Basic Scheduling

Peter Stuckey

Basic Scheduling

- Scheduling is an important class of discrete optimisation problems

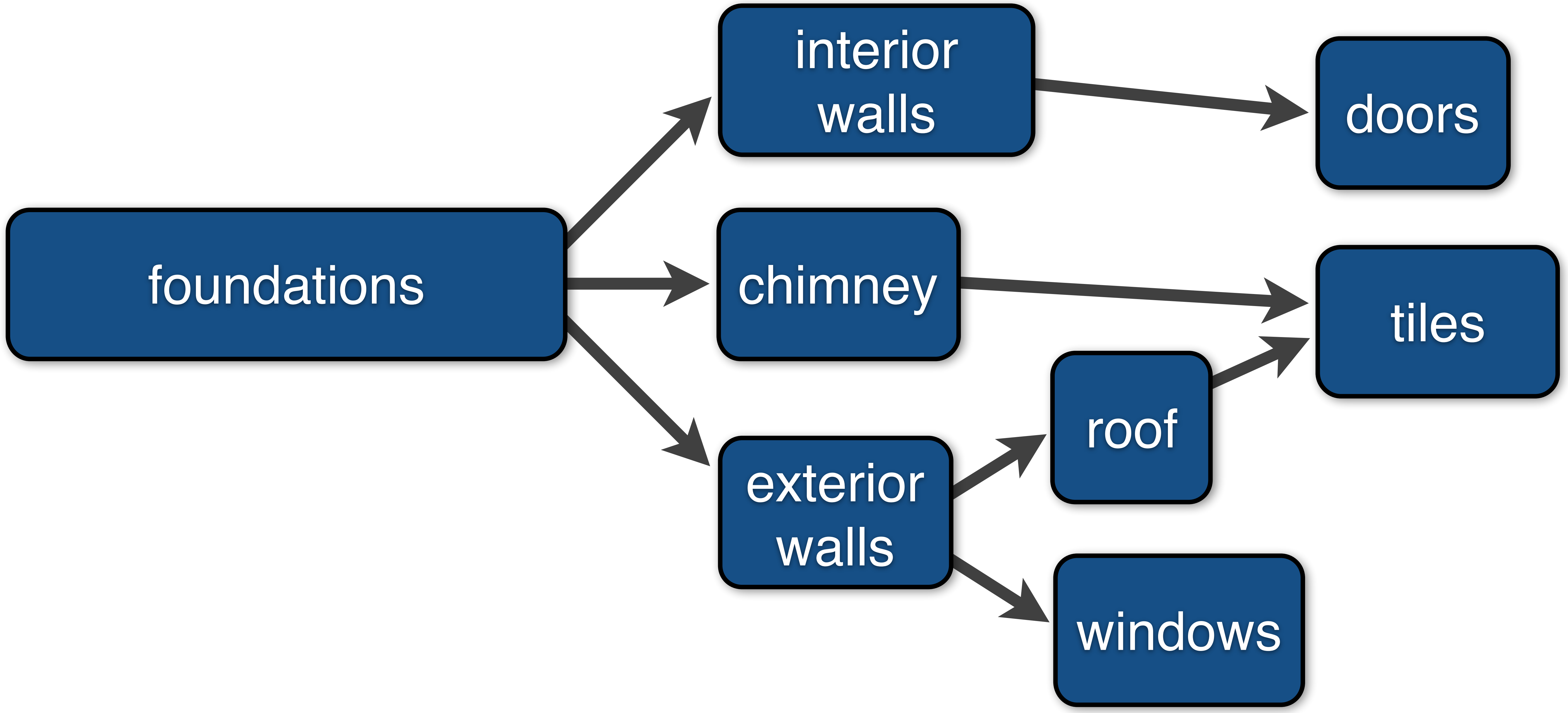
- Basic scheduling involves:
  - tasks with durations
  - precedences between tasks
    - one task must complete before another starts

- The aim is to schedule the tasks
  - usually to minimize the latest end time

Project Scheduling

- Building a house involves a number of tasks, and precedences where one task may not be started until another is completed. Each task has a duration. We need to determine the schedule that minimises the total time to build the house
  - Task (duration): foundations (7), interior walls (4), exterior walls (3), chimney (3), roof (2), doors (2), tiles (3), windows (3).
  - walls and chimney need foundations finished, roof and windows after exterior walls, doors after interior walls, tiles after chimney and roof

Project Scheduling



- Length indicates durations
- Arcs indicate precedences

Project Scheduling

- Data

```
int: n = 8; % no of tasks max
set of int: TASK = 1..n;
int: f = 1; int: iw =2; int: ew = 3;
int: c = 4; int: r = 5; int: d = 6;
int: t = 7; int: w = 8;
array[TASK] of int: duration =
  [7,4,3,3,2,2,3,3];
int: p = 8; % number of precedences
set of int: PREC = 1..p;
array[PREC] of TASK: pre =
  [f,f,f,ew,ew,iw,c,r];
array[PREC] of TASK: post =
  [iw,ew,c,r,w,d,t,t];
```

Project Scheduling

- Decisions

```
int: s = sum(duration);
array[TASK] of var 0..s: start;
```

- Constraints

```
forall(i in PREC)
  (start[pre[i]] + duration[pre[i]]
   <= start[post[i]]);
```

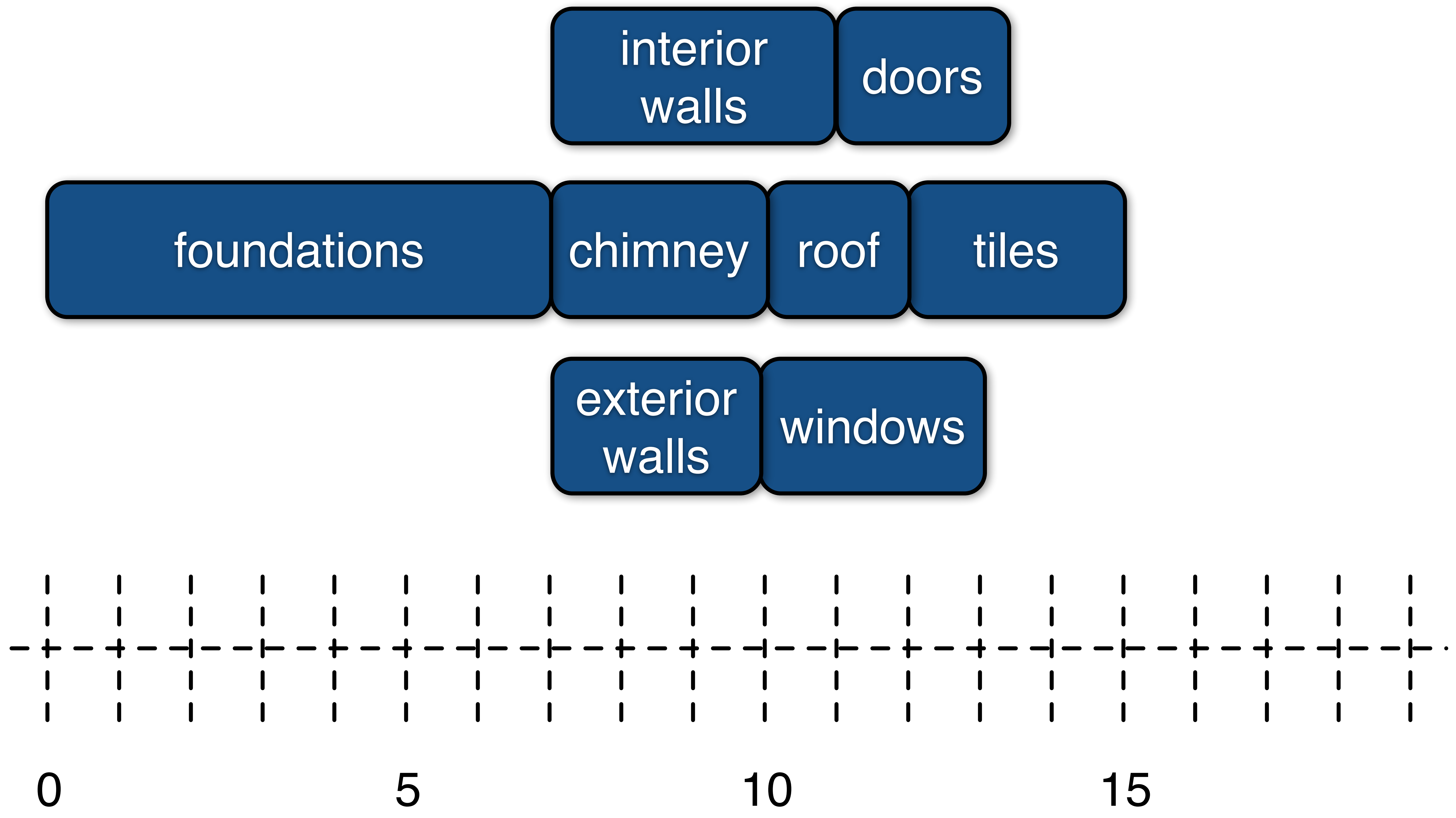
- Objective

```
var 0..s: makespan;
forall(t in TASK)
      (start[t] + duration[t] <= makespan);
solve minimize makespan;
```

Project Scheduling

- Constraints generated
  - s[f] + 7 <= s[iw]
  - s[f] + 7 <= s[ew]
  - s[f] + 7 <= s[c]
  - s[ew] + 3 <= s[r]
  - s[ew] + 3 <= s[w]
  - s[iw] + 4 <= s[d]
  - s[c] + 4 <= s[t]
  - s[r] + 2 <= s[t]
  - s[d] + 2 <= makespan
  - s[t] + 3 <= makespan
  - s[w] + 3 <= makespan

Project Scheduling Solution




```
makespan      f  iw  ew   c   r   d   t   w
15          = [0, 7, 7, 7, 10, 11, 12, 10]
```

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

Overview

- Basic scheduling problems
  – tasks with precedences

  are a common part of many complex discrete optimisation problems

- 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

EOF