Multiple Modeling: Permutation

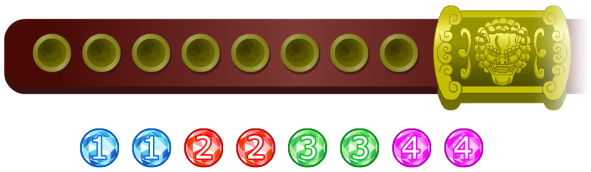
Jimmy Lee & Peter Stuckey

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

The Belt Problem

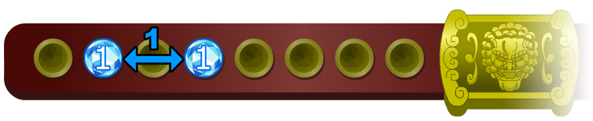
2

The Belt Problem



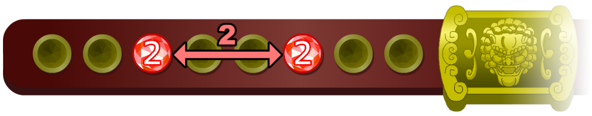
3

The Separation Constraint



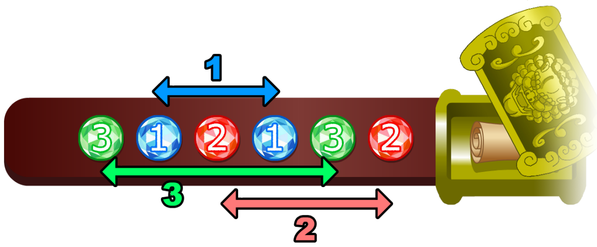
4

The Separation Constraint



Belt Problem Example

- B(m,n): Given m copies of each of the numbers 1..n, find a sequence of these numbers where there are k digits between every two consecutive copies of digit k



An Assignment Problem

- How is this an assignment problem?

- Map DOM = 1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 4_1, 4_2
  - the ordered copies of the digits
- to COD = 1,2,3,4,5,6,7,8
  - the position in the sequence

Belt Problem Model (beltPos.mzn)

- Data and Decision

```
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;
```

- Constraints

```
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]);
```

The Inverse Belt Problem

- DOM is the positions and COD is the digit-copies
- We need to map DIG x COPY to a single integer: $d_c = m^*(d\text{-}1) + c$
  - $1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 4_1, 4_2 = 1, 2, 3, 4, 5, 6, 7, 8$

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

- The inverse alldifferent constraint is easy to express

```
alldifferent([dc[p] | p in POS]);
```


The Inverse Separation Constraint

- How do we model the inverse separation constraints?
  - $dc[p] = d_c \iff po[d,c] = p$

```
forall(d in DIG, c in 1..m-1)
   (po[d,c+1] = po[d,c] + d + 1);
```

  - thus the inverse constraint is

```
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);
```

- Terrible encoding!
  - if position p has d_c then p+d+1 has d_{c+1}

A Note About the Inverse Constraints

```
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);
```

- Notice that we are accessing positions outside the *dc* array, e.g. $l + d + 1$
- But this is correct
  - None but the last copy of a digit d_m can occur in the last *d* positions of the sequence
  - Relational semantics requires $dc[i] = j$ evaluates to false when $i > l$ since ***dc*[*i*] does not exist**

11

A Note About the Inverse Constraints

- Safer to avoid out of array access
  - but clumsy in this instance

```
forall(d in DIG, c in 1..m-1,
      p in POS)
    (dc[p] = m*(d-1) + c <->
       if p+d+1 in POS then
          dc[p+d+1] = m*(d-1) + c + 1
       else false endif);
```

12

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

Inverse Belt Problem Model (beltDig.mzn)

```
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;
```

13

Two Distinct Models

- Belt:
  - *po*[*d*,*c*] = position of d_c
- Inverse Belt
  - *dc*[*p*] = the digit d_c in position *p*

- Which runs faster?

- Which one is easier to print the Belt sequence?

```
output["\((dc[p]-1) div m + 1) " |
    p in POS];
4 1 3 1 2 4 3 2
```

14

Combined Belt Model

- We can combine the models
  - Omit the `alldifferent` constraints
    - they are implied by `inverse`
  - Omit the inverse models encoding of the constraints
    - they are implied by the equations on *po*[*d*,*c*]
- CP solvers can solve the combined model better
  - searching on *po* and *dc* variables simultaneously

15

Combined Belt Model (beltComb.mzn)

```
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]);

solve satisfy;
output["\((dc[p]-1) div m + 1) " | p in POS];
```

16

Summary

- A further example to illustrate modeling from different viewpoints
- The Belt problem is an adaptation and generalisation of the well-known Langford's Problem, which is a mathematical puzzle with applications in circuit design and others
- While it is possible to describe requirements of the example completely in either viewpoint, some requirements are more naturally described in certain viewpoints

17

Image Credits

All graphics by Marti Wong, ©The Chinese University of Hong Kong and the University of Melbourne 2016

18