数组和推导式

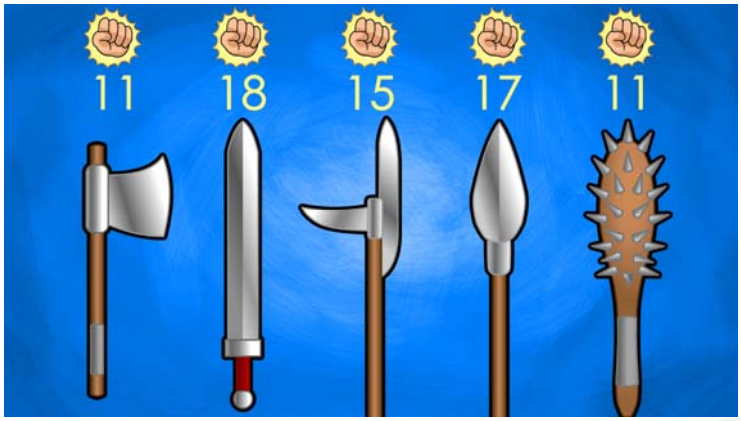
李浩文、彼得·斯塔基

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

兵器生产

2

兵器



3

资源

4

资源消耗

	1.5	2.0	1.5	0.5	0.1
	1.0	0.0	0.5	1.0	2.5
	1.0	2.0	1.0	0.9	0.1
	1.0	0.0	1.0	1.5	2.5

5

资源可用量约束

6

最大化战斗力

7

生产规划

- **我**们应当注意此类问题和之前的一些问题很**相似**
- **每个**“产品”**都消耗掉一些**资源
- 约束和目标
  - 资源的限制
  - **最大化利**润
- 组建军队
  - 资源 = 预算，产品 = **士兵**
- **桃园宴会**
  - 资源 = **桌子空**间，产品 = **菜品**
- **我**们可以为这些问题建立一个**一般化的模型**

8

生产规划 数据 (prod-plan.mzn)

```
% products
enum PRODUCT;
% Profit per unit for each product
array[PRODUCT] of float: profit;

% resources
enum RESOURCE;
% Amount of each resource available
array[RESOURCE] of float: capacity;

% Units of each resource required to produce
%       1 unit of product
array[PRODUCT,RESOURCE] of float: consumption;
```

两维数组声明

9

生产规划 约束 (prod-plan.mzn)

```
% Variables: how much should we make of each product
array[PRODUCT] of var int: produce;

% Must produce a non-negative amount
constraint forall(p in PRODUCT)
                 (produce[p] >= 0);

% Production can only use the available resources:
constraint forall(r in RESOURCE)(
   sum (p in PRODUCT)
      (consumption[p, r] * produce[p]) <= capacity[r]
);

% Maximize profit
solve maximize sum(p in PRODUCT)
   (profit[p]*produce[p]);

output ["\(p): \(produce[p])\n" | p in PRODUCT];
```

二维数组查找

10

数组

- 一个数组可以是多维的，可如下声明为
  - `array[`*下标集合1*`, `*下标集合2*`, ...] of` 类型
- 数组的下标集合必须是
  - 一个整型范围或者枚举类型
  - 或者是固定值的集合表达式，而它的值则是一个范围
- 数组的元素可以是任何类型，但不可以是另外一个数组，例如，
  - `array[PRODUCT,RESOURCE] of int: consumption;`
- 内建函数`length`返回一维数组的长度

11

数组（续）

- 一维数组使用列表的形式来初始化
  - `profit = [400, 500];`
  - `capacity = [4000, 6, 2000, 60, 50];`
- 二维数组使用二维数组语法来初始化
  - 起始于`[|`
  - `|` 用来分隔行（第一维）
  - 结束于`|]`

```
consumption = [| 1.5, 1.0, 1.0, 1.0
               | 2.0, 0.0, 2.0, 0.0
               | 1.5, 0.5, 1.0, 1.0
               | 0.5, 1.0, 0.9, 1.5
               | 0.1, 2.5, 0.1, 2.5 |];
```

12

数组（续）

- 对任何维度的数组（≤ 6）**都可以使用**array*n*d **族的函数**进行初始化。它们把一个一维数组转换为一个*n* 维数组，例如，同等地

```
consumption = array2d(1..5, 1..4,
   [1.5, 1.0, 1.0, 1.0,
    2.0, 0.0, 2.0, 0.0,
    1.5, 0.5, 1.0, 1.0,
    0.5, 1.0, 0.9, 1.5,
    0.1, 2.5, 0.1, 2.5];
```

13

数组推导式

- MiniZinc**提供数**组推导式（类似Haskell和ML）

- **数**组推导式有以下形式
  - [表达式 | *生成*器*1*, *生成*器*2*, …]
  - [表达式 | *生成*器*1*, *生成*器*2*, … where 测试]

- **例如** `[i + j | i, j in 1..4 where i < j]`

```
= [1+2, 1+3, 1+4, 2+3, 2+4, 3+4]
= [3, 4, 5, 5, 6, 7]
```

14

数组（续）

- 对任何维度的数组（≤ 6）**都可以使用**array*n*d **族的函数**进行初始化。它们把一个一维数组转换为一个*n* 维数组，例如，同等地

```
consumption = array2d(1..4, 1..5,
 [1.5,1.0,1.0,1.0,2.0,0.0,2.0,0.0,1.5,0.5,
  1.0,1.0,0.5,1.0,0.9,1.5,0.1,2.5,0.1,2.5]);
```

- array*n*d **把一个一**维数组转换为一个*n* 维数组；**我**们也可以使用推导式来把一个*n* 维数组展平为一个列表（一维数组），例如，

```
array[1..20] of int: list =
   [consumption[i,j] | i in 1..5, j in 1..4];
```

15

迭代

- MiniZinc**提供了各种**对列表或者集合进行操**作的内建函数**
  - **数字列表**：`sum, product, min, max`
  - 约束列表：`forall, exists`
- **在**调用这些（以及其他的生成器函数）时，MiniZinc**提供了特殊的**语法，例如，

```
forall(i,j in 1..10 where i < j)
      (a[i] != a[j])
```

等价于（单个**参数的**forall)

```
forall([a[i] != a[j]
     | i,j in 1..10 where i < j])
```

16

兵器生产 (prod-plan-weapon.dzn)

- 数据

```
PRODUCT = {AXE, SWORD, PIKE, SPEAR, CLUB};
profit =  [11.0, 18.0, 15.0, 17.0, 11.0];
RESOURCE = {IRON, WOOD, SMITH, CARPENTER};
capacity = [5000, 7500, 4000, 3000];
consumption = [| 1.5, 1.0, 1.0, 1.0
               | 2.0, 0.0, 2.0, 0.0
               | 1.5, 0.5, 1.0, 1.0
               | 0.5, 1.0, 0.9, 1.5
               | 0.1, 2.5, 0.1, 2.5 |];
```

二维数组常量

每行对应一种PRODUCT
每列对应一种RESOURCE

17

兵器生产

- 根据数据求解兵器生产模型，得到

```
AXE: 0
SWORD: 632
PIKE: 2340
SPEAR: 440
CLUB: 0
PROFIT: 53956.0
----------
==========
```

- 英雄们已经准备好去武装他们的军队了！

18

其他生产规划例子

- 组建军队 (prod-plan-army.dzn)

```
PRODUCT = {F, L, Z, J};
profit = [6.0, 10.0, 8.0, 40.0];
RESOURCE = {MONEY};
capacity = [10000.0];
consumption = [| 13.0 | 21.0 | 17.0 | 100.0 |];
```

- 桃园宴会 (prod-plan-banquet.dzn)

```
PRODUCT = {SNAKESOUP, GONGBAOFROGS, MAPOTOFU};
profit = [29.0, 19.0, 8.0];
RESOURCE = {SIZE};
capacity = [18.0];
consumption= [| 8.0 | 5.0 | 3.0 |];
```

19

小结

- **真正的模型可以**应用到不同**大小的数据上**
- MiniZinc使用
  - **枚**举类型为对象命名
  - **数**组来捕捉（存放）对象的信息
  - **推**导式**用来**创建适用于不同规模的数据的
    - 约束，以及
    - **表达式**

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