Planning and Scheduling with Time and Resources

Section 11.1

Outline

- Scheduling problems vs. planning problems
- Scheduling with time constraints
- Scheduling with resource constraints

Additional references used for the slides:

Smith, D.E, Frank, J. and Jonsson, A. K. (2000). Bridging the Gap Between Planning and Scheduling. *Knowledge Engineering Review*, 15(1).

Kambhampati, S. (2000). AI Planning tutorial notes. *AAAI-2000*.

Planning vs. scheduling

- Planning
  - Involves choice of actions
  - Cannot deal with time and resource constraints
- Scheduling
  - Can easily represent time and resource constraints
  - Cannot deal with action choices
- Most real world problems are optimization problems that involve continuous time, resources, metric quantities, and a complex mixture of action choices and ordering decisions.

Planning vs. scheduling

Planning problem	Scheduling problem
Initial state, goals	set of jobs (possibly partially ordered)
action descriptions	temporal constraints on jobs (EST, LFT, duration) resource constraints
Synthesize a sequence of actions	Assign optimal start times and resources

Dealing with time

- *EST*: earliest start time
- *LFT*: latest finish time
- *duration*
- *CPM*: critical path method. A *path* is a sequence of actions that depend on each other. A *critical path* is the longest path. Delaying it would delay the entire plan.

Example

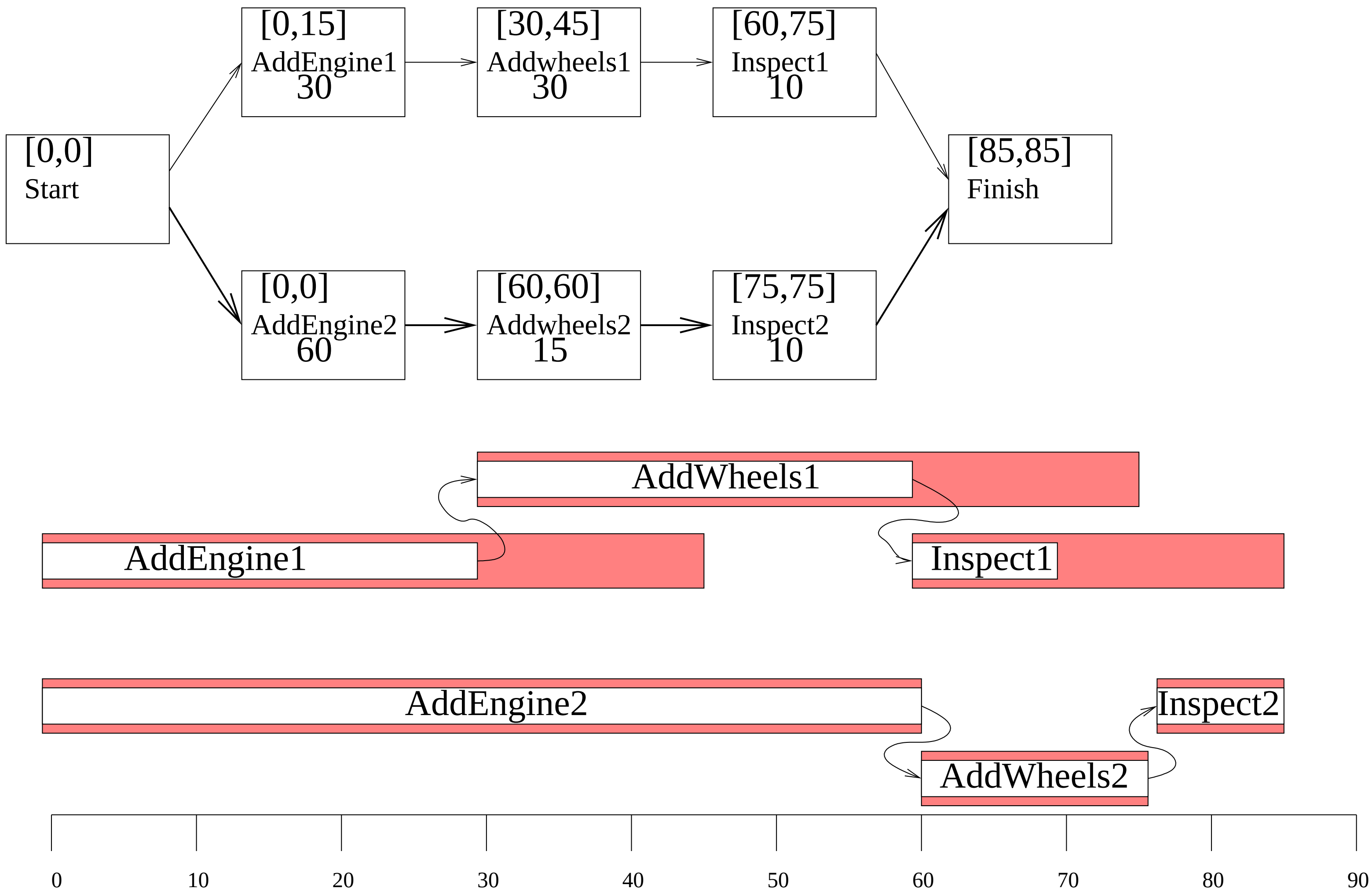
Init (Chassis(C_1) ∧ Chassis(C_2) ∧
Engine(E_1,C_1,30) ∧ Engine(E_2,C_2,60) ∧
Wheels(W_1,C_1,30) ∧ Wheels(W_2,C_2,15))
Goal(Done(C_1) ∧ Done(C_2))

Action(AddEngine(e,c),
PRECOND: Engine(e,c,d) ∧ Chassis(c) ∧¬ EngineIn(c)
EFFECT: EngineIn(c) ∧ Duration(d))

Action(AddWheels(w,c),
PRECOND: Wheels(w,c,d) ∧ Chassis(c) ∧ EngineIn(c)
EFFECT: WheelsOn(c) ∧ Duration(d))

Action(Inspect(c),
PRECOND: EngineIn(c) ∧ WheelsOn(c) ∧ Chassis(c)
EFFECT: Done(c) ∧ Duration(10))

Example



[0,0]
Start
[0,15]
AddEngine1
30
[30,45]
Addwheels1
30
[60,75]
Inspect1
10
[85,85]
Finish
[0,0]
AddEngine2
60
[60,60]
Addwheels2
15
[75,75]
Inspect2
10
AddWheels1
AddEngine1
Inspect1
AddEngine2
Inspect2
AddWheels2
0
10
20
30
40
50
60
70
80
90

Dealing with resources

- *reusable resource*: is occupied during an action, and is freed afterwards
- *aggregation of resources*: group indistinguishable resources into quantities
- *Minimum slack algorithm*: a greedy algorithm

Example

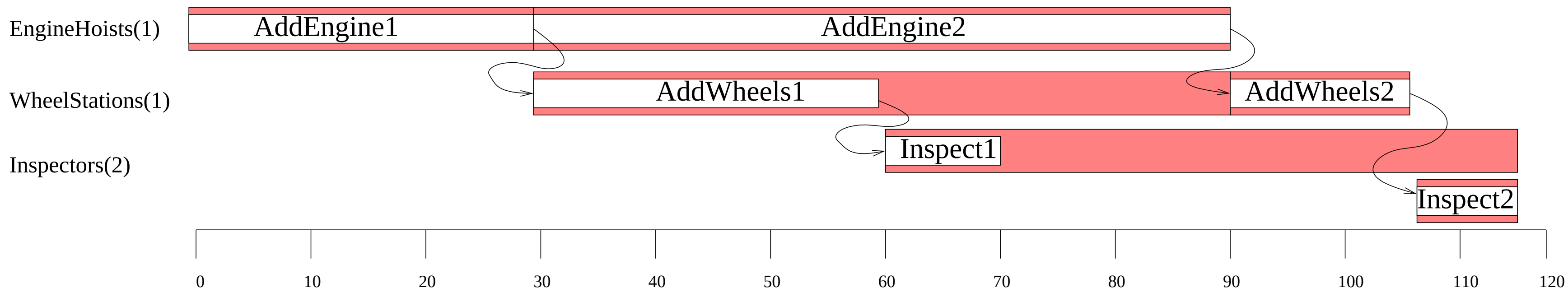
Init (Chassis(C_1) ∧ Chassis(C_2) ∧ Engine(E_1,C_1,30) ∧
Engine(E_2,C_2,60) ∧ Wheels(W_1,C_1,30) ∧ Wheels(W_2,C_2,15) ∧
EngineHoists(1) ∧ WheelStations(1) ∧ Inspectors(2))
Goal(Done(C_1) ∧ Done(C_2))

Action(AddEngine(e,c),
PRECOND: Engine(e,c,d) ∧ Chassis(c) ∧¬ EngineIn(c)
EFFECT: EngineIn(c) ∧ Duration(d)
RESOURCE: EngineHoists(1))

Action(AddWheels(w,c),
PRECOND: Wheels(w,c,d) ∧ Chassis(c) ∧ EngineIn(c)
EFFECT: WheelsOn(c) ∧ Duration(d)
RESOURCE: WheelStations(1))

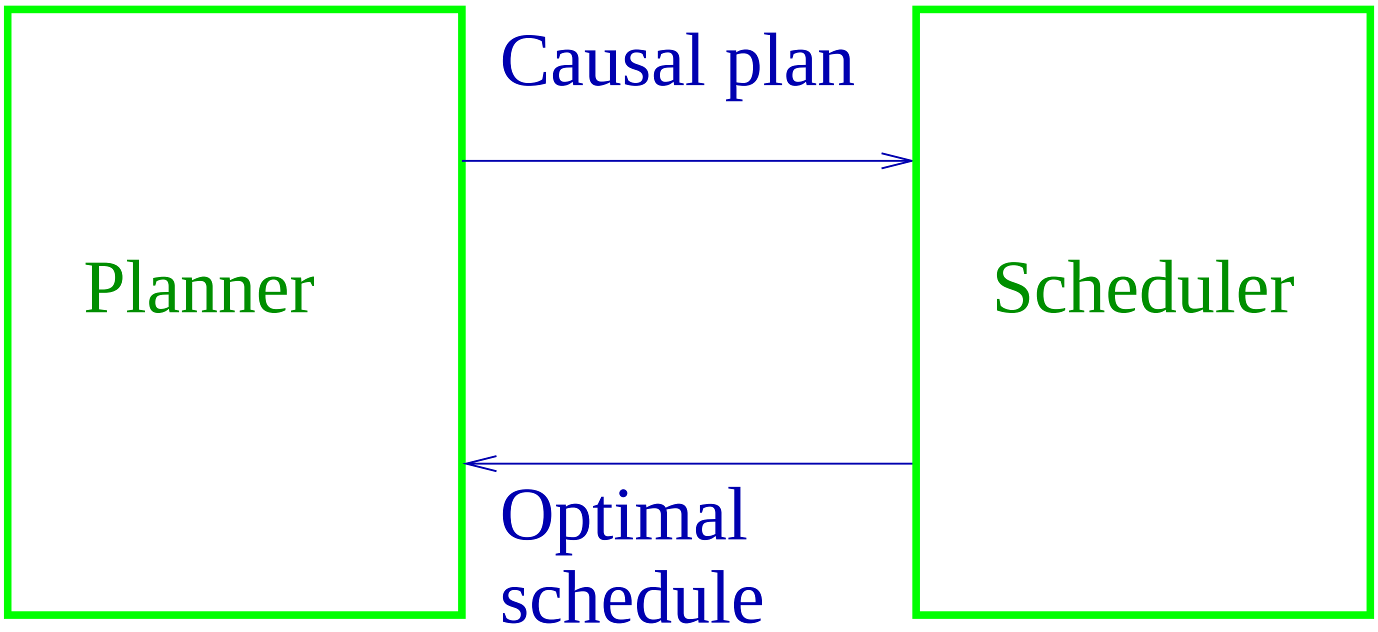
Action(Inspect(c),
PRECOND: EngineIn(c) ∧ WheelsOn(c) ∧ Chassis(c)
EFFECT: Done(c) ∧ Duration(10)
RESOURCE: Inspectors(1))

Example



EngineHoists(1)
WheelStations(1)
Inspectors(2)
AddEngine1
AddEngine2
AddWheels1
AddWheels2
Inspect1
Inspect2
0
10
20
30
40
50
60
70
80
90
100
110
120

Planner-scheduler interface



Each can do its own job. The big question is how best to couple them to avoid inter-module trashing.

The second big question is which planners are most suitable for coupling.