Basic Scheduling

Peter Stuckey & Guido Tack

MONASH
University

Basic Scheduling

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

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
- We typically only care about
  - earliest time, or
  - latest time
- when reasoning (not about all possible times)

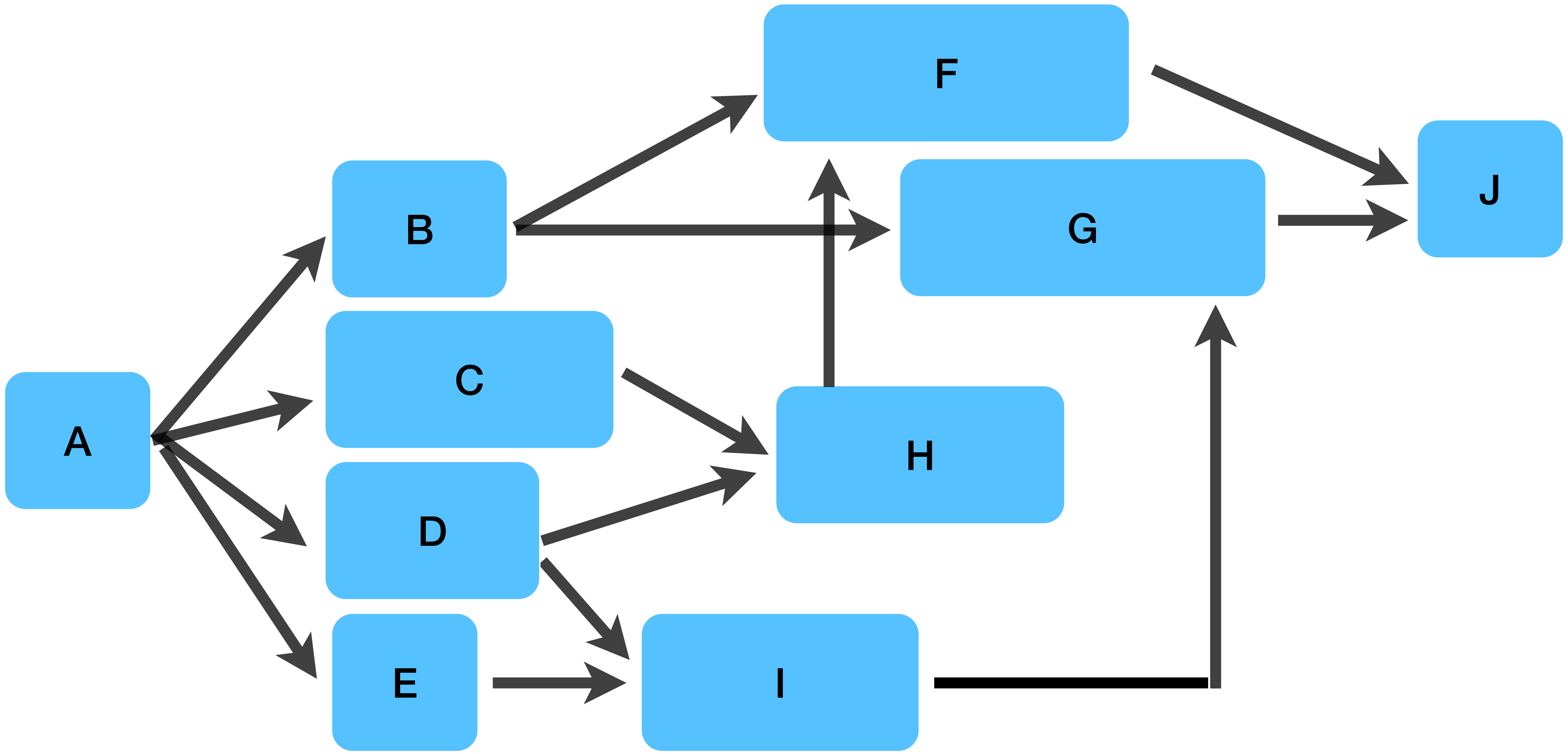
Basic Scheduling Data & Decisions

```
enum TASK;

array[TASK] of int: duration;
array[int,1..2] of TASK: pre;
set of int: PREC = index_set_1of2(pre);

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

Basic Scheduling Instance



- Length indicates durations
- Arcs indicate precedences

Basic Scheduling Data & Decisions

```
TASK = {A, B, C, D, E, F, G, H, I, J};

duration = [10,10,20,15,10,25,20,20,20,10];

pre =
  [| A, B | A, C
   | A, D | A, E
   | B, F | B, G
   | F, J | G, J
   | C, H | D, H
   | D, I | E, I
   | H, F | I, G |];
```

Constraints & Objective

- We can force the precedences of the tasks using

```
constraint forall(i in PREC)
  (start[pre[i,1]] + duration[pre[i,1]]
     <= start[pre[i,2]]);
```

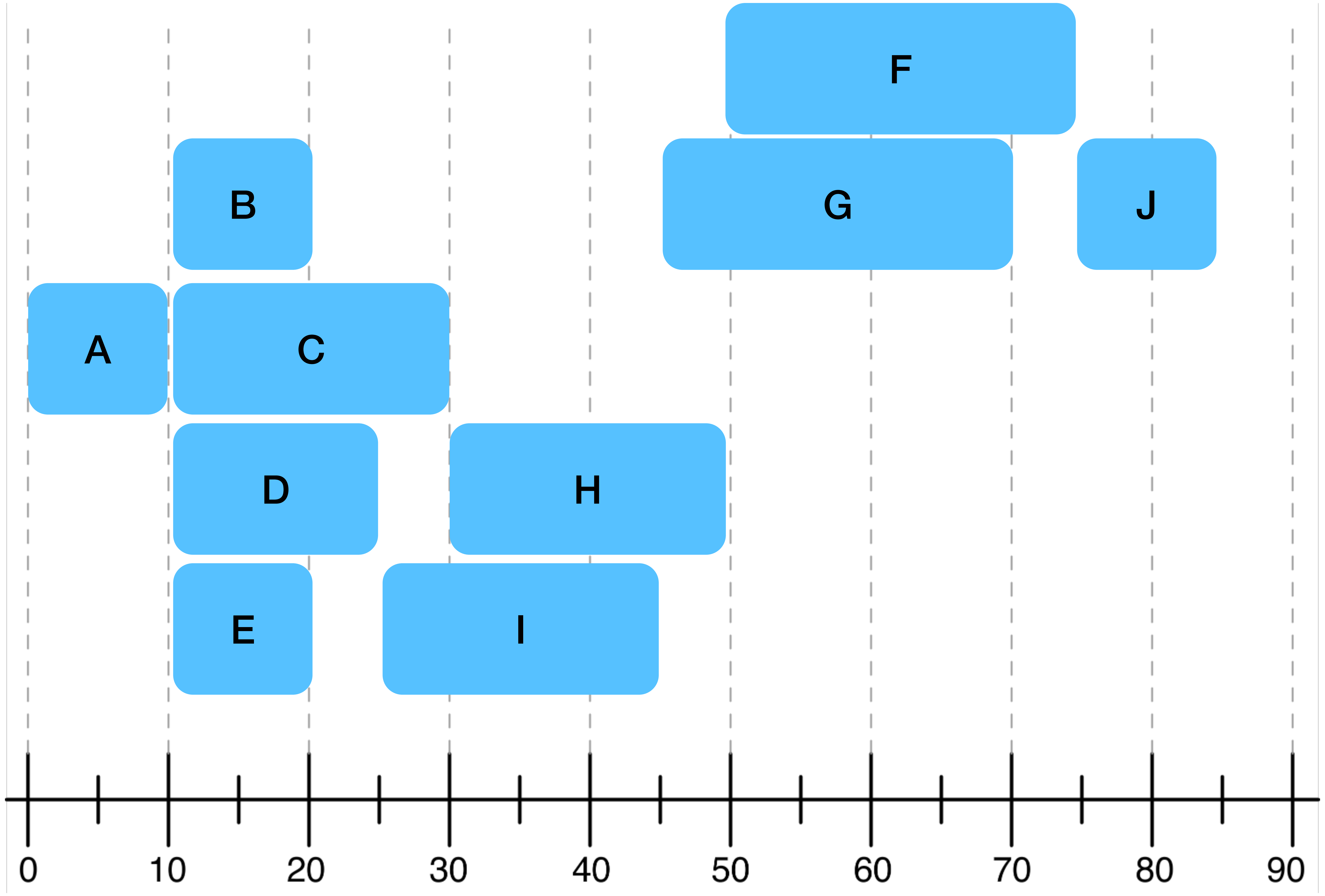
- To minimize the total time taken we use

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

Constraints Generated

```
start[A] + 10 <= start[B]
start[A] + 10 <= start[C]
start[A] + 10 <= start[D]
start[A] + 10 <= start[E]
start[B] + 10 <= start[F]
start[B] + 10 <= start[G]
start[F] + 25 <= start[J]
start[G] + 20 <= start[J]
start[C] + 20 <= start[H]
start[D] + 15 <= start[H]
start[D] + 15 <= start[I]
start[E] + 10 <= start[I]
start[H] + 20 <= start[F]
start[I] + 20 <= start[J]
```

Basic Scheduling Solution



```
makespan     A    B   C    D    E    F    G    H    I    J
85         = [0, 10, 10, 10, 10, 50, 45, 30, 25, 75]
```

Difference Logic Constraints

- **Difference logic constraints** take the form $x + d \leq y$, where d is constant

- Note $x + d = y \leftrightarrow x + d \leq y \wedge y + (-d) \leq x$

- A problem consists solely of difference logic constraints can be rapidly solved as a **longest/shortest path problem**

- But adding extra constraints means this advantage disappears
    - e.g., at most two tasks can run simultaneously

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**

EOF