Choosing a Fixed Cardinality Set

Jimmy Lee & Peter Stuckey

SetSelect Question Revised (baguaCard-10-8.dzn)

- Given a subset of numbers `1..nSpots` for each symbol in `SYMB`, choose a subset of `1..nSpots` of size `size` which includes at most one from each subset and maximize the damage points of the chosen set

```
nSpots = 10;
damage = [10, 8, 4, 2, 6, 9, 5, 3, 8, 10];
size = 3;
SYMB = {'天','澤','火','雷','風','水','山','地'};
group = [{1,4,6}, {1,2,6,7}, {1,3,6,8}, {1,2,3},
    {2,9,10}, {5,6,8,10}, {7,8,10}, {1,3,5}];
```

2

SetSelect Revised Addition (baguaCardSet.mzn)

- Additional constraint

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

- Executing the model

```
attacks: {5,7,9} & damage: 19;
```

- But we can model a set with **known** cardinality differently!

3

Deciding a Fixed Cardinality Set

- Instead of a set variable

```
var set of SPOT: attacks;
```

with cardinality constraint

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

- An array of `size` elements

```
array[1..size] of var SPOT: attacks;
```

  - and some other constraints …
- Why: suppose nSpots = 1000, size = 4
- First representation: 1000 Boolean variables
- Second representation: 4 integer variables

4

Deciding a Fixed Cardinality Set

- Consider
  `array[1..3] of var 1..10: x;`
- How many possible values? 1000
- And `var set of 1..10: x;` with
  `card(x) = 3;`
  - 10 * 9 * 8 / 3 * 2 * 1 = 120
- First issue: some array solutions are not sets of cardinality 3
  - e.g. [1,1,1] = {1}, [1,2,1] = {1,2}
- Solution: ensure all different

```
forall(i,j in 1..u where i < j)
    (x[i] != x[j]);
```

5

Deciding a Fixed Cardinality Set

- Consider

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

- How many possible values?
  - 10 * 9 * 8 = 720
- Second issue: multiple representations of the same set
  - 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]);
```

6

Deciding a Fixed Cardinality Set

- Consider

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

- How many possible values?
  - 10 * 9 * 8 = 720
- Second issue: multiple representations of the same set
  - 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]);
```

7

Critical Issues in Modeling Decisions

- Decisions in the problem
  - are not necessarily the decisions in the model
- If possible make them identical but
  - what if you are deciding
    - a path in a graph,
    - a tree structure
- Critical modeling issue (1)
  - decisions in the model (satisfying constraints)
  - are valid decisions in the problem
- Add constraints to make this so
  - e.g. add constraints forcing array elements ≠

8

Critical Issues in Modeling Decisions

- Multiple decisions in the model
  - reflect the same decision in the problem
  - e.g. x = [1,2,7], x = [7,2,1] for x = {1,2,7}
- Critical modeling issue (2)
  - try to have only **one** set of decisions in the model corresponding to each solution
  - reflect valid decisions in the problem
- Add constraints to remove all but one set
  - e.g. add constraints forcing array elements <

9

SetSelect Question Revised (baguaCard-10-8.dzn)

- Given a subset of numbers `1..nSpots` for each symbol in `SYMB`, choose a subset of `1..nSpots` of size `size` which includes at most one from each subset and maximize the damage points of the chosen set

```
nSpots = 10;
damage = [10, 8, 4, 2, 6, 9, 5, 3, 8, 10];
size = 3;
SYMB = {'天','澤','火','雷','風','水','山','地'};
group = [{1,4,6}, {1,2,6,7}, {1,3,6,8}, {1,2,3},
    {2,9,10}, {5,6,8,10}, {7,8,10}, {1,3,5}];
```

10

Bagua Fixed Cardinality Model (baguaCardInt.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

array[1..size] of var SPOT: attacks;

constraint forall(i in 1..size-1)
   (attacks[i] < attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(i in 1..size)(damage[attacks[i]]);
solve maximize (totalDamages);
```

11

Bagua Fixed Cardinality Model (baguaCardInt.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

array[1..size] of var SPOT: attacks;

constraint forall(i in 1..size-1)
   (attacks[i] < attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(i in 1..size)(damage[attacks[i]]);
solve maximize (totalDamages);
```

Decisions

12

Bagua Fixed Cardinality Model (baguaCardInt.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

array[1..size] of var SPOT: attacks;

constraint forall(i in 1..size-1)
   (attacks[i] < attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(i in 1..size)(damage[attacks[i]]);
solve maximize (totalDamages);
```

Valid representations

13

Bagua Fixed Cardinality Model (baguaCardInt.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

array[1..size] of var SPOT: attacks;

constraint forall(i in 1..size-1)
   (attacks[i] < attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(i in 1..size)(damage[attacks[i]]);
solve maximize (totalDamages);
```

At most one intersection

14

Bagua Fixed Cardinality Model (baguaCardInt.mzn)

```
int: nSpots;
set of int: SPOT = 1..nSpots;
array[SPOT] of int: damage;
enum SYMB;
array[SYMB] of set of SPOT: group;
int: size;

array[1..size] of var SPOT: attacks;

constraint forall(i in 1..size-1)
   (attacks[i] < attacks[i+1]);
constraint forall(s in SYMB)(sum(i in 1..size)
   (attacks[i] in group[s]) <= 1);

var int: totalDamages =
   sum(i in 1..size)(damage[attacks[i]]);
solve maximize (totalDamages);
```

Objective

15

Solving the Model

- Executing the model

```
attacks: [5,7,9] & damage: 19;
```

16

Summary

- There are multiple ways to represent fixed cardinality sets
  - var set of OBJ + cardinality constraint
    - good if the solver natively supports sets
    - good when OBJ is not too big
  - array[1..u] of var OBJ
    - good when u is small
- Two critical issues in modelling decisions
  - ensure each solution to the model is a solution of the problem
  - try to ensure each solution of the problem only has one solution in the model (symmetry)

17

Image Credits

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

18