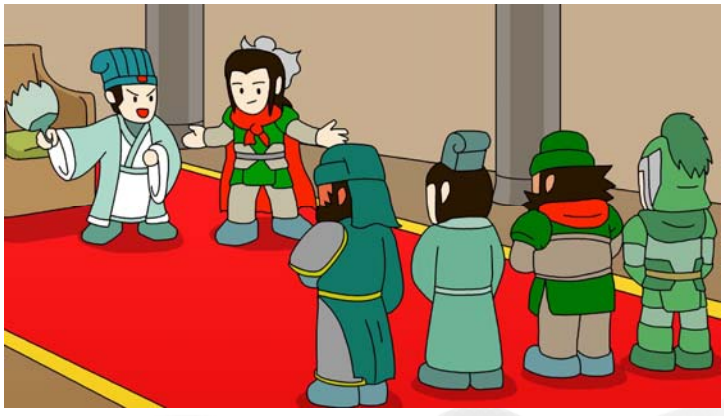
Basic Scheduling

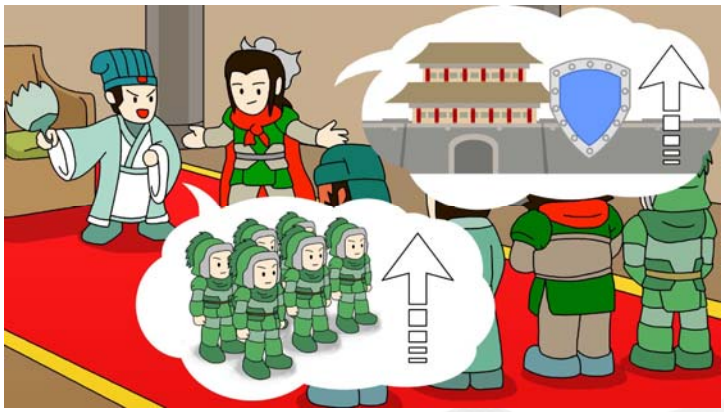
Jimmy Lee & Peter Stuckey

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

Strengthening the Defense

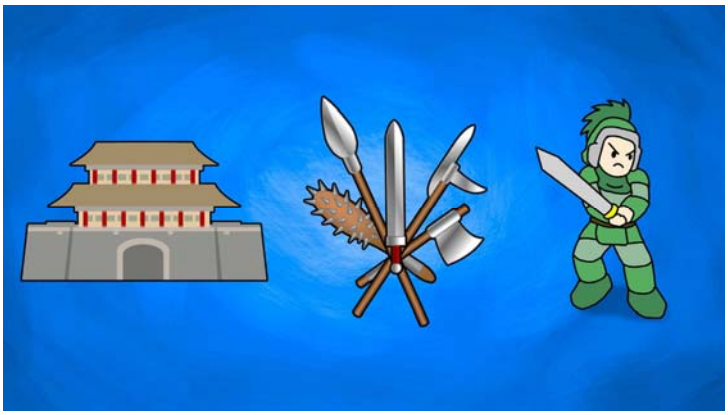


Strengthening the Defense



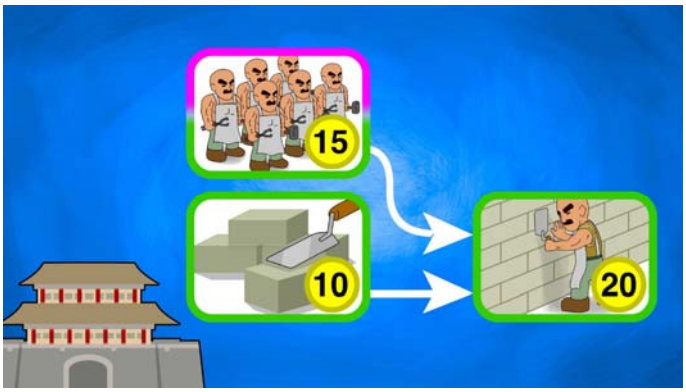
3

Three Types of Tasks



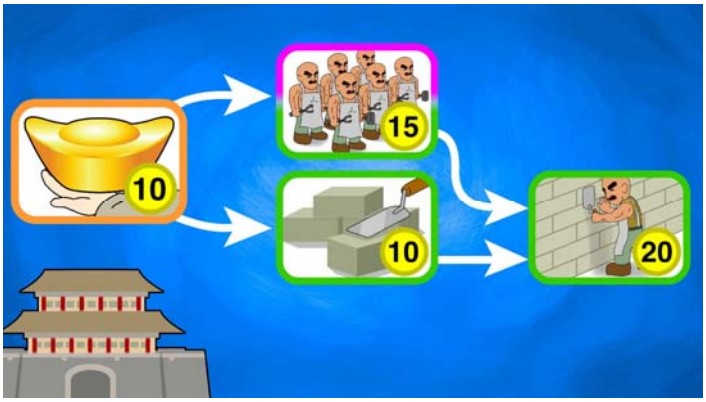
4

Strengthening the Wall



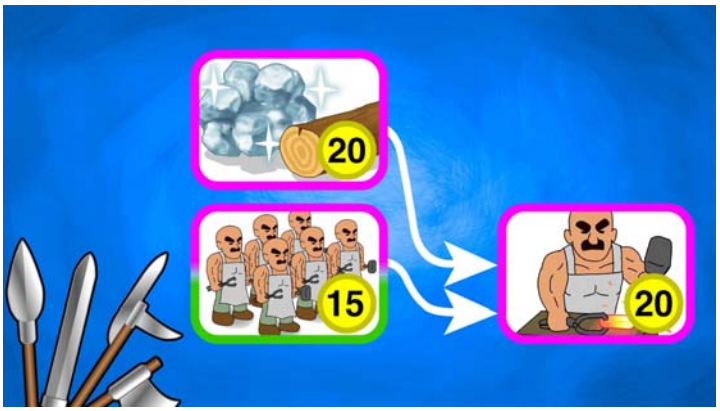
5

Strengthening the Wall



6

Making More Weapons

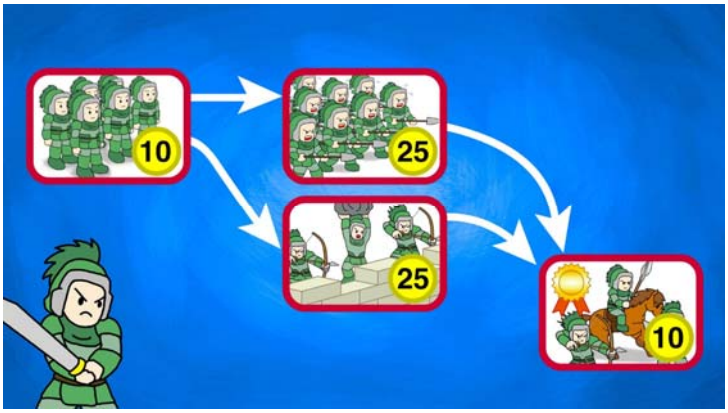


7

Making More Weapons

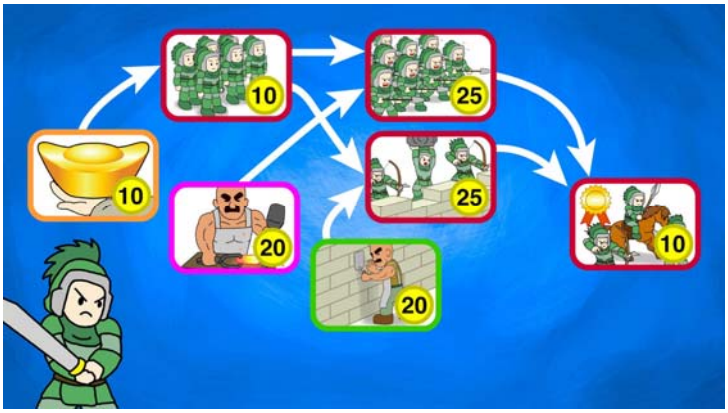
8

Training an Elite Army



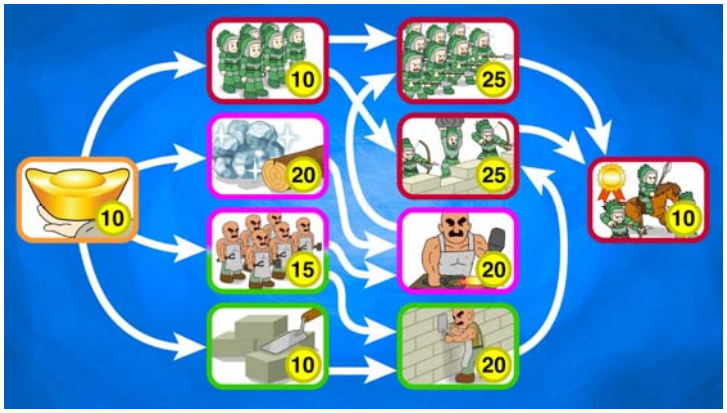
9

Training an Elite Army



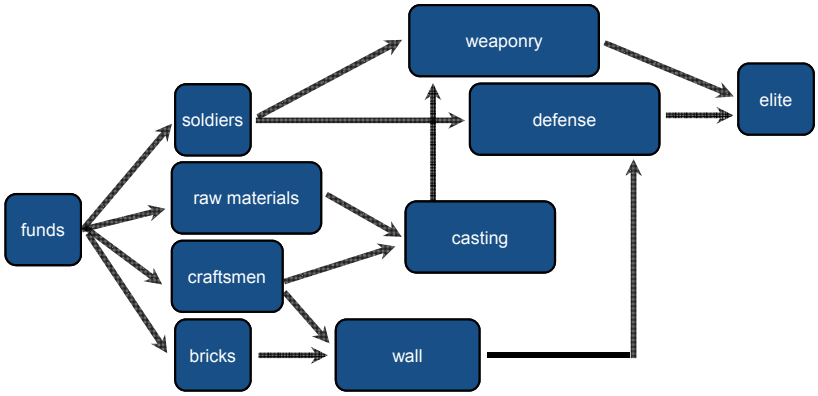
10

A Non-Trivial Scheduling Task



11

Basic Scheduling



- Length indicates durations
- Arcs indicate precedences

12

Basic Scheduling

- Scheduling is an important class of discrete optimisation problems
- Basic scheduling involves:
  - tasks with durations
  - precedences between tasks
    - one task must complete before another starts
- The aim is to schedule the tasks
  - usually to minimize the latest end time

13

Modeling Time

- In discrete optimization
  - time is modeled by integers (not continuous)
- Time variables
  - tend to have VERY large ranges
    - e.g. start times on the minute for a 7 day schedule
  - typically only care about
    - earliest time, or
    - latest time
  - when reasoning (not about all possible times)

14

Basic Scheduling Data & Decisions (basic_sched.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;
```

15

Basic Scheduling Data File (basic_sched.dzn)

```
TASK = {FUNDS, SOLDIERS, DEFENSE, WEAPONRY,
ELITEARMY, RAW_MATERIALS, CRAFTSMEN, CASTING,
BRICKS, WALL};

duration = [10,10,25,25,10,20,15,20,10,20];
p = 14; % number of precedences
pre =
  [| FUNDS, SOLDIERS          | FUNDS, RAW_MATERIALS
   | FUNDS, CRAFTSMEN         | FUNDS, BRICKS
   | SOLDIERS, WEAPONRY       | SOLDIERS, DEFENSE
   | WEAPONRY, ELITEARMY      | DEFENSE, ELITEARMY
   | RAW_MATERIALS, CASTING   | CRAFTSMEN, CASTING
   | CRAFTSMEN, WALL          | BRICKS, WALL
   | CASTING, WEAPONRY        | WALL, DEFENSE |];
```

16

Constraints & Objective (basic_sched.mzn)

- Constraints

```
constraint 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;
```

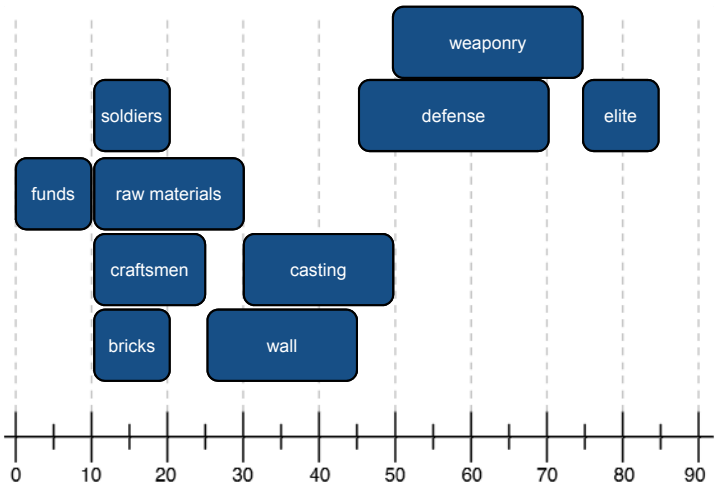
17

Constraints Generated

```
start[FUNDS] + 10 <= start[SOLDIERS]
start[FUNDS] + 10 <= start[RAW_MATERIALS]
start[FUNDS] + 10 <= start[CRAFTSMEN]
start[FUNDS] + 10 <= start[BRICKS]
start[SOLDIERS] + 10 <= start[WEAPONRY]
start[SOLDIERS] + 10 <= start[DEFENSE]
start[WEAPONRY] + 25 <= start[ELITEARMY]
start[DEFENSE] + 25 <= start[ELITEARMY]
start[RAW_MATERIALS] + 20 <= start[CASTING]
start[CRAFTSMEN] + 15 <= start[CASTING]
start[CRAFTSMEN] + 15 <= start[WALL]
start[BRICKS] + 10 <= start[WALL]
start[CASTING] + 20 <= start[WEAPONRY]
start[WALL] + 20 <= start[DEFENSE]
```

18

Basic Scheduling Solution




```
makespan    F   S   D  We   E   R  Cr  Ca   B  Wa
85        = [0, 10, 45, 50, 75, 10, 10, 30, 10, 25]
```

19

Difference Logic Constraints

- Difference logic constraints take the form
  - $x + d \leq y$ d is constant
- Note $x + d = y \leftrightarrow x + d \leq y \wedge y + (-d) \leq x$
- A problem that is representable as a conjunction of difference logic constraints can be solved very rapidly
  - longest/shortest path problem
- But adding extra constraints means this advantage disappears
  - e.g. at most two tasks can run simultaneously

20

Summary

- Basic scheduling problems are a common part of many complex discrete optimisation problems
  - tasks with precedences
- The constraints needed to model this are a simple form of linear constraints
  - difference logic constraints
- Problems involving only these constraints can be solved very efficiently

21

Image Credits

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

22