ELEYANG DESIGN

La Vita E Bella

Mio

MiniZinc

ELEYANG DESIGN

1

Content

内容

Review

复习记录

/ / / / /

Chapter

Title week1 / PPT 7 Date 6 / 1

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

I Array

1D→nD: 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]);

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

II Set

operator: in, union, intersect, subset, superset, diff, symdiff

III exercise

Exercise: What does b =?

```
set of int: COL = 1..5;
set of int: ROW = 1..2;
array[ROW,COL] of int: c =
    [| 250, 2,  75, 100,  0
     | 200, 0, 150, 150, 75 |];
b = array2d(COL, ROW,
   [c[j, i] | i in COL, j in ROW]);
```

- b is the transpose of c

```
[c[j, i] | i in COL, j in ROW ] =
[ 250, 200, 2, 0, 75,
  150, 100, 150, 0, 75]

b = [| 250,200
     |   2,  0
     |  75,150
     | 100,150
     |   0, 75
     |];
```

IV alldifferent

include "alldifferent.mzn";

constraint alldifferent(...);

Content Index 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

2

Content
内容

集合选择

There are multiple ways to represent sets

`var set of OBJ`

- good if the solver natively supports sets
- good when OBJ is not too big

`array[OBJ] of var bool / 0..1`

- good when OBJ is not too big

`array[1..u] of var OBJ`

- only for fixed cardinality u
- good when u is small

`array[1..u] of var OBJx`

- need to represent the "null" object

Review
复习记录

/ / / / /

Chapter

Title Week 2 PPT1 Date 6/1

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

Select a set 、 bool2int

如何表示是否选中了一组对象？

一. 0-1 int

eg1. 使用 int 0..1

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

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

二. bool2int

eg2 使用bool2int

```
enum MOVES;
array[MOVES] of var bool : occur
```

或者 var int 0..1

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

会自动强制转化

三. var set

eg3 使用var set形式

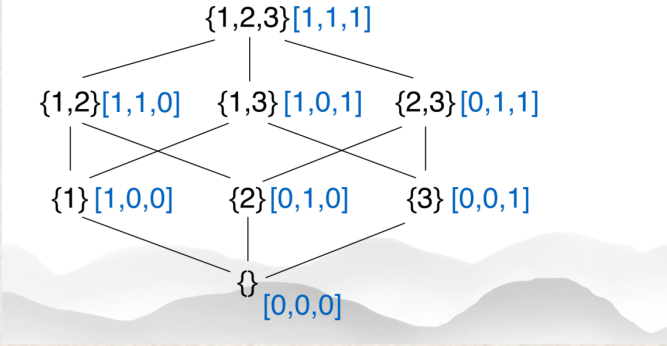
```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;
var set of MOVES: occur;
constraint (sum(i in occur)(duration[i])) <= timeBound;
solve maximize sum(i in occur)(power[i]);
```

Content Index 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

Keywords 关键词 | Notes 笔记

- Other set representations that model the possible values of a set variable, e.g.

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



- set operator
  - in (membership e.g. x in S)
  - subset, superset
  - intersect (intersection)
  - union
  - card (cardinality)
  - diff (set difference, e.g x diff y = x \ y)
  - symdiff (symmetric difference)
    - e.g. {1, 2, 5, 6} symdiff {2, 3, 4, 5} = {1, 3, 4, 6}

Summary 总结

Chapter

Title week 2 PPT 3

Date 6/1

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

choose a fixed set

如何确保每个解决方案在模型中只有一个？

1. consider

   array[1..3] of var 1..10: xi

   有 $10^3 = 1000$ 种可能的值

2. consider

   var set of 1..10: x

   card (x) = 3

   有 $10 \times 9 \times 8 / 3 \times 2 \times 1 = 120$ 种

   ensure all different

   forall (i,j in 1..u where i<j)

   (x[i] != x[j])

   有 $10 \times 9 \times 8 = 720$ 种

3. ensure ordered

   forall (i in 1..u-1) (x[i] < x[i+1])

Content Index

Chapter

Title week 2 PPT 4 Date /

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

OBJ x

```
int: size;
int: nSpots;
Set of int: SPOT = 1..nSpots;
var set of SPOTx: attacks;
```

Set of int: SPOTx = {0} union SPOT;

```
constraint card(attacks) <= size;
```

- Each solution in the model represent a solution in the problem:
  - 除0以外m值不可被重复!

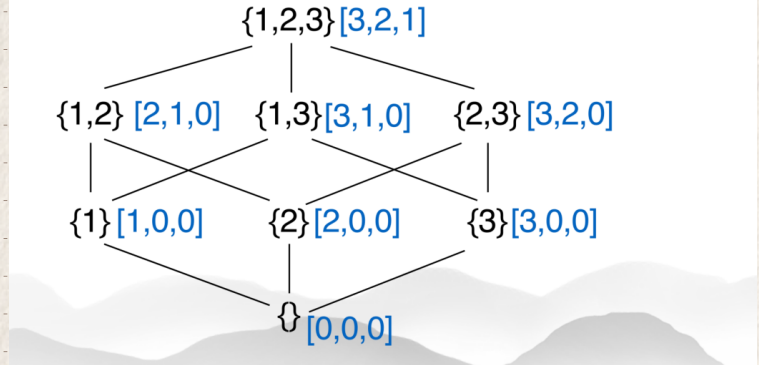
    [3,0,3] x

    [0,2,0] ✓

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

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

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



Content Index

ELEYANG DESIGN

3

Content
内容

Review
复习记录

/ / / / /

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

分配问题

DOM ↔ COD

问题：士兵轮班问题，为了增加营地的安全性，刘备需要安排士兵值班，晚班需要有且只有0个士兵巡逻，早上不需要巡逻，傍晚需要l~u个士兵巡逻，一个士兵不能连续值晚班超过2晚，一个士兵不能值完下午班后值晚班，为了保证安全性，在给定的士兵，给定的巡逻天数的情况下，使得在这几天里，参与巡逻的人数最大化，其实就是最大化下午班人数的数量(同一士兵参与不同值班可重复计数)，一个士兵不违反规定的值班安排表

例子如下：

代码：

```
enum SOLDIER;
enum SHIFT = { OFF, EVE, NIGHT };
int: nDays;
set of int: DAY = 1..nDays;
int: o;
int: l;
int: u;
```

为每个士兵的某一天，分配给定天数中的值班时间段

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

一个士兵不能连续值晚班超过2晚：

```
forall(s in SOLDIER, k in 1..nDays - 2)(
 sum(d in k..k + 2)(roster[s, d] = NIGHT) <= 2
);
```

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

⇒这样写更好

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

一个士兵不能值完下午班后值晚班

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

人数上的限制

```
% 每天都有0士兵值夜班
constraint forall(d in DAY)(
    sum (s in SOLDIER)((roster[s,d] = NIGHT)) = 0
);

% 每个晚班都有l~u个士兵巡逻
constraint forall(d in DAY)(
    sum (s in SOLDIER)((roster[s,d] = EVE)) >= l
);
constraint forall(d in DAY)(
    sum (s in SOLDIER)((roster[s,d] = EVE)) <= u
);
```

目标：晚上应该有尽可能多的士兵巡逻

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

solve maximize (tOnEve);
```

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Global cardinality

I 优化

有公共的部分：

```
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);
```

可以使用中间变量存储被重用的表达式的值：

```
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);
```

或者

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

目标为

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

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

Ⅱ Global cardinality

1. global_cardinality

约束global_cardinality(x, v, c):

v和c的大小相同

约束的表达式为 ci = Σ(j in 1..n)(xj = vi)

收集计数，并限制vi出现次数的计数，例如：约束x中，1出现2次，2出现1次

global_cardinality(x,[1,2],[2,1])

所以可以把代码

```
forall(d in DAY)(sum(s in SOLDIER)
    (roster[s,d] = NIGHT) = 0);
  forall(d in DAY)(sum(s in SOLDIER)
    (roster[s,d] = EVE) >= 1);
  forall(d in DAY)(sum(s in SOLDIER)
    (roster[s,d] = EVE) <= 4);
```

替换为

```
array[DAY] of var 1..4: onEve;
constraint forall(d in DAY) (
          global_cardinality(
      [roster[s,d] | s in SOLDIER],
      [NIGHT, EVE],
      [0, onEve[d]]
        )
);
```

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

2. global_cadinality_closed (x, v, c)

x中的所有数都必须出现在v中

3. global_cardinality_low_up (x, v, lo, hi)

4. global_cardinality_low_up_closed (x, v, lo, hi)

x中的所有数都必须出现在v中

ELEYANG DESIGN

4

Content
内容

Review
复习记录

/ / / / /

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

Inverse

限制：将每道菜与不同的饮料搭配

目标：最大限度地提高烹饪(和政治)的乐趣

两个互补模型：inverse

inverse(eat, drink)

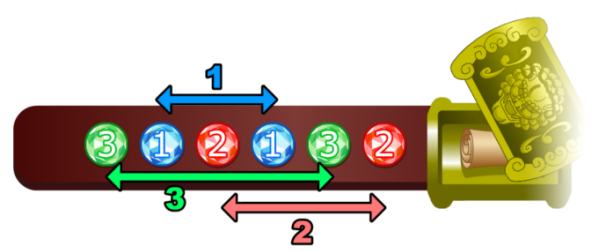
or inverse(drink, eat)

```
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]]);
```

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Permutation 置换

一、问题：B(m,n):给出每个数字1..n的m个拷贝。找到这些数字的序列，其中每两个k的连续副本中有k个数字



二、分析：

Map DOM = 1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 4_1, 4_2 数字的有序拷贝

三、数据和决策

```
int: n;
set of int: DIG = 1..n;
int: m;
set of int: COPY = 1..m;
int: L = m*n;
set of int: POS = 1..L;
array[DIG,COPY] of var POS: po;
约束
forall(d in DIG, c in 1..m-1)
      (po[d,c+1] = po[d,c] + d + 1);
alldifferent([po[d,c] |d in DIG, c in COPY]);
```

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

四、逆向考虑：

DOM是位置，COD是数字副本

我们需要将DIG × COPY映射为单个整数：dc = m*(d-1) + c

1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 4_1, 4_2 = 1, 2, 3, 4, 5, 6, 7, 8

set of int: DIGCOP = 1..L; array[POS] of var DIGCOP: dc;

约束alldifferent([dc[p] | p in POS]);

进行建模dc[p] = dc <=> po[d,c] = p

正向思考建模代码：

forall(d in DIG, c in 1..m-1)

(po[d,c+1] = po[d,c] + d + 1);

逆向约束：

forall(d in DIG, c in 1..m-1, p in POS)

完整代码

```
include "globals.mzn";
int: n;
set of int: DIG = 1..n;
int: m;
set of int: COPY = 1..m;
int: L = m*n;
set of int: POS = 1..L;
set of int: DIGCOP = 1..L;
array[POS] of var DIGCOP: dc;
constraint forall(d in DIG, c in 1..m-1, p in POS)
(dc[p] = m*(d-1) + c <-> dc[p+d+1] = m*(d-1) + c + 1);
constraint alldifferent([dc[p] | p in POS]);
solve satisfy;
```

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

五、引入inverse

```
include "globals.mzn";
int: n;
set of int: DIG = 1..n;
int: m;
set of int: COPY = 1..m;
int: l = m*n;
set of int: POS = 1..l;
array[DIG,COPY] of var POS: po;
set of int: DIGCOP = 1..l;
array[POS] of var DIGCOP: dc;
constraint forall(d in DIG, c in 1..m-1)(
      po[d,c+1] = po[d,c] + d + 1
);
constraint inverse(dc, [po[d,c]|d in DIG, c in COPY]);
output["\((dc[p]-1) div m + 1) " | p in POS];
solve satisfy;
```

六、总结

- 排列问题至少要有两种观点
- 选择可能/容易表达约束和目标的观点
- 否则，选择两个视点并添加反向约束

例子：假设有n个人被排队拍照片，照片的友好值定义为一行中相邻的每对人之间的友好度之和

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

```
int:n ;
set of int: PERSON = 1..n;
set of int: POS = 1..n;
array[PERSON,PERSON] of int: friend;
```

如果变量为每个人的位置array[PERSON] of var POS: x;

Constraints: alldifferent(x);

目标:solve maximize …难以表达

如果变量为每个人array[POS] of var PERSON: y;

Constraints alldifferent(y);

目标: solve maximize sum(i in 1..n-1) (friend[y[i],y[i+1]]);

5

Content
内容

Debug

- No solutions
- Too many solutions
- Missing solutions

How to avoid?

- Assertions
- Tracing a model
- Relational semantics

Review
复习记录

/ / / / /

Chapter

Title Week5 PPT1 Date 6/3

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Assertion

一、定义

① assert(boolexp,stringexp)

- returns true if boolexp holds,
- otherwise prints stringexp and aborts

For example

eg1

```
array[RESOURCE] of int: capacity;
constraint assert(
    forall(r in RESOURCE)(capacity[r] >= 0),
    "Error: negative capacity");
```

eg2

Write an assertion to check that consumption is non-negative where

```
array[PRODUCT,RESOURCE] of int: consumption;
constraint forall(p in PRODUCT, r in RESOURCE)
        (assert(consumption[p,r] >= 0,
                "consumption[\(p),\(r)] < 0!"));
```

② 另一种 assertion：

assert(boolexp,stringexp,exp)

- returns exp if boolexp holds,
- otherwise prints stringexp and aborts

当不是所有的模型都将被执行时，特别是当我们稍后引入谓词和用户定义函数时，这很有用

Content Index

Keywords 关键词 | Notes 笔记

eg

```
int: n = 5;
array[1..n] of var 1..n: a;
array[1..n] of 1..n: b = [3,5,2,3,1];
constraint forall(j in 1..n, i in b[n-j]..n)
    (a[j] < i);
```

Error message

```
MiniZinc: result of evaluation is undefined:
debug1.mzn:5:
  in call 'forall'
  in array comprehension expression
     with j = 5
  in binary '..' operator expression
  in array access
  array access out of bounds
```

溢出下界，j=5时，i in 0..5

⇓ 利用assertion

```
int: n = 5;
array[1..n] of var 1..n: a;
array[1..n] of 1..n: b = [3,5,2,3,1];
constraint forall(j in 1..n)
  (assert(n-j in 1..n, "b[\(n-j)]"));
```

Error message

```
debug.mzn:5
  In constraint.
  In 'forall' expression.
  In comprehension.
  j = 5
  In comprehension head.
  In assert expression
  Assertion failure: "b[0]"
```

Summary 总结

始终添加断言以检查输入数据的假设

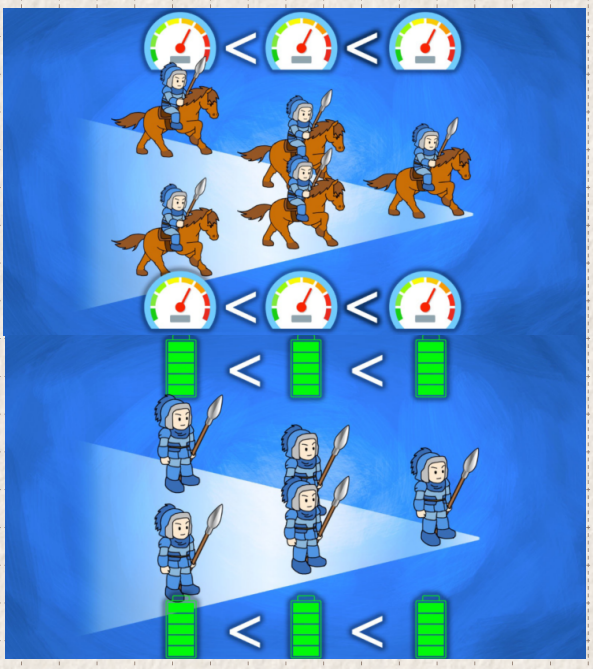
Chapter

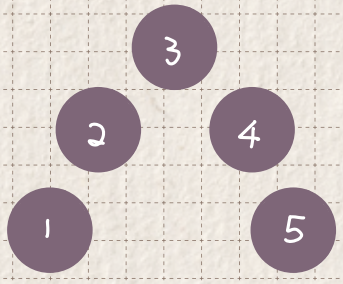
Title week 5 PPT 2 Date 6/3

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Tracing models

一、问题描述：通过将奇数的部队和士兵排列成楔形来建立编队。为了保持这种阵型，越前的马速度需要越快。并且越靠前的士兵有更大的机会与敌人战斗，所以越靠前的士兵需要越大的耐力。一些马和士兵相性极差/极好，不能/可以组合在一起。确定最佳的楔形编队，以最大限度的提高强度。



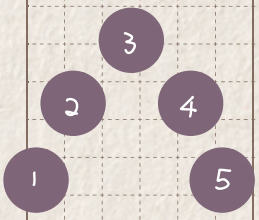


二、代码：

```
% Data Declarations
enum HORSE;
enum RIDER;
array[HORSE] of int: speed;
array[RIDER] of int: endur;
array[RIDER] of int: strength;
array[HORSE] of set of RIDER: compat;
int: n; % size of wedge (should be odd)
assert(n mod 2 = 1,"n must be odd");
set of int: POS = 1..n;

% Decisions
array[POS] of var HORSE: h;
array[POS] of var RIDER: r;
```

Content Index 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24




```
% Constraints
include "alldifferent.mzn";
alldifferent(h);
alldifferent(r);
forall(i in 1..n div 2)
    (speed[h[i]] < speed[h[i+1]] /\
    endur[r[i]] < endur[r[i+1]]);
forall(i in n div 2..n)
    (speed[h[i]] > speed[h[i+1]] /\
    endur[r[i]] > endur[r[i+1]]);
forall(i in POS)(r[i] in compat[h[i]]);
% Objective
solve maximize sum(i in POS)(strength[r[i]]);
output["h = \(h)\nr = \(r)\n"];
```

理想输出为下方代码：(但实际为 unsatisfied)

```
HORSE = {H1, H2, H3, H4, H5, H6, H7, H8, H9, H10};
RIDER = {R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11};
speed = [10, 9, 8, 7, 6, 5, 7, 4, 3, 2];
endur = [8, 4, 3, 2, 6, 4, 2, 6, 7, 5, 3];
strength = [5, 2, 8, 9, 4, 2, 1, 3, 4, 5, 9];
compatible = [ {R2, R3, R11}, {R5, R6}, {R8}, {R1, R5}, {R4}, {R2,
R7}, {R1, R3}, {R9, R1, R10}, {R11, R3}, {R9, R10, R7} ];
n = 5;
```

三、Trace

The builtin trace function prints out things during model compilation

trace(stringexp, exp)

- prints the value stringexp
- and then returns exp

Keywords 关键词	Notes 笔记

eg.

```
forall(i in 1..n div 2)(trace(
        "speed[h[\(i)]] < speed[h[\(i+1)]]\n"
++      "endur[r[\(i)]] < endur[r[\(i+1)]]\n",
        speed[h[i]] < speed[h[i+1]] /\
        endur[r[i]] < endur[r[i+1]]));
forall(i in n div 2..n)(trace(
        "speed[h[\(i)]] > speed[h[\(i+1)]]\n"
++      "endur[r[\(i)]] > endur[r[\(i+1)]]\n",
        speed[h[i]] > speed[h[i+1]] /\
        endur[r[i]] > endur[r[i+1]]));
```

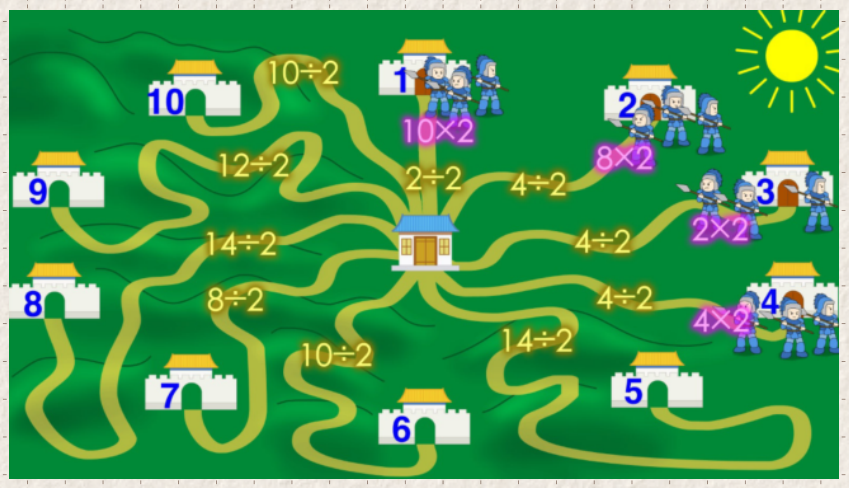
Chapter

Title Week5 PPT3 Date 6/3

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Retional Semantics
关系语义

问条描述：有10个出口，沿着这些出口步行需要花费不同的时间。有四个出口有士兵把手，所以需要额外的时间躲避士兵。逃跑时间是步行时间和躲避士兵时间之和。下雨天的时候，没有任何影响。晴天的时候，不幸事件减少一半，躲避士兵时间增加一倍。暴雨天气无法逃跑。



一、Data

```
set of int: PATH = 1..10;
set of int: POST = 1..4;
array[PATH] of int: time;
array[POST] of int: guard;
var 0..2: weather;
% 0 = storm, 1 = rain, 2 = sun
var PATH: path;
var int: t = time[path] div weather + guard[path] * weather;
solve minimize t;
time = [ 2, 4, 4, 4, 14, 10, 8, 14, 12, 10];
guard = [10, 8, 2, 4];
```

Content Index
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

Keywords 关键词 | Notes 笔记

二、问题

对于time[path] div weather, weather=0将不是解决方案的一部分
并且guard[path]会越界

替换

用中间变量extra存储(可能)溢出的值

```
var int: t = .. + extra * weather;
var int: extra;
path in POST -> extra = guard[path];
not(path in POST) -> extra = 0;
```

或者

```
var int: t = time[path] div weather +
  if path in POST then guard[path] else 0
  endif * weather;
```

三、其他

`not(x div y = 1)` not same as `x div y != 1`

(x,y) = (0,0) is a solution of the first

```
% Use this editor as a MiniZinc scratch book
int: x=5;
int: y=0;
bool: q1 = not(x div y = 1);
bool: q2 = x div y != 1;


output["q1=\(q1),q2=\(q2)"]

```

```
Output
Hide all  default
  Playground:4.1-8
    in variable declaration for 'q1'
    in unary 'not' operator expression
    in binary '=' operator expression
    in binary 'div' operator expression
Warning: undefined result becomes false in Boolean context
    (division by zero)

  Playground:5.1-8
    in variable declaration for 'q2'
    in binary '!=' operator expression
    in binary 'div' operator expression
Warning: undefined result becomes false in Boolean context
    (division by zero)
q1=true,q2=false
```

未定义(越界)返回 False

Summary 总结

relational semantics:

- division by zero
- array access out of bounds

these are just statements with no solution

尽量避免 →

partial function applications:

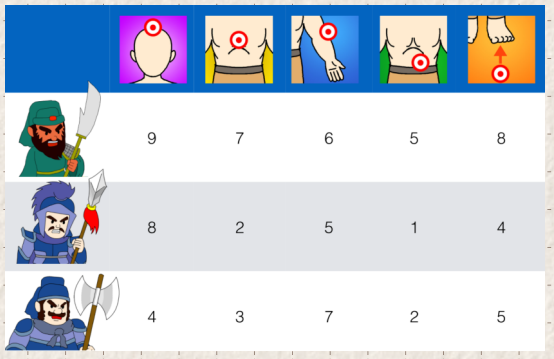
①x div y assure y != 0

②x[i] assure variable or parameter i is in the index_set of x

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

To many solutions

问题描述：关、张、徐各攻击一个将军的不同部位，而第一个将军有一个无法穿透的头盔，所以如果他们攻击第一个将军的下半身，就不能攻击头部，第二个将军将能够完美地保护下半身，所以他们不能攻击他的下半身。找出他们应该攻击的部位，以最大限度地增加对将军的总伤害。



	9	7	6	5	8
	8	2	5	1	4
	4	3	7	2	5


```
%data
enum HERO;
enum SPOT;
%decisions
array[HERO,SPOT] of int: damage;
array[HERO] of var SPOT: pos1;
array[HERO] of var SPOT: pos2;
%objective
solve maximize sum(h in HERO)
(damage[h,pos1[h]] +
damage[h,pos2[h]]);
%每一个都在不同的位置打击每一个将军
alldifferent(pos1);
alldifferent(pos2);
```

Keywords 关键词 | Notes 笔记

输出：

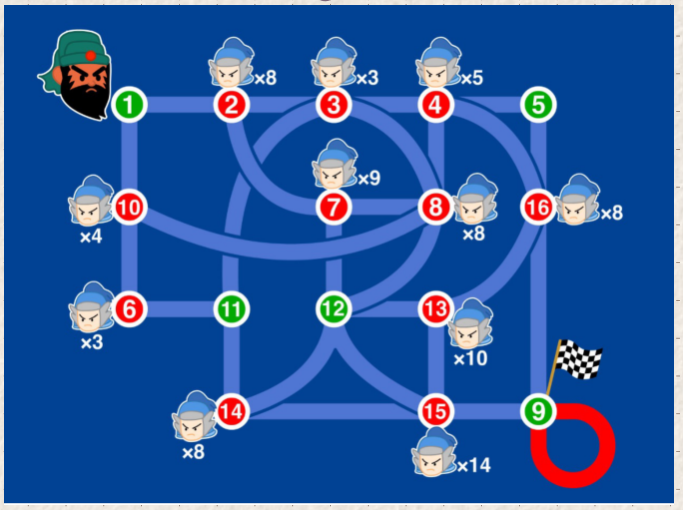
```
pos1 = [YONGQUAN, BAIHUI, QUCHI]
pos2 = [DANZHONG, BAIHUI, QUCHI]
```

修改，添加

```
forall(h in HERO)
    (pos1[h] != BAIHUI /\
      (pos1[h] in LO -> pos2[h] in HI));
```

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Missing solution



一.代码

```
%Data
int: n; set of int: NODE = 1..n;
array[NODE] of int: guard;
int: m; set of int: EDGE = 1..m;
array[EDGE,1..2] of NODE: edge;
NODE: start;
NODE: dest;
int: rest; % resting every rest junctions
 % Decisions
int: maxstep;
var int: step; % the actual number of steps
set of int: STEP = 1..maxstep;
array[STEP] of var NODE: path;
```

二. Table Global Constraint

table(array[1..n] of var int: x, array[1..T,1..n] of int: t)
强制x的值是从表t的同一行中获取的值
E.g.

table([x,y,z], [| 0,0,0 | 1,2,3 | 4,2,0 |])
either x = 0∧ y = 0∧ z = 0
or x = 1∧ y = 2∧ z = 3
or x = 4∧ y = 2∧ z = 0

Keywords 关键词 | Notes 笔记

eg2.

```
array[1..2*m,1..2] of NODE: uedge =
  array2d(1..2*m,1..2,
    [ edge[i,j] | i in EDGE, j in 1..2] ++
    [ edge[i,3-j] | i in EDGE, j in 1..2]);
```

三. Sliding Sum

```
sliding_sum(int: low,
            int: up,
            int: seq,
            array [int] of var int: vs)
```

强制每个子序列vs[i]..vs[i+seq-1]总和介于low和up之间
它用于强制序列的属性
E.g. sliding_sum(4, 8, 4, x)

x = [1,4,2,0,0,3,4] ✅

x = [1,4,3,0,1,0,2] ❌

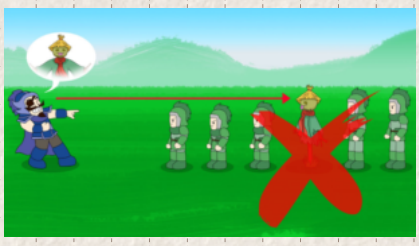
Chapter

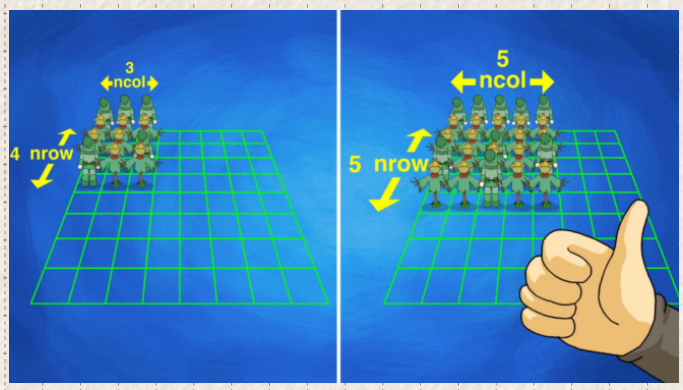
Title Week5 PPT6 Date 6/3

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

Model Improvement

问题描述：听说关羽要回来了，刘备带着几个部队跑出来迎接他。但夏侯敦紧随其后。刘备命令士兵列队，增加稻草士兵，使他们看起来更大。列队中有nrow行和ncol列。所有稻草士兵都有身高稻草高度。每个稻草士兵都有一个真正的士兵相邻。稻草士兵前面不能存在身高小于或等于它的士兵。目标是最大限度地增加表观士兵人数 (nrow*ncol)





一.代码

```
%Data
int: nsoldier; int: strawheight;
set of int: SOLDIER = 1..nsoldier;
set of int: SOLDIER0 = 0..nsoldier;
array[SOLDIER] of int: height;
int: maxr; % max rows
set of int: ROW = 1..maxr;
int: maxc; % max columns
set of int: COL = 1..maxc;
%Decisions (positions and size of parade)
 %soldier in position, or 0 = straw or empty
array[ROW,COL] of var SOLDIER0: x;
var ROW: nrow; var COL: ncol;
```

Content Index

Keywords 关键词 | Notes 笔记

其中注意：

```
%Each real soldier appears at most once
forall(i in 1..maxr*maxc)
   (let {int: r1 = (i-1) div ncol + 1;
      int: c1 = (i-1) mod ncol + 1; } in
   x[r1,c1] = 0 \/
   forall(j in i+1..maxr*maxc)
      (let {int: r2 = (j-1) div ncol + 1;
         int: c2 = (j-1) mod ncol + 1; } in
      x[r1,c1] != x[r2,c2] ));
```

可以用全局约束替换

二. alldifferent - except - 0

```
include "alldifferent_except_0.mzn";
alldifferent_except_0(
   [ x[r,c] | r in ROW, c in COL ]);
```

三.

```
% Each real soldier appears at least
once
forall(s in SOLDIER)
(exists(r in ROW, c in COL)
(x[r,c] = s));
```

或者可以写成

```
include "global_cardinality.mzn";
global_cardinality(
   [x[r,c] | r in ROW, c in COL],
   [s | s in SOLDIER],
   [1 | s in SOLDIER]);
```

或者可以写成

```
sum(r in ROW, c in COL)(x[r,c] != 0) =
nsoldier;
```

Content Index

Keywords 关键词 | Notes 笔记

```
%Each real soldier appears in the nrow*ncol area
%Force positions outside the parade to be empty
forall(r in ROW, c in COL)
( (r > nrow -> x[r,c] = 0)
/\ (c > ncol -> x[r,c] = 0));
%每个稻草士兵都有一个真正的士兵相邻(右、左、后、前)
%注意使用if-then-else以避免越界
forall(r in ROW, c in COL)
   ((r <= nrow /\ c <= ncol /\ x[r,c] = 0) ->
     ( if c < maxc then x[r,c+1] != 0 else false endif
       \/ if c > 1 then x[r,c-1] != 0 else false endif
       \/ if r < maxr then x[r+1,c] != 0 else false endif
       \/ if r > 1 then x[r-1,c] != 0 else false endif));
%没有一个稻草士兵面前只有身高小于或等于的士兵
not exists(r1 in ROW, c in COL)
     ((r1 <= nrow /\ c <= ncol /\ x[r1,c] = 0) ->
       forall(r2 in 1..r1-1)
          (x[r2,c] = 0 \/
          height[x[r2,c]] <= strawheight))
```

⑪. improvement

- negation `not C`
  - solvers definitely do not like negation
  - replace negation with negated form:
    - `not x = y` becomes `x != y`
- disjunctions `C1 \/ C2`
  - they will be necessary
  - try to make them as simple as possible
- implications `C1 -> C2`
  - make C1 especially simple
    - it's in a negated context
- existential loops `exists([ .. ])`

6

Content
内容

Review
复习记录

/ / / / /

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

Predicate

- Predicate Definition

```
predicate form(var set of ELITE: team) =
   card(team intersect ARCHER) >= 1
/\ card(team intersect CAVALRY) >= 2
/\ card(team intersect INFANTRY) >= 2
/\ card(team) = 6;
```

- Predicate Use

```
form(Liu); form(Guan); form(Zhang);
```

- More correctly (slides often omit keyword)

```
constraint form(Liu);
constraint form(Guan);
constraint form(Zhang);
```

Let - in

- Format

```
let {<type>:<varname> [ = <expr>] ;
          …
     <type>:<varname> [ = <expr>] [;]} in
<expr>
```

- Parameters introduced must have a defining expression, but variables do not
- New vars/pars visible **only** in remainder of let-in expression

- All tunnel exits different

```
let {array [POINT] of var 1..size*size:
   points = [(ptR[i]-1)*size + ptC[i] |
          i in POINT]} in
     alldifferent(points);
```

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

- All huts must be covered

```
predicate covered(var int: x, var int: y) =
   let {var POINT: i, var int: dist =
      abs(x-ptR[i]) + abs(y-ptC[i])} in
         dist <= mDist;

constraint let {
   array [1..(size div 2)] of 1..size-1:
      huts = [i*2|i in 1..(size div 2)]} in
         forall(i,j in huts)(covered(i,j));
```

7

Content
内容

Review
复习记录

/ / / / /

Keywords 关键词	Notes 笔记	Review 复习记录 week 7 Flatting / /

Flatting

I Flatting Expression Example

(i) model:

```
int: i=3; int: j=2;
Var int x; var 0..2:y; var 0..3:z;
x*y+ y*z<=i*j;
```

the resulting flatting model

```
Var int: x;
var 0..2: y;
var 0..3: z;
var 0..6: INTO1;
var int: INTO2;
INTO1 = y*z;
INTO2= x*y;
INTO1 + INTO2 <=6
```

in a more strict way → FlatZinc

```
Var int: x;
var 0..2: y;
var 0..3: z;
var 0..6: INTO1::is_defined_var
var int: INTO2:: is_defined_var
int_times(y,z,INTO1):: defines_var
(INTO1)
int_times(x,y,INTO2):: defines_var
(INTO2)
int_lin_le([1,1],[INTO1,INTO2],6);
```

(ii) model

```
int: i=3; int: j=3;
var 0..5: x ; var 0..2: y; var 0..3: z;
(x-i)*(x-j)+y+z+i+j >=0
```

common expression ← (x-i)*(x-j)

the resulting flatting model

```
var 0..5: x ; var 0..2: y; var 0..3: z;
Var -3..2: INTO1;
var -6..9: INTO2;
INTO1= x-3;
INTO2= INTO1*INTO1;
INTO2+ y+z+6 >=0
```

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

in a more strict way: FlatZinc

```
var 0..5: x ; var 0..2: y; var 0..3: z;
var -3..2: INT01 :: is_defined_var;
var -6..9: INT02 :: is_defined_var;
int_lin_eq([1,-1],[x,INT01],3):: dv(INT01);
int_times (INT01,INT02,INT02) :: dv(INT02);
int_line_le([-1,-1,-1],[z,y,INT02],6);
```

(iii) model

$(x-y)*(y-x) \geq y-x;$

flat:

```
INT01: x-y;
INT02: y-x;
INT03: INT01 * INT02;
INT03 >= INT02
```

FlatZinc →

```
int_lin_eq([1,-1,-1],[x,y,INT01],0);
int_lin_eq([1,-1,-1],[y,x,INT02],0);
int_times(INT01,INT02,INT03);
int_le(INT02,INT03);
```

II Linear Expression

对于 int: k=4;

constraint $x+z*(y-x)+z \leq k*z$

并不是 naively flat，而是要简化线性表达式：

$(-1)*x+2*y+(-3)*z<=0$

flatZink:

```
constraint int_lin_le([-1,2,-3],[x,y,z],0);
```

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

Ⅲ Aarray traslation

(i) 转换成一维 (start from index 1)

eg. array[l1..u1, l2..u2] of int: x;

↓ FlatZinc

array[1..(u1-l1+1)*(u2-l2+1)] of int: x;

x[(i-l1)*(u2-l2+1)+(j-l2+1)]

2D traslate to 1D: (start from index 1)

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

eg.

Example 2D array

```
array[0..2,0..2] of var 0..2: x;
constraint sum(i in 0..2)(x[i,i]) <= 1;
constraint x[x[1,1],1] = 2;
```

Flattening

```
array[0..2,0..2] of var 0..2: x;
constraint x[0,0] + x[1,1] + x[2,2] <= 1;
var int: INT01 = x[1,1];
constraint x[INT01,1] = 2;
```

Converting to 1D

```
array[1..9] of var 0..2: x;
constraint x[1] + x[5] + x[9] <= 1;
var int: INT01 = x[5];
var int: INT02 = INT01 * 3 + (1 + 1);
constraint x[INT02] = 2;
```

(ii) element constraint

array_int_element(index, array, result)

eg. constraint x[INT02]=2;

Becomes:

constraint array_int_element(INT02, x, 2)

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

IV if-then-endif

- Flattening `if b then t else e endif`
  - evaluate *b* (assuming it is fixed)
  - if true then replace with *t*
  - else replace with *e*
- When *b* is not fixed
  - replace with [e,t][bool2int(b)+1] and flatten

```
constraint if b then x else y endif >= 0;
```

- becomes

```
constraint [y,x][bool2int(b)+1] >= 0;
```

- becomes

```
constraint bool2int(b,INT00);
constraint int_plus(INT00,1,INT01);
constraint array_int_element(INT01,[y,x],INT02);
constraint INT02 >= 0;
```

V. Flattening Boolean Expressions

(1) Consider the express:

$x > 0 \rightarrow bool2int(y > 0 \wedge z > 0) + t >= u;$

then flattening is analogous to other expressions

```
constraint BOOL01 <-> x > 0;
constraint BOOL02 <-> y > 0;
constraint BOOL03 <-> z > 0;
constraint BOOL04 <-> BOOL02 /\ BOOL03
constraint INT01 = bool2int(BOOL04)
constraint BOOL05 <-> INT01 + t >= u;
constraint BOOL01 -> BOOL05
```

Flatzinc:

```
constraint int_le_reif(1,x,BOOL01);
constraint int_le_reif(1,y,BOOL02);
constraint int_le_reif(1,z,BOOL03);
constraint
array_bool_and([BOOL02,BOOL30],BOOL04);
constraint bool2int(BOOL04,INT01);
constraint int_lin_le_reif([-1,-1,1],
          [INT01,t,u],0,BOOL05);
constraint bool_le(BOOL01,BOOL05);
```

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

(ii) 避免否定

- 否定将否定推向最后层

VI.

当我们看到包含变量in表达式时，我们用参数展开它，then flatten

```
predicate far_or_equal(var int:x1, var int:y1,
                       var int:x2, var int:y2)=
          man_dist(x1,y1,x2,y2) >= 4 \/
         (x1 = x2 /\ y1 = y2);

function var int: man_dist(var int:u1,
         var int:v1, var int:u2, var int:v2) =
         abs(u1 - u2) + abs(v1 - v2);

constraint far_or_equal(a,b,c,d);
```

↓

constraint man_dist(a,b,c,d) >= 4 \/ (a=c /\ b=d);

↓

abs(a-c) + abs(b-d) >= 4 \/ (a=c /\ b=d);

↓ flatten

```
constraint INT01 = a - c;
constraint INT02 = abs(INT01);
constraint INT03 = b - d;
constraint INT04 = abs(INT03);
constraint BOOL01 <-> INT02 + INT04 >= 4
constraint BOOL02 <-> a = c;
constraint BOOL03 <-> b = d;
constraint BOOL04 <-> BOOL02 /\ BOOL03
constraint BOOL01 \/ BOOL04;
```

8

Content
内容

此章节主要讲了

1: Basic scheduling

每个任务有持续时间

任务有先后顺序，一个任务结束必须在另一个任务开始之前

目标为最小化做完所有任务的结束时间

2: disjunction scheduling

在basic scheduling的基础上涉及了overlap

3: cumulative scheduling

资源有限的scheduling

4: sequence dependent scheduling1

single channel problem

schedule取决于在该资源上哪些任务优先于另一任务

下一任务的开始时间会因上一个任务而延迟

5: sequence dependent scheduling1

double channel problem

Review
复习记录

/ / / / /

Basic Scheduling

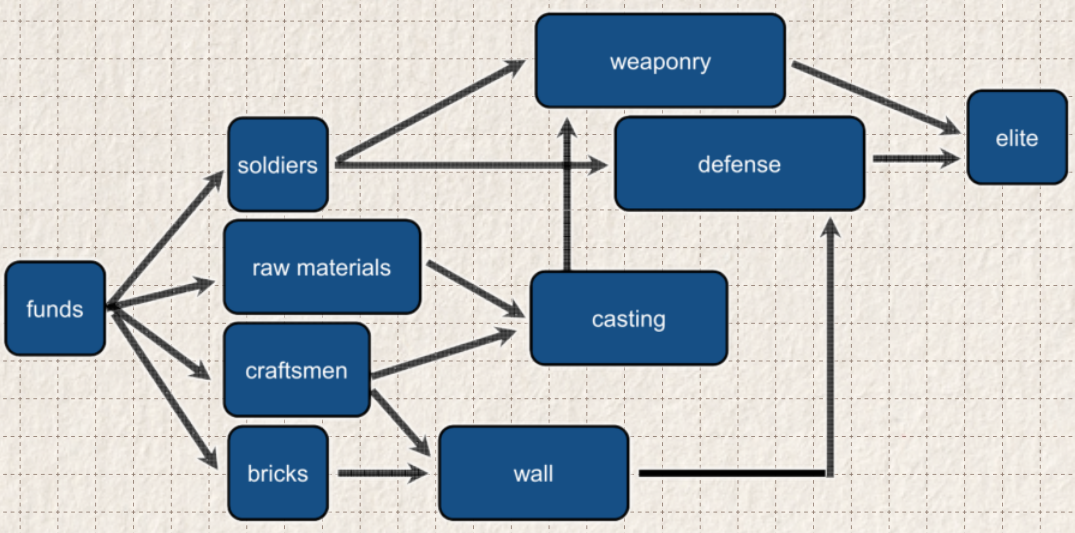
——— Schedule the tasks to minimize the end time

题目描述：现有三个任务：Building a wall, producing more weapons, and selecting elite sodiers.

10$ → 15$ hire craftsman → 20$ building the wall
10$ → 10$ make bricks →

10$ → 20$ purchase materials → 20$ produce weapon
10$ → 15$ hire craftsman →

10$ → 10$ recruit sodiers → $25 join weaponry → 10$ identify elite soldiers
$25 defense training
20$ produce weapon
$20 building wall



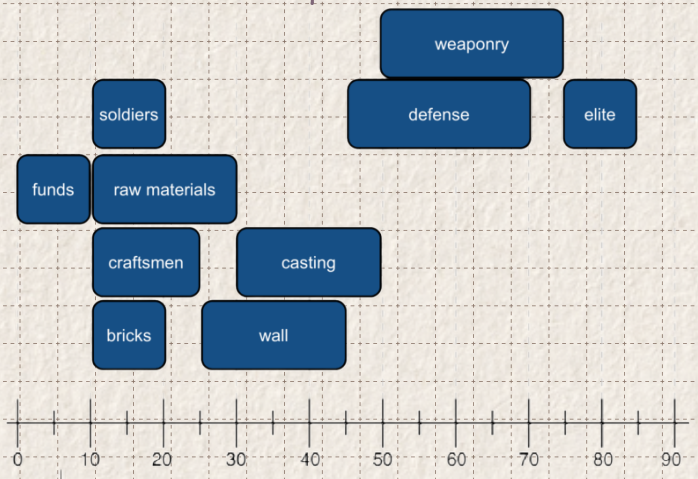
```
enum TASK;

array[TASK] of int: duration;
int: p; % number of precedences
set of int: PREC = 1..p;
array[PREC,1..2] of TASK: pre;

int: t = sum(duration);
array[TASK] of var 0..t: start;
```

```
% constraints
constraints forall(i in PREC)
        (start[pre[i,1]] + duration[pre[i,1]]
            <= start[pre[i,2]]);
% objective
var 0..t: makespan = max(t in Task)
                (start[t] + duration[t]);
solve minimize makespan;
```

$x + d \leq y$
(d is constant)



Scheduling

What is the given information when solving a basic scheduling problem?

- ○ (a) order of tasks, and (b) precedences between tasks.
- ◉ (a) duration times of tasks, and (b) precedences between tasks.
- ○ (a) latest end time of tasks, and (b) duration of precedences.
- ○ (a) duration times of tasks, and (b) start times of tasks.

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Disjunctive Scheduling

I Nonoverlap

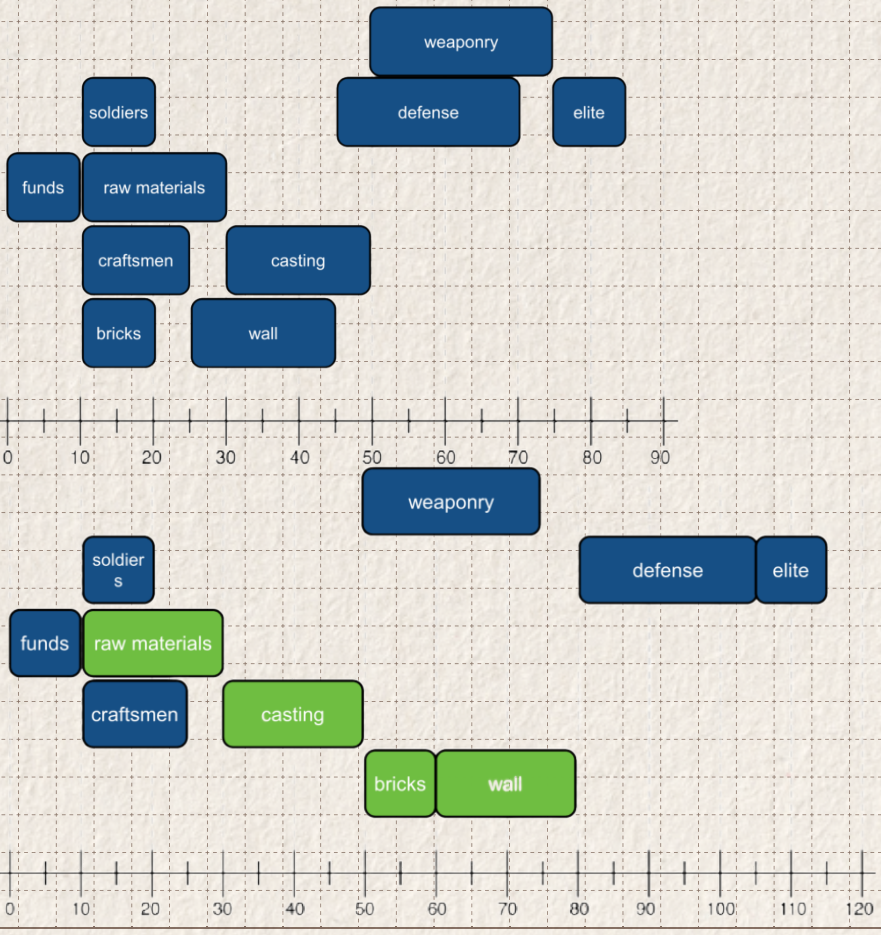
题目描述：刘备负责四个任务不可在时间上重叠，

None of the tasks can overlap in time.

```
predicate nonoverlap(var int:s1, var int:d1
                     var int:s2, var int:d2)
    s1 + d1 <= s2 \/ s2 + d2 <= s1

set of TASK: LIU = {CASTING, RAW_MATERIALS, BRICKS, WALL};

forall(t1, t2 in LIU where t1 < t2)
    (nonoverlap(start[t1], duration[t1],
                start[t2], duration[t2]));
```



Keywords 关键词 | Notes 笔记

题目描述：相同兵，张飞，关羽负责训练也不可以在时间上重叠。

set of TASK: LIU = {CASTING, RAW_MATERIALS, BRICKS, WALL};

Set of TASK: Zhang = {SODIERS, RAW_MATERIALS, BRICKS, CRAFTSMAN};

set of TASK: GUAN = {WEAPONRY, DEFENSE, ELITEARMY, WALL};

More nonoverlaps:

```
predicate nonoverlap(var int:s1, var int:d1
                     var int:s2, var int:d2)
    s1+d1 <= s2 \/ s2+d2 <= s1
predicate exclusive(set of TASK: tasks) =
    forall(t1,t2 in tasks where t1<t2)
        (nonoverlap(start[t1], duration[t1],
                    start[t2], duration[t2]));
constraint exclusive(LIU);
constraint exclusive(ZHANG);
constraint exclusive(GUAN);
```

不足之处：nonoverlap 只保证3两两之间不重叠。

我们可以用 disconjuctive

disconjuctive(<start time array>, <duration time array>)

```
predicate disjunctive(array[int] of var int:s,
                      array[int] of var int:d) =
   forall(i1,i2 in index_set(s) where i1 < i2)
        (nonoverlap(s[i1],d[i1],s[i2],d[i2]));
```

```
include "disjunctive.mzn";
predicate exclusive(set of TASK: tasks) =
   let {array[int] of var int:
      ss = [start[t] | t in tasks];
        array[int] of int:
      dd = [duration[t] | t in tasks];}
   in disjunctive(ss,dd);
```

Content Index

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Cumulative scheduling

资源受限项目调度

- 有一系列任务 $t \in TASK$
- 任务有先后顺序 $p \in PREC$

  pre[p,1] precedes pre[p,2]
- 有一些资源
- 每个任务t在执行期间需要 res[r,t] 资源
- 每个资源有限 L[r]
- 寻找最短时间完成任务

```
include "globals.mzn";

enum TASK;

array[TASK] of int: duration;
int: p; % number of precedences
set of int: PREC = 1..p;
array[PREC,1..2] of TASK: pre;

int: t = sum(duration);
array[TASK] of var 0..t: start;

enum RESOURCE;
array[RESOURCE] of int: L; % resource limit
array[RESOURCE,TASK] of int: res;
```

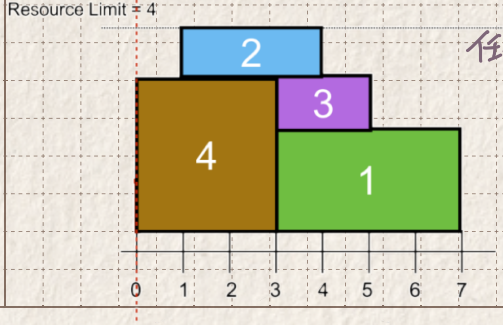
I Time Decomposition 时间分解

forall(i in time)(Sum(t in TASK)

((Start[t]<=i /\ start[t]+duration[t]>i)

*res[t])<=L);

任务t是否在i时刻运行，是布尔值



检查点：0,1,2,3,4,5,6,7

Keywords 关键词 | Notes 笔记

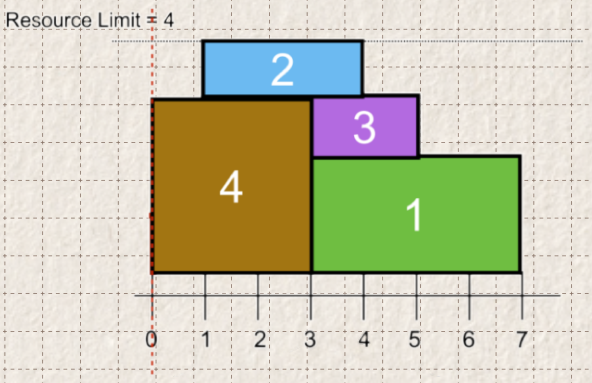
缺点：每次检查每个时间点 → large

修改：only check start times

```
forall (t2 in TASk) ( Sum( t in TASk)
    (( Start[t] <= start[t2] /\ Start[t] + duration[t]
    > start[t2] ) * res[t] <= L);
```

或者这种更清楚地表达：

```
forall (t2 in TASk) ( Sum( t in TASk where t != t2)
    (( Start[t] <= start[t2] /\ Start[t] + duration[t]
    > start[t2] ) * res[t] <= L);
```



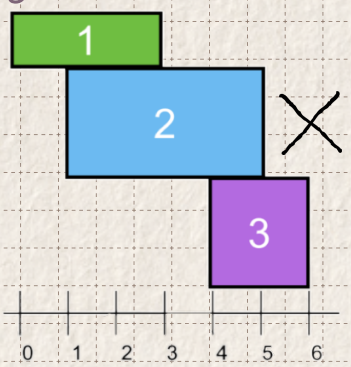
检查点：0,1,3

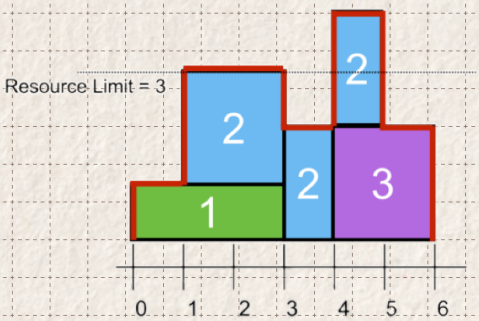
II cumulative global constrains

cumulative (<start time array>, <duration time array>,
<resource usage array>, <limit>)

确保在执行任务期间的任何时候使用的资源不超过限制

eg. cumulative ([0,1,4],[3,4,2],[1,2,2],3)





Content Index
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

start time possibilities are given as ranges

- cumulative([0..3,0..3,2..3,0..4],[4,3,2,3],[2,1,2,3],4)

■ Decisions

```
array[TASK] of var TIME: start;
```

■ Constraints

```
forall(p in PREC)
      (start[pre[i,1]] + duration[pre[i,1]]
         <= start[pre[i,2]]);
forall(r in RESOURCE)
      (cumulative(start, duration,
         [res[r,t] | t in TASK], L[r]));
```

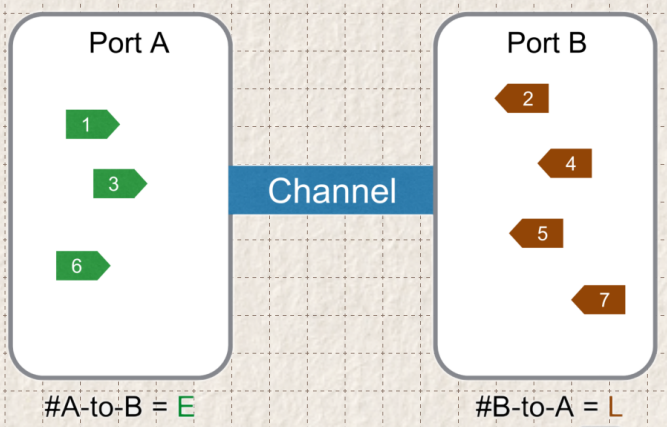
■ Objective

```
solve minimize max(t in TASK)
   (start[t] + duration[t]);
```

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Sequence Dependent Scheduling I

问题描述：两城之间有若干船，须将这些船全部运送武器至对方城。船有一定行驶速度。河道只允许一个方向的船通过，因为只有一条河道。两船之间m安全距离为2,000m。船发出时间须晚于开船时间，不可m在装货前离开。河道总长32000m。



Data

```
int: len;

int: nS; % number of ships
set of int: SHIP = 1..nS;
array[SHIP] of int: speed;    % 1000m time
array[SHIP] of int: desired; % desired time
int: atob = 1; int: btoa = 2;
array[SHIP] of atob..btoa: dirn;

int: leeway;  % leeway between 2 ships
int: maxt;         % maximum time
set of int: TIME = 0..maxt;
```

Keywords 关键词 | Notes 笔记

I Decision

- 在河道上添加一个dummy ship作为最后一个船
- 确保每个船运行都有下一个船

```
set of int: SHIPE = 1..nS+1; % add dummy
int: dummy = 3;
array[SHIPE] of atob..dummy: kind
           = dirn ++ [ dummy ];
array[SHIPE] of int: speede = speed++[0];
array[SHIPE] of var TIME: start;
array[SHIPE] of var TIME: end;

array[SHIP] of var SHIPE: next; % next ship
```

- dummy ship is the last

```
start[nS+1]=maxt;
end[nS+1]=maxt;
```

- relationship between start and end

```
forall(s in SHIP)
    (end[s] = start[s]+ len * speed[s]);
```

- the next ships are alldifferent

```
alldifferent(next);
```

II single channel

- 开始和结束时间受到约束

```
forall(s in SHIP)
    % 相反方向的船
    (if kind[s] + kind[next[s]]=3 then
        end[s] <= start[next[s]]
    else  % 相同方向的船
        start[s] + speed[s] * leeway <= start[next[s]] \/
        end[s] + speede[next[s]] * leeway <= end[next[s]]
    endif);
```

Content Index

Keywords 关键词 | Notes 笔记

- 不能在发船之前开船

```
forall (s in SHIP) (start[s] >= desired[s]);
```

- objective

```
solve minimize max (s in SHIP) (end[s]);
```

注意：需要考虑有补情况！

Scheduling

The *makespan* is the time it takes for all tasks to finish. Assuming that we start at time 0, which of the following will calculate the makespan? Given the following:

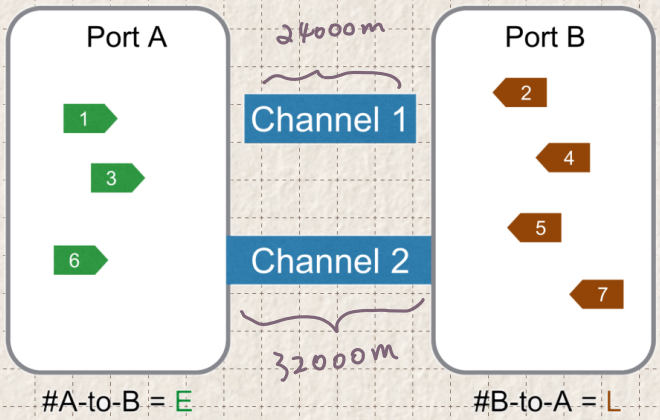
- `TASK` is the set of all tasks,
- `dur[t]` is the duration of a task `t`
- `start[t]` is the time a task `t` starts
- `end[t]` is the time a task `t` ends

- ○ `forall (t in TASK) (start[t] + dur[t]);`
- ◙ `max (t in TASK) (end[t]);`
- ○ `max (t in TASK) (start[t]) + max (t in TASK) (dur[t]);`
- ○ `sum (t in TASK) (end[t]);`

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Sequence Dependent Scheduling 2

问题描述河道拓宽为2条，其他条件不变。



Data

```
int: nC; % number of channels
array[1..nC] of int: len;

int: nS; % number of ships
set of int: SHIP = 1..nS;
array[SHIP] of int: speed;    % 1000m time
array[SHIP] of int: desired; % desired time
int: atob = 1; int: btoa = 2;
array[SHIP] of atob..btoa: dirn;

int: leeway;  % leeway between 2 ships
int: maxt;          % maximum time
set of int: TIME = 0..maxt;
```

1 Decisions

·添加n个dummy ships作为最后一个船

Keywords 关键词 | Notes 笔记

```
Set of int: SHIPE = 1..ns+nc;  % add dummies
int: dummy = 3;
array[SHIPE] of atob.. dummy: kind
                = dim ++ [dummy | i in 1..nc];
array [SHIPE] of int : speede
                = Speed ++ [0 | i in 1..nc]
array [SHIPE] of var TIME: start;
array [SHIPE] of var TIME: end;
array [SHIPE] of var 1..nC: channel;
array [SHIP] of var SHIPE: next  % next ship
```

Ⅱ Constraint

- dummy ship 的开始与结束时间均为maxt

```
forall (s in ns+1 .. ns+nc)
        (start[s] = maxt ∧ end[s] = maxt);
forall (s in ns+1 .. ns+nc)
        (channel(s) = s - ns);   % 确保每个dummy在不同channel
```

- Relationship between start and end

```
forall (s in SHAP) (end[s] = start[s] +
                        len[channel[s]] * speed[s]);
```

- The next ships are all different

```
alldifferent (next);
```

- The relationship between a ship and its next ship

```
forall (s in SHIP)
    % 相反方向a船
    (if kind[s] + kind[next[s]] = 3 then
        end[s] <= start[next[s]]
    else  % 相同方向a船
        start[s] + speed[s] * leeway <= start[next[s]] ∧
        end[s] + speede[next[s]] * leeway <= end[next[s]]
    endif);
```

Content Index

Keywords 关键词	Notes 笔记
	• next ship 是上一个 ship 是在相同 in channel. forall (s in SHIP) (channel[next[s]] = channel[s];) • Cannot leave before desired time forall (s in SHIP) (start[s] >= desired[S]); • Objective Solve minize max(s in SHIP)(end[s]);

ELEYANG DESIGN

9

Content
内容

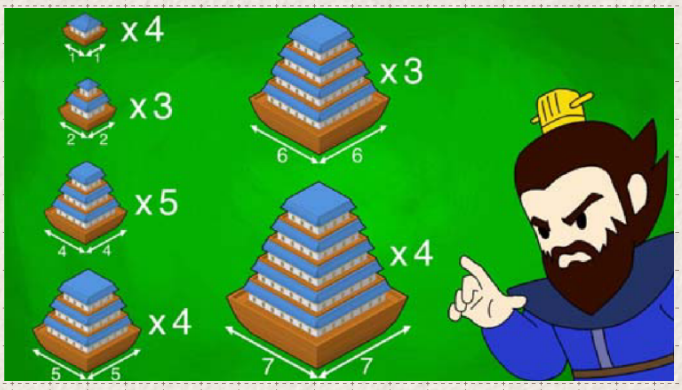
Review
复习记录

/ / / / /

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Square packing

问题描述，将正方形打包成最小的矩形

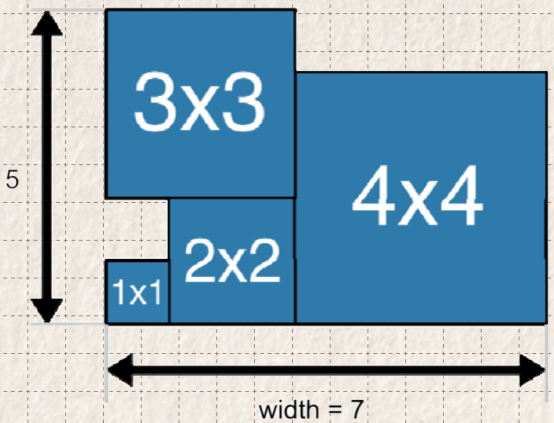


I Data

```
int: n; % number of square sizes
set of int: SQUARE = 1..n;
array[SQUARE] of int: ncopy;
int: maxl = sum(i in SQUARE)(i*ncopy[i]);%最大的长度就是所有正方形长度和
int: mina = sum(i in SQUARE)(i*i*ncopy[i]); %最小的面积就是把所有的面积加起来

var n..maxl: height; var n..maxl: width;
var mina .. n*maxl: area = height * width;
int: nsq = sum(i in SQUARE)(ncopy[i]); %总共square的数量
set of int: NSQ = 1..nsq;
array[NSQ] of var 0..maxl: x;
array[NSQ] of var 0..maxl: y;
```

- Note the tight bounds on variables



area = 5x7 = 35

Keywords 关键词 | Notes 笔记

II 添加辅助变量

```
array[NSQ] of var SQUARE: size;
%计算每个方块的型号size
include "global_cardinality_mzn";
global_cardinality(size, [i | i in SQUARE], ncopy);
%确保数组中每个方块都有正确数量的副本
forall(i in 1..nsq-1)
    (size[i] <= size[i+1]);
```

例如

```
ncopy = [3,2,5,4,3]
size = [1,1,1, 2,2, 3,3,3,3,3, 4,4,4,4, 5,5,5]
```

III Constraint & Objective

```
forall(s in NSQ)(x[s] + size[s] <= width);
forall(s in NSQ)(y[s] + size[s] <= height);
```

方块不重叠

```
forall(s1, s2 in NSQ where s1<s2)
( x[s1]+ size[s1] <= x[s2] \/
  y[s1]+ size[s1] <= y[s2] \/
  x[s2]+ size[s2] <= x[s1] \/
  y[s2]+ size[y2] <= y[s1] );
```

Objective

```
Solve minimize area;
```

缺点：效率太低，运行时间太久

提高效率的办法

- Global constraints
- Redundant constraints
- Symmetry breaking

IV Global constraints

(i) 使用diffn

- The diffn global constraint captures exactly 2d non overlap (it should be called diff2)
  - diffn([x_1, …, x_n], [y_1, …, y_n],
  - [dx_1, …, dx_n], [dy_1, …, dy_n])
    - –ensure no two objects at positions (x_i,y_i) with dimensions (dx_i,dy_i) overlap

```
predicate diffn(array[int] of var int: x,
                array[int] of var int: y,
                array[int] of var int: dx,
                array[int] of var int: dy);
```

- Squares do not overlap (multisqpackimp.mzn)

```
include "diffn.mzn";
diffn(x, y, size, size);
```

↑替换

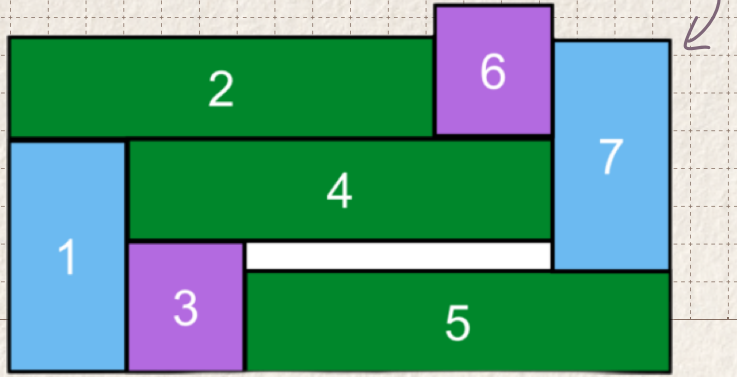
```
forall (S1, S2 in NSQ where S1 < S2)
( x[S1] + size[S1] <= x[S2] \/
  y[S1] + size[S1] <= y[S2] \/
  x[S2] + size[S2] <= x[S1] \/
  y[S2] + size[y2] <= y[S1] );
```

(ii) 使用cumulative.

- If there is a packing
  - then the cumulative constraint must hold!
- We can add **redundant** cumulative constraints to packing problems
  - improves propagation (and hence solving)
- Squares do not overlap in the x and y dimension respectively (multisqpackimp.mzn)

```
cumulative(x, size, size, height);
cumulative(y, size, size, width);
```

但是cummulative不会强制Packing



Keywords 关键词 | Notes 笔记

V. Symmetries

lex_greater([x_1,...,x_n],[y_1,...,y_n])

确保 (x_1,...x_n) > (y_1,...y_n)

将方块的顺序进行排序

严格排序：(x_1,y_1) > (x_2,y_2)
- $x_1 > x_2$; or
- $x_1 = x_2$, $y_1 > y_2$

找到每个size的起始索引（左下角）

```
array[SQUARE] of int: base =
    [ if i=1
      then 0
      else sum(j in 1..i-1)(ncopy[j])
      endif
    | i in SQUARE ];
```

排到相同大小的正方形

```
include "lex_greater.mzn"
constraint forall( i in SQUARE)
    (forall( j in 1..ncopy[i]-1)(
        lex_greater([x[base[i]+j], y[base[i]+j]],
        [x[base[i]+j+1], y[base[i]+j+1]] ));
```

- The array `size` is a variable solved by constraints

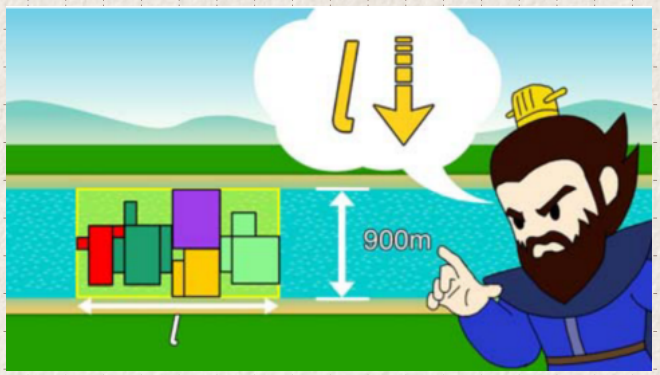
```
array[NSQ] of var SQUARE: size;
include "global_cardinality.mzn";
global_cardinality(size,
   [i | i in SQUARE], ncopy);
forall(i in 1..nsq-1)
   (size[i] <= size[i+1]);
```

- Can do without constraint solving

```
array[NSQ] of SQUARE: size;
size = [max(j in SQUARE)(j*(i > base[j]))
   | i in NSQ];
```

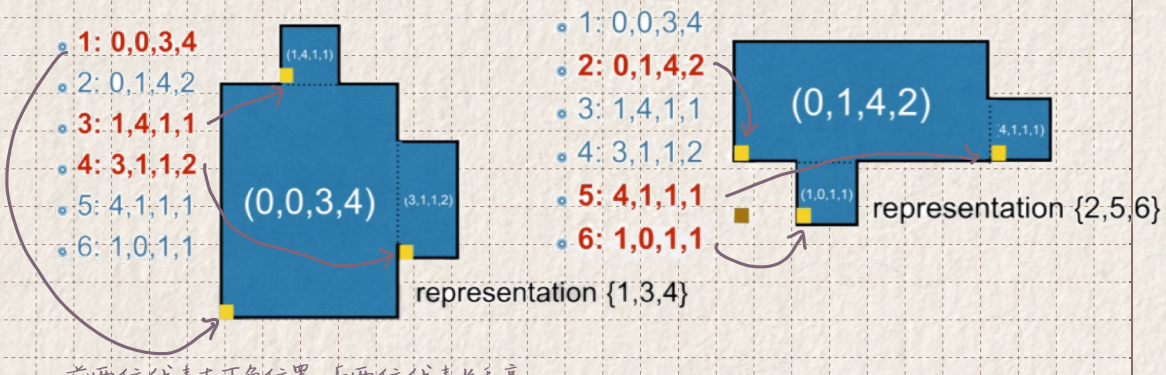
Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Rectilinear Packing without Rotation



1. 表示块状图形

用每个方块左下角偏移量表示



前两位代表左下角位置，后两位代表长和高

```
int: n; % number of blocks
set of int: BLOCK = 1..n;
int: m; % number of rectangles/offsets
set of int: ROFF = 1..m;
array[ROFF,1..4] of int: d; % defns
array[BLOCK] of set of ROFF: shape;
int: h; % width of river
int: maxl; % maximum length of river
```

Keywords 关键词 | Notes 笔记

```
n = 5; m = 12; h = 9; maxl = 16;
d = [| 1,0,2,5 % (xoffset,yoffset,xsize,ysize)
     | 3,4,1,1
     | 0,3,1,1
     | 1,4,2,2
     | 0,1,1,3 % shared by blocks 2 & 3
     | 1,0,4,4
     | 1,5,1,2
     | 1,0,3,5
     | 4,1,1,4
     | 0,0,1,3
     | 1,0,3,4
     | 0,0,4,5 |];
shape = [{1,2,3}, {4,5,6}, {5,7,8,9}, {12},
         {10,11}];
```

II Packing decision + objective

```
array[BLOCK] of var 0..maxl: x;
array[BLOCK] of var 0..h: y;

var 0..maxl: l; %便利河流宽度
solve minimize l;
```

```
%每个方块都要fits the river
forall (i in BLOCK)( forall r in ROFF)
    (r in shape[i] ->
                ( x[i] + d[r,1] + d[r,3] <=1 /\
                  y[i] + d[r,2] + d[r,4] <= h)
    ));
```

小矩形相对于BLOCK的坐标

```
%nonoverlap部分
constraint forall(i,j in BLOCK where i < j)
  (forall(r1,r2 in ROFF)
  (r1 in shape[i] /\
   r2 in shape[j] ->
(x[i] + d[r1,1] + d[r1,3] <= x[j] + d[r2,1]
                  \/
 x[j] + d[r2,1] + d[r2,3] <= x[i] + d[r1,1]
                  \/
 y[i] + d[r1,2] + d[r1,4] <= y[j] + d[r2,2]
                  \/
 y[j] + d[r2,2] + d[r2,4] <= y[i] + d[r1,2])
  ));
```

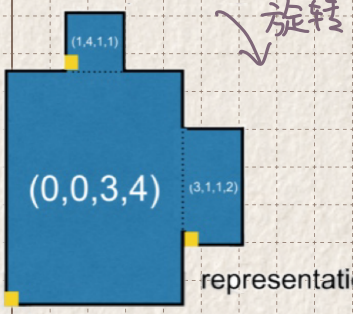
右边界 左边界 上边界 下边界

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Rectilinear Packing with Rotation

工 表示方块形状

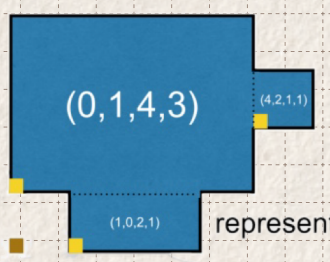
- 1: 0,0,3,4
- 2: 0,1,4,2
- 3: 1,4,1,1
- 4: 3,1,1,2
- 5: 4,1,1,1
- 6: 1,0,1,1



旋转

representation {1,3,4}

- 1: 0,0,3,4
- 2: 0,1,4,3
- 3: 1,4,1,1
- 4: 3,1,1,2
- 5: 4,2,1,1
- 6: 1,0,2,1



representation {2,5,6}

```
int: n; % number of blocks
set of int: BLOCK = 1..n;
int: m; % number of rectangles/offsets
set of int: ROFF = 1..m;
array[ROFF,1..4] of int: d; % defns
set of int: ROT = 1..4;  for each rotation
array[BLOCK,ROT] of set of ROFF: shape;
int: h; % width of river
int: maxl; % maximum length of river
```

Keywords 关键词

Notes

```
n = 5; m = 45; h = 9; maxl = 16;
% (xoffset,yoffset,xsize,ysize)
d = [| 1,0,1,1 | 0,1,5,2 | 0,3,1,1 | 1,0,2,5
     | 3,4,1,1 | 3,1,1,1 | 0,0,1,1 | 3,3,1,1
     | 4,0,1,1 | 0,0,4,4 | 1,4,3,1 | 4,2,2,2
     | 0,2,4,4 | 4,2,1,3 | 2,0,2,2 | 1,0,4,4
     | 0,1,1,3 | 1,4,2,2 | 2,1,4,4 | 2,0,3,1
     | 0,1,2,2 | 1,5,1,2 | 1,0,3,5 | 4,1,1,4
     | 5,3,2,1 | 0,1,5,3 | 1,0,4,1 | 3,0,3,1
     | 0,1,2,1 | 2,1,5,3 | 2,4,4,1 | 4,3,1,3
     | 3,0,1,2 | 1,2,3,5 | 0,2,1,4 | 0,0,1,3
     | 1,0,3,4 | 0,3,3,1 | 0,0,4,3 | 1,0,3,1
     | 0,1,4,3 | 3,1,1,3 | 0,0,3,4 | 0,0,5,4
     | 0,0,4,5 |];
% Each shape is a list of 4 sets (rotations)
shape = [|
    {1,2,3}, {3,4,5}, {6,4,7}, {8,2,9}
  | {10,11,12}, {13,14,15}, {16,17,18},
    {19,20,21}
  | {17,22,23,24}, {11,25,26,27}, {28,29,30,31},
    {32,33,34,35}
  | {44}, {45}, {}, {}
  | {36,37}, {38,39}, {40,41}, {42,43}
|];
```

II Decision + Objective

```
array[BLOCK] of var 0..maxl: x;
array[BLOCK] of var 0..h: y;
array[BLOCK] of var ROT: rot;
var 0..maxl: l; % length of river used
solve minimize l;
```

?没明白这句是什么意思

```
constraint forall(i in BLOCK)(kind[i] in shapeind[i]);
```

BLOCK中的每一个矩形 fits in the river

```
constraint forall(i in BLOCK)(forall(r in ROFF)
  (r in shape[i,rot[i]] ->
   (x[i] + d[r,1] + d[r,3] <= l /\
    y[i] + d[r,2] + d[r,4] <= h)));
```

none overlap →

```
constraint forall(i,j in BLOCK where i < j)
  (forall(r1,r2 in ROFF)
  (r1 in shape[i,rot[i]] /\
   r2 in shape[j,rot[j]] ->
  (x[i] + d[r1,1] + d[r1,3] <= x[j] + d[r2,1]
                    \/
   x[j] + d[r2,1] + d[r2,3] <= x[i] + d[r1,1]
                    \/
   y[i] + d[r1,2] + d[r1,4] <= y[j] + d[r2,2]
                    \/
   y[j] + d[r2,2] + d[r2,4] <= y[i] + d[r1,2]
  )));
```

Content Index

III Globals diffn_K

diffn_K强制约束对象不重叠，同时考虑旋转

→ geost global constraints

```
predicate geost_bb(int: k,                    → 2维就是2
                   array[int,int] of int: rect_size,
                   array[int,int] of int: rect_offset,
                   array[int] of set of int: shape,
                   array[int,int] of var int: x,
                   array[int] of var int: kind,
                   array[int] of var int: l,
                   array[int] of var int: u)
```

Arguments

- k = number of dimensions
- rectangle sizes: row = rectangle, col = dimension
- rectangle offsets: row = rect, col = dim
- shape definitions (sets of rectangle/offsets)
- position of each object
- kind (shape) of each object
- lower and upper bounds on each dimension

geost Data

```
int: n; % number of blocks
set of int: BLOCK = 1..n;
int: m; % number of rectangle/offsets
set of int: ROFF = 1..m;
array[ROFF,1..4] of int: d; % defns
array[int] of set of ROFF: shape;
int: h; % width of river
int: maxl; % maximum length of river

array[BLOCK] of var 0..maxl: x;
array[BLOCK] of var 0..h: y;
var 0..maxl: l; % length of river used

solve minimize l;
```

转成一维

```
% All shape+orientations are in a 1-d array
shape = [
  {1,2,3}, {3,4,5}, {6,4,7}, {8,2,9},
  {10,11,12}, {13,14,15}, {16,17,18}, {19,20,21},
  {17,22,23,24}, {11,25,26,27}, {28,29,30,31},
      {32,33,34,35},
  {44}, {45},
  {36,37}, {38,39}, {40,41}, {42,43}
];

% Define the shapes and the associated rotations
shapeind = [ {1,2,3,4}, {5,6,7,8}, {9,10,11,12},
  {13,14}, {15,16,17,18} ];
```

Keywords 关键词

Notes 笔记

```
% extract the offsets and sizes
array[ROFF,1..2] of int: rsize =
   array2d(ROFF, 1..2,
      [d[i,j] | i in ROFF, j in 3..4]);
array[ROFF,1..2] of int: roff =
   array2d(ROFF, 1..2,
      [d[i,j] | i in ROFF, j in 1..2]);

% pack the x and y coordinates
array[BLOCK,1..2] of var int: coord;
constraint forall(i in BLOCK)
   (coord[i,1] = x[i] /\ coord[i,2] = y[i]);

% set up the "kind" constraints
array[BLOCK] of var int: kind;
array[BLOCK] of int: nemptycount =
   [ sum(j in 1..4*i)(shapeorient[j] != {}) | i in BLOCK];
array[BLOCK] of set of int: shapeind =
   [ if i = 1 then { j | j in 1..nemptycount[i]}
        else {j | j in nemptycount[i-1]+1..nemptycount[i]}
     endif | i in BLOCK ];
constraint forall(i in BLOCK)(kind[i] in shapeind[i]);

include "geost.mzn";
constraint geost_bb(2,
        rsize,
        roff,
        neshapeorient,
        coord,
        kind,
        [ 0,0 ],
        [ L,h ]);
```

Global

- diffn_k(for k dimensional packing)
- geost(for flexible k dimensional packing)

Summary 总结

Content Index

ELEYANG DESIGN

10

Content
内容

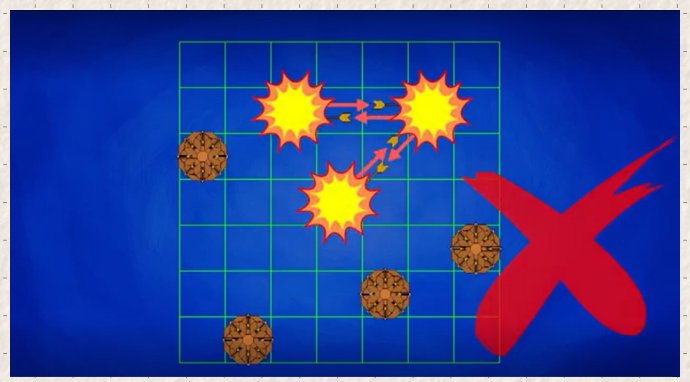
Review
复习记录

/ / / / /

Keywords 关键词 | Notes 笔记 | Review 复习记录 / / / / /

Symmetries and LexLeader

问题描述诸葛亮布置横梁，每个横梁有8个发射方向。一个陷井是由多个簇列排在一个方形网络中组成的。如果横梁处于对面位置，那么它们有可能相互攻击和破坏。



8 directions:
N, S, E, W, NE, SE, NW, SW

I Decisions

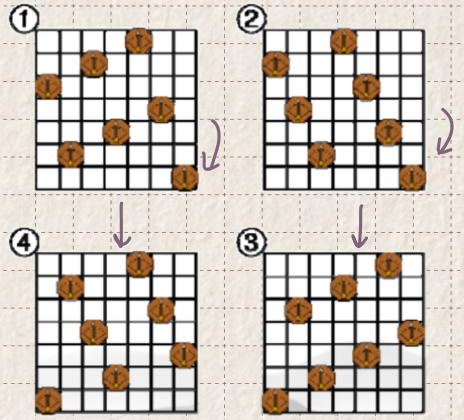
```
int: n;
set of int: N = 1..n;
array[N,N] of var bool: t;
constraint sum(i,j in N)(t[i,j]) = n;
solve satisfy;
```

Keywords 关键词 | Notes 笔记

Ⅱ constraints

```
% no two traps on the same row
constraint forall(i in N)(sum(j in N)(t[i,j]) <= 1);
% no two traps on the same column
constraint forall(j in N)(sum(i in N)(t[i,j]) <= 1);
% no two traps on same diagonal
constraint forall(k in 1-n..n-1)
      (sum(i,j in N where i-j=k)(t[i,j])<= 1);
constraint forall(k in 2..2*n)
      (sum(i,j in N where i+j=k)(t[i,j])<= 1);
```

n=7时 Solution:

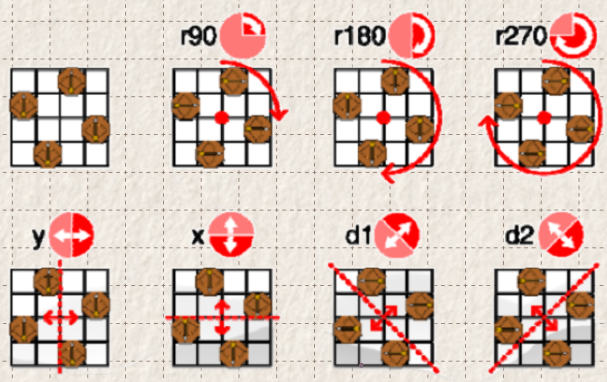


由①可以推断④

由②可以推断③

如果solution A 可以通过旋转或翻转Solution B获得，

那么Solution A is symmetric geometrically (几何对称) to B



Ⅲ variable Symmetries → is a bijection

$[x_1=v_1, \ldots, x_n=v_n]$ is a solution

⇓ bijection 映射

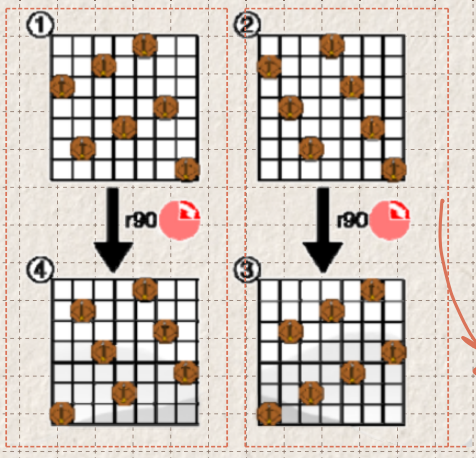
$[x_{g(1)}=v_1, \ldots, x_{g(n)}=v_n]$ is a solution

所以，题中的geometric symmetries 是 variable Symmetries

bijection 就是旋转、翻转

Content Index

Keywords 关键词 | Notes 笔记

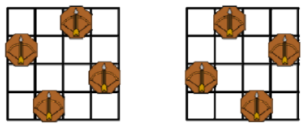


All the solutions that can be obtained from one another by symmetry transformations form a symmetry class

Symmetry class

- Zhuge Liang wants only one solution in each symmetry class
- Choose the lexicographically least solution (字典最小)
- Example: using 0 = false, 1 = true

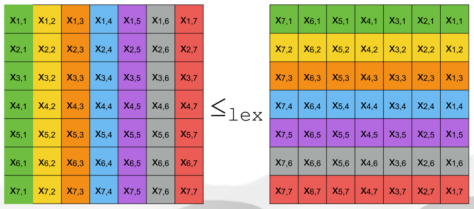
0010 1000 0001 0100 $<_{lex}$ 0100 0001 1000 0010



- The left solution is lexicographically smaller

IV LexLeader Symmetry Breaking (字典最小)

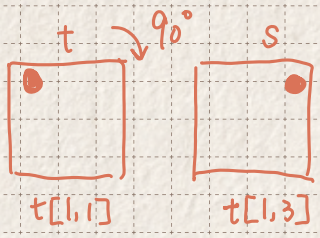
- Add constraints to the model to require that
  - the solution is lexicographically smaller than all its symmetric versions: $[x_1, \ldots, x_n] \leq_{lex} [x_{g(1)}, \ldots, x_{g(n)}]$, for **each** symmetry g
- For example, g is r90



约束：

```
% solution lex less than r90 version
constraint let { array[N,N] of var bool: s; } in
    forall(i,j in N)(s[i,j] = t[j,n+1-i]) /\
    lex_lesseq(array1d(t), array1d(s));
```

变量数组索引对称计算



↓

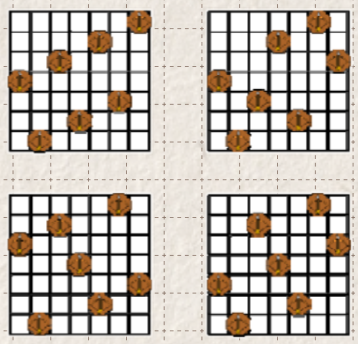
order them lexicographically

use Global lex_lesseq for lexicographically order

Content Index
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

Keywords 关键词 | Notes 笔记

- r90, r180, r270, x, y, d1, d2



7种 Symmetries class

① var_sqr_sym(t);
② card(x);

minizinc有一个公共的对称库，可以直接使用

var_sqr_sym(array[int,int] of var int: x) 约束了对于旋转对称的字典最小

include "var_sqr_sym.mzn";
① var_sqr_sym(t);

eg

- Consider

```
array[1..3] of var 1..10: x;
forall(i,j in 1..u where i < j
   (x[i] != x[j]);
```

for modeling a fixed cardinality set again 同一集合的多表示

- Issue: multiple representations of the same set (variable interchangeability)
  - e.g. {1,6,10}: [1,6,10], [10,1,6], [10,6,1], [1,10,6], [6,10,1], [6,1,10]
- Solution: ensure ordered

```
forall(i in 1..u-1)(x[i] < x[i+1]);
```

- Another solution: use a set variable! 但是解对称问题的解或它的一部分
- Reformulate the problem by a set variable

```
var set of 1..10: x;
```

with cardinality constraint

② `card(x) = 3;`

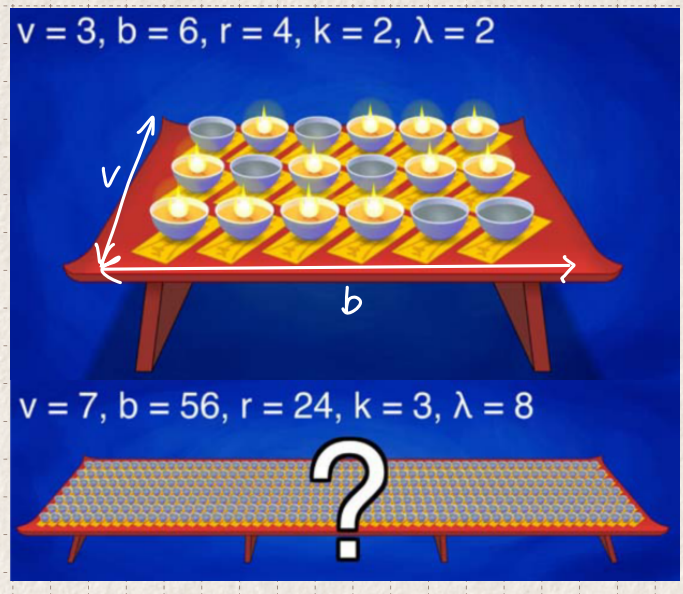
Symmetries and Dominance

Which of the following best explains **variable symmetry**?

- ○ Given a solution, if we can scramble the assigned values of the variables (not necessarily resulting in the same values in total), and we still have a solution, then we have a variable symmetry.
- ○ Given a solution, if we can scramble the assigned values of the variables, so that we have the same values in total (but not necessarily associated with the same variable as before), and we still have a solution, then we have a variable symmetry.
- ○ If we swap the position of the variables, and we get a solution, then we have a variable symmetry.
- ○ If it doesn't matter how we swap the values of the variables, we still end up with a solution, then we have a variable symmetry.

Keywords 关键词	Notes 笔记	Review 复习记录 / / / / /

Matric Model Symmetries



v是行数
b是列数
r是每行点亮的灯数
k是每列点亮的灯数
λ是between any two distinct rows, the number of columns containing two lit lamps

■ Data

```
int: v;
set of int: ROW = 1..v;
int: b;
set of int: COL = 1..b;
int: r;
int: k;
int: lambda;
```

■ Decisions: which lamps are lit

```
array[ROW,COL] of var bool: m;
solve satisfy;
```

```
constraint forall(i in ROW)(sum(j in COL)(m[i,j]) = r);
constraint forall(j in COL)(sum(i in ROW)(m[i,j]) = k);
constraint forall(i1, i2 in ROW where i1 < i2)
                (sum(j in COL)
                    (m[i1,j] /\ m[i2,j]) = lambda);
```

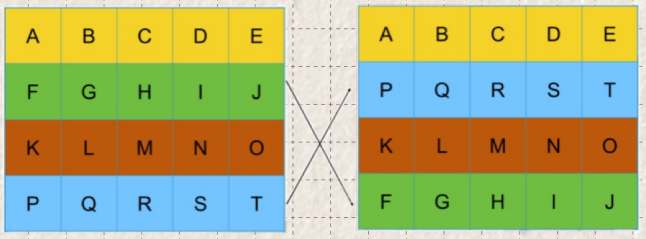
每行点亮r盏灯
每列点亮k盏灯
λ

Keywords 关键词 | Notes 笔记

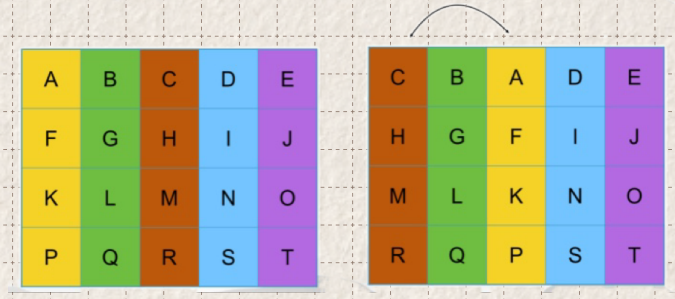
too many Semmetries!

交换任意数量的行可以得到另一种解决方案

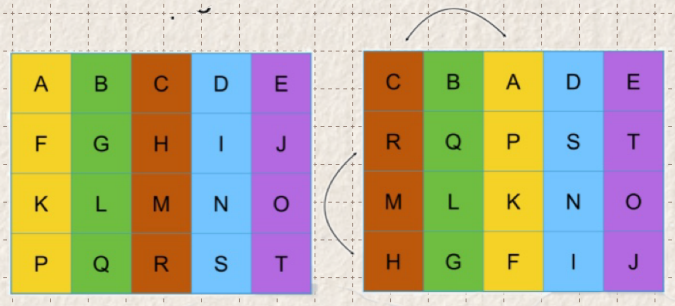
eg:



交换行



交换列



交换行和列

One lex leader constraint per symmetry

A	B	C
D	E	F

1. ABCDEF ≤lex ABCDEF
2. ABCDEF ≤lex ACBDFE
3. ABCDEF ≤lex BACEDF
4. ABCDEF ≤lex CBAFED
5. ABCDEF ≤lex BCAEFD
6. ABCDEF ≤lex CABFDE
7. ABCDEF ≤lex DEFABC
8. ABCDEF ≤lex DFEACB
9. ABCDEF ≤lex EDFBAC
10. ABCDEF ≤lex FEDCBA
11. ABCDEF ≤lex EFDBCA
12. ABCDEF ≤lex FDECAB

一共有 n!m! 个

Breaking all symmetries 需要 n!m! 个约束

Choose a subset

选择 2.3.7

Content Index

Keywords 关键词 | Notes 笔记

- One lex leader constraint per selected symmetry

A	B	C
D	E	F

- ABCDEF $\leq_{lex}$ DEFABC
- ABCDEF $\leq_{lex}$ ACBDFE
- ABCDEF $\leq_{lex}$ BACEDF

- Simplify the constraints, e.g.
  - ABCDEF $\leq_{lex}$ ACBDFE
  - BCEF $\leq_{lex}$ CBFE removing same positions
  - BE $\leq_{lex}$ CF $XY \leq_{lex} YX \Rightarrow X \leq Y$

- One lex leader constraint per selected symmetry

A	B	C
D	E	F

- ABCDEF $\leq_{lex}$ DEFABC $\Leftrightarrow$ ABC $\leq_{lex}$ DEF
- ABCDEF $\leq_{lex}$ ACBDFE $\Leftrightarrow$ BE $\leq_{lex}$ CF → does not break all symmetry
- ABCDEF $\leq_{lex}$ BACEDF $\Leftrightarrow$ AD $\leq_{lex}$ BE
- Does not break all symmetries, e.g.
  - ABCDEF = 011100
  - Now 011 $\leq_{lex}$ 100, 10 $\leq_{lex}$ 10, 01 $\leq_{lex}$ 10
  - but not $\leq_{lex}$ 001110 = FEDCBA

Innovation、Conclusion —到此

Intrucduction Objective

BERT + CNN

LSTM