Selecting a Set

Jimmy Lee & Peter Stuckey

香港中文大學
The Chinese University of Hong Kong

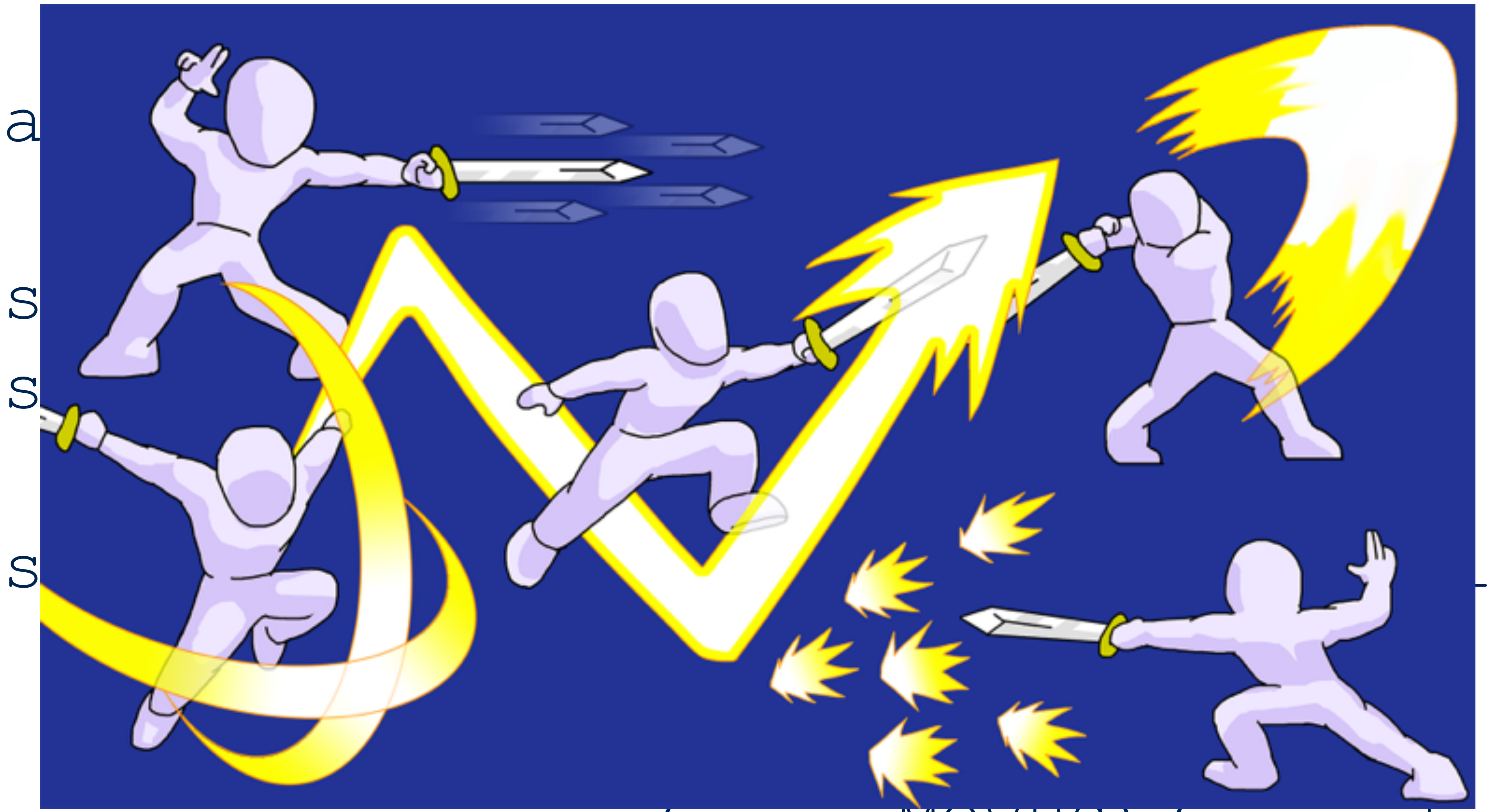
THE UNIVERSITY OF
MELBOURNE

0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
output [show(occur)];
```

0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

arra

cons                                          >= 0);
cons                                          <= 1);

cons                                         ] *



solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;
```

Power	6	8	5	3	4
Duration	4	5	3	2	3

0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

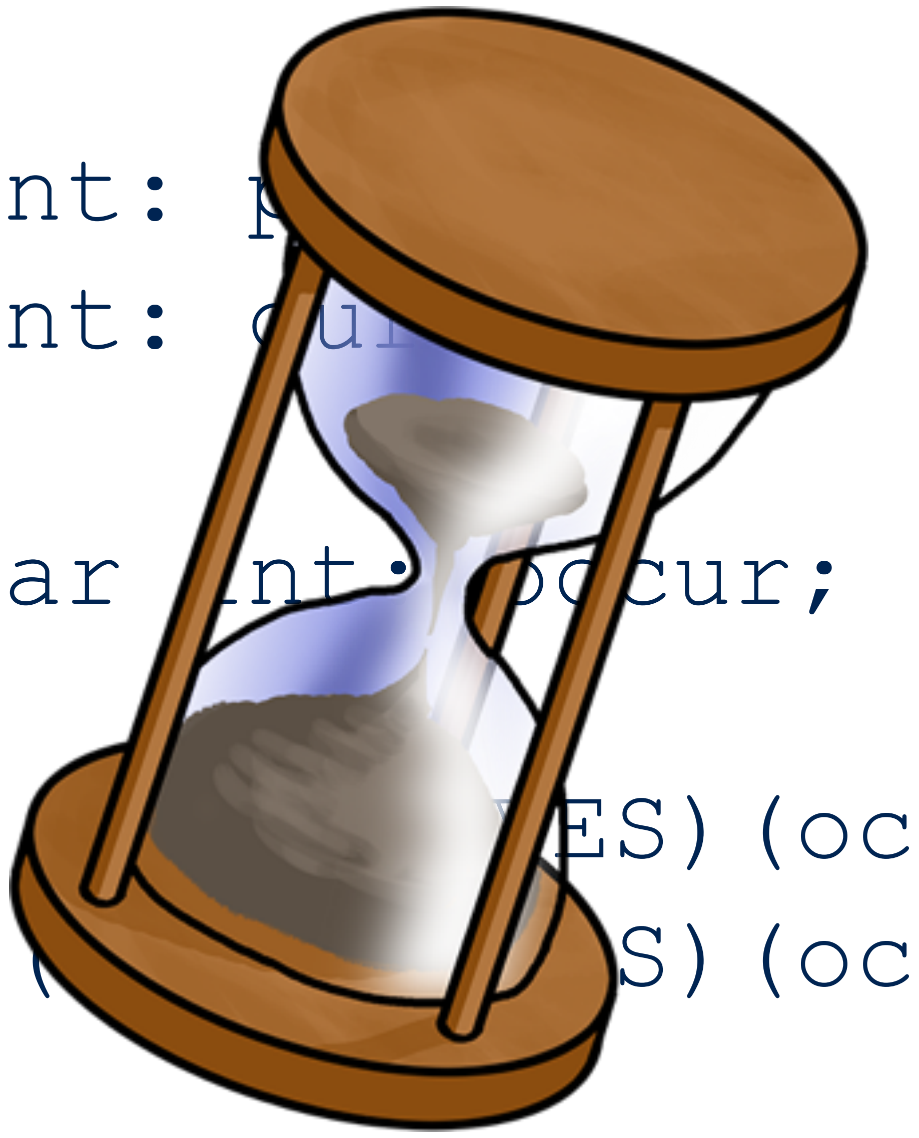
```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

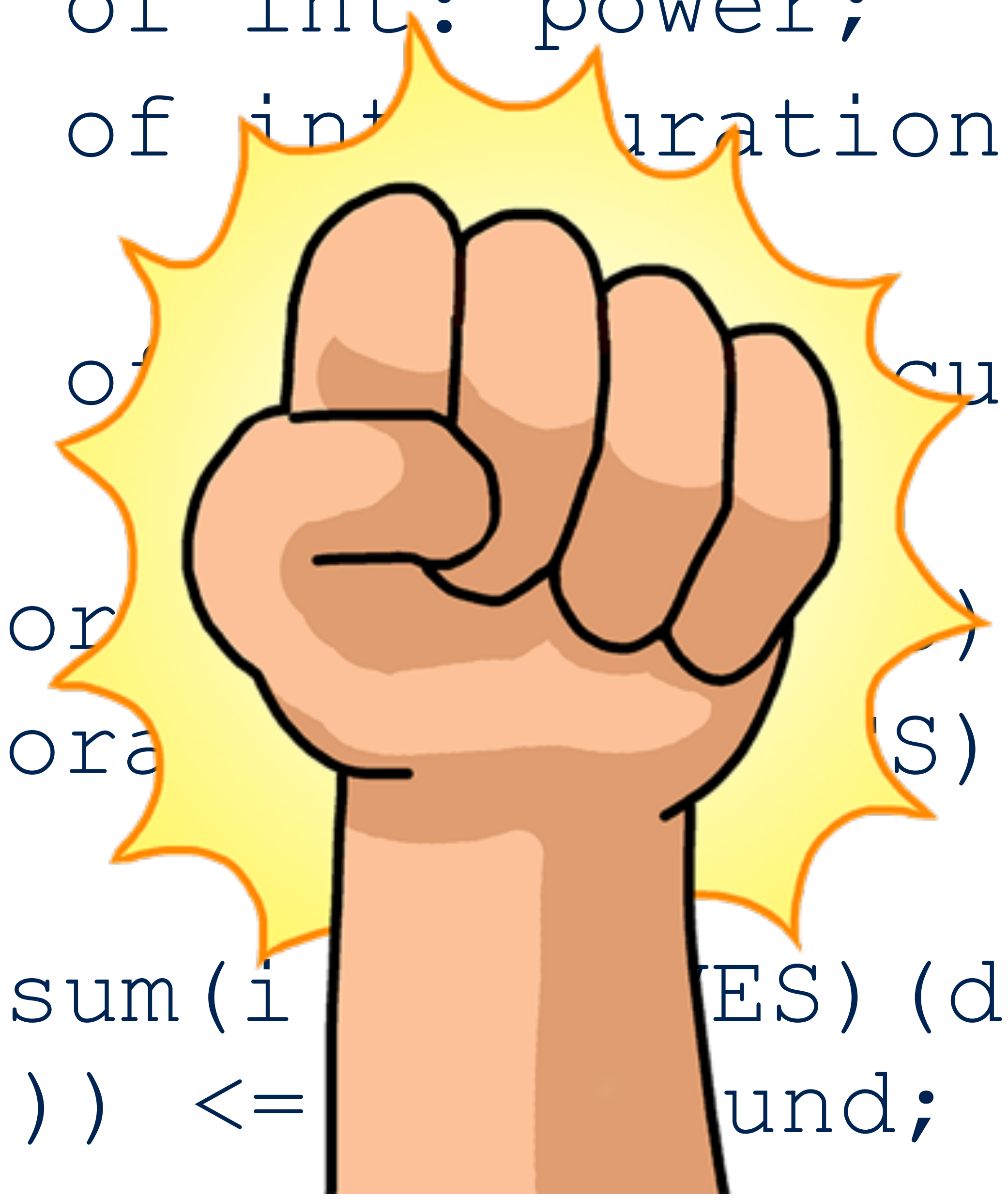
0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: p
array[MOVES] of int: du

array[MOVES] of var int: occur;

constraint forall(      ES)(occur[i] >= 0);
constraint forall(       S)(occur[i] <= 1);

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```



0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of in         uration;

array[MOVES] of             cur;

constraint for              ) (occur[i] >= 0);
constraint fora             S) (occur[i] <= 1);

constraint (sum(i          ES) (duration[i] *
    occur[i])) <=          und;
```



```
solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

0-1 Basic Yellow Turban Model (yellow01Basic.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var int: occur;

constraint forall(i in MOVES)(occur[i] >= 0);
constraint forall(i in MOVES)(occur[i] <= 1);

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

0-1 Yellow Turban Model (yellow01.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

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

~~constraint forall(i in MOVES)(occur[i] >= 0);~~
~~constraint forall(i in MOVES)(occur[i] <= 1);~~

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

solve maximize sum(i in MOVES)(power[i] *
    occur[i]);
```

0-1 Boolean Yellow Turban Model (yellow01Bool.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var bool: occur;

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

solve maximize sum(i in MOVES)(power[i] *
    bool2int(occur[i]));
```

bool2int

- Casting a Boolean to an integer
    - bool2int(false) = 0
    - bool2int(true) = 1

- Many solvers will use the same internal representation for 0-1 integers and Booleans

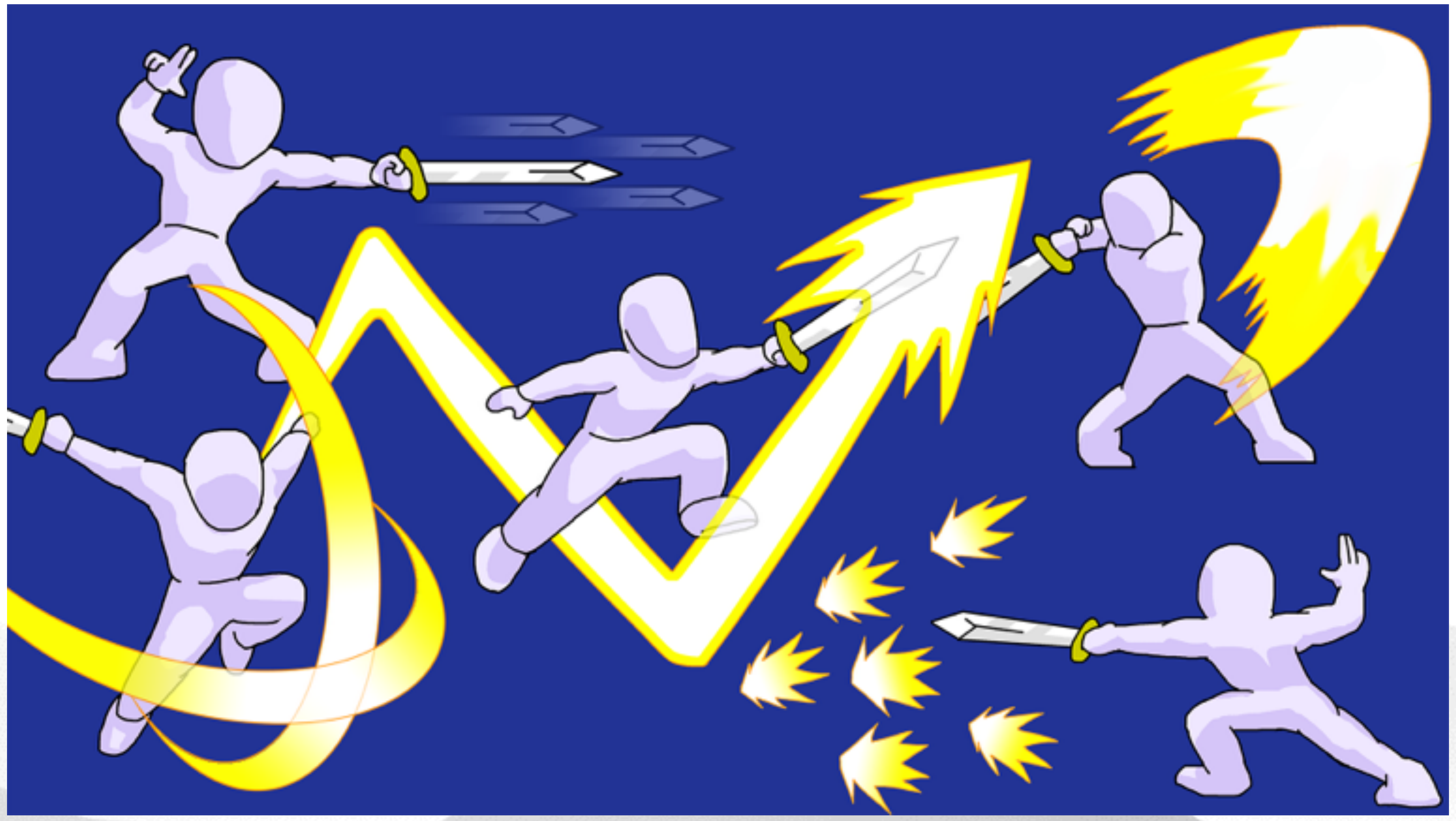
- When you use a Boolean where MiniZinc expects an integer, MiniZinc will automatically “add” the bool2int function

0-1 Boolean Yellow Turban Model (yellow01Bool.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var bool: occur;

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

solve maximize sum(i in MOVES)(power[i] *
    ~~bool2int(~~occur[i]~~)~~);
```

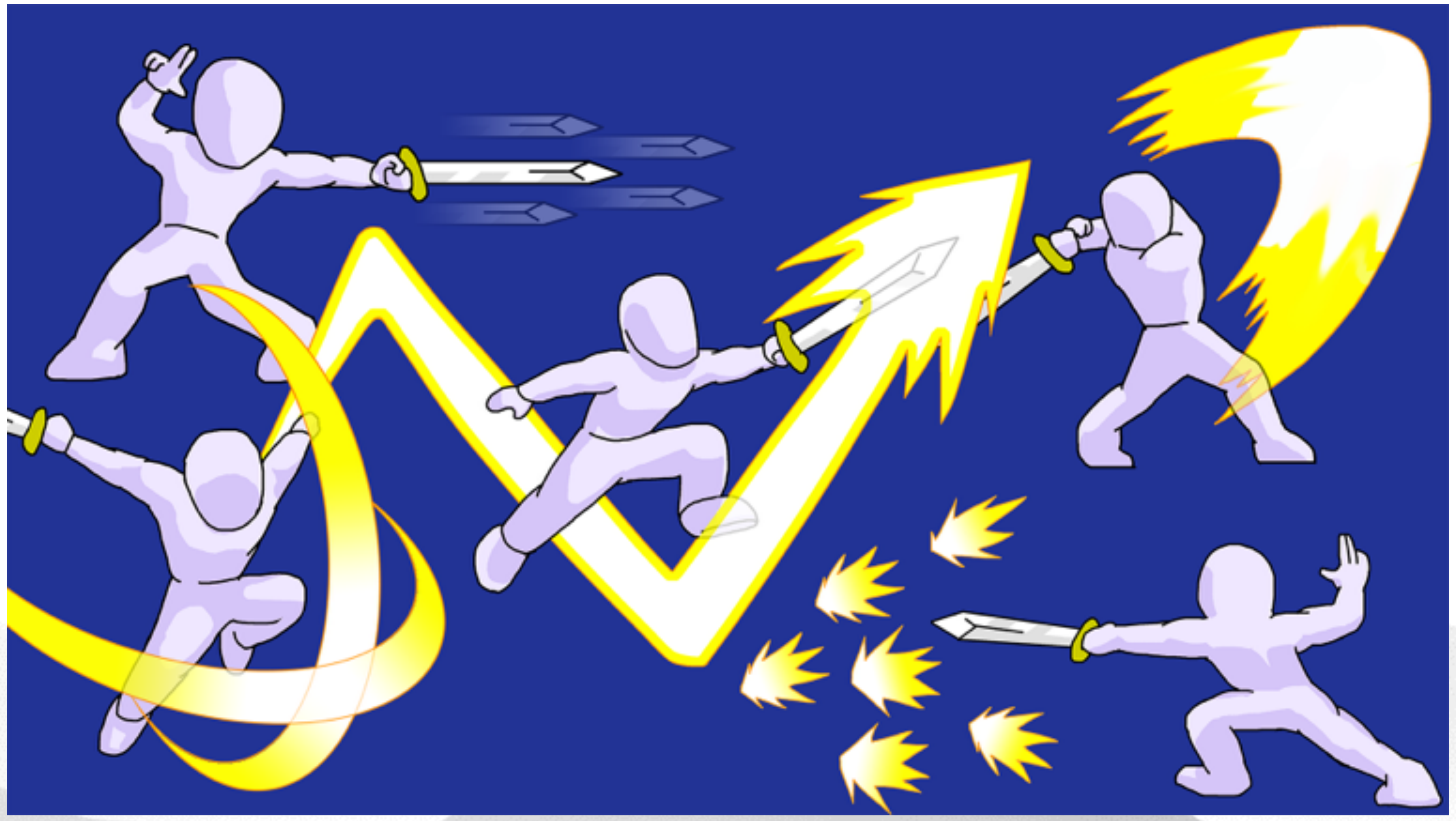
Choosing from a Set of Objects

- The Yellow Turban Rebellion story is a typical problem requiring us to select a subset from a set of objects that
  - Meets some criteria; and
  - Optimizes some objective functions



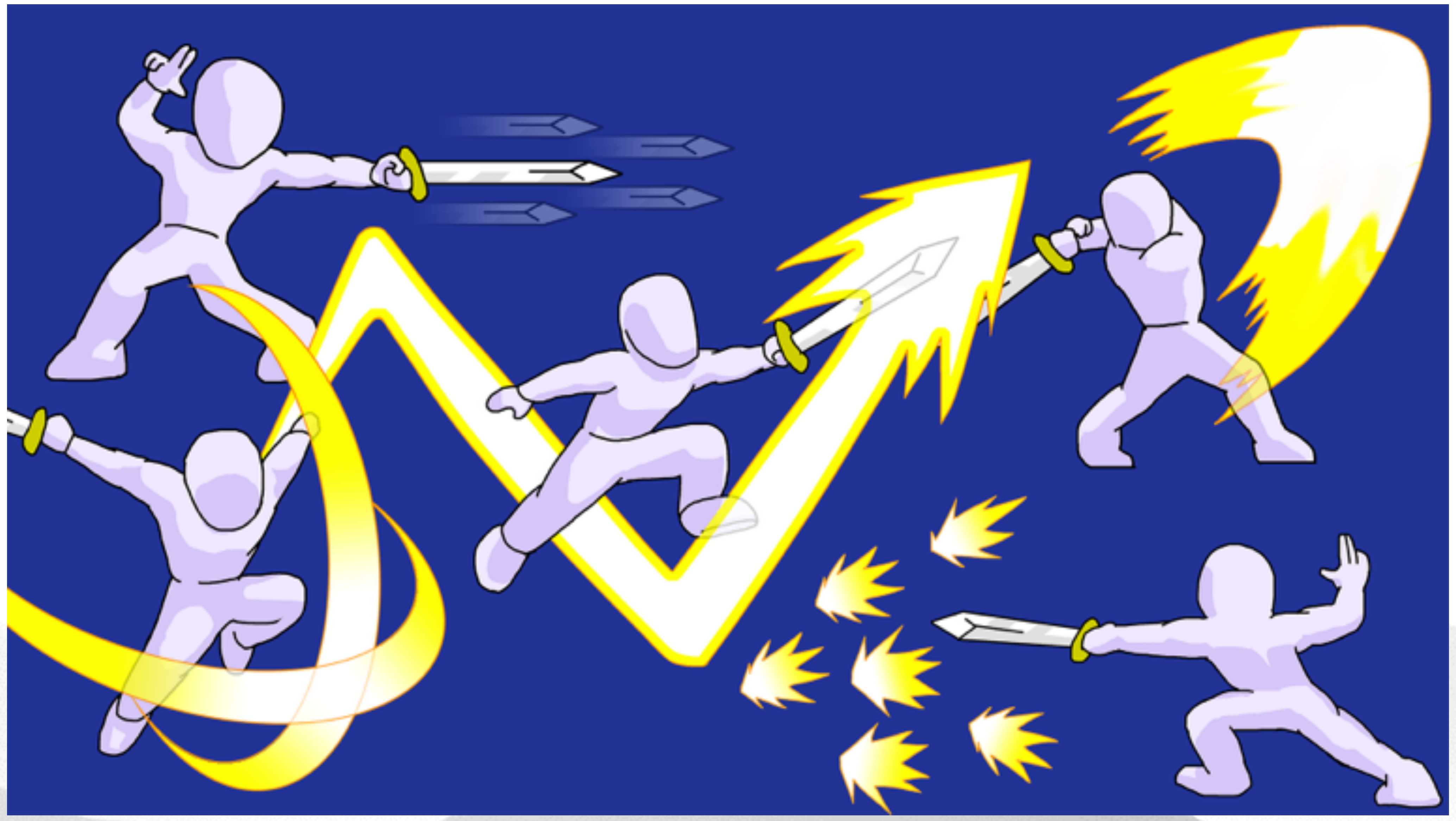
Choosing from a Set of Objects

- The Yellow Turban Rebellion story is a typical problem requiring us to select a subset from a set of objects that
  - **Meets** some criteria; and
  - Optimizes some objective functions



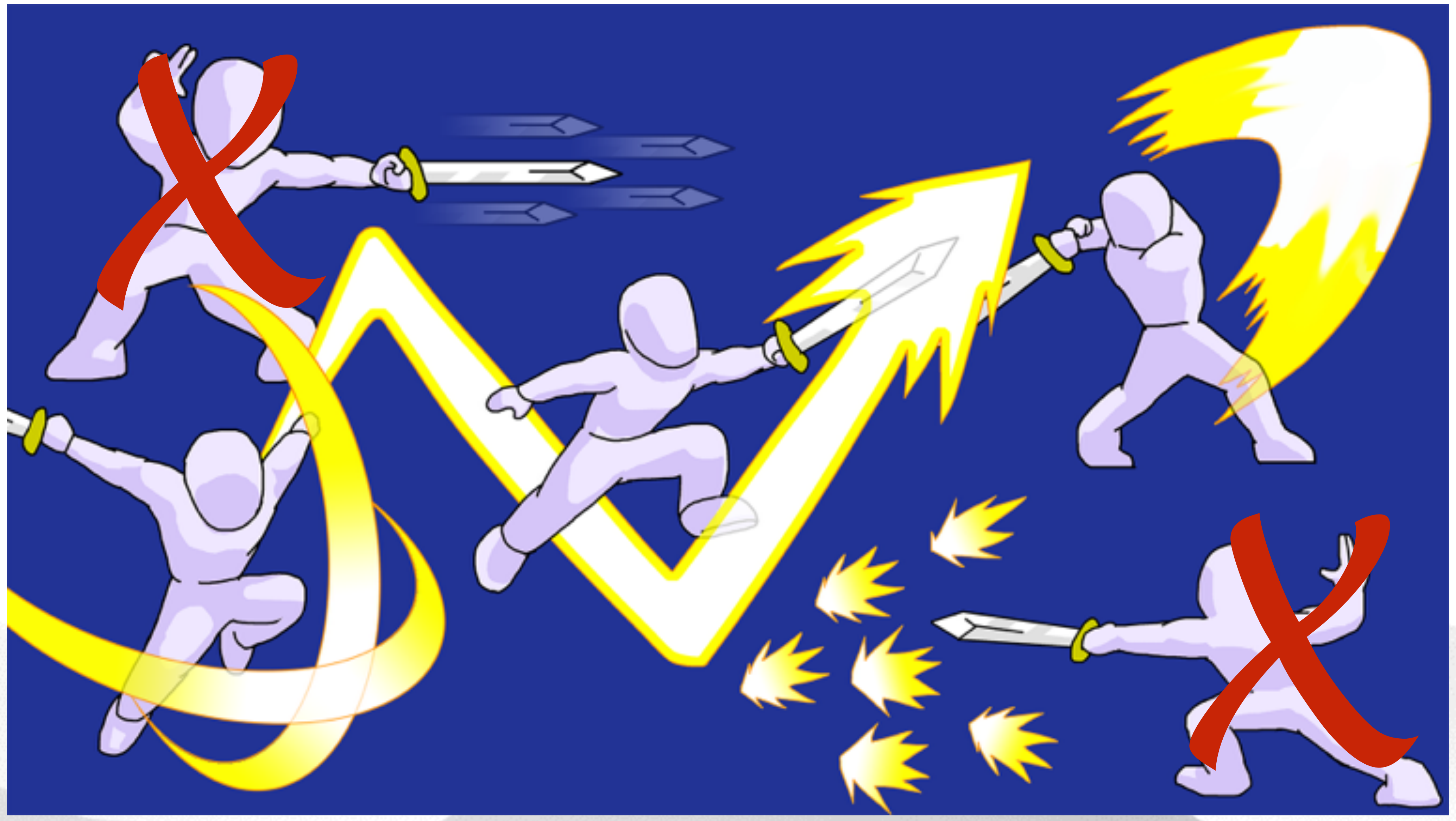
Choosing from a Set of Objects

- The Yellow Turban Rebellion story is a typical problem requiring us to select a subset from a set of objects that
  - Meets some criteria; and
  - **Optimizes** some objective functions



Choosing from a Set of Objects

- The Yellow Turban Rebellion story is a typical problem requiring us to select a subset from a set of objects that
  - Meets some criteria; and
  - Optimizes some objective functions



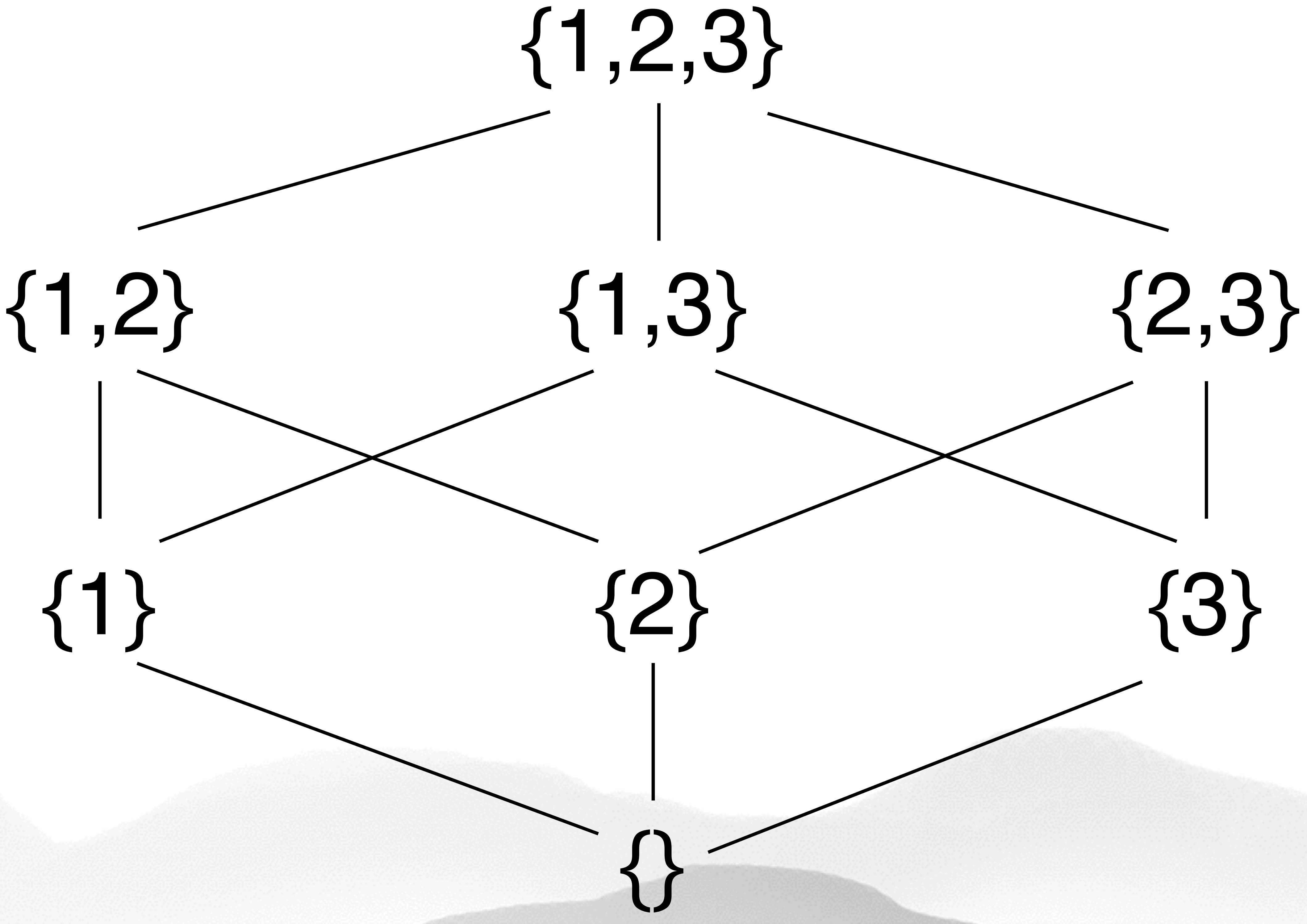
0-1/Set Selection Problem

- An array of 0-1 variables
- An array of Boolean variables
- A **set** variable

Set Variables

- Set variables in MiniZinc choose a set from a given fixed superset, e.g.

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



{1,2,3}

{1,2} {1,3} {2,3}

{1} {2} {3}

{}

0-1 Boolean Yellow Turban Model (yellow01Bool.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

array[MOVES] of var bool: occur;

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

solve maximize sum(i in MOVES)(power[i] *
    ~~bool2int(~~occur[i]~~)~~);
```

0-1 Set Yellow Turban Model (yellow01Set.mzn)

```
enum MOVES;
int: timeBound;
array[MOVES] of int: power;
array[MOVES] of int: duration;

var set of MOVES: occur;

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

solve maximize sum(i in MOVES)(power[i] *
    (i in occur));
```

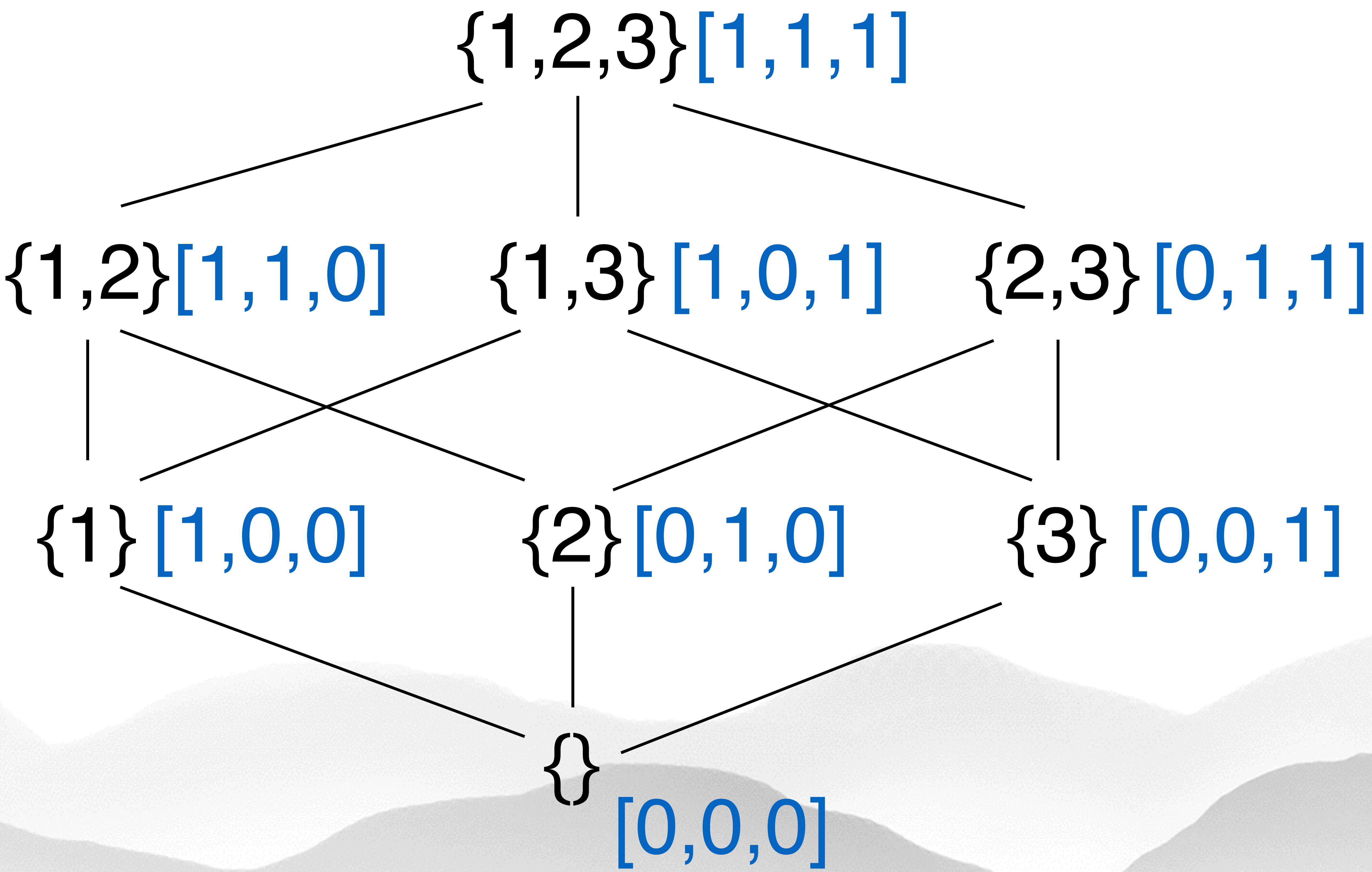
Concise Yellow Turban Model (yellow01SetConcise.mzn)

```
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]);
```

Other Set Representations

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

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



{1,2,3}[1,1,1]

{1,2}[1,1,0] {1,3}[1,0,1] {2,3}[0,1,1]

{1}[1,0,0] {2}[0,1,0] {3}[0,0,1]

{}[0,0,0]

Set Operators

- MiniZinc provides (infix) set operations
  - `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}

Which Model is the Best?

- Most solvers treats each model the same
  - CP solvers may treat the last model better since they can combine cardinality reasoning with other set reasoning

- Model whichever makes it easier to express the constraints
  - the 0-1 integer first version

- Model using the highest level model
  - the set last version

Summary

- Modeling with sets is common for combinatorial problems
- The Yellow Turban Rebellion story is actually an adaptation of the well-known 0-1 Knapsack Problem
  - it appears frequently in the real world for e.g : selection of investments, least wasteful cutting of raw materials, and knapsack cryptosystems
- There are at least **3** modeling approaches
  - Indicator variables: 0-1 or bool variables
  - Native set variables

Image Credits

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