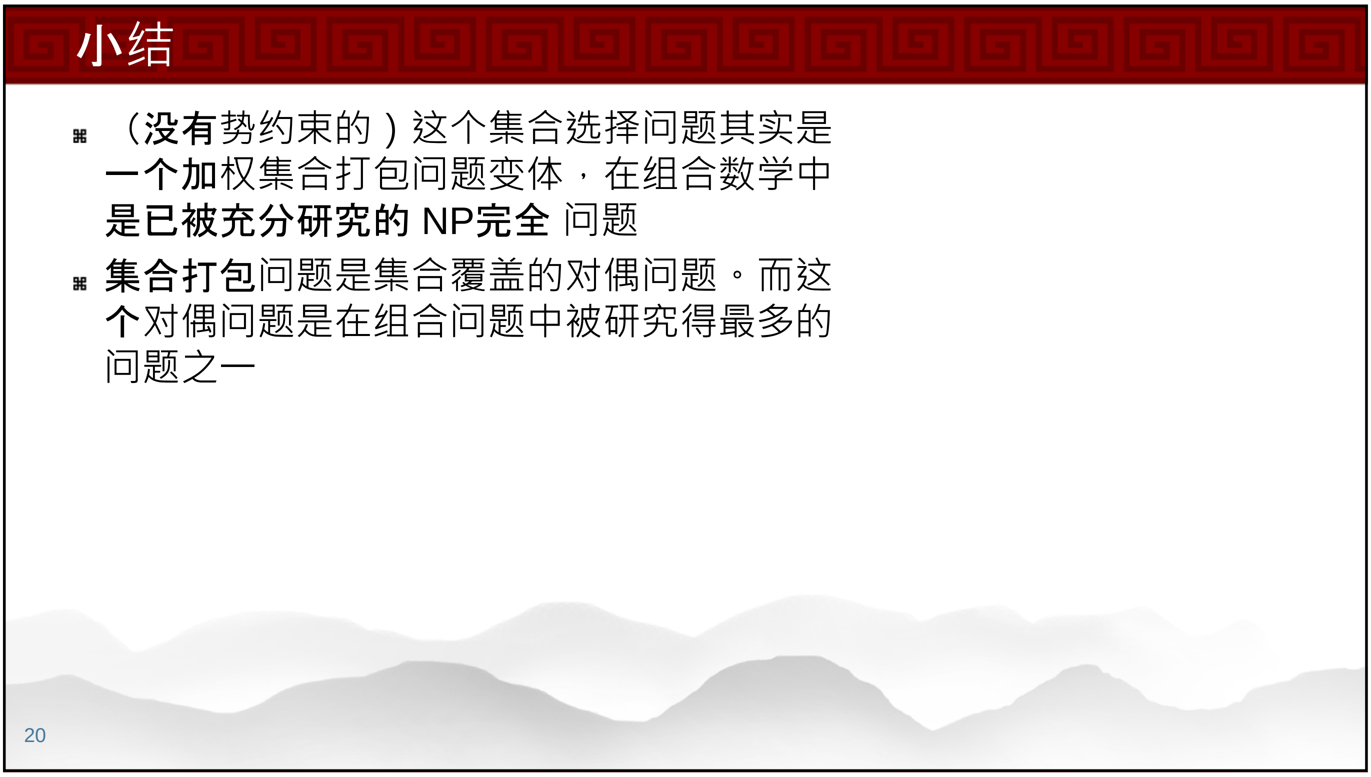
18



小结

- 有多种方式去表示集合
  - var set of OBJ
    - 适用情况：求解器本身支持集合
    - 适用情况：OBJ不是太大
  - array[OBJ] of var bool / 0..1
    - 适用情况：OBJ不是太大
  - array[1..u] of var OBJ
    - 只用于固定势u
    - 适用情况：当u比较小
  - array[1..u] of var OBJx
    - 需要表示“无”这个元素

19



小结

- （**没有**势约束的）这个集合选择问题其实是**一个加权**集合打包问题变体，在组合数学中**是已被充分研究的 NP完全** 问题
- **集合打包**问题是集合覆盖的对偶问题。而这**个**对偶问题是在组合问题中被研究得最多的问题之一

20

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

21