全局势约束

李浩文、彼得·斯塔基

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

求解模型

- 求解 … 求解 … 求解 …
- 5分钟后还是没有任何结果
- 在10.5分时终于得到了一个结果
- 更改天数为6，只需要7秒这个模型就可以找到一个最优解

哪个地方不对？

2

排班需求约束

- 每个夜晚有o个士兵值班

```
forall(d in DAY)
   (sum(s in SOLDIER)
      (roster[s,d] = NIGHT) = o);
```

- 每个傍晚有l到u个士兵值班

```
forall(d in DAY)
   (sum(s in SOLDIER)
      (roster[s,d] = EVE) >= l);
forall(d in DAY)
   (sum(s in SOLDIER)
      (roster[s,d] = EVE) <= u);
```

3

排班需求约束

- 每个夜晚o个士兵值班

```
forall(d in DAY)
   (sum(s in SOLDIER)
      (roster[s,d] = NIGHT) = o);
```

- 每个傍晚有l到u个士兵值班

```
forall(d in DAY)
   (sum(s in SOLDIER)
      (roster[s,d] = EVE) >= l);
forall(d in DAY)
   (sum(s in SOLDIER)
      (roster[s,d] = EVE) <= u);
```

共同的子表达式

4

中间变量

- 中间变量
  - 存储会再次用到的表达式的值
  - 依赖于决策变量
  - (注意: 中间参数也是!)

```
array[DAY] of var 0..card(SOLDIER): onEve;
onEve = [sum(s in SOLDIER)(roster[s,d] = EVE)
   | d in DAY];
forall(d in DAY)
   (onEve[d] >= l /\ onEve[d] <= u);
```

- 给中间变量选择一个好的界限
- **或者直接地**

```
array[DAY] of var l..u: onEve;
onEve = [sum(s in SOLDIER)(roster[s,d] = EVE)
      | d in DAY];
```

5

另外一个子表达式

- 目标

```
var int: tOnEve = sum(d in DAY)
   (sum(s in SOLDIER)(roster[s,d] = EVE));
solve maximize (tOnEve);
```

6

巡逻模型 (patrolV2.mzn)

```
enum SOLDIER;
enum SHIFT;
int: nDays;
set of int: DAY = 1..nDays;
int: o;
int: l;
int: u;

array[SOLDIER,DAY] of var SHIFT: roster;

constraint forall(d in 1..(nDays-2),
   s in SOLDIER)((roster[s,d] = NIGHT) /\
      (roster[s,d+1] = NIGHT)
   -> (roster[s,d+2] != NIGHT));
constraint forall(d in 1..(nDays-1),
   s in SOLDIER)((roster[s,d] = EVE) ->
      (roster[s,d+1] != NIGHT));
```

7

巡逻模型 (patrolV2.mzn)

```
constraint forall(d in DAY)(sum (s in SOLDIER)
   ((roster[s,d] = NIGHT)) = o);
array[DAY] of var l..u: onEve;
constraint onEve = [sum (s in SOLDIER)
   (roster[s,d]=EVE) | d in DAY];

solve maximize sum(onEve);
```

8

求解新模型

- 除去共同子表达式后的新模型在 4.5 秒内求解完毕

9

划分问题

- **很多**时候当我们划分一个集合时，我们必须**要限定每个划分**类的大小
  - **例如**，`#NIGHT = 0, 1 ≤ #EVE ≤ u`
- **我**们有特殊的约束来限定划分类的大小
  - `global_cardinality(x, v, c)`
  - **v和c的大小一样**
  - 约束 $c_i = \Sigma_{j \text{ in } 1..n}(x_j = v_i)$
  - **收集出**现次数，限定v_i**出**现的次数
  - **例如**，`global_cardinality(x,[1,2],[2,1])`
    `x = [1,1,2,3]` ✓, `[1,2,3,4]` ✗

10

全局势约束

- 替换

```
forall(d in DAY)(sum(s in SOLDIER)
   (roster[s,d] = NIGHT) = o);
forall(d in DAY)(sum(s in SOLDIER)
   (roster[s,d] = EVE) >= l);
forall(d in DAY)(sum(s in SOLDIER)
   (roster[s,d] = EVE) <= u);
```

- 为

```
array[DAY] of var l..u: onEve;
constraint forall(d in DAY)
   (global_cardinality(
      [roster[s,d] | s in SOLDIER],
      [NIGHT, EVE],
      [o, onEve[d]]));
```

11

全局势约束的变种

- **全局**势约束有很多个变种

- **收集出**现次数，要求每个值都出现
  - `global_cardinality_closed(x, v, c)`
  - x_i 的值都要从 v 来: 就是 $\forall i.\exists j.\ x_i = v_j$

- **限定出**现次数的上限和下限
  - `global_cardinality_low_up(x, v, lo, hi)`
  - 约束 $lo_i \le (\Sigma_{j \text{ in } 1..n} (x_j = v_i)) \le hi_i$
- **限定出**现次数的上限和下限，要求每个值都出现

`global_cardinality_low_up_closed(x,v,l,u)`

12

巡逻模型 (patrolV3.mzn)

```
enum SOLDIER;
enum SHIFT;
int: nDays;
set of int: DAY = 1..nDays;
int: o;
int: l;
int: u;

array[SOLDIER,DAY] of var SHIFT: roster;

constraint forall(d in 1..(nDays-2),
   s in SOLDIER)((roster[s,d] = NIGHT) /\
      (roster[s,d+1] = NIGHT)
   -> (roster[s,d+2] != NIGHT));
constraint forall(d in 1..(nDays-1),
   s in SOLDIER)((roster[s,d] = EVE) ->
      (roster[s,d+1] != NIGHT));
```

13

巡逻模型 (patrolV3.mzn)

```
array[DAY] of var l..u: onEve;
constraint forall(d in DAY)(global_cardinality(
   [roster[s,d] | s in SOLDIER],
   [NIGHT, EVE], [o, onEve[d]]));

solve maximize sum(onEve);
```

14

求解全局约束模型

- 全局约束版本的模型在 1.5 秒内求解完毕
- 在接下来的课程中，我们会解释建模是如何影响求解效率的

15

小结

- 很多离散优化问题都涉及到
  - 确定一个函数 f: DOM ➔ COD
  - 这个可以看成是划分DOM
- 从划分的角度来看问题在我们推理集合F(c)时是很有帮助的 (即 f 的伪逆函数)
- 在势约束下进行划分时，子结构可以由
  - `global_cardinality` 族 来表达
- 共同子表达式：
  - 中间变量
- 逻辑联结词

16

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

17