多重建模

李浩文、彼得·斯塔基

煮酒论英雄

2

食物和酒配对的欢愉程度

3

另一个纯分配问题

- 再次遇到确定函数 f: DOM ➔ COD的问题
  - DOM = 食物 以及 COD = 酒

- 约束：为每盘菜配不同的酒
- 目标：最大化宴席（政治）场合的欢愉度

4

煮酒问题 (foodToWine.mzn)

```
include "globals.mzn";

enum FOOD;
enum WINE;
array[FOOD, WINE] of int: joy;

array[FOOD] of var WINE: drink;

constraint alldifferent(drink);

solve maximize sum(f in FOOD)(joy[f, drink[f]]);
```

5

求解煮酒问题

```
Pair Food CHILIFISHHEAD with Wine HUADIAO
Pair Food MAPOTOFU with Wine GRAPE
Pair Food SNAKESOUP with Wine RICE
Pair Food GONGBAOFROG with Wine GAOLIANG


Joy: 21
----------
==========
```

6

多重模型

- 离散优化问题通常有
  - 对同一个问题的多重视角
- 我们可以建立两个（或多个）完全不同的模型来解决同一个问题
- 我们也可以结合它们

7

视角

- 在以下情况下，函数 f: DOM ➔ COD 是特殊的
  - |DOM| = |COD|
  - 函数 f 是双射的
- 某一视角会从一个特定的角度看待问题的决策
- 这是一个完全的匹配：
  - 每个DOM中的*d* 被匹配给不同的COD中的 *c*
  - 或者，等价地，每个COD中的 *c* 被匹配给不同的DOM中的 *d*
  - 两个不同的视角 ==> 两个互补的模型

8

完全匹配

- 一个双射函数有两个视角
- （通常的）函数
  ```
  array[DOM] of var COD: f;
  ```
- 和它的逆函数
  ```
  array[COD] of var DOM: finv;
  ```

- 就跟我们可以为食物配对酒一样，我们也可以为酒配对食物
- 在煮酒问题中，逆函数是
  ```
  array[WINE] of var FOOD: eat;
  ```

9

煮酒问题 (wineToFood.mzn)

```
include "globals.mzn";

enum FOOD;
enum WINE;
array[FOOD, WINE] of int: joy;

array[WINE] of var FOOD: eat;

constraint alldifferent(eat);

solve maximize sum(w in WINE)(joy[eat[w], w]);
```

10

求解逆向煮酒问题模型

```
Pair Food MAPOTOFU with Wine GRAPE
Pair Food SNAKESOUP with Wine RICE
Pair Food GONGBAOFROG with Wine GAOLIANG
Pair Food CHILIFISHHEAD with Wine HUADIAO


Joy: 21
----------
==========
```

11

配对食物和酒

- 哪一个模型可能会更好？
  - 初始

```
alldifferent(drink);
maximize sum(f in FOOD)(joy[f, drink[f]]);
```

  - 运用逆函数

```
alldifferent(eat);
maximize sum(w in WINE)(joy[eat[w], w]);
```

12

连通约束

- 问题的有些约束可能用函数或者它的逆函数**更容易表达**
- 为什么不两个都用**呢!?**

- **我**们可以把两个模型结合
- **通**过使用连通约束，我们使两个函数一致

```
forall(w in WINE, f in FOOD)
      (eat[w] = f <->
       drink[f] = w);
```

13

使用inverse来结合两个模型

- **此**连通可以使用以下全局约束来描述
  - `inverse(eat, drink)`
  - **或** `inverse(drink, eat)`
  - **注意到**`inverse`**已经使**`alldifferent`**约束变得冗余，我**们可以将之**去掉**

- **我**们为什么要结合模型？

14

煮酒问题 (combined.mzn)

```
include "globals.mzn";

enum FOOD;
enum WINE;
array[FOOD, WINE] of int: joy;

array[FOOD] of var WINE: drink;
array[WINE] of var FOOD: eat;

constraint inverse(eat, drink);

solve maximize sum(f in FOOD)(joy[f, drink[f]]);
% solve maximize sum(w in WINE)(joy[eat[w], w]);
```

15

煮酒问题 (combined.mzn)

```
include "globals.mzn";

enum FOOD;
enum WINE;
array[FOOD, WINE] of int: joy;

array[FOOD] of var WINE: drink;
array[WINE] of var FOOD: eat;

constraint inverse(eat, drink);

solve maximize sum(f in FOOD)(joy[f, drink[f]]);
% solve maximize sum(w in WINE)(joy[eat[w], w]);
```

16

为什么要结合模型？

- **如果做得合适，基于CP的求解器可以从模型结合中获益，提高求解效率**
- **方便**约束表达
  - **煮酒**问题，正如你看到的那样，是一个纯匹配问题。**不是的！**
  - **但是**还有**其他**约束**呢？**

17

其他约束举例

- **味道的丰富程度**

3 2 1 4

```
array[FOOD] of int: taste;
```

- **和高粱配**对的菜肴必须比和葡萄酒配对的菜肴有更丰富的味道

18

其他约束举例

- **味道的丰富程度**

3 2 1 4

```
array[FOOD] of int: taste;
```

- **和高粱配**对的菜肴必须比和葡萄酒配对的菜肴**有更丰富的味道**

```
taste[eat[GRAPE]] < taste[eat[GAOLIANG]];
```

- **即使并非不可能**，这个约束在食物-**酒模型中很难表达**

19

其他约束举例

- **酒精**浓度

1 3 4 2

```
array[WINE] of int: alcohol;
```

- **和**宫爆田鸡配对的酒必须比和蛇羹配对的酒**有更高的酒精**浓度

20

其他约束举例

- **酒精**浓度

1 3 4 2

```
array[WINE] of int: alcohol;
```

- **和**宫爆田鸡配对的酒必须比和蛇羹配对的酒**有更高的酒精**浓度

```
alcohol[drink[SNAKESOUP]] <
   alcohol[drink[GONGBAOFROG]];
```

- **即使并非不可能**，这个约束在酒-**食物模型中**很难**表达**

21

其他约束举例

- **当且**仅当蛇羹和花雕配对时，麻婆豆腐才会**跟米酒配**对

22

其他约束举例

- **当且**仅当蛇羹和花雕配对时，麻婆豆腐才会**跟米酒配**对

```
eat[RICE] = MAPOTOFU <->
   eat[HUADIAO] = SNAKESOUP;
```

或

```
drink[MAPOTOFU] = RICE <->
   drink[SNAKESOUP] = HUADIAO;
```

或

```
eat[RICE] = MAPOTOFU <->
   drink[SNAKESOUP] = HUADIAO;
```

或

```
drink[MAPOTOFU] = RICE <->
   eat[HUADIAO] = SNAKESOUP;
```

23

小结

- 问题的多重视角导致了
  - **多重模型**
- **不同的**视角可以表达不同的约束
  - **更自然也更**简洁
  - 对求解器更好（通常越简洁越好）
- 连通约束使多个视角的约束表达一致，并结**合他**们
- 结合模型有时候可以比单独用一个模型有更**高的求解效率**

24

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

25