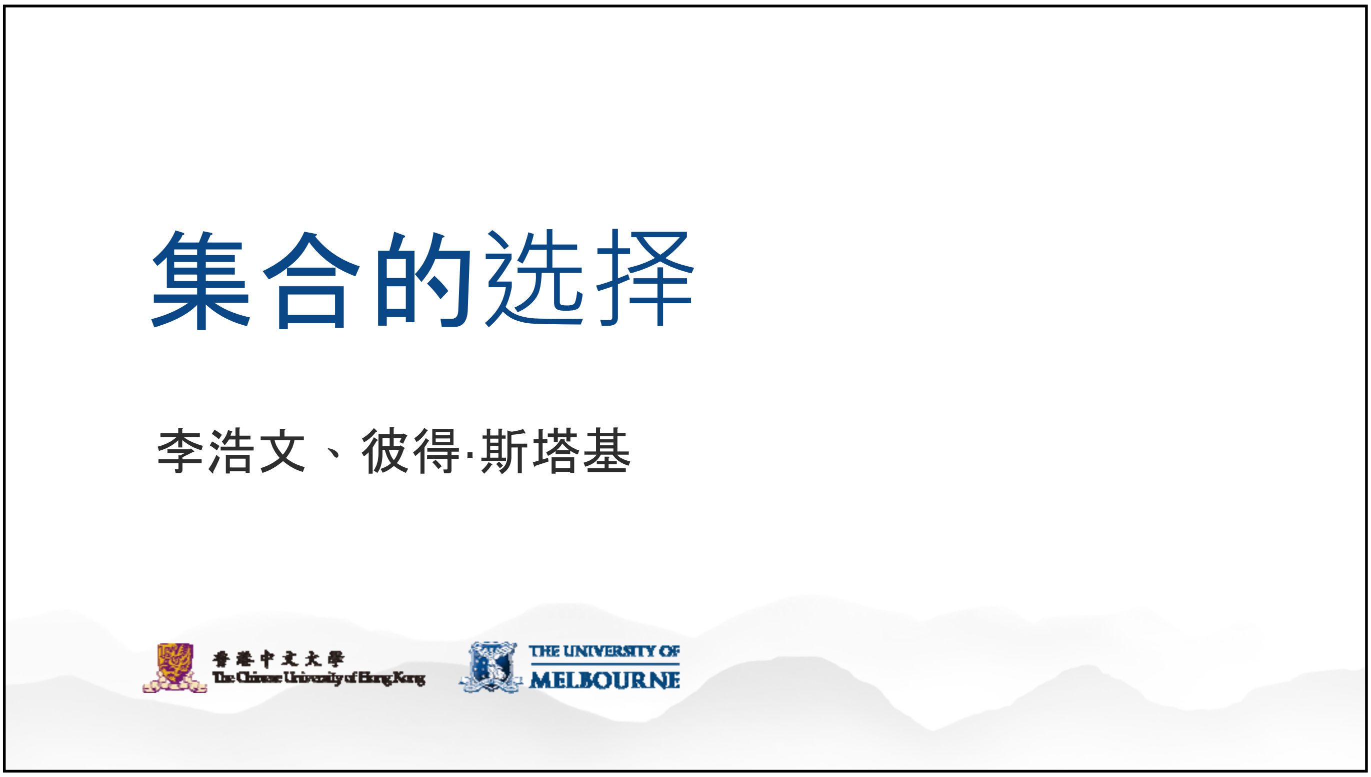
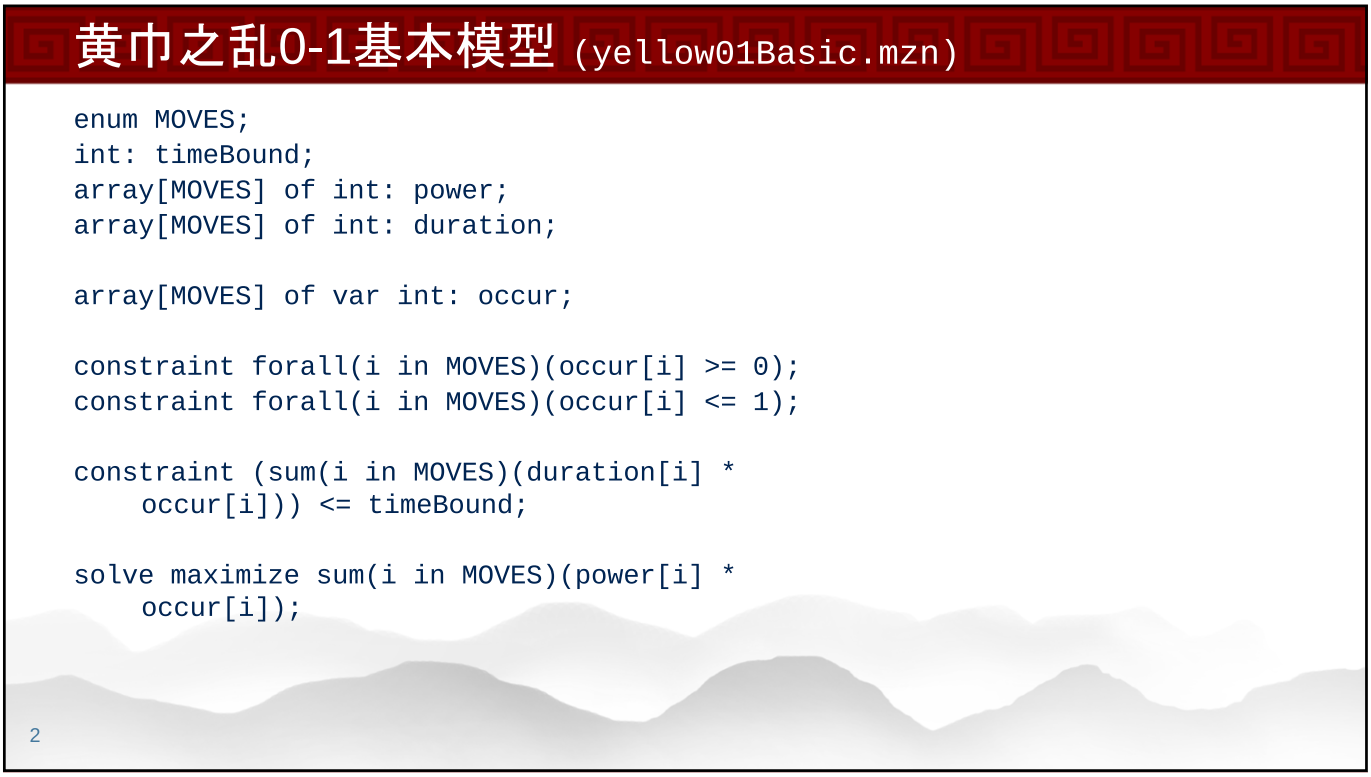


集合的选择

李浩文、彼得·斯塔基



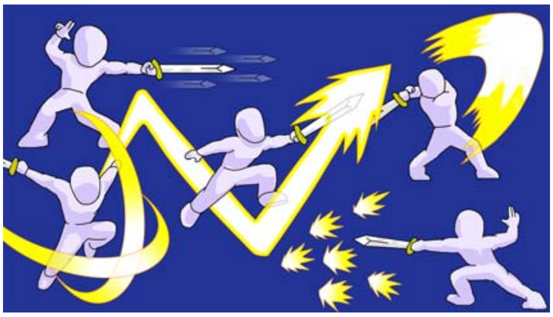
黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

2

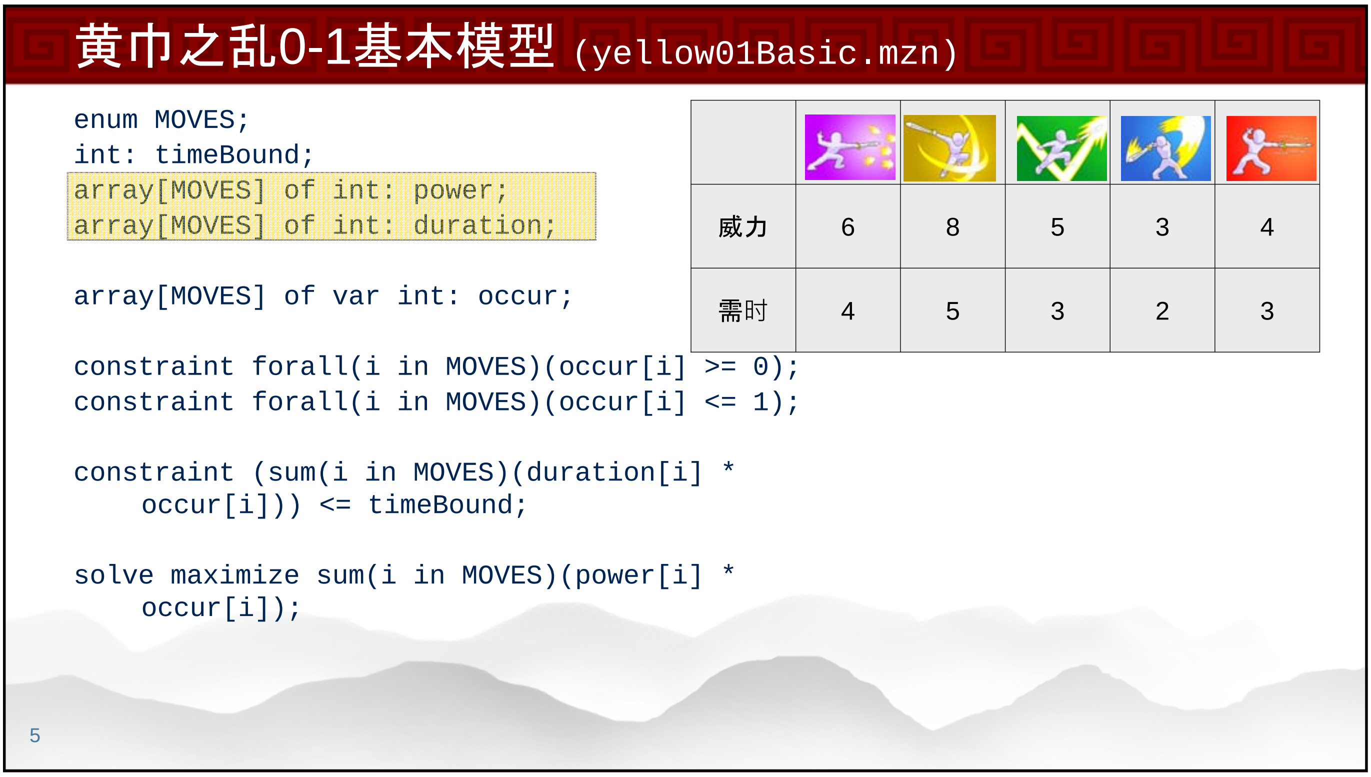
黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



黄巾之乱0-1基本模型 (yellow01Basic.mzn)

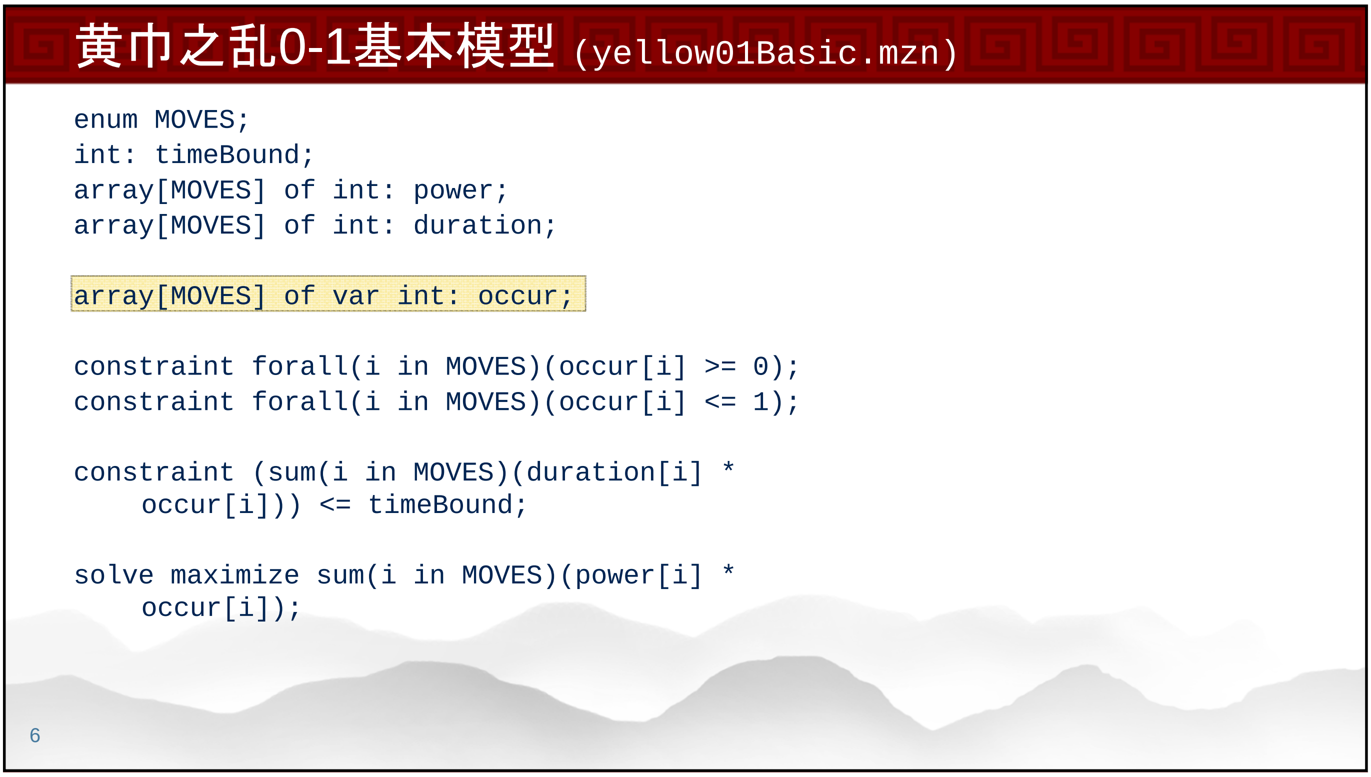
```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



黄巾之乱0-1基本模型 (yellow01Basic.mzn)

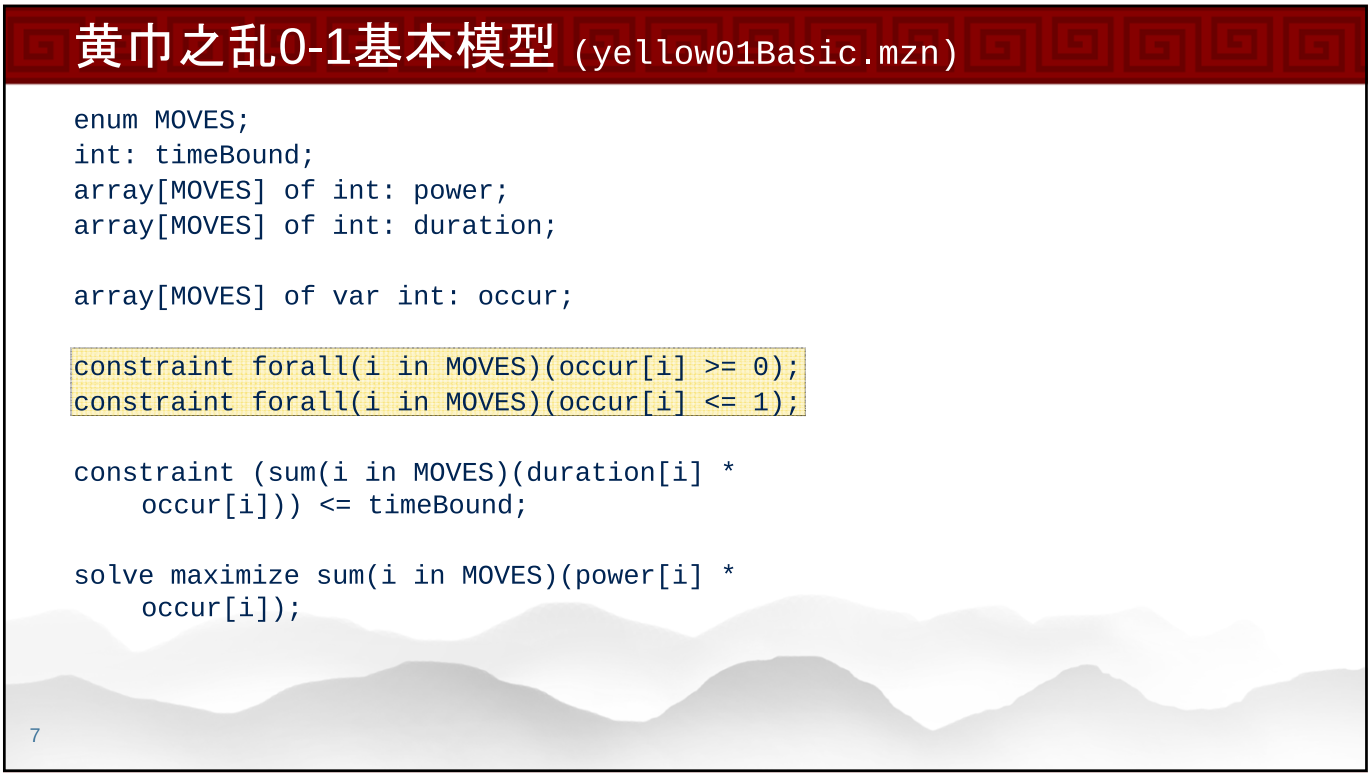
```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

威力	6	8	5	3	4
需时	4	5	3	2	3



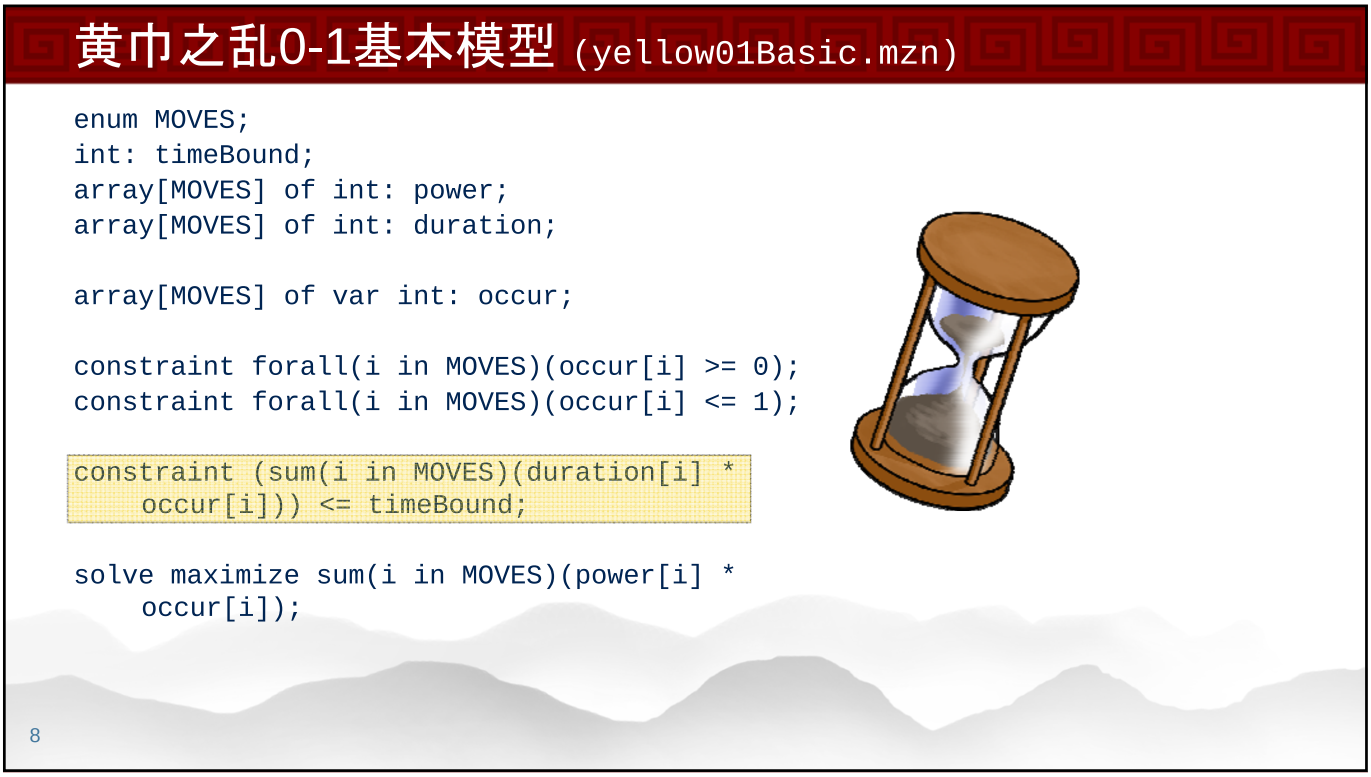
黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



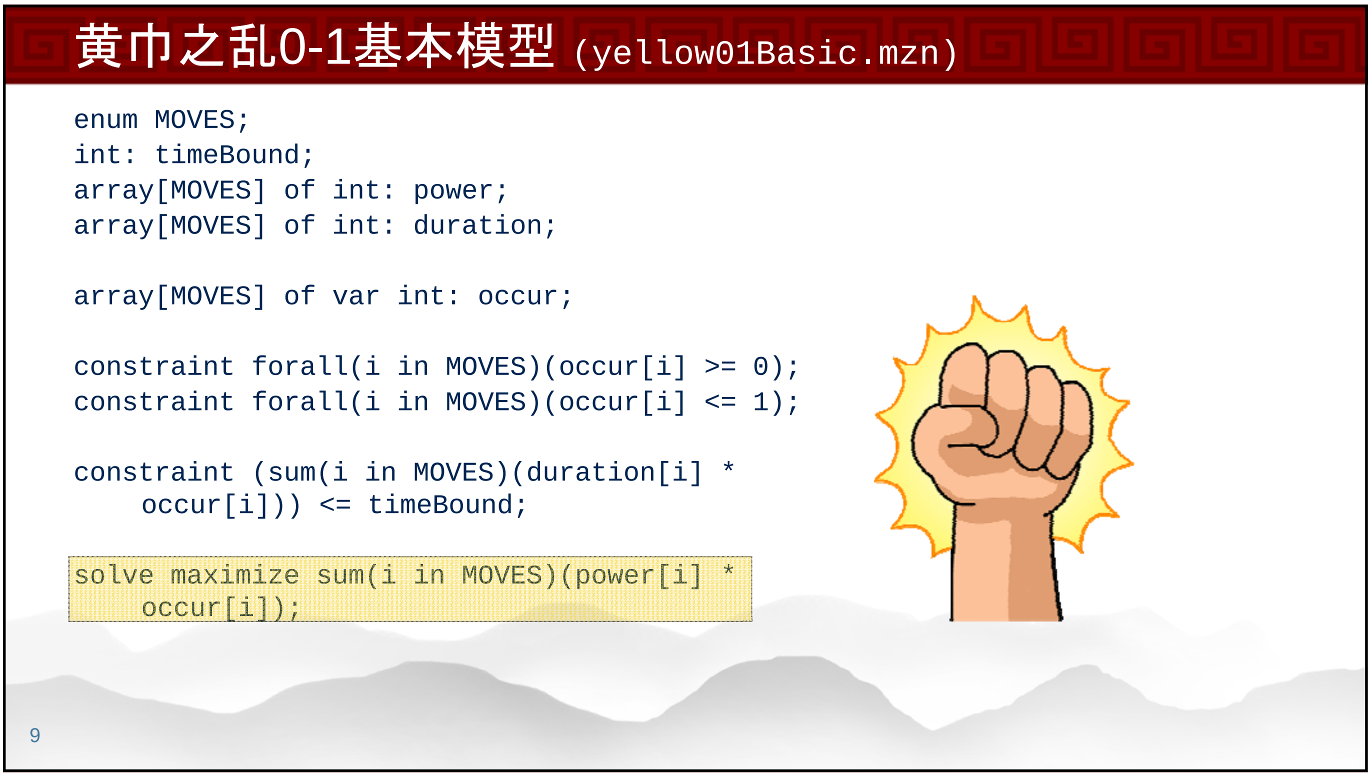
黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



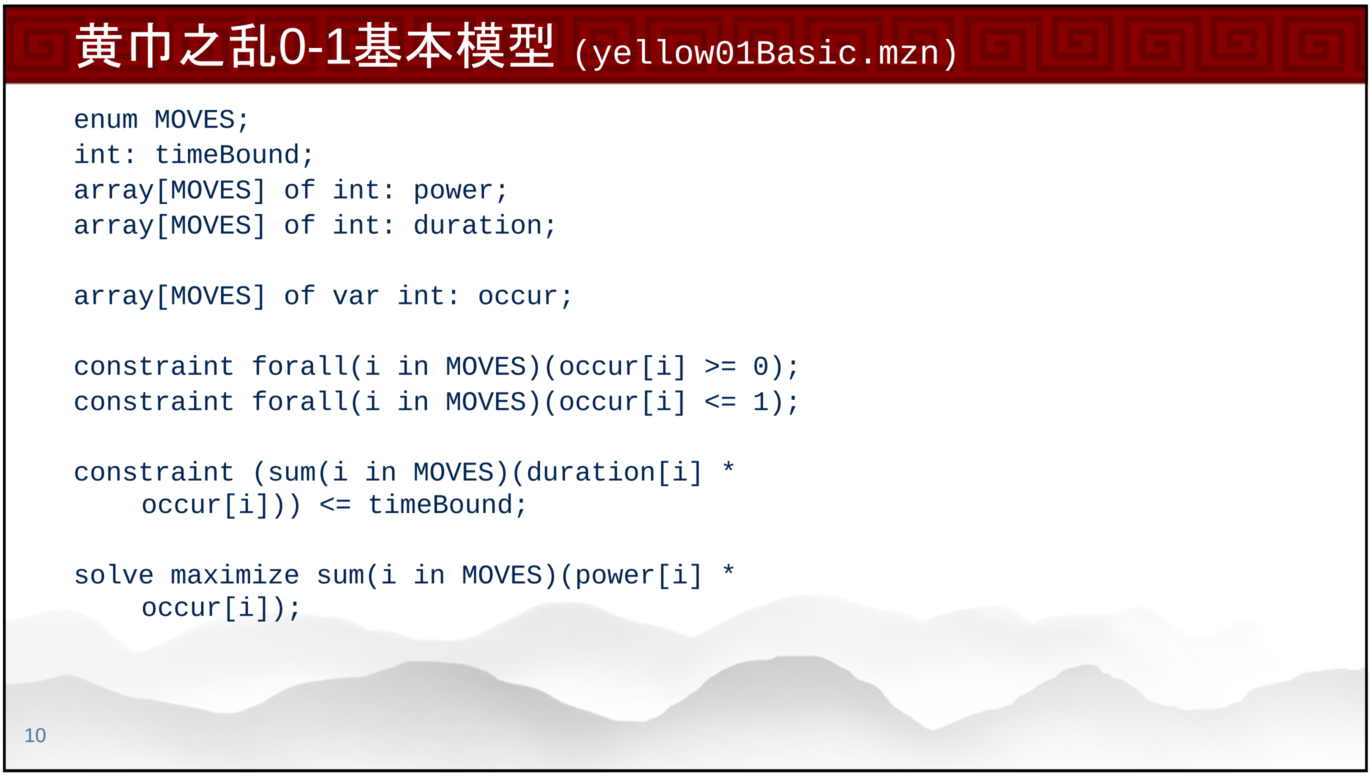
黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



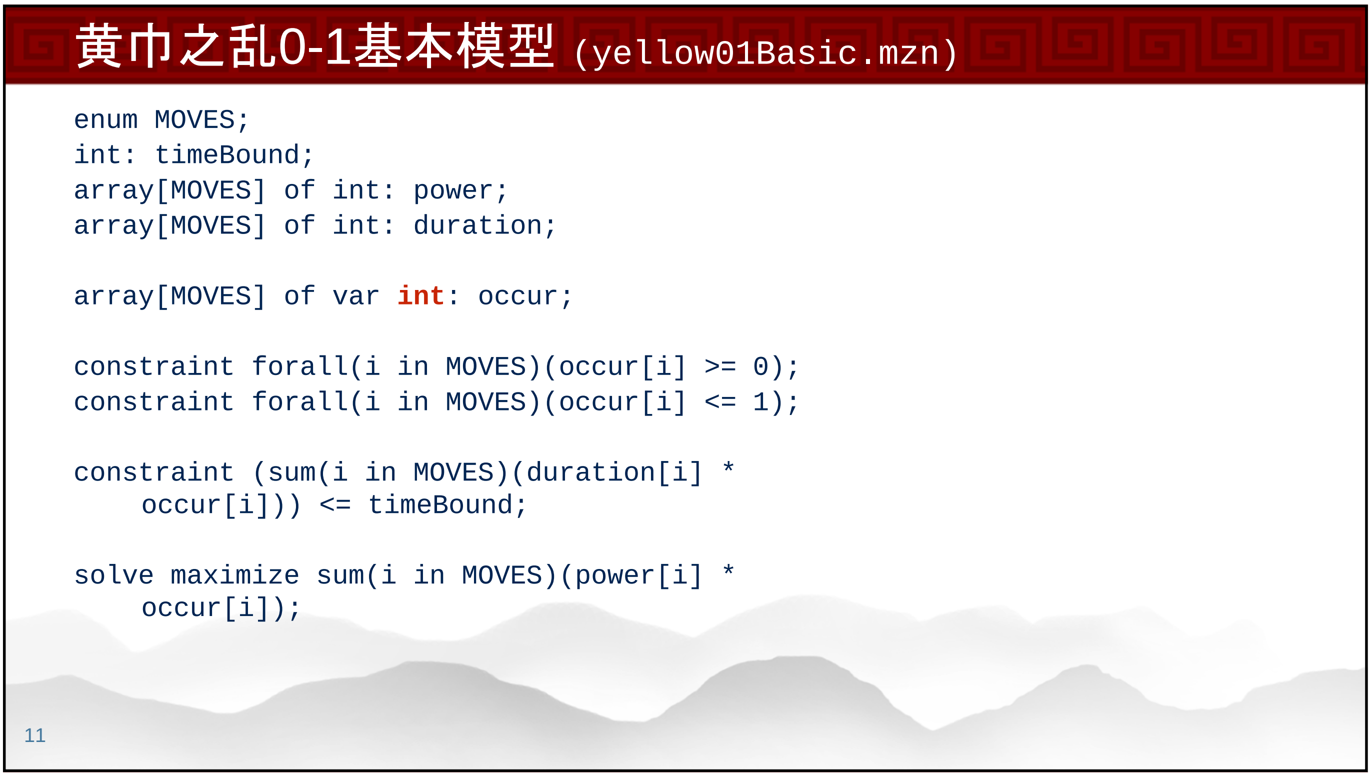
黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



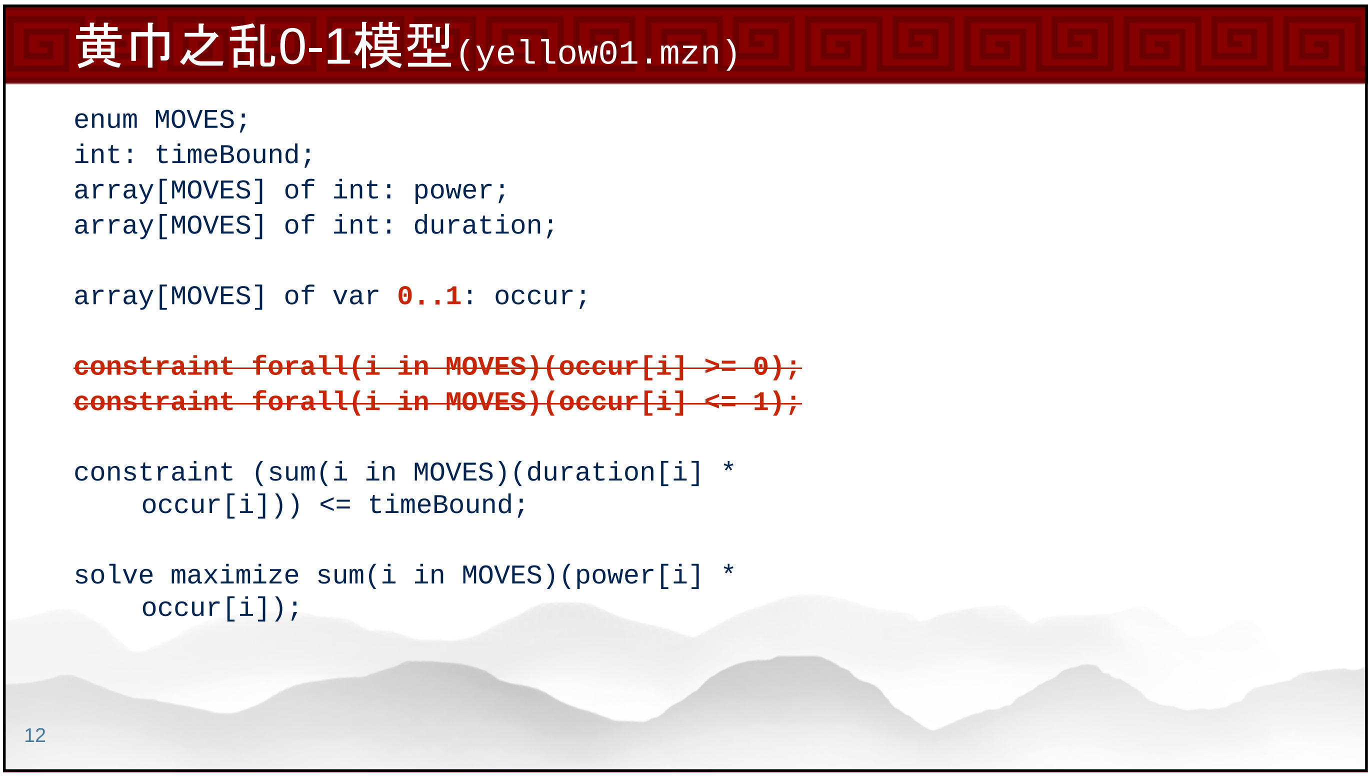
黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



黄巾之乱0-1基本模型 (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

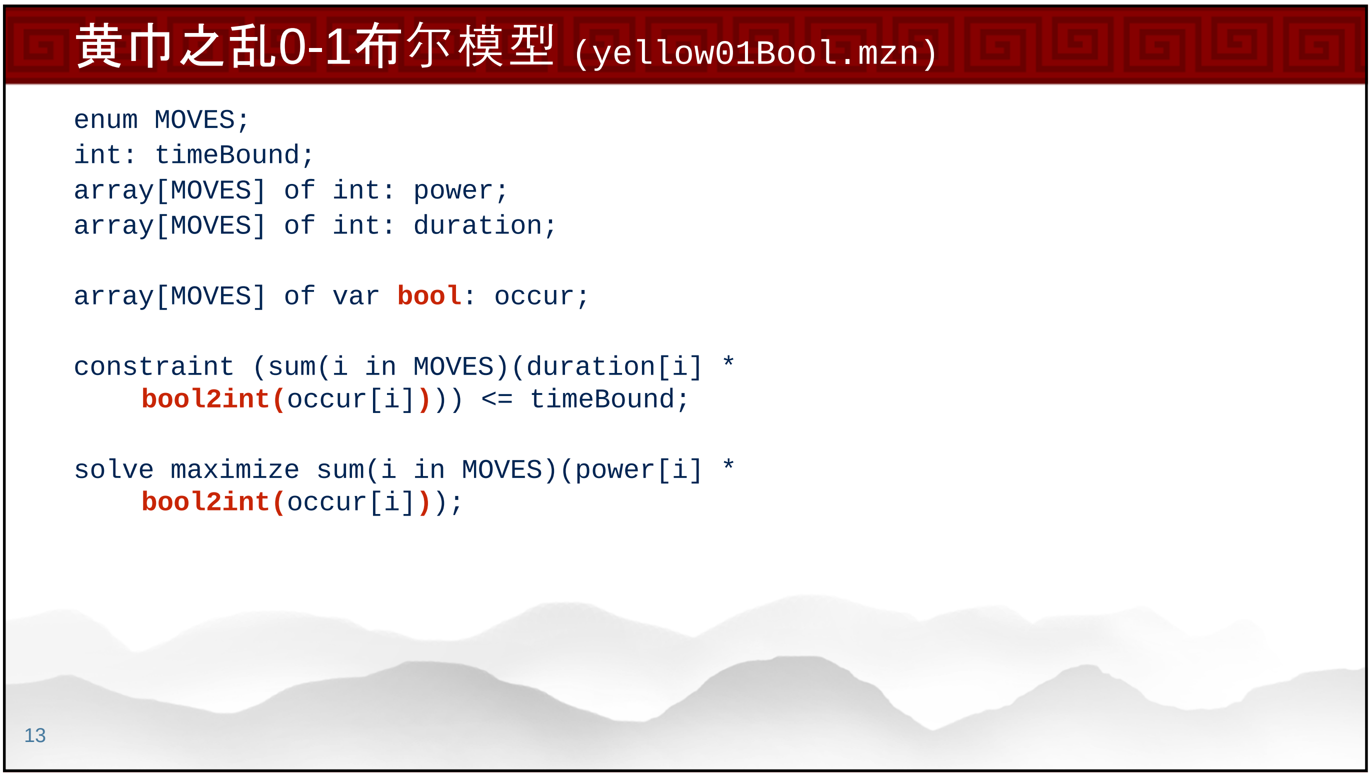


黄巾之乱0-1模型(yellow01.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var 0..1: occur;
```

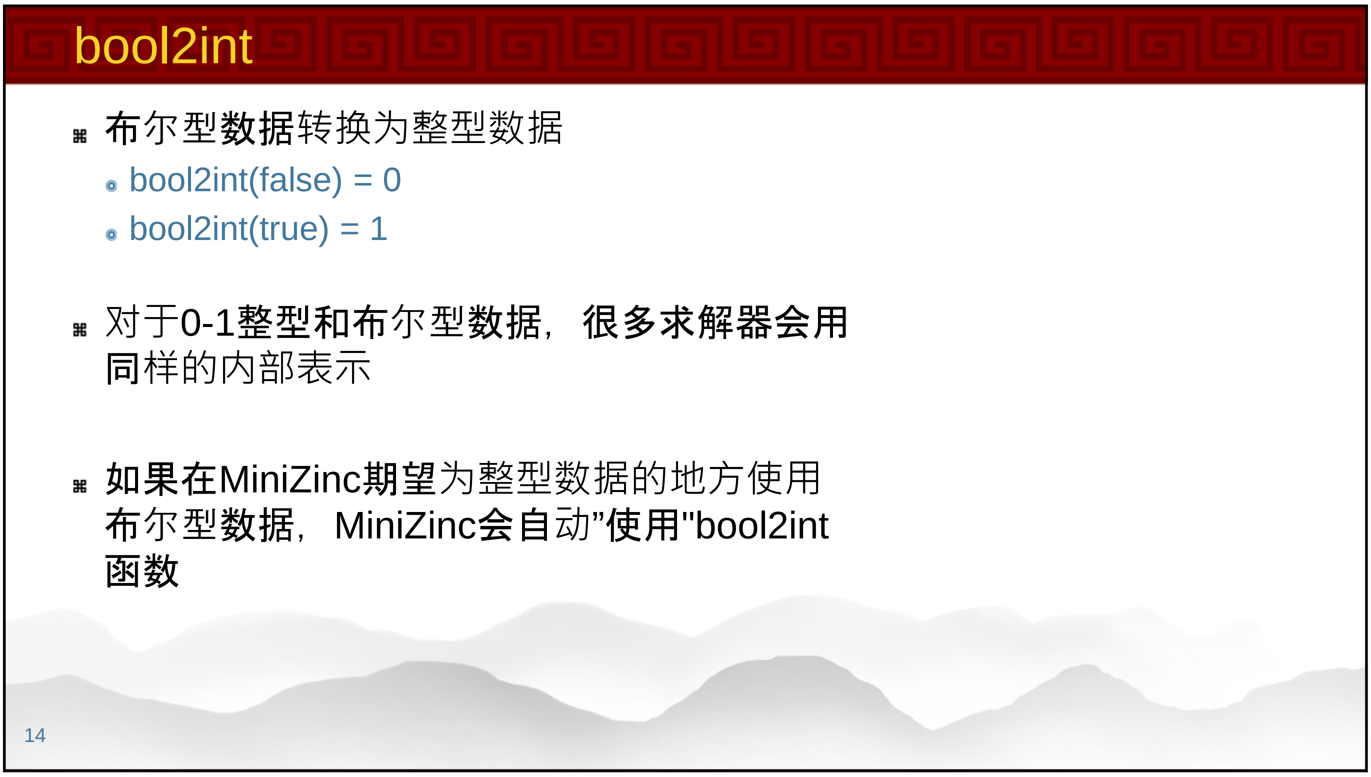
~~constraint forall(i in MOVES)(occur[i] >= 0);~~
~~constraint forall(i in MOVES)(occur[i] <= 1);~~

```
constraint (sum(i in MOVES)(duration[i] *
    occur[i])) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



黄巾之乱0-1布尔模型 (yellow01Bool.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var bool: occur;

constraint (sum(i in MOVES)(duration[i] *
    bool2int(occur[i]))) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    bool2int(occur[i]));
```

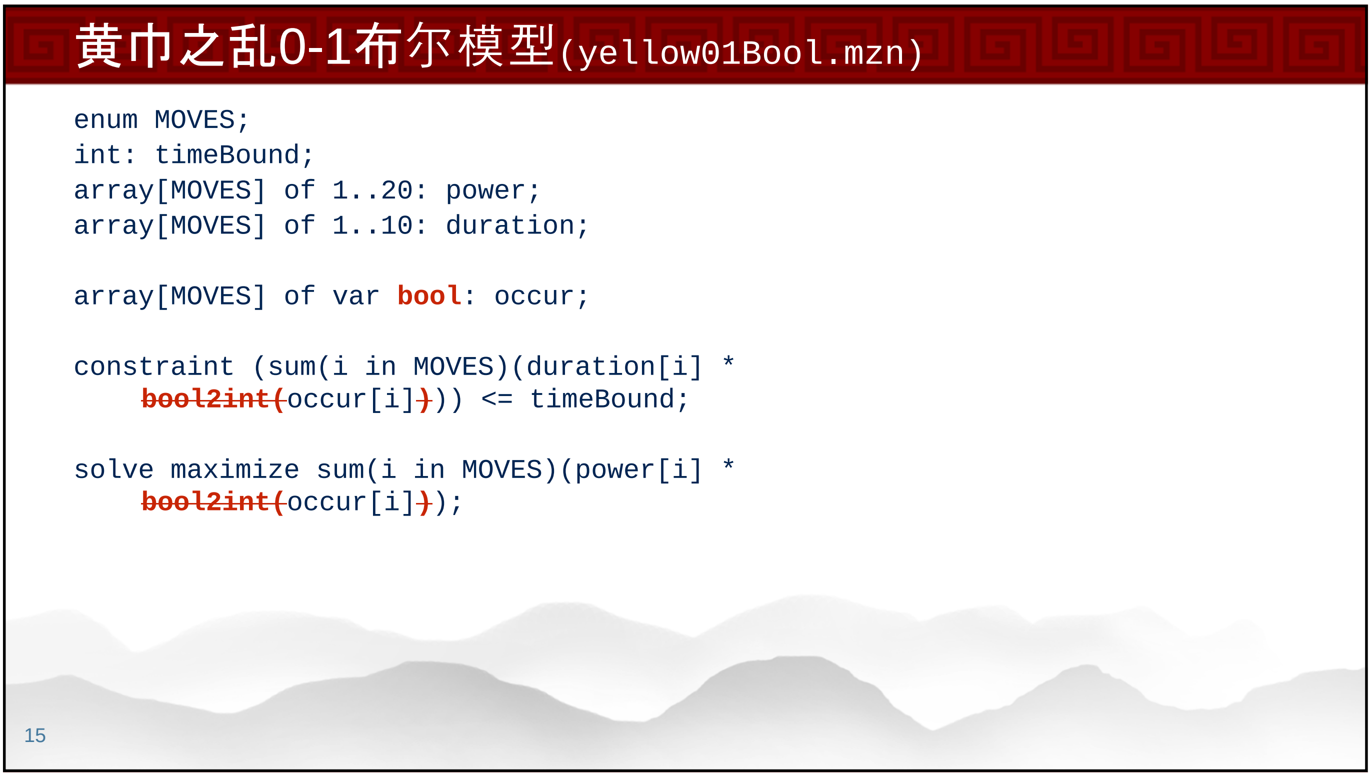


bool2int

- 布尔型数据转换为整型数据
  - bool2int(false) = 0
  - bool2int(true) = 1

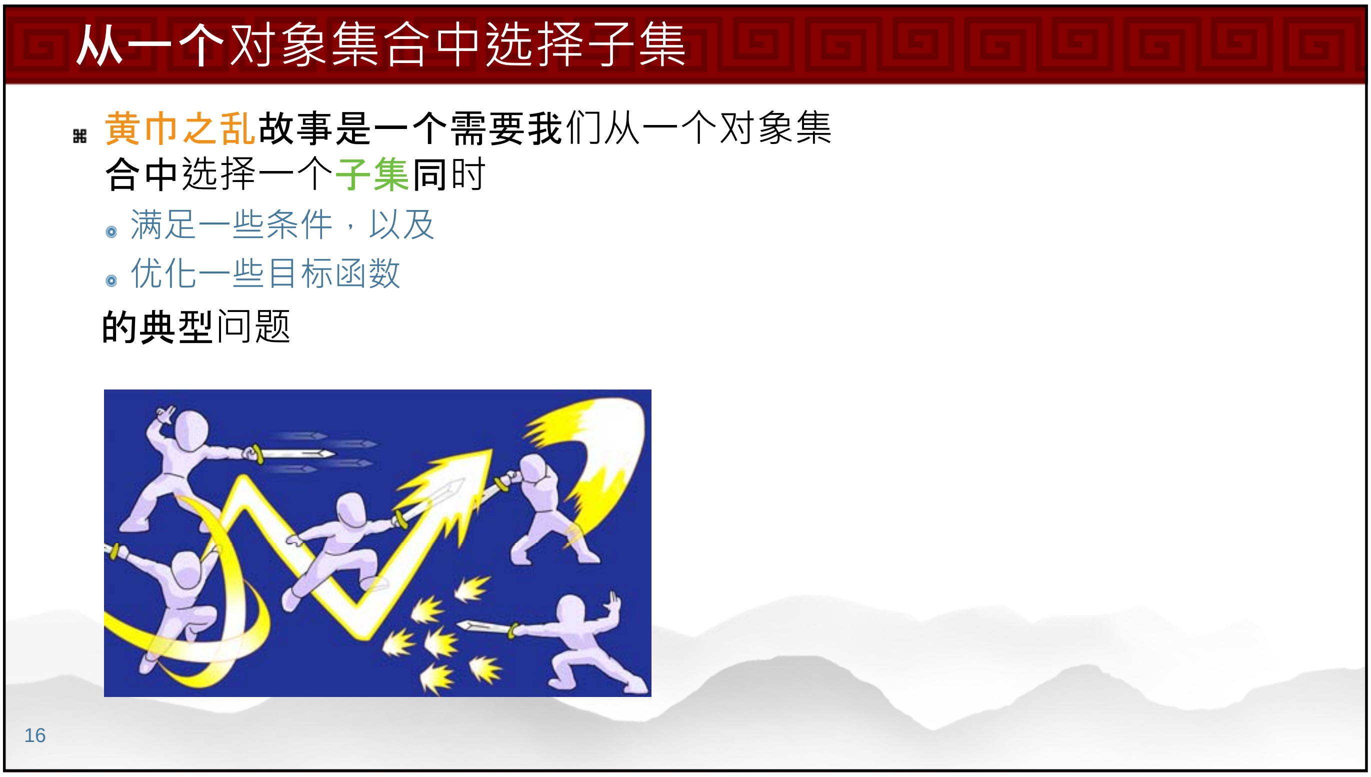
- 对于0-1整型和布尔型数据，很多求解器会用同样的内部表示

- 如果在MiniZinc期望为整型数据的地方使用布尔型数据，MiniZinc会自动"使用"bool2int函数



黄巾之乱0-1布尔模型(yellow01Bool.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of 1..20: power;
array[MOVES] of 1..10: duration;

array[MOVES] of var bool: occur;

constraint (sum(i in MOVES)(duration[i] *
    ~~bool2int(~~occur[i]~~)~~)) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    ~~bool2int(~~occur[i]~~)~~);
```



从一个对象集合中选择子集

- 黄巾之乱故事是一个需要我们从一个对象集合中选择一个子集同时
  - 满足一些条件，以及
  - 优化一些目标函数

  的典型问题

从一个对象集合中选择子集

- 黄巾之乱故事是一个需要我们从一个对象集合中选择一个子集同时
  - 满足一些条件，以及
  - 优化一些目标函数

  的典型问题

17

从一个对象集合中选择子集

- 黄巾之乱故事是一个需要我们从一个对象集合中选择一个子集同时
  - 满足一些条件，以及
  - 优化一些目标函数

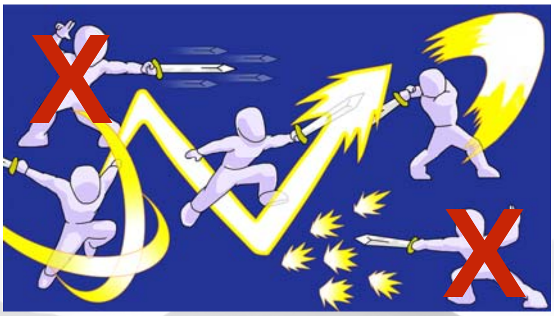
  的典型问题

18

从一个对象集合中选择子集

- 黄巾之乱故事是一个需要我们从一个对象集合中选择一个子集同时
  - 满足一些条件，以及
  - 优化一些目标函数

  的典型问题



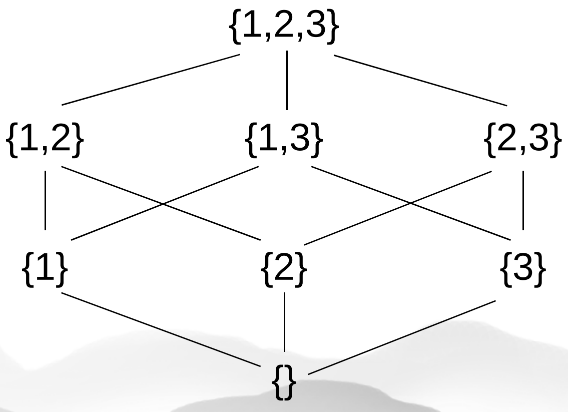
0-1/集合选择问题

- 0-1整型变量数组
- 布尔型变量数组
- 一个集合变量

集合变量

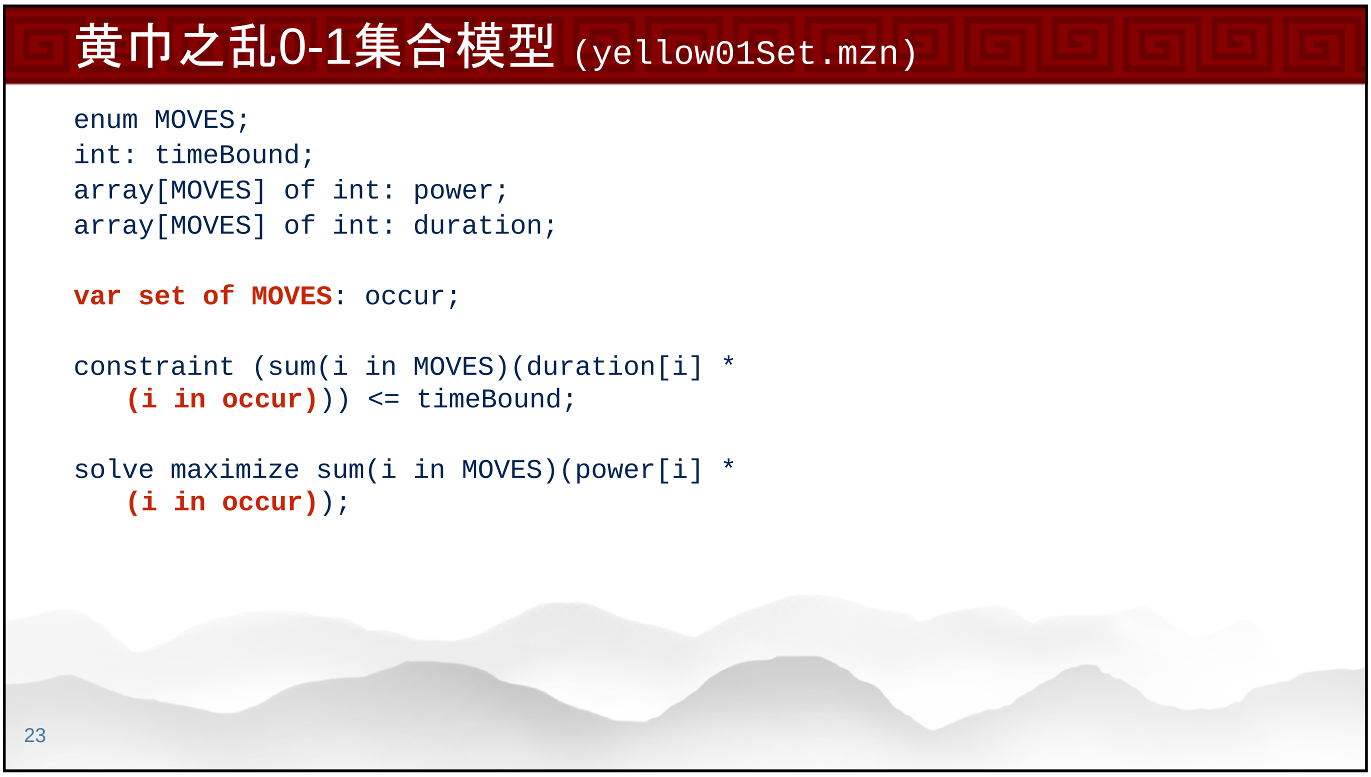
- MiniZinc中的集合变量从一个给定的固定超集中选择一个子集，例如：

```
var set of {1,2,3}: x;
```



黄巾之乱0-1布尔模型 (yellow01Bool.mzn)

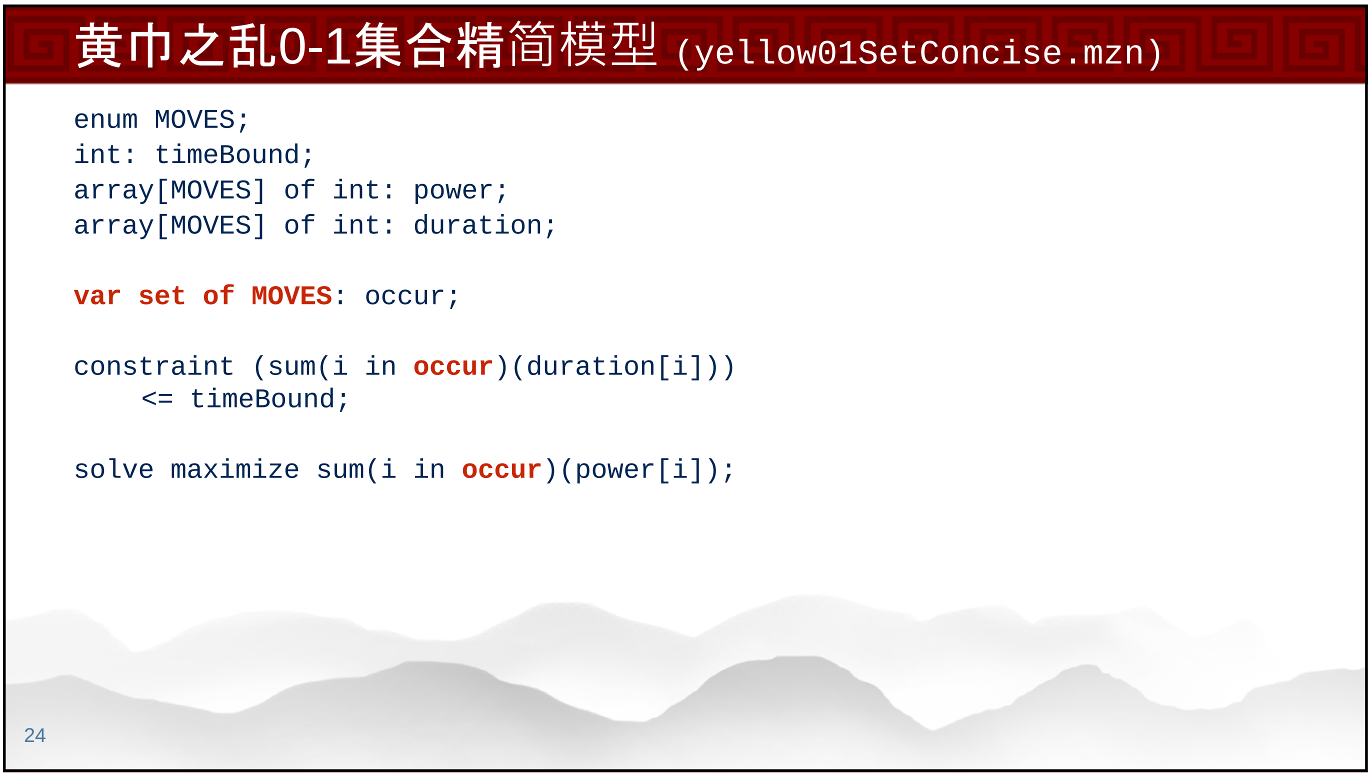
```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var bool: occur;

constraint (sum(i in MOVES)(duration[i] *
    ~~bool2int(~~occur[i]~~)~~)) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    ~~bool2int(~~occur[i]~~)~~);
```



黄巾之乱0-1集合模型 (yellow01Set.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

var set of MOVES: occur;

constraint (sum(i in MOVES)(duration[i] *
    (i in occur))) <= timeBound;

solve maximize sum(i in MOVES)(power[i] *
    (i in occur));
```

23



黄巾之乱0-1集合精简模型 (yellow01SetConcise.mzn)

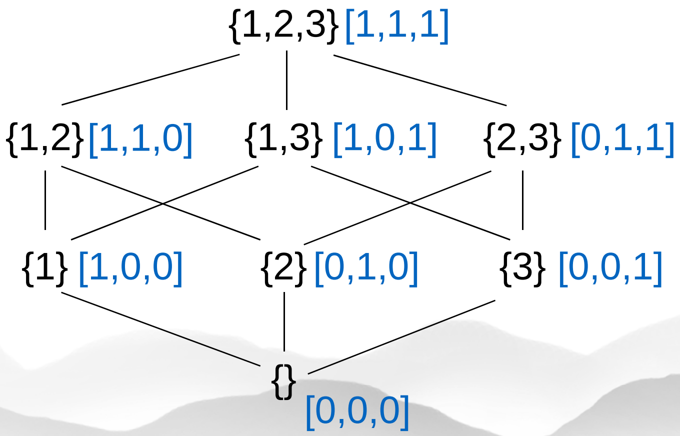
```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

var set of MOVES: occur;

constraint (sum(i in occur)(duration[i]))
     <= timeBound;

solve maximize sum(i in occur)(power[i]);
```

24

其他的集合表示方式

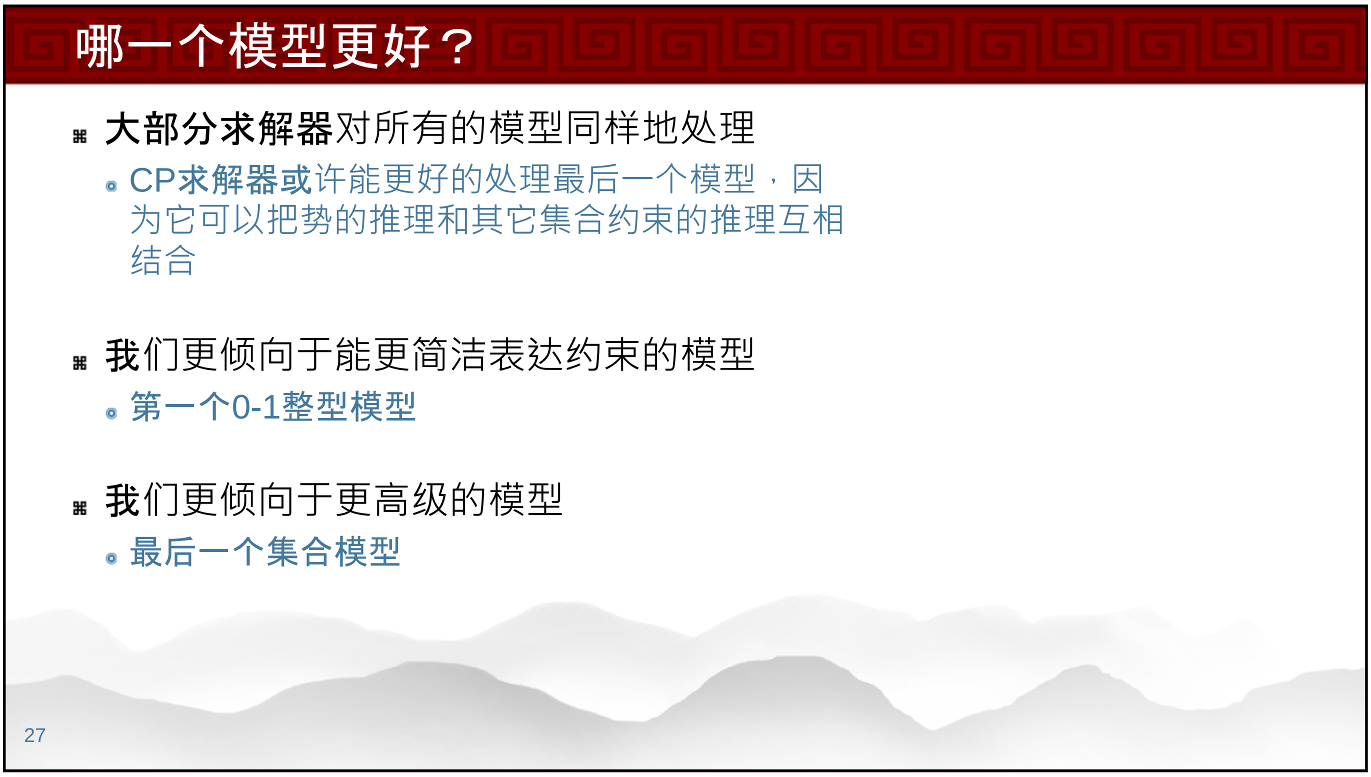
- 其他可对集合变量的可选值建模的集合表示方式，例如：

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



集合操作符

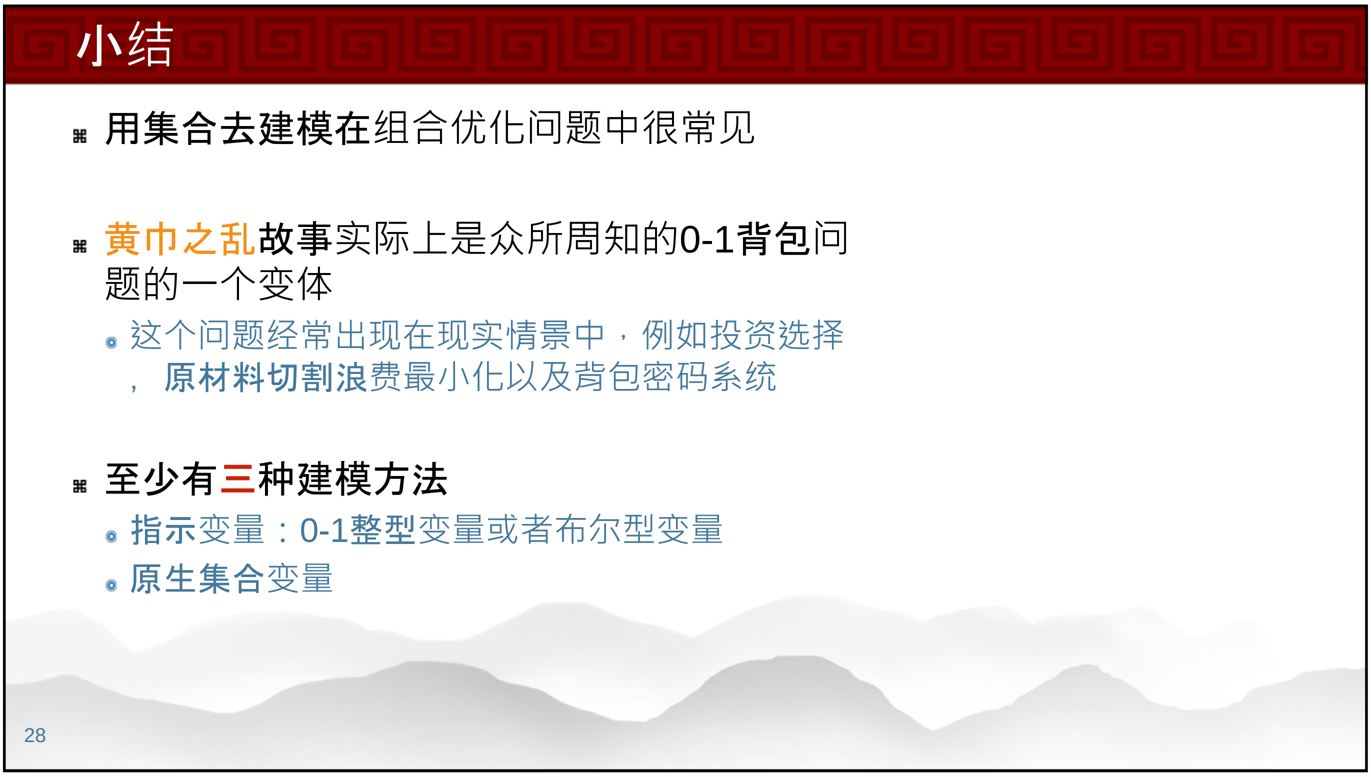
- MiniZinc提供了（中缀）集合操作符
  - in（集合中的元素 例如： x in s）
  - subset, superset（子集， 超集）
  - intersect（交集）
  - union（并集）
  - card（集合势）
  - diff（差运算， 例如：x diff y = x \ y）
  - symdiff（对称差）
    - 例如： {1, 2, 5, 6} symdiff {2, 3, 4, 5} = {1, 3, 4, 6}



哪一个模型更好？

- **大部分求解器**对所有的模型同样地处理
  - **CP求解器或**许能更好的处理最后一个模型，因为它可以把势的推理和其它集合约束的推理互相结合
- 我们更倾向于能更简洁表达约束的模型
  - **第一个0-1整型模型**
- 我们更倾向于更高级的模型
  - **最后一个集合模型**

27



小结

- **用集合去建模在**组合优化问题中很常见
- **黄巾之乱故事**实际上是众所周知的**0-1背包**问题的一个变体
  - 这个问题经常出现在现实情景中，例如投资选择，**原材料切割浪**费最小化以及背包密码系统
- **至少有三种建模方法**
  - **指示变量：0-1整型**变量或者布尔型变量
  - **原生集合**变量

28

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

29