Introduction to Artificial Intelligence

COMP307
Planning and Scheduling 2: Static Scheduling

Yi Mei

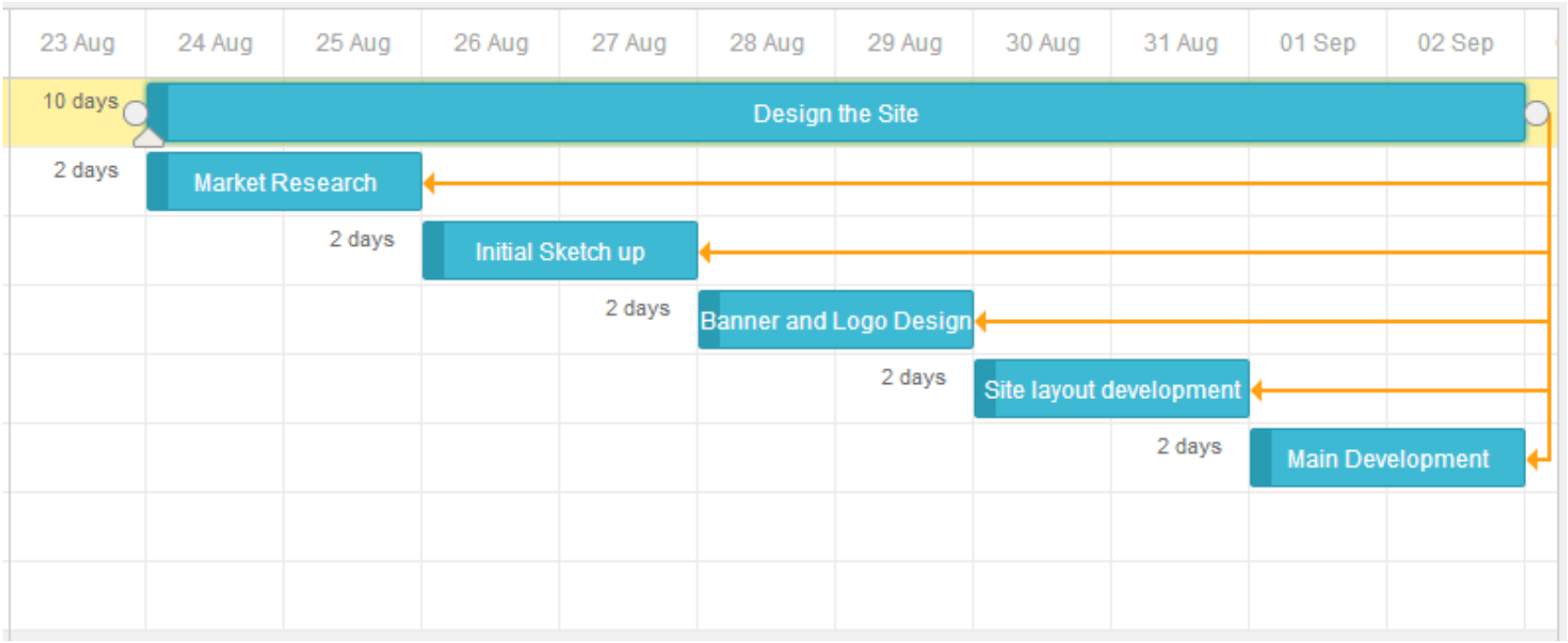
yi.mei@ecs.vuw.ac.nz

Outline

- Why Scheduling?
- What is scheduling?
- Job Shop Scheduling
- Solution as Schedule
- Job Shop Scheduling Algorithms

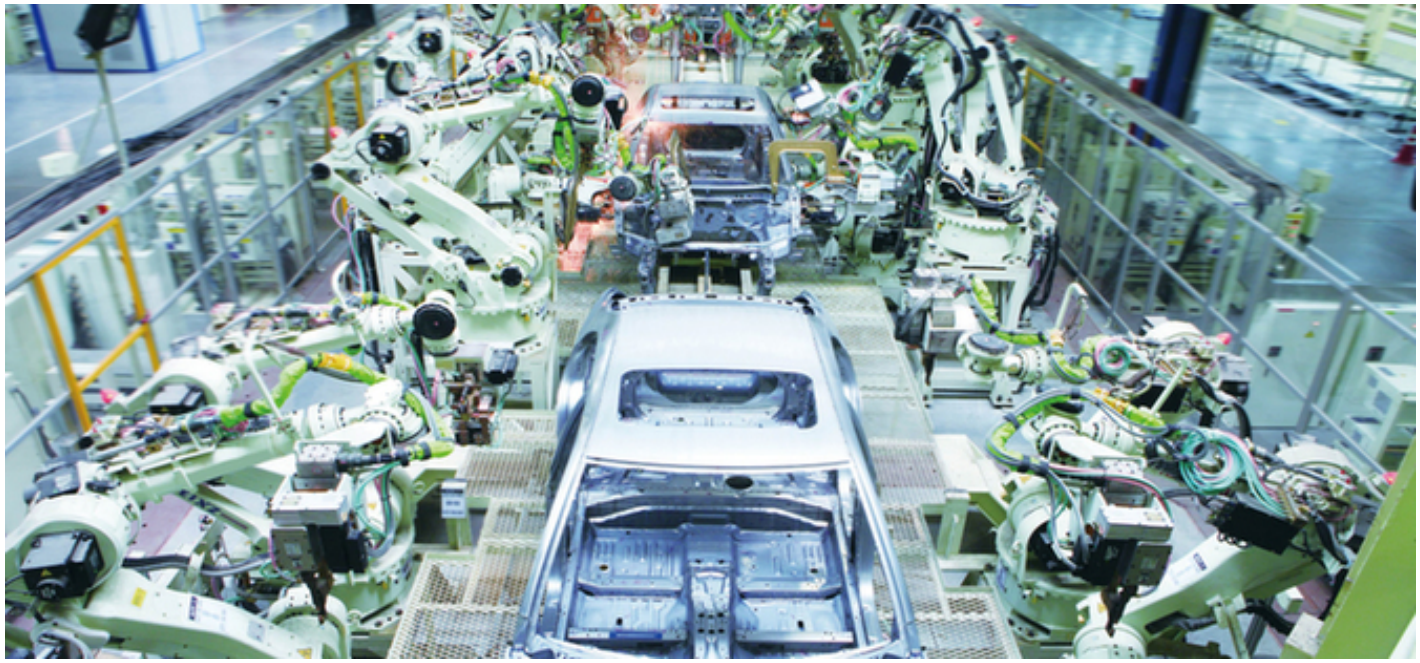
Why Scheduling

- Planning does not consider time
  - States have no time stamp
  - Actions have no starting time
  - Actions have no duration
- Scheduling deals with time in planning



Why Scheduling

- Real-world applications need scheduling
  - Cloud computing/Resource allocation
  - Project management
  - Exam/course timetabling
  - Manufacturing
  - …



What is Scheduling

- Scheduling v.s. Planning
  - Planning is to find a sequence of actions to achieve the goal state from the initial state
  - In Scheduling, each action has a starting and finishing time
  - s' = RESULT(s,a) is reached after the action is finished (it takes time from s to s')

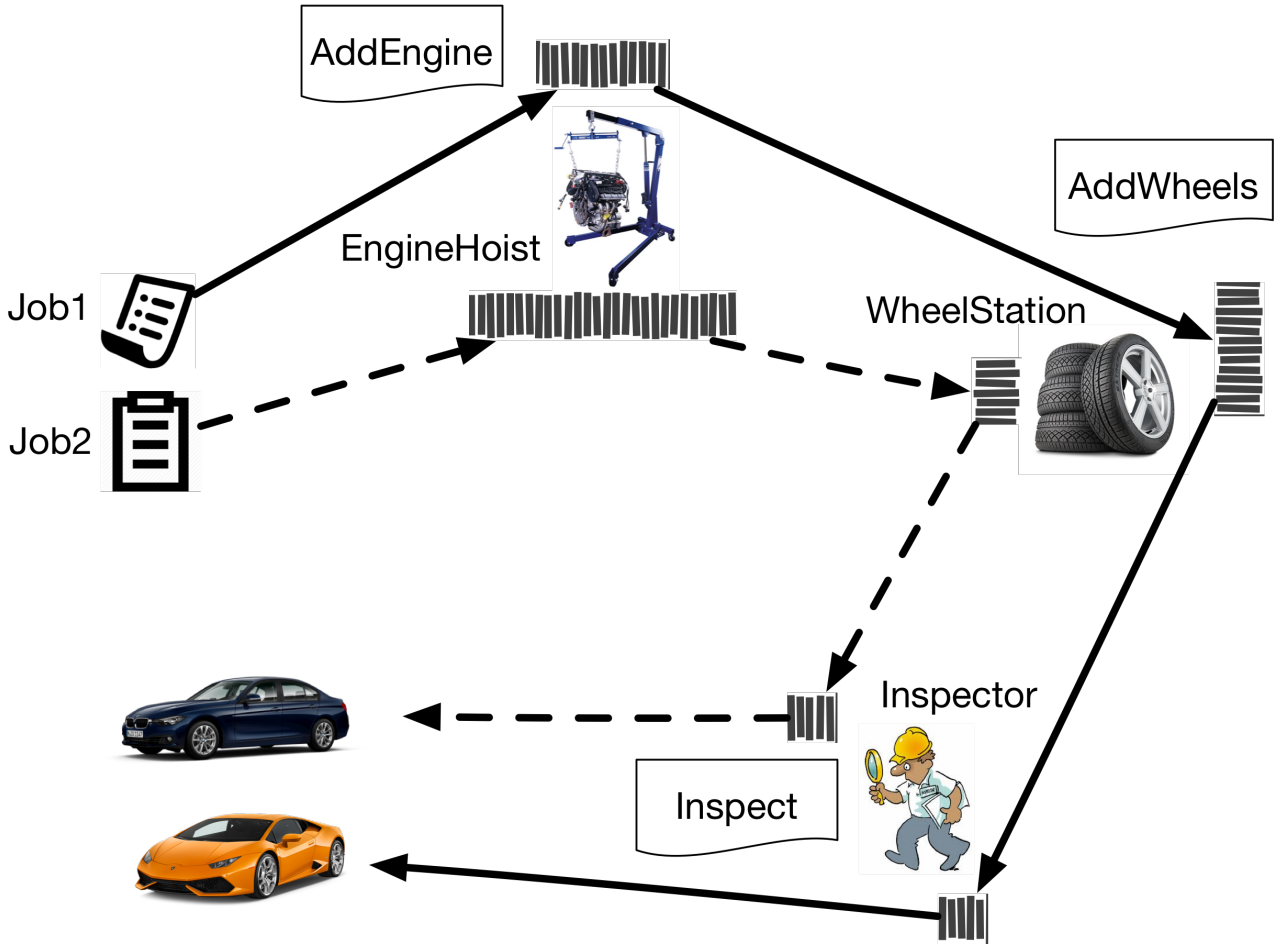
- An example: **Job Shop Scheduling**

Job Shop Scheduling

- A set of jobs to be processed
- Each job has a sequence of operations, following the ordering constraint
  - An operation cannot start until its preceding operations have completed
  - The first operation can start at any time
- Each operation has a processing time, and occupies a machine
  - (Resource constraint) Each machine can process only one operation at a time
- **Objective**: process the operations with the machines to minimise the total duration (makespan) of the plan

JSS: An Example

- A car manufacturing factory
  - Two types of cars
  - Three operations (simplified)
    - Add engine
    - Add wheels
    - Inspect



JSS: Representation

- Jobs({AddEngine1 $\prec$ AddWheels1 $\prec$ Inspect1},
- {AddEngine2 $\prec$ AddWheels2 $\prec$ Inspect2})
- Machines(EngineHoist, WheelStation, Inspector)
- Operation(AddEngine1, ProcTime: 30, Use: EngineHoist)
- Operation(AddEngine2, ProcTime: 60, Use: EngineHoist)
- Operation(AddWheels1, ProcTime: 30, Use: WheelStation)
- Operation(AddWheels2, ProcTime: 15, Use: WheelStation)
- Operation(Inspect_i, ProcTime: 10, Use: Inspector)

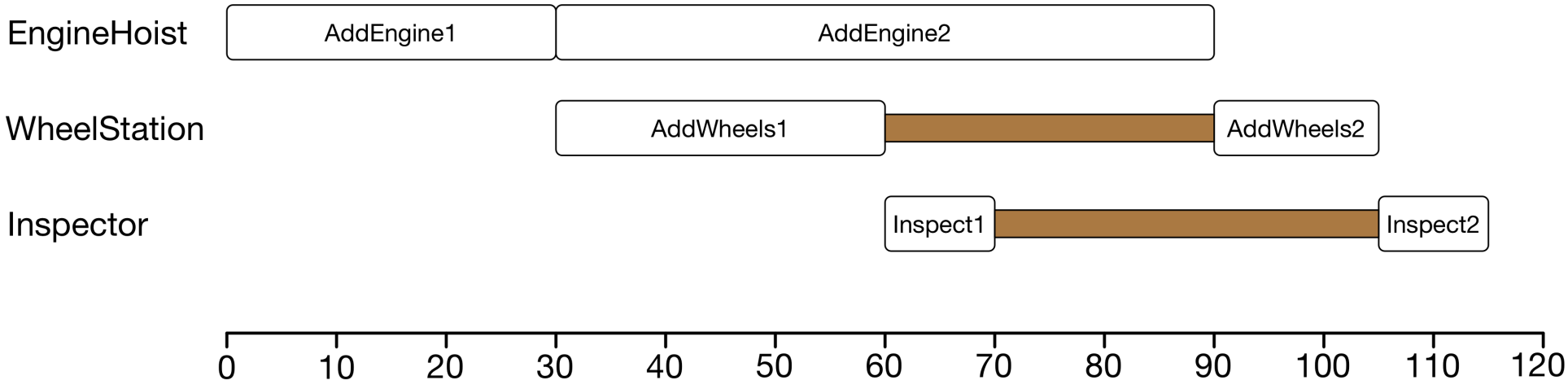
JSS: Table

Job	Operation	Machine	ProcTime
1	AddEngine1	EngineHoist	30
	AddWheels1	WheelStation	30
	Inspect1	Inspector	10
2	AddEngine2	EngineHoist	60
	AddWheels2	WheelStation	15
	Inspect2	Inspector	10

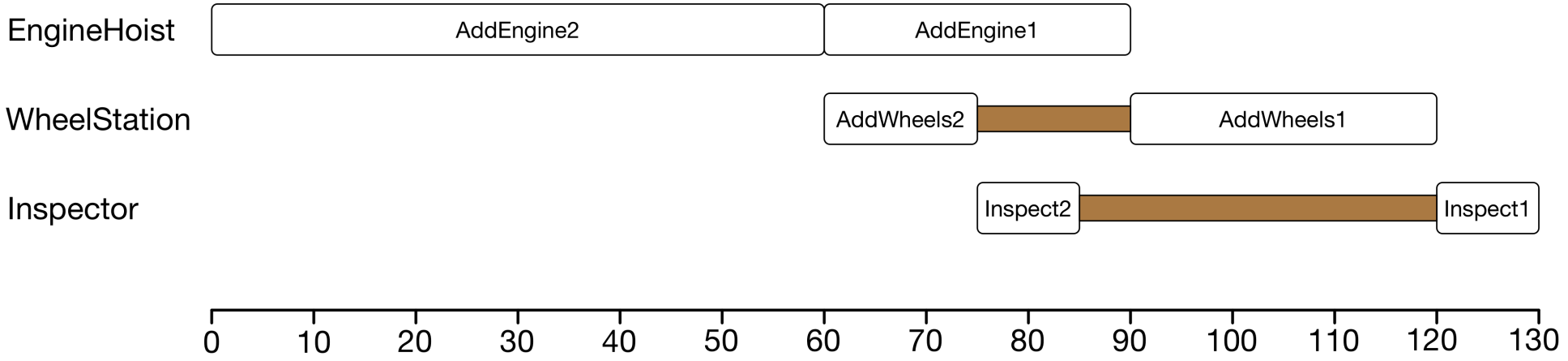
- A solution is a schedule that processes these jobs with the machines (gantt chart)

Two different solutions

- Job 1 first: total duration = 115



- Job 2 first: total duration = 130



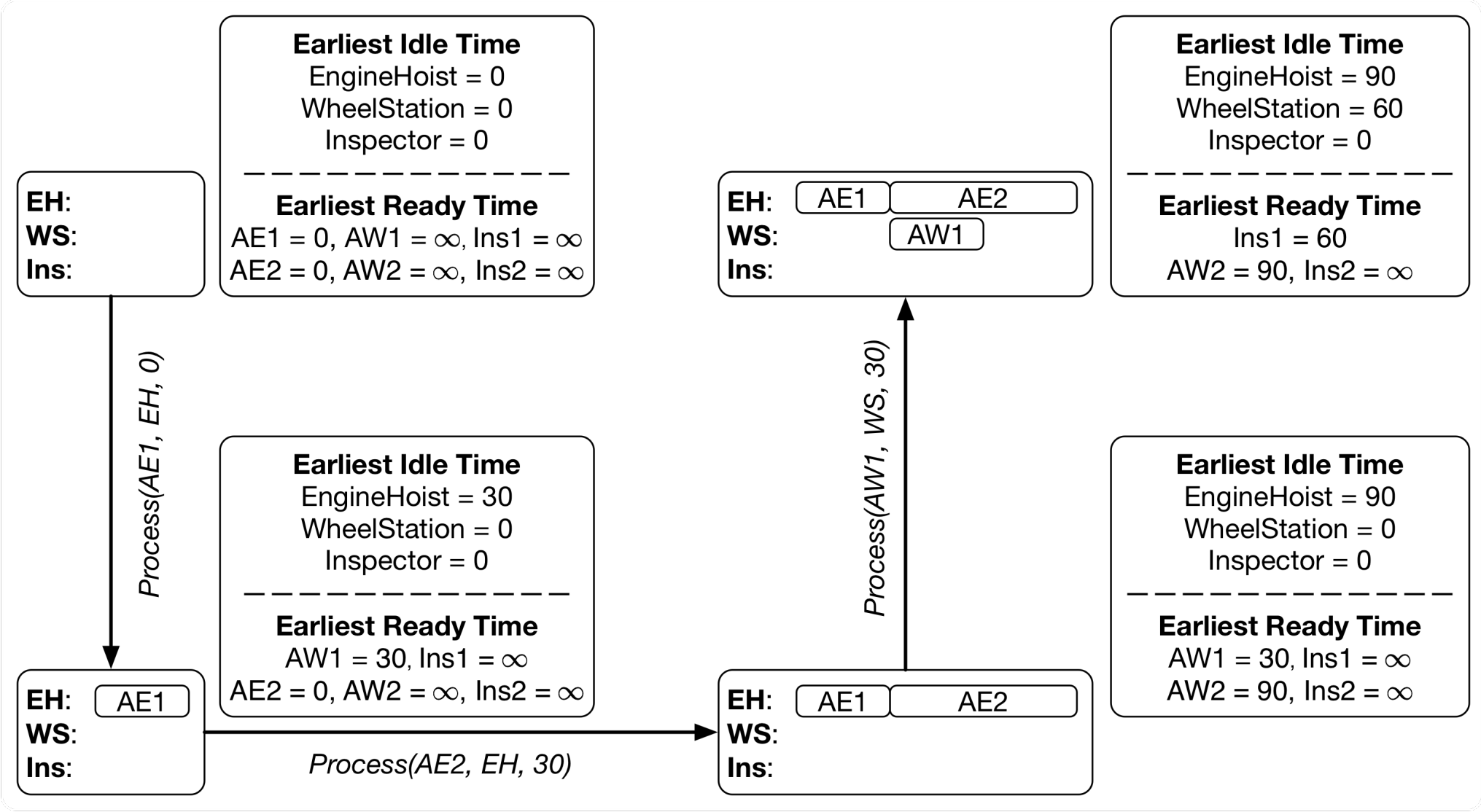
Search for Schedules

- Initial state
  - Empty schedule, t=0, all operations unprocessed
  - The first operation of each job is ready at time 0, all the other operations are not ready
  - All machines are idle at time 0
- Goal state: all operations processed
- Actions: Process(o, m, t)
  - Start processing operation o with machine m at time t
  - Precondition:
    - o unprocessed, and is ready at time t
    - m is idle at t
  - Effect:
    - o processed
    - next(o) (if exists) is ready at time t+ProcTime(o), and is idle at t+ProcTime(o)
- How to decide t?

Deciding Starting Time of Action

- **Non-delay**: start the action as soon as possible
  - Operation earliest ready time
  - Machine earliest idle time
  - Earliest starting time: the later between the above two
- Find operation earliest ready time
  - Initial: 0 for the first operation, and infinity for others
  - When Process(o,m,t) is scheduled, then the earliest ready time of next(o) becomes t+ProcTime(o)
- Find machine earliest idle time
  - Initial: 0
  - When Process(o,m,t) is scheduled, then the earliest idle time of machine m becomes t+ProcTime(o)

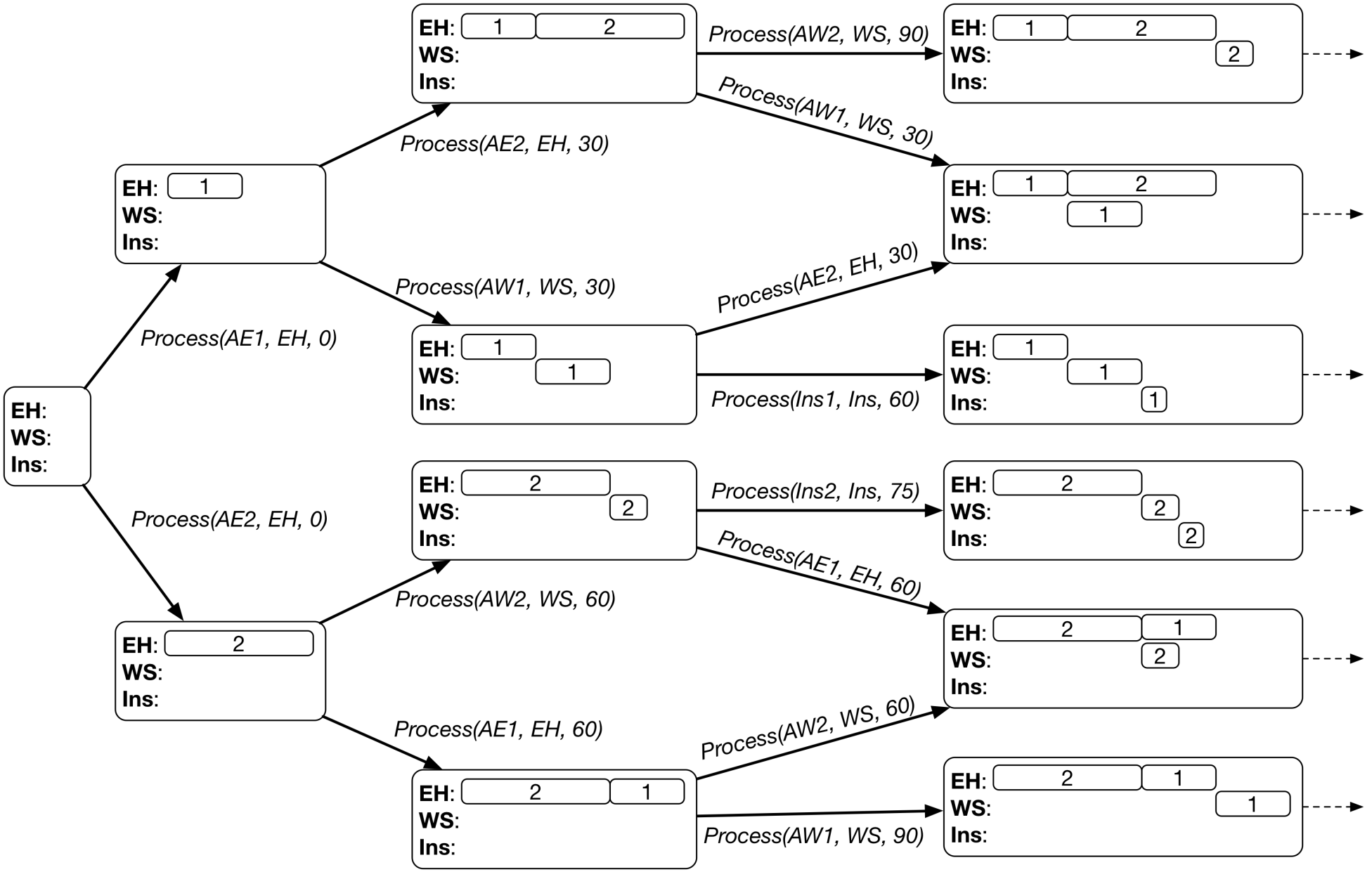
Update Earliest Ready and Idle Time



Forward State-Space Search

- Start from the initial state
  - Empty schedule, t=0, all operations unprocessed
  - The first operation of each job is ready at time 0, all the other operations are not ready
  - All machines are idle at time 0
- Examine all the applicable actions Process(o,m,t)
  - Enumerate each unprocessed operation o and its machine m
  - Calculate the earliest starting time t
  - Applicable if $t < \infty$
- All the leaf nodes are goal states (all operations processed)
- Each schedule is a path from the root node to a leaf node

Forward State-Space Search



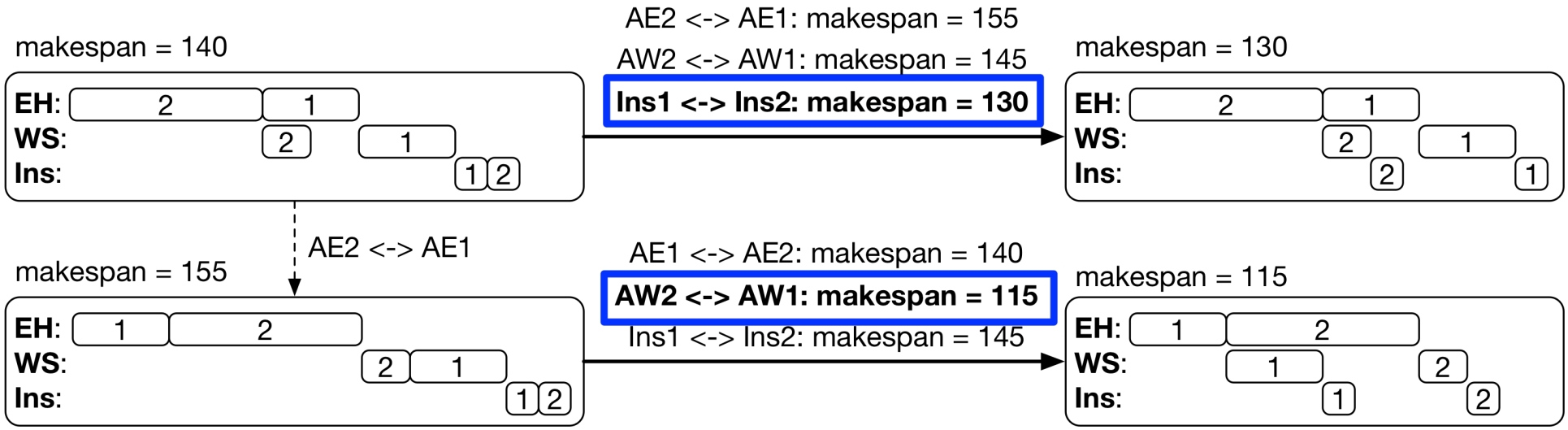
Local Search (Hill Climbing)

- **Step 1**. Random generate a scheduling s;
- **Step 2**. Examine all the neighbours in the neighbourhood of s, and select the best neighbour s';
- **Step 3**. If s' is better than s, set s ← s', and go to Step 2. Otherwise return s.

- How to define the neighbourhood?
  - Local/Subtle modification
  - Still feasible
  - An example: swap operations on the same machine
  - **Shift the starting time of all actions**

Local Search (Hill Climbing)

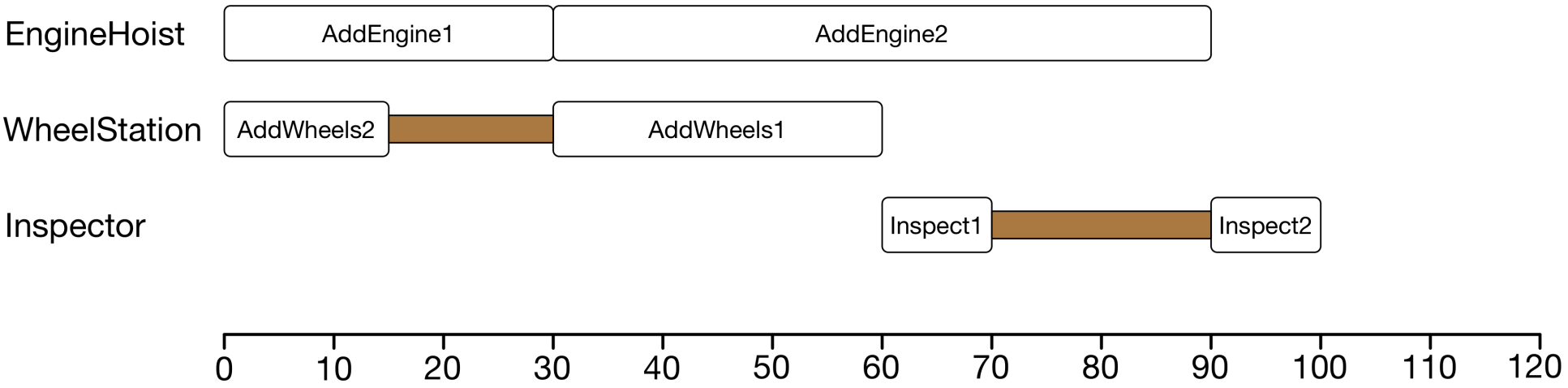
- Step 1. Random generate a scheduling s;
- Step 2. Examine all the neighbours in the neighbourhood of s, and select the best neighbour s';
- Step 3. If s' is better than s, set s ← s', and go to Step 2. Otherwise return s.



- Jump out of local optima: simulated annealing, genetic algorithms, …

Jobs with Different Sequences

Job	Operation	Machine	ProcTime
1	AddEngine1	EngineHoist	30
	AddWheels1	WheelStation	30
	Inspect1	Inspector	10
2	AddWheels2	WheelStation	15
	AddEngine2	EngineHoist	60
	Inspect2	Inspector	10



Summary

- What is Scheduling? – Temporal planning (actions have starting/finishing time)
- Solution as schedule
- Search for schedules: state-space search, local search, …

- Suggested Reading: Textbook, Chapter 11

- Next: Dynamic scheduling