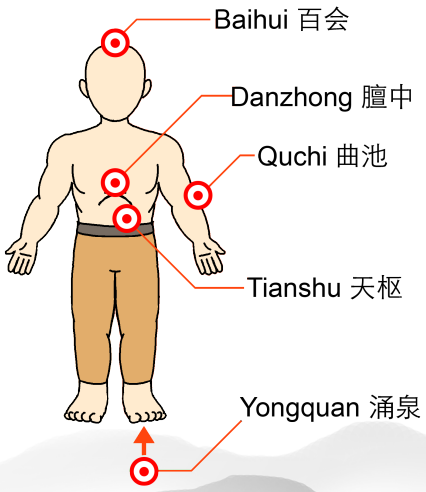
# Modeling Functions

Jimmy Lee & Peter Stuckey

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

## Three Heroes Combating Lü Bu

## Lü Bu's Weak Spots




3

## Damage Matrix

| |  |  |  |  |  |
|---|---|---|---|---|---|
|  | 7 | 1 | 3 | 4 | 6 |
|  | 8 | 2 | 5 | 1 | 4 |
|  | 4 | 3 | 7 | 2 | 5 |

4

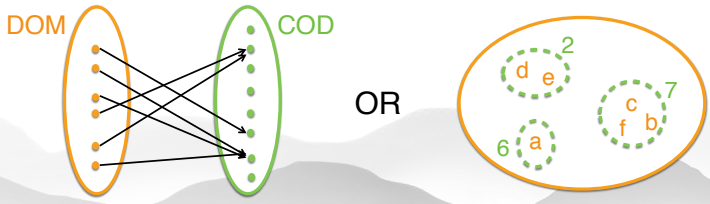
## Lü Bu and the Pure Assignment Problem

- Liu, Guan and Zhang will each attack a **different** spot of Lü Bu to distract him
- Find out the spots they should attack in order to maximize the damage to Lü Bu
- The Lü Bu Problem takes the form of a Pure Assignment Problem
  - 3 heroes and 5 spots
  - **Assign** each hero to a spot so as to maximise damage

5

## Deciding Functions

- Many combinatorial problems have the form:
  - assign to each object in one set DOM (domain)
  - a value from another set COD (codomain)
- We can interpret this as
  - Defining a function DOM ➔ COD
  - Or partitioning the set DOM (in sets labelled by COD)




6

## Deciding Functions

- This function could be
  - injective: assignment problem
  - bijective (DOM = COD): matching problem
- In the Lü Bu Problem, DOM is the heroes and COD is the weak spots of Lü Bu



## Deciding Functions

- This function could be
  - injective: **assignment problem**
  - bijective (DOM = COD): matching problem
- In the Lü Bu Problem, DOM is the heroes and COD is the weak spots of Lü Bu

## Lü Bu Problem Data and Decisions (lvbu.mzn)

- Data

```
enum HERO;
enum SPOT;
array[HERO,SPOT] of int: damage;
```

- What are the decisions?

```
array[HERO] of var SPOT: pos;
```

- What is the objective?

```
var int: tDamages = sum(h in HERO)
   (damage[h,pos[h]]);
solve maximize tDamages;
```



## Lü Bu Problem Constraints (lvbu.mzn)

- Each spot is assigned to at most one hero

```
forall(s in SPOT)
   (sum(h in HERO)
      (pos[h] = s) <= 1);
```

- Alternatively
- Each two heroes are assigned different spots to attack

```
forall(h1, h2 in HERO where h1 < h2)
   (pos[h1] != pos[h2]);
```

- Which is better?

## Global Constraints

- Let's not choose (?)
  - Different solvers will prefer different representations
- Record the structure of the problem
- Let the solver determine the best way it knows how to handle this substructure
- In modelling these substructures, we use

# global constraints



## Alldifferent

- Global constraint version

```
alldifferent(pos);
```

- Enforces that each hero is assigned a different spot
- Solvers can make use of the substructure to solve better.
- The first example of a global constraint
- Captures the
  - assignment substructure, or alternatively
  - deciding an injective function

## The Lü Bu Assignment Model (lvbu.mzn)

```
enum HERO;
enum SPOT;
array[HERO,SPOT] of int: damage;

array[HERO] of var SPOT: pos;

include "alldifferent.mzn";
alldifferent(pos);

var int: tDamages = sum(h in HERO)
   (damage[h,pos[h]]);
solve maximize tDamages;

output ["\(h): \(pos[h])\n" | h in HERO] ++
   ["Total Damages: \(tDamages)"];
```



## Assignment Problem

- The pure assignment problem is very well studied
  - specialized polynomial (fast) algorithms
  - maximal weighted matching
- If you have a pure assignment problem
  - use a specialized algorithm
- BUT the real world is never pure
  - add some side constraints and these specialised algorithms almost always break!

THE UNIVERSITY OF MELBOURNE

香港中文大學
The Chinese University of Hong Kong

## Summary

- Deciding a (finite) function is common
- Deciding an injective function is a
  - assignment (sub)problem
- The global constraint `alldifferent` captures this

- Global constraints
  - names of combinatorial substructures
  - solvers can use their best method for capturing this
  - plenty more to come …



## Image Credits

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