The need for continuous planning, real-time planning, non-disruptive planning, etc

by Geoffrey De Smet

OptaPlanner lead

"Plans are of little importance, but planning is essential."

Winston Churchill
(Prime Minister UK 1940-1945)

"Everybody has a plan until they get punched in the mouth…"

Mike Tyson
(Boxer, heavyweight champion 1987-1990)

What kind of planning disruptions?

- Local disruption
  - Impact limited to one department/region
- Global disruption
  - Impact across the board

Local disruption

Global disruptions

- 2001 - September 11 attacks
  - Air-traffic shut-down in North America for 2 days

Global disruptions

- 2001 - September 11 attacks
  - Air-traffic shut-down in North America for 2 days
- 2010 - Eyjafjallajökull volcano eruption
  - Air-traffic shut-down in Europe for 8 days

Global disruptions

- 2001 - September 11 attacks
  - Air-traffic shut-down in North America for 2 days
- 2010 - Eyjafjallajökull volcano eruption
  - Air-traffic shut-down in Europe for 8 days
- 2020 - COVID-19
  - Flight cancellations

Planning agility paradox

"We can *not* automate planning
because the plans will change."

Planning agility paradox

"We can *not* automate planning
because the plans will change."

⇒

"We *need* to automate (re)planning
because the plans will change."

Planning agility paradox

"We can *not* automate planning
because the plans will change."

⇒

"We *need* to automate (re)planning
because the plans will change."

But how?

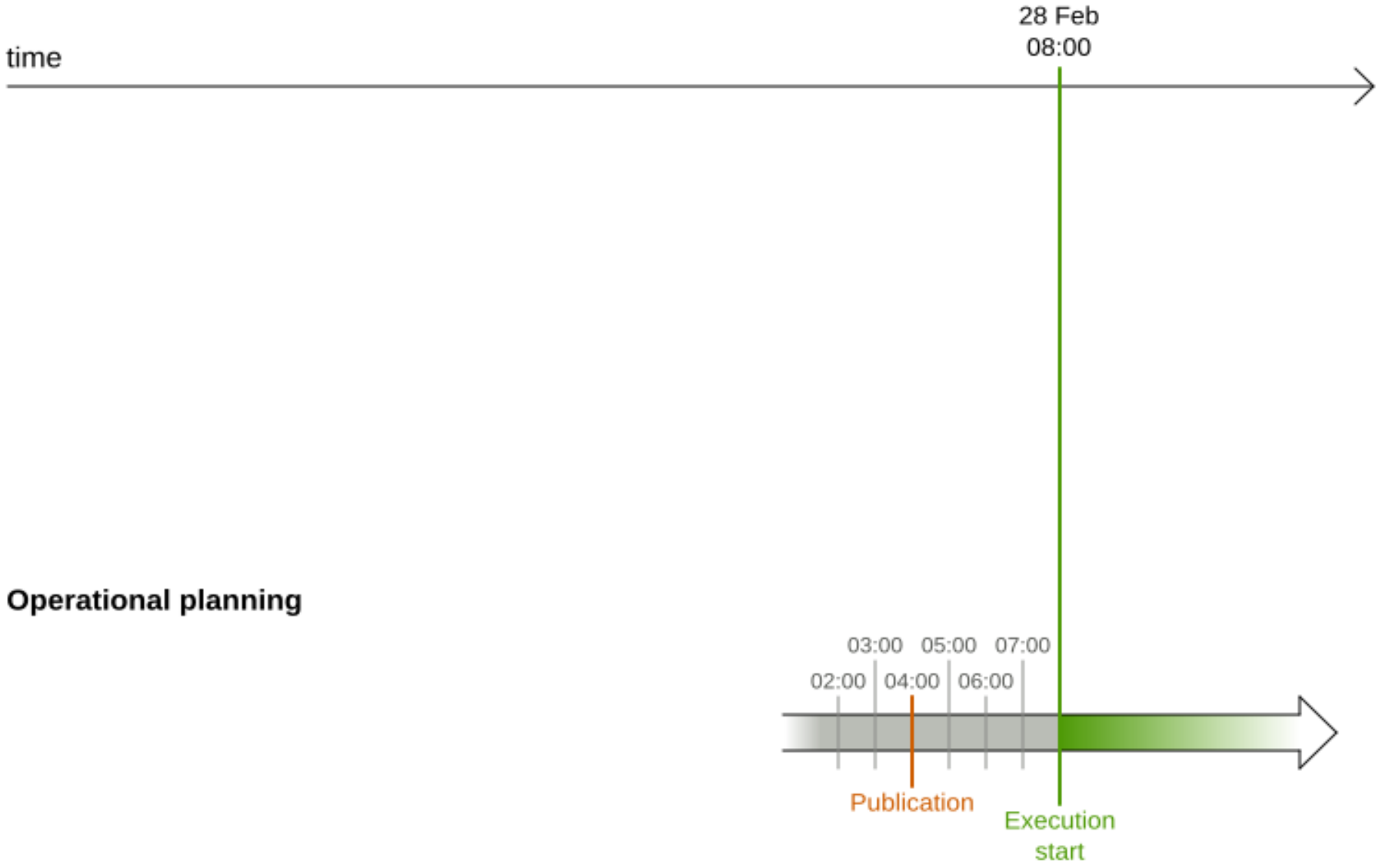
1) What is your publication deadline?

Publication

- Tell employees when/how/where to work
- Publication deadline
  - Technician routing: 1 hour before departure
  - School timetabling: before end of vacation
  - Guards: 4 weeks before shift
  - Nurses: 6 weeks before shift
  - Facility location: 6 months before opening

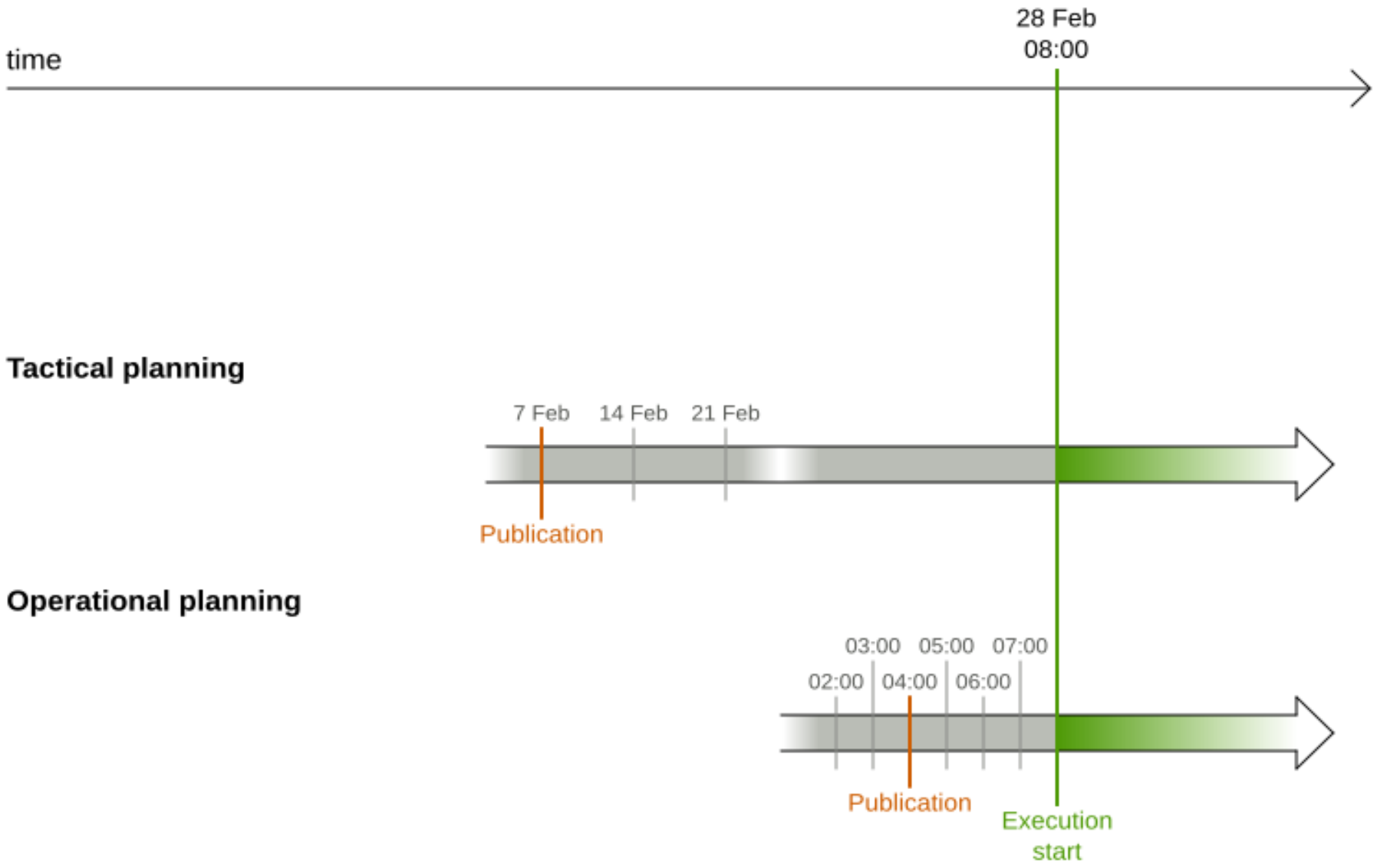
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



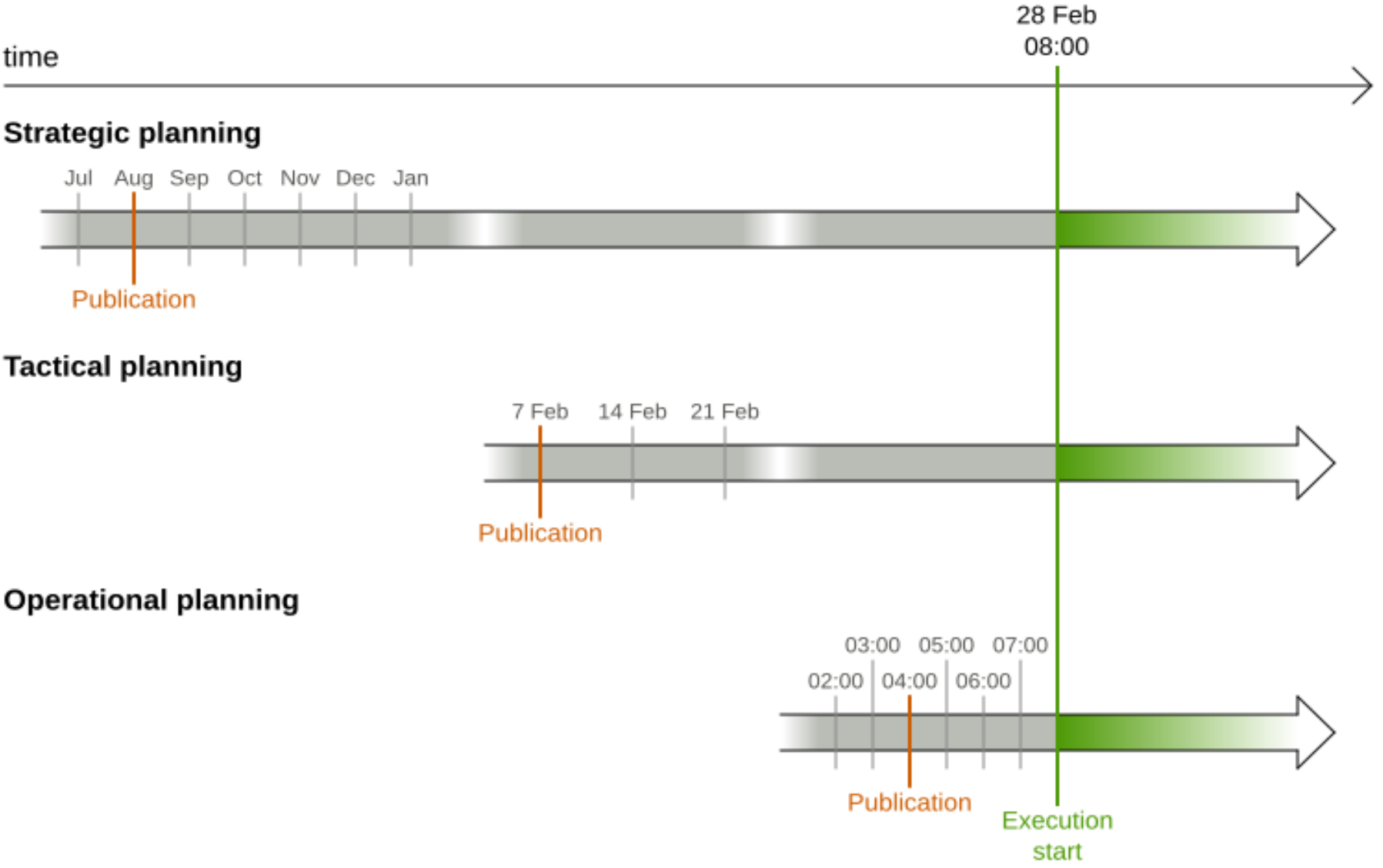
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



Multi-stage planning

- Strategic: expand the maternity ward?
- Tactical: hire a respiratory specialist?
- Operational: assign a Monday shift to Ann?

Each planning stage feeds into the next.

Multi-stage planning

- Strategic: expand the maternity ward?
- Tactical: hire a respiratory specialist?
- Operational: assign a Monday shift to Ann?

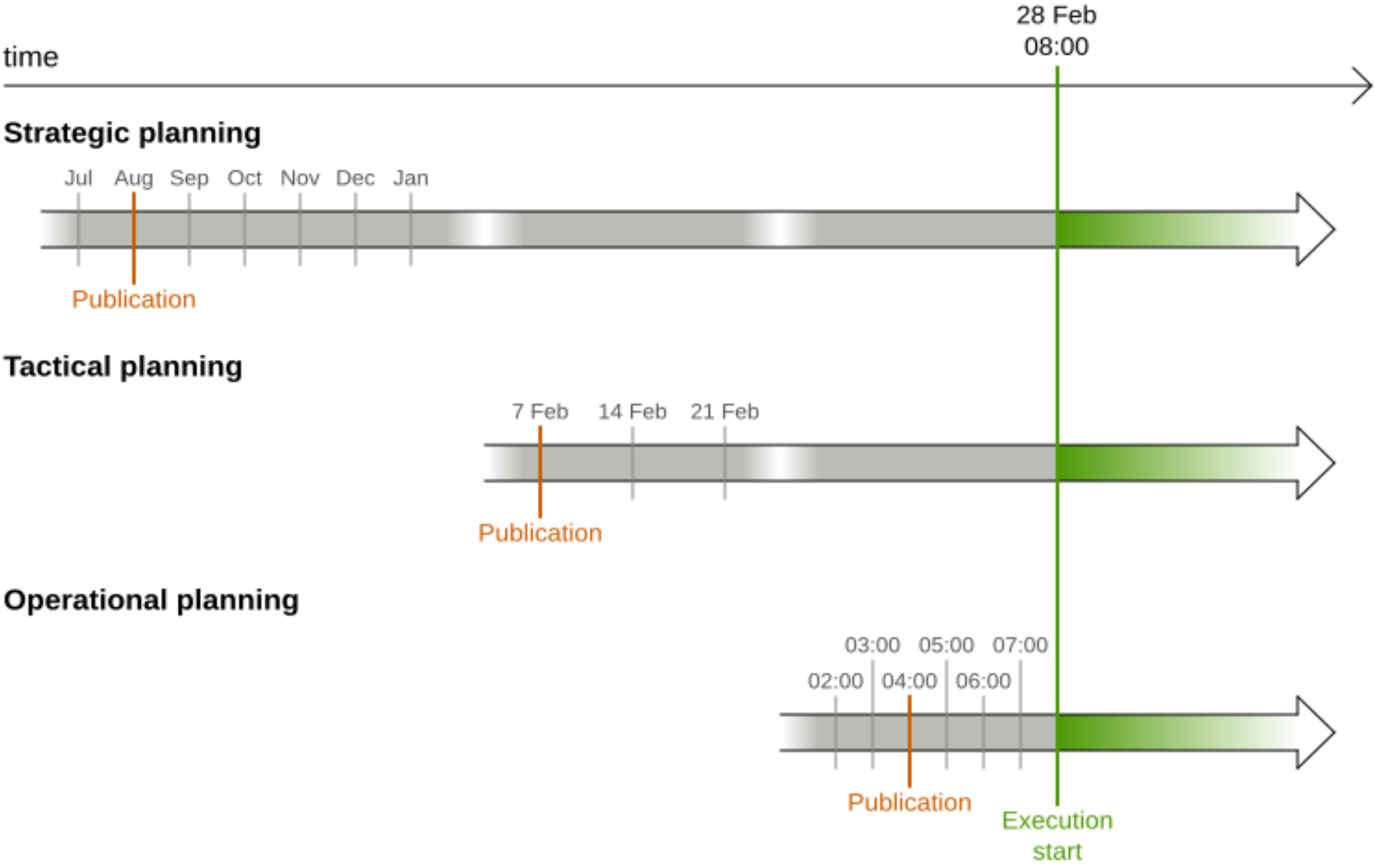
Each planning stage feeds into the next.

Especially for trains, airplanes, etc

- Strategic: add a stop in Philadelphia?
- Tactical: depart at 7 AM?
- Operational: Which locomotive and wagons?

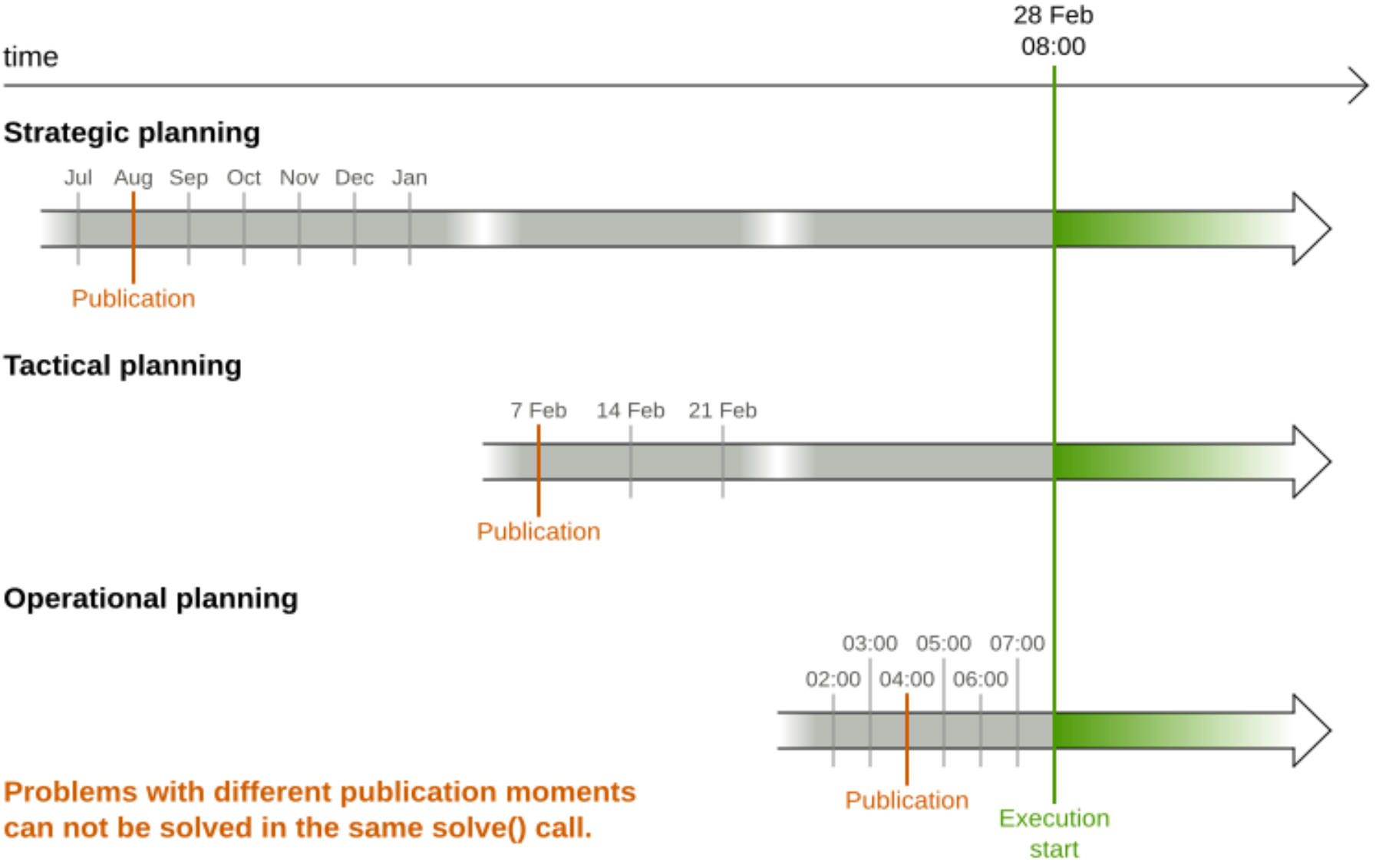
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



Problems with different publication moments can not be solved in the same solve() call.

Conway's law

"Any organization that designs a (software) system
will produce a design whose structure
is a copy of the organization's communication structure."

Conway's law

"Any organization that designs a (software) system
will produce a design whose structure
is a copy of the organization's communication structure."

⇒

Planning problems solved by different groups
should use different Solver instances
(during the first year in production).

In practice

```
@Path("/trainWagon")
public class TrainWagonDeciderResource {

  @Inject
  SolverManager<TrainWagonSolution, ...> wagonSolverManager;

  ... wagonSolverManager.solve(...)
}
@Path("/trainPlatform")
public class TrainPlatformDeciderResource {

  @Inject
  SolverManager<TrainCrewSolution, ...> platformSolverManager;

  // Uses TrainWagonSolution's output as problem facts:
  // Length of train impacts eligble platforms
  ... platformSolverManager.solve(...)
}
```

2) Do you assign to time?

No, domain lacks time data

No `java.time` import

Yes, Domain has time but planning does not assign time

```
import java.time.LocalDate;

@PlanningEntity
public class Patient {

  private String name;
  private LocalDate arrivalDate;
  private LocalDate departureDate;
  ...
  @PlanningVariable(...)
  private Bed bed;
  ...
}
```

Hospital bed planning does not change a patient's arrival or departure date.

By the way...

Don't use `java.util.Date` for time manipulation.
It's like asking your bartender how to treat cancer.

By the way...

Don't use `java.util.Date` for time manipulation.
It's like asking your bartender how to treat cancer.

Don't use `java.util.Calender` either,
It's like asking your dog how to treat cancer.

By the way...

Don't use `java.util.Date` for time manipulation.
It's like asking your bartender how to treat cancer.

Don't use `java.util.Calender` either,
It's like asking your dog how to treat cancer.

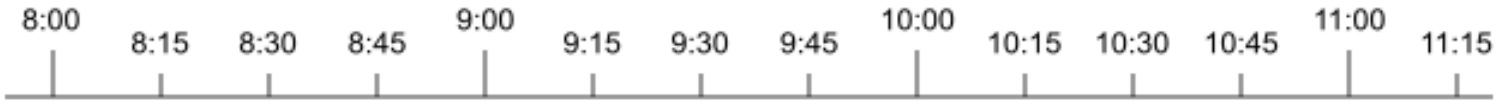
Use `java.time.LocalDate` or
`java.time.LocalDateTime` instead!

Yes, Domain has time and planning assigns to time

Pick a good domain model.

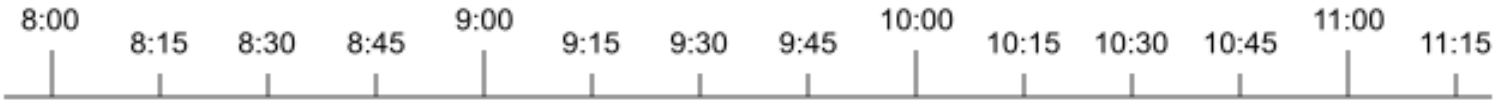
Assigning time to planning entities 1/2

There are several design patterns to deal with time, depending on your use case.

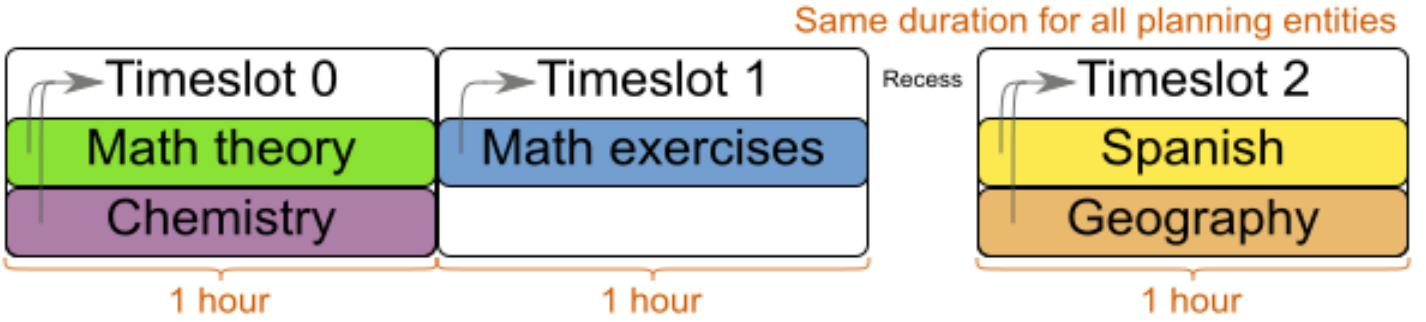


Assigning time to planning entities 1/2

There are several design patterns to deal with time, depending on your use case.



Timeslot pattern

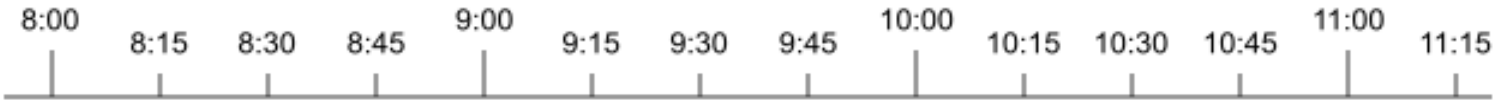


Planning assigns to time

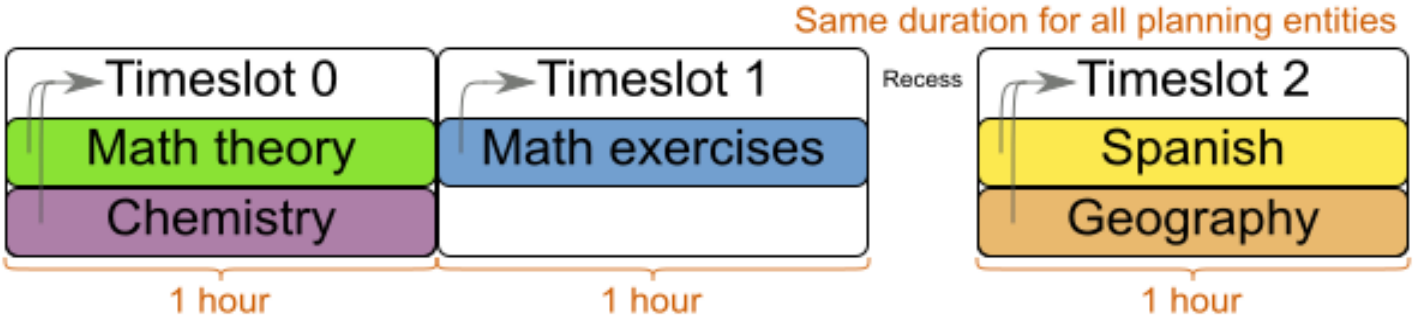
```
@PlanningEntity
public class Lesson {

  private String subject;
  ...
  @PlanningVariable(...)
  private Timeslot timeslot;
  ...
}

import java.time.DayOfWeek;
import java.time.LocalTime;

public class Timeslot {

  private DayOfWeek dayOfWeek;
  private LocalTime startTime;
  private LocalTime endTime;
  ...
}
```

Assigning time to planning entities 1/2

There are several design patterns to deal with time, depending on your use case.

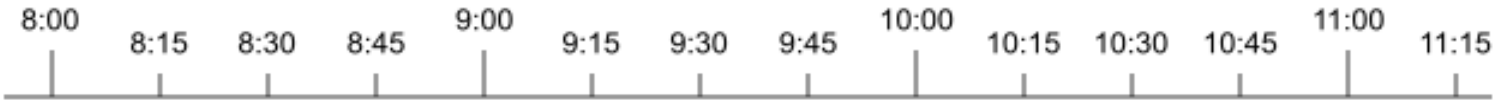


Timeslot pattern

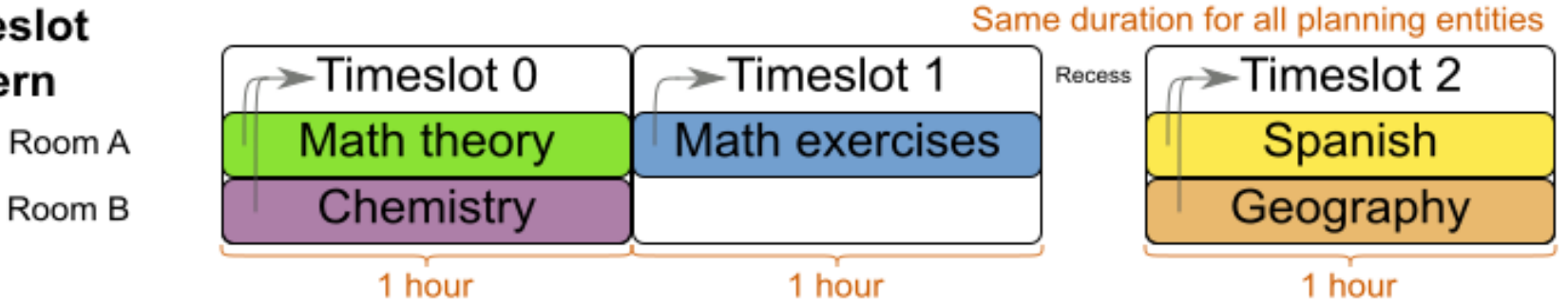


Assigning time to planning entities 1/2

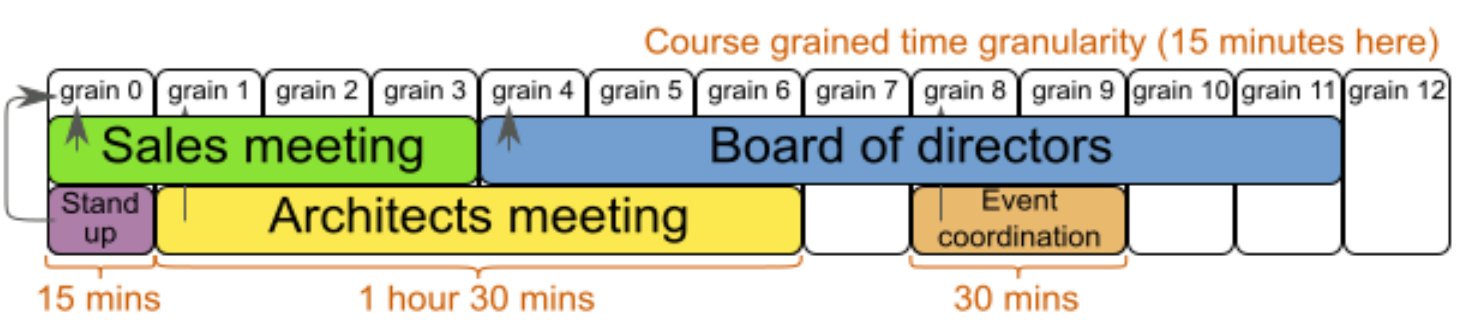
There are several design patterns to deal with time, depending on your use case.



Timeslot pattern

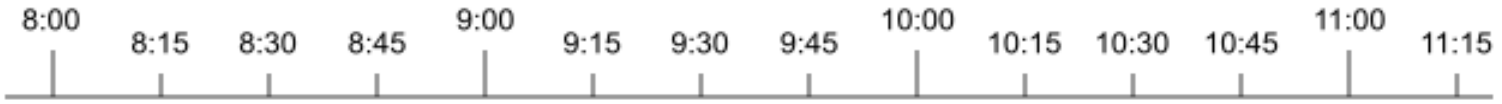


TimeGrain pattern

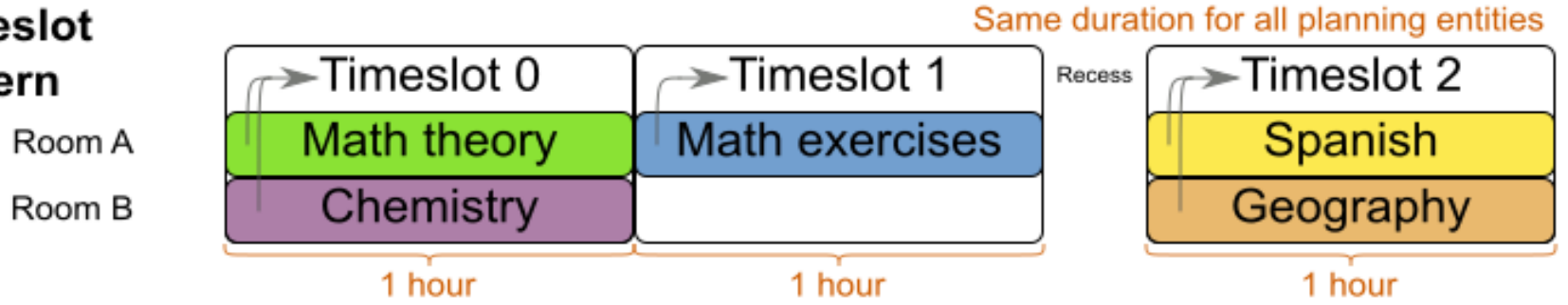


Assigning time to planning entities 1/2

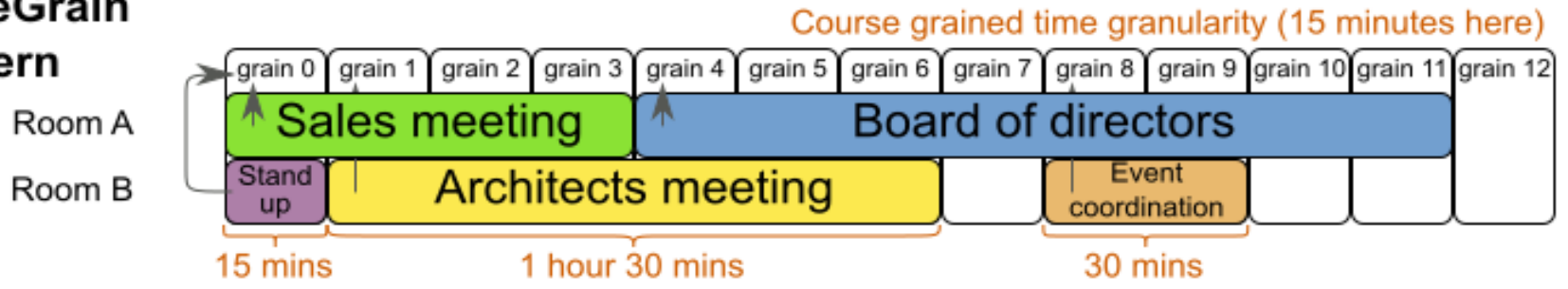
There are several design patterns to deal with time, depending on your use case.



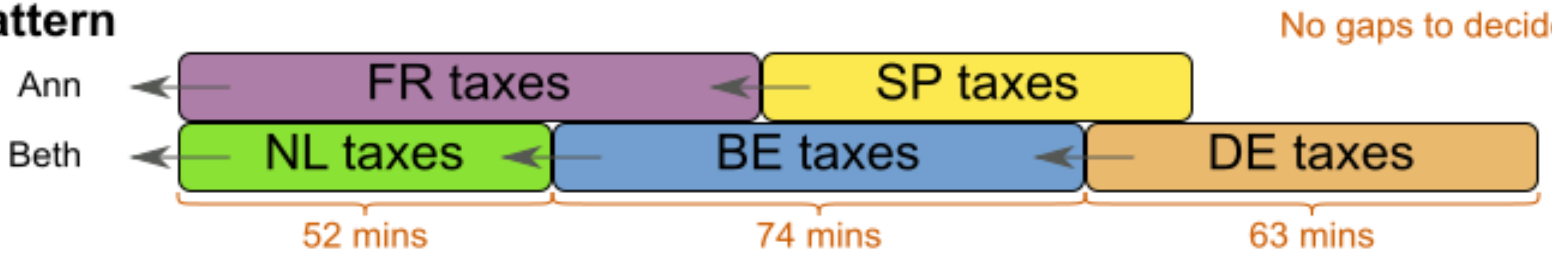
Timeslot pattern



TimeGrain pattern



Chained through time pattern



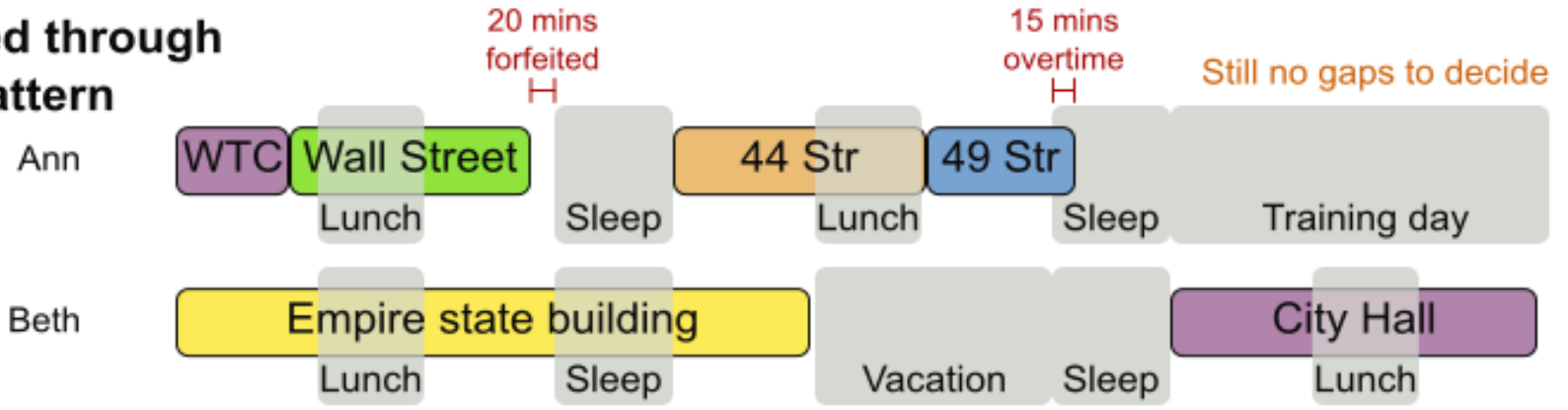
Assigning time to planning entities 2/2

There are several design patterns to deal with time, depending on your use case.

Assigning time to planning entities 2/2

There are several design patterns to deal with time, depending on your use case.

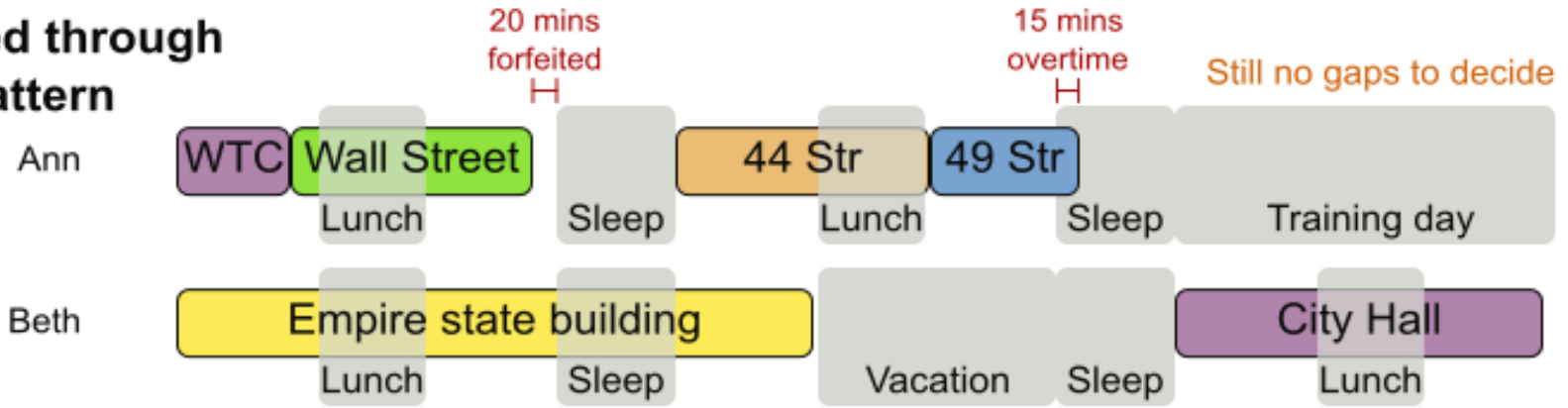
Chained through time pattern



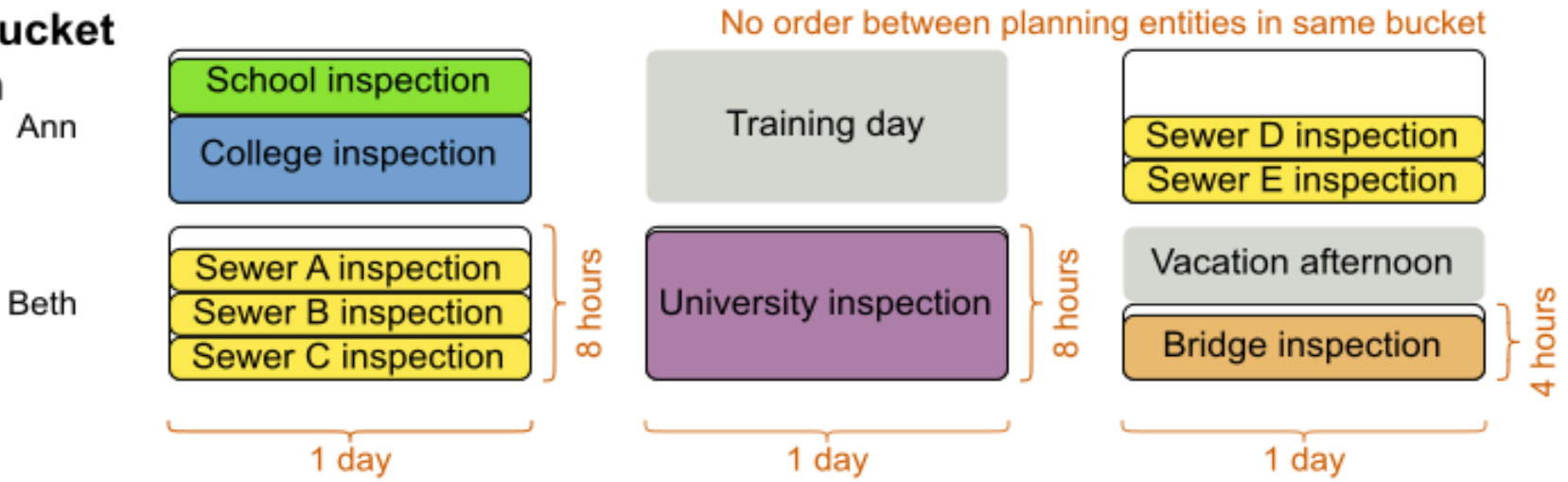
Assigning time to planning entities 2/2

There are several design patterns to deal with time, depending on your use case.

Chained through time pattern



Time bucket pattern



3) Do you replan every week, day or hour?

Continuous planning

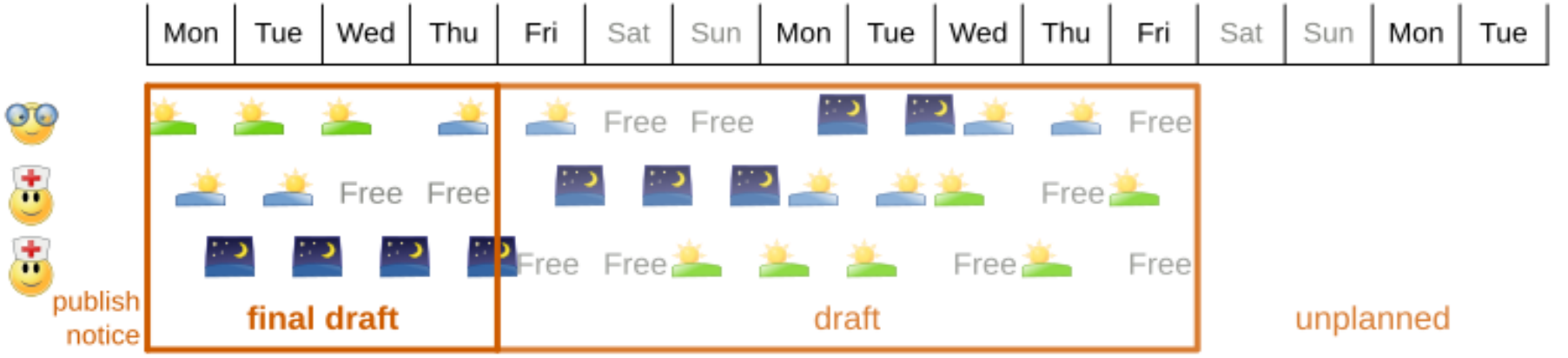
What is continuous planning?

Continuous planning

Replan at the start of every period (usually a week). Plan 3 periods, but only publish the first period.

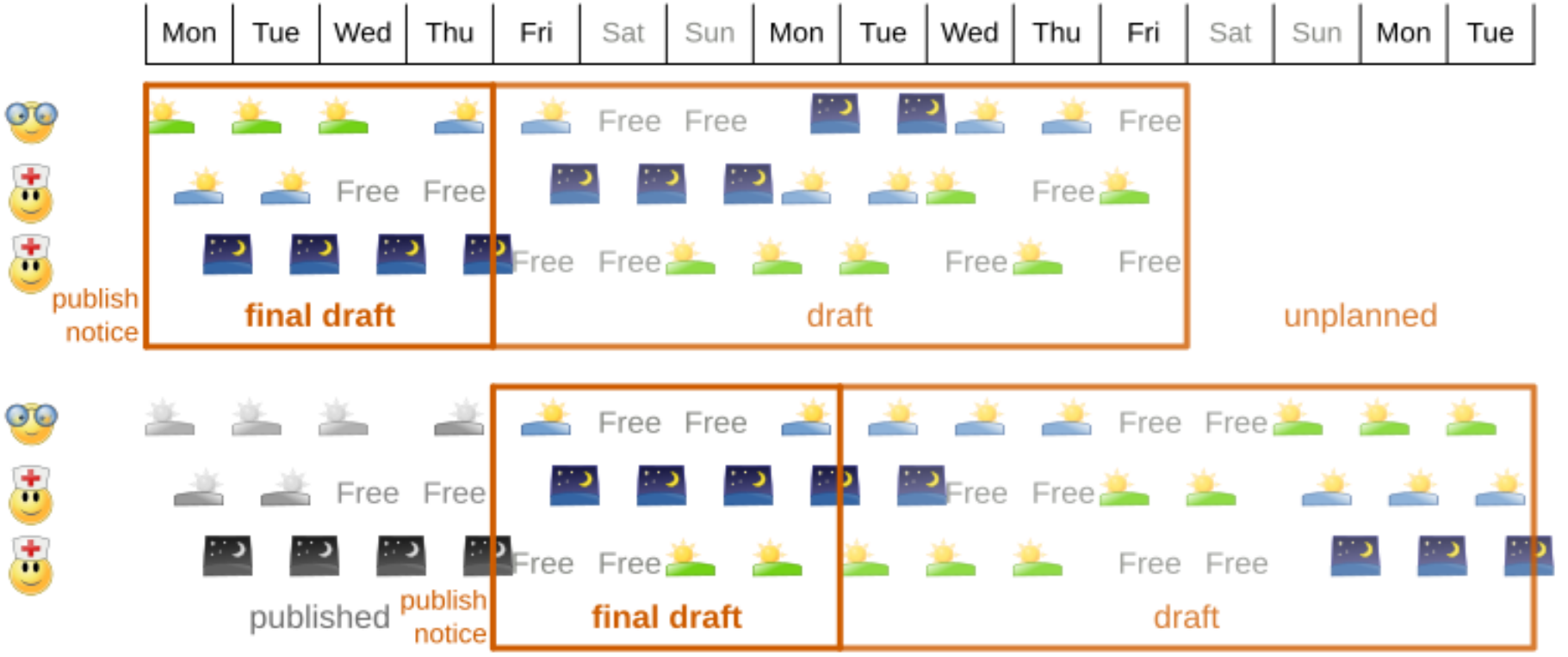
Continuous planning

Replan at the start of every period (usually a week). Plan 3 periods, but only publish the first period.



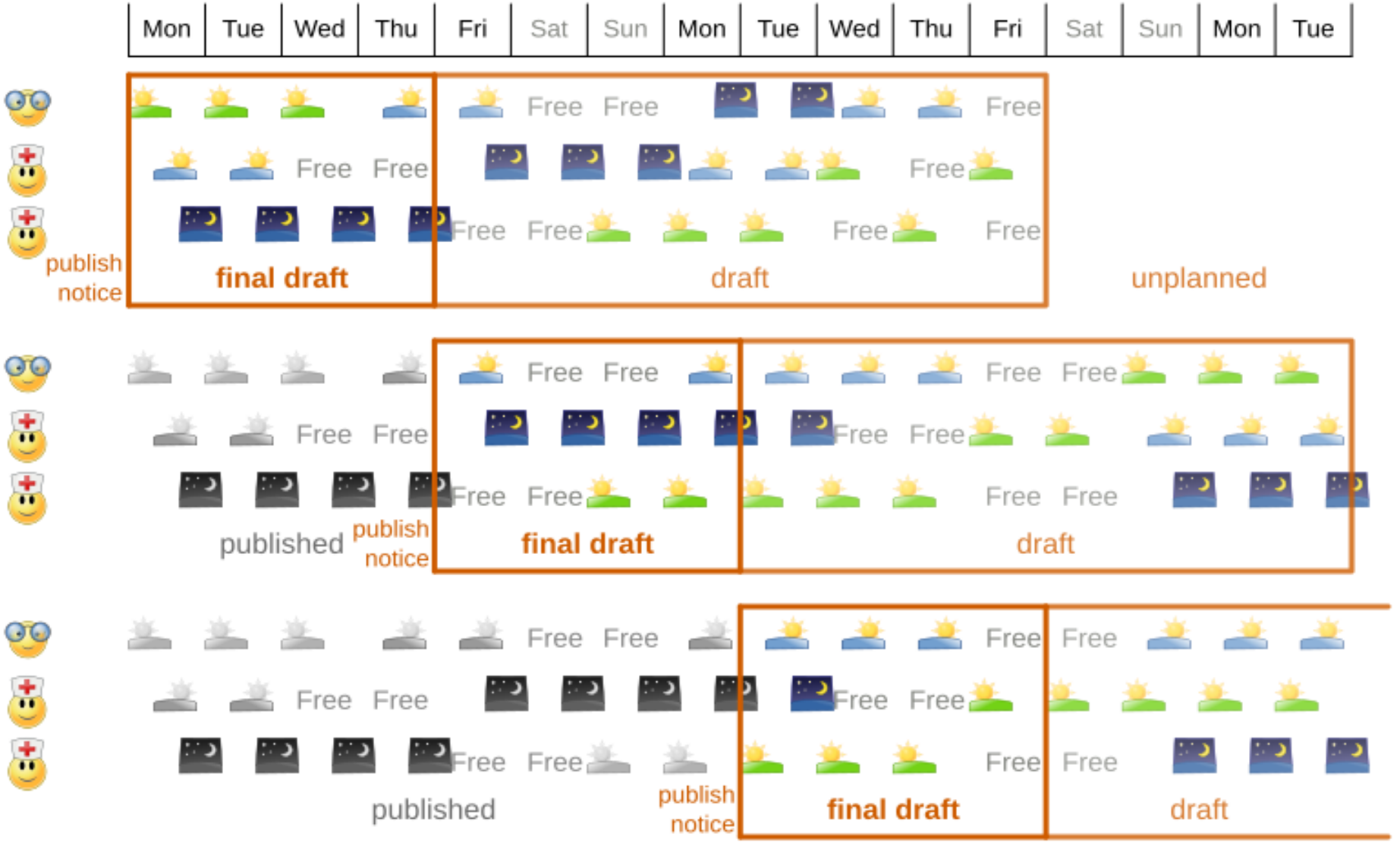
Continuous planning

Replan at the start of every period (usually a week). Plan 3 periods, but only publish the first period.



Continuous planning

Replan at the start of every period (usually a week). Plan 3 periods, but only publish the first period.



Don't change history

```
@Entity
public class Shift {

  private LocalDateTime start;
  private LocalDateTime end;

  @PlanningPin
  private boolean history;
  ...
  @PlanningVariable(...)
  private Employee employee;
  ...
}
```

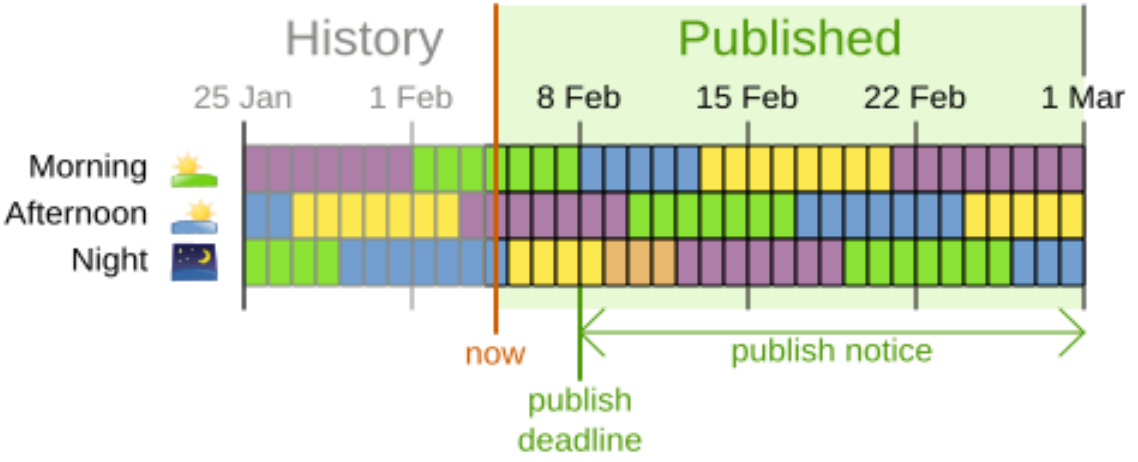
Don't change published shifts?

```
@PlanningEntity
public class Shift {

  private LocalDateTime start;
  private LocalDateTime end;

  @PlanningPin
  // every historic shift is published
  private boolean published;
  ...
  @PlanningVariable(...)
  private Employee employee;
  ...
}
```

Maybe

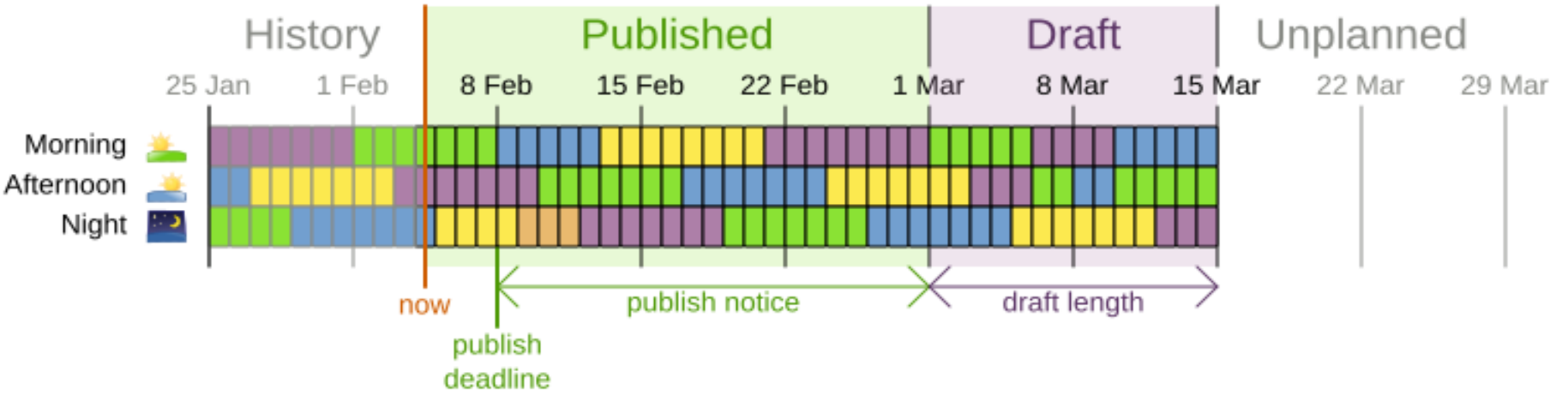
Continuous publishing with rotation

In this example, the schedule is published every week, at least 3 weeks in advance.



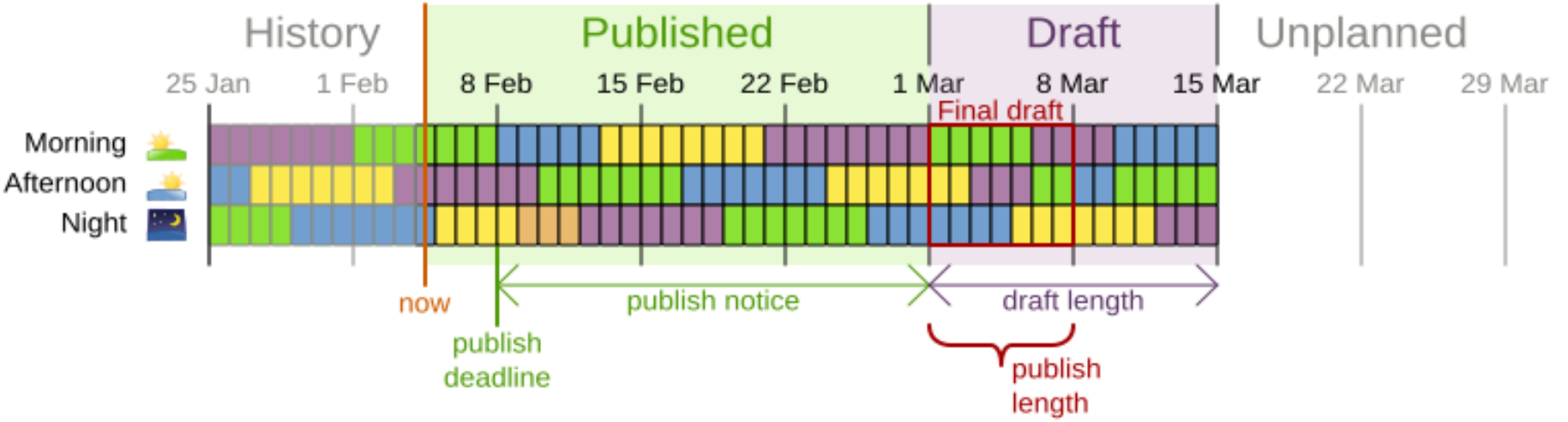
Continuous publishing with rotation

In this example, the schedule is published every week, at least 3 weeks in advance.



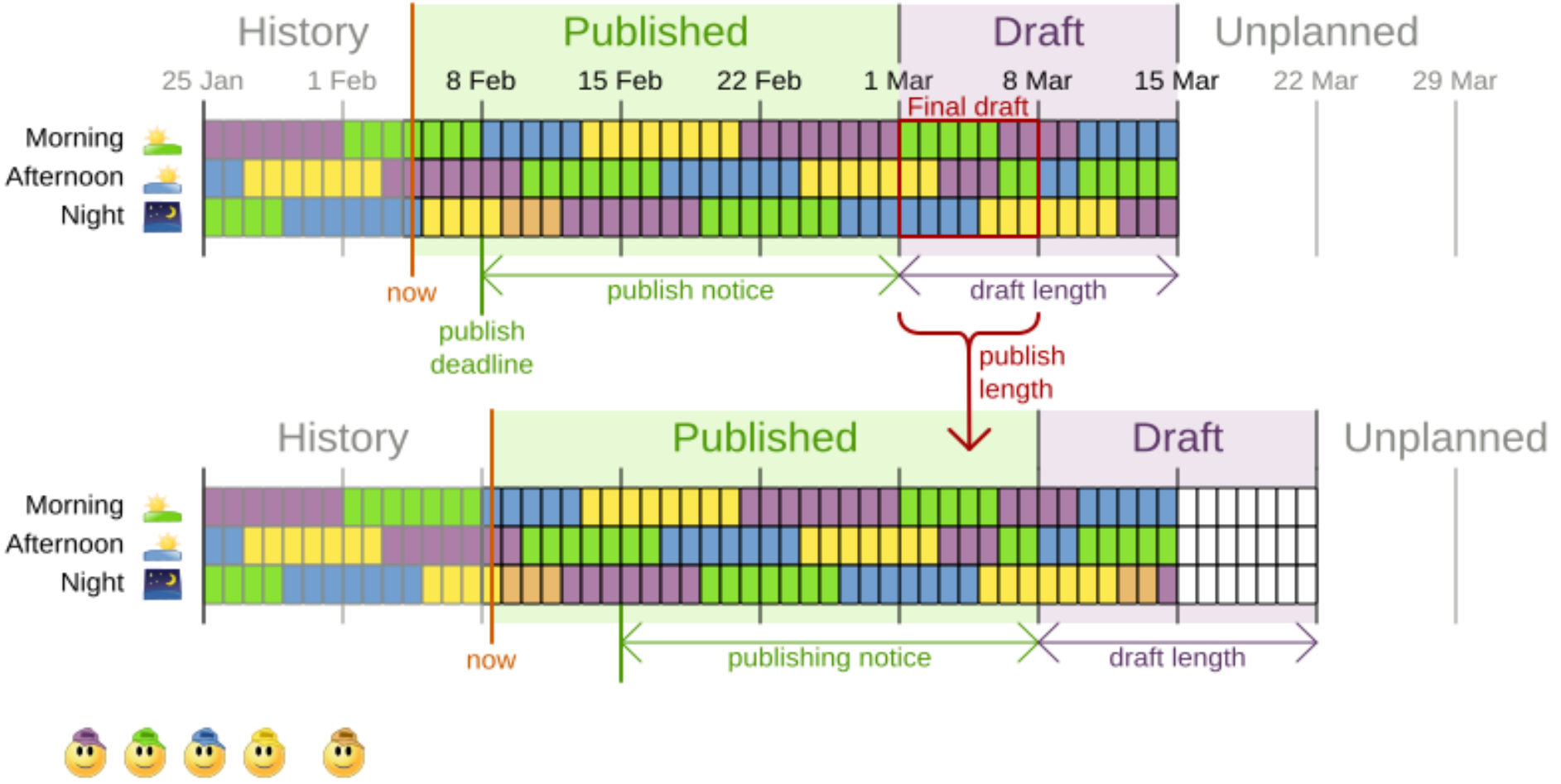
Continuous publishing with rotation

In this example, the schedule is published every week, at least 3 weeks in advance.



Continuous publishing with rotation

In this example, the schedule is published every week, at least 3 weeks in advance.



How to paint yourself in a corner

Assign all respiratory specialists
to the *last* shift of your planning window
(even in other wards as normal nurses).

How to paint yourself in a corner

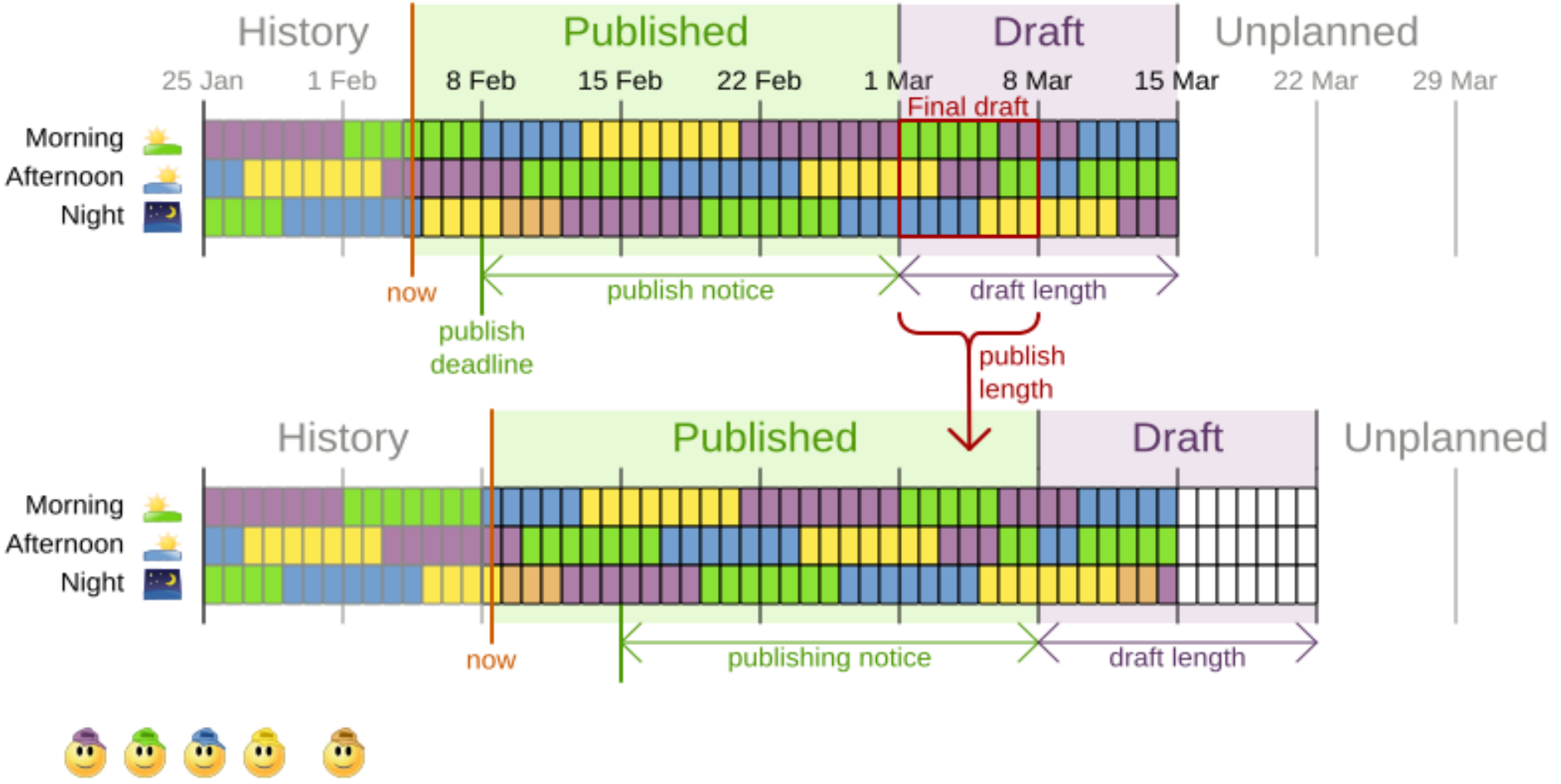
Assign all respiratory specialists
to the *last* shift of your planning window
(even in other wards as normal nurses).

⇒

draft length >= 2 * publish length

Continuous publishing with rotation

In this example, the schedule is published every week, at least 3 weeks in advance.



Continuous planning

- Publish notice ranges from long to short
  - Weeks: employee shift rostering, pharmacy on duty scheduling, ...
  - Seconds: runway scheduling, gate scheduling, platform scheduling, ...

Continuous planning

- Publish notice ranges from long to short
  - Weeks: employee shift rostering, pharmacy on duty scheduling, ...
  - Seconds: runway scheduling, gate scheduling, platform scheduling, ...
- **Domain must contain dates or timestamps data**
  - So domain probably has `import java.time`
  - Regardless if planning assigns to time or not.

Continuous planning

- Publish notice ranges from long to short
  - Weeks: employee shift rostering, pharmacy on duty scheduling, ...
  - Seconds: runway scheduling, gate scheduling, platform scheduling, ...
- **Domain must contain dates or timestamps data**
  - So domain probably has `import java.time`
  - Regardless if planning assigns to time or not.
- **Planning windows must overlap**

Continuous planning

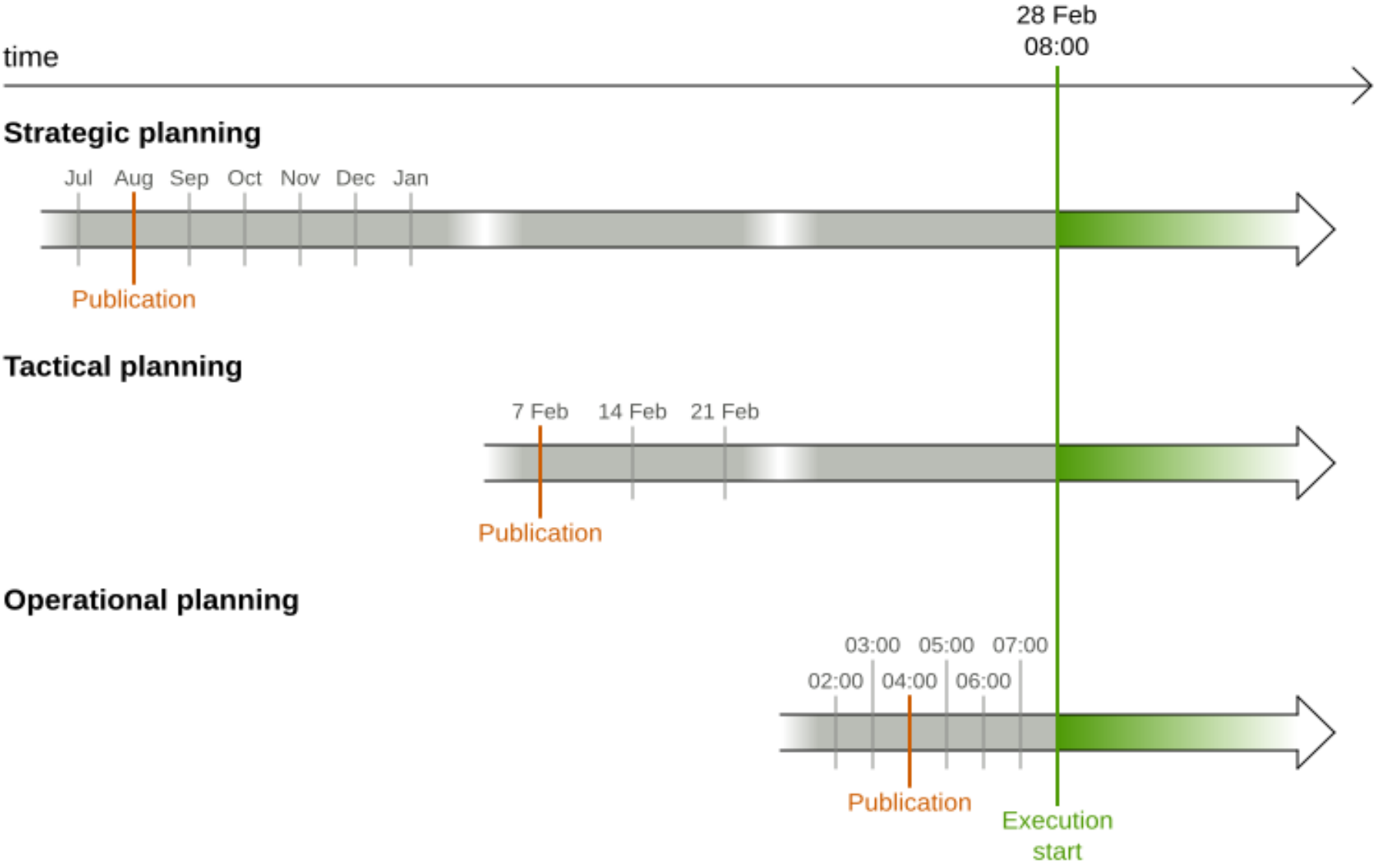
- Publish notice ranges from long to short
  - Weeks: employee shift rostering, pharmacy on duty scheduling, ...
  - Seconds: runway scheduling, gate scheduling, platform scheduling, ...
- **Domain must contain dates or timestamps data**
  - So domain probably has `import java.time`
  - Regardless if planning assigns to time or not.
- **Planning windows must overlap**
  - ⇒ School timetabling is not continuous planning

4) Will published plan change?

Non-disruptive Replanning

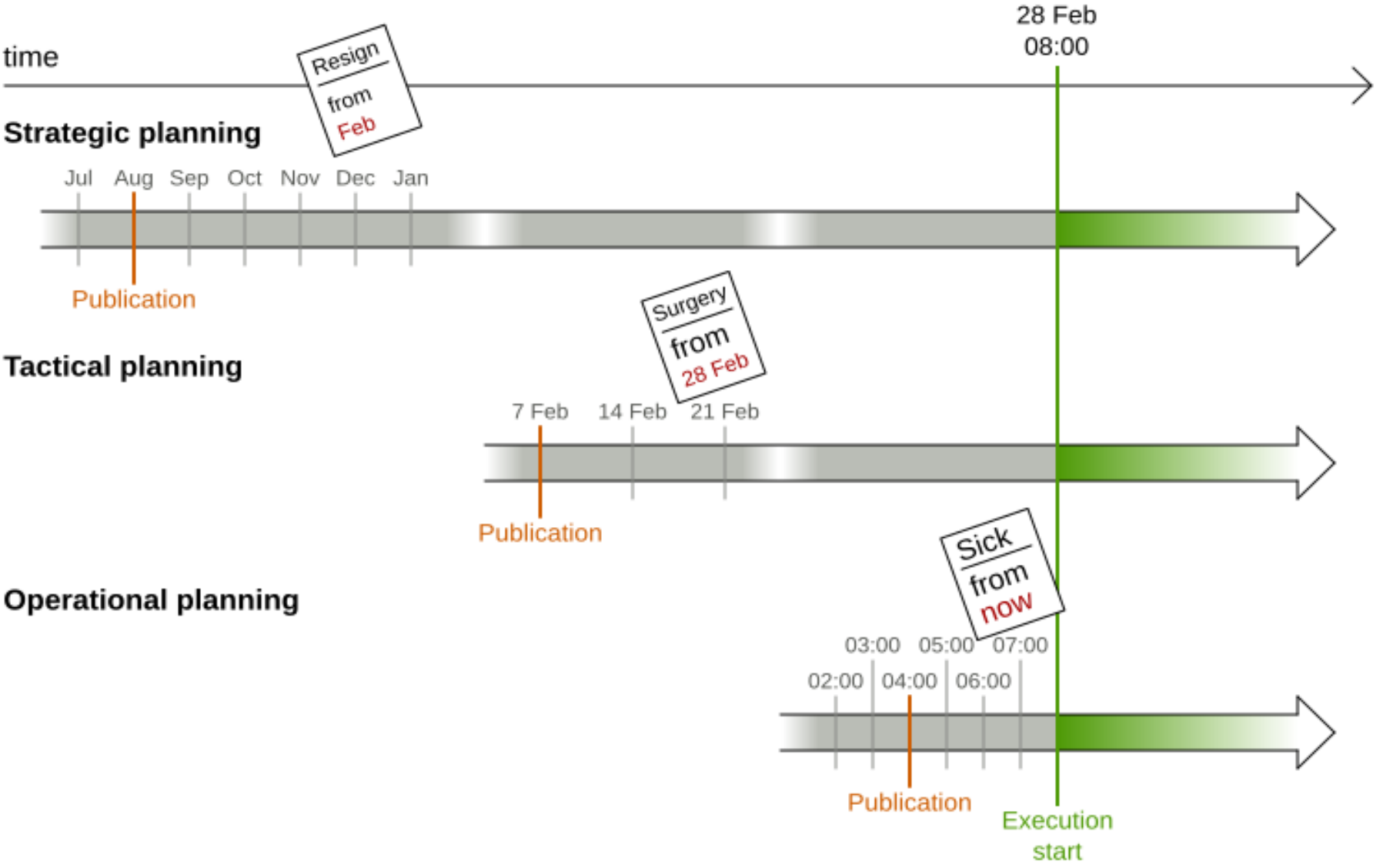
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



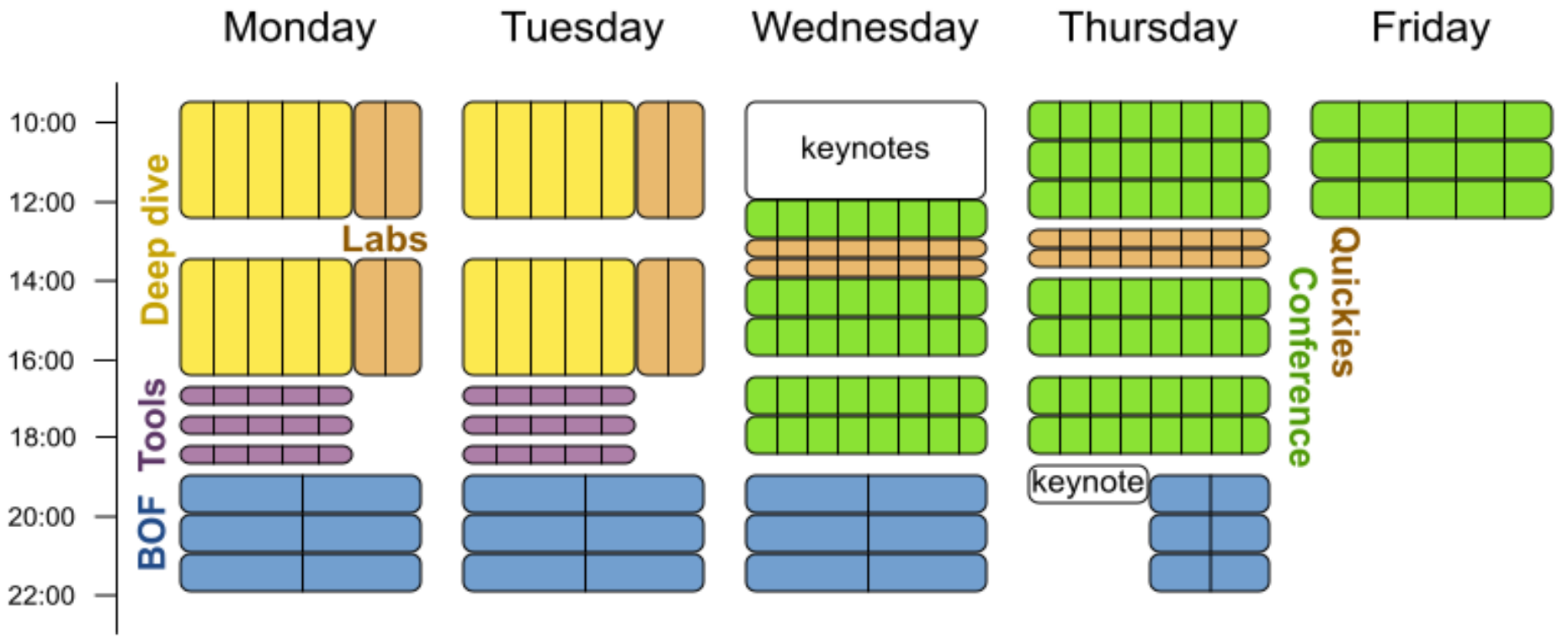
Multi-stage planning

Strategic planning, tactical planning and operational planning are seperated by time.



Conference scheduling problem

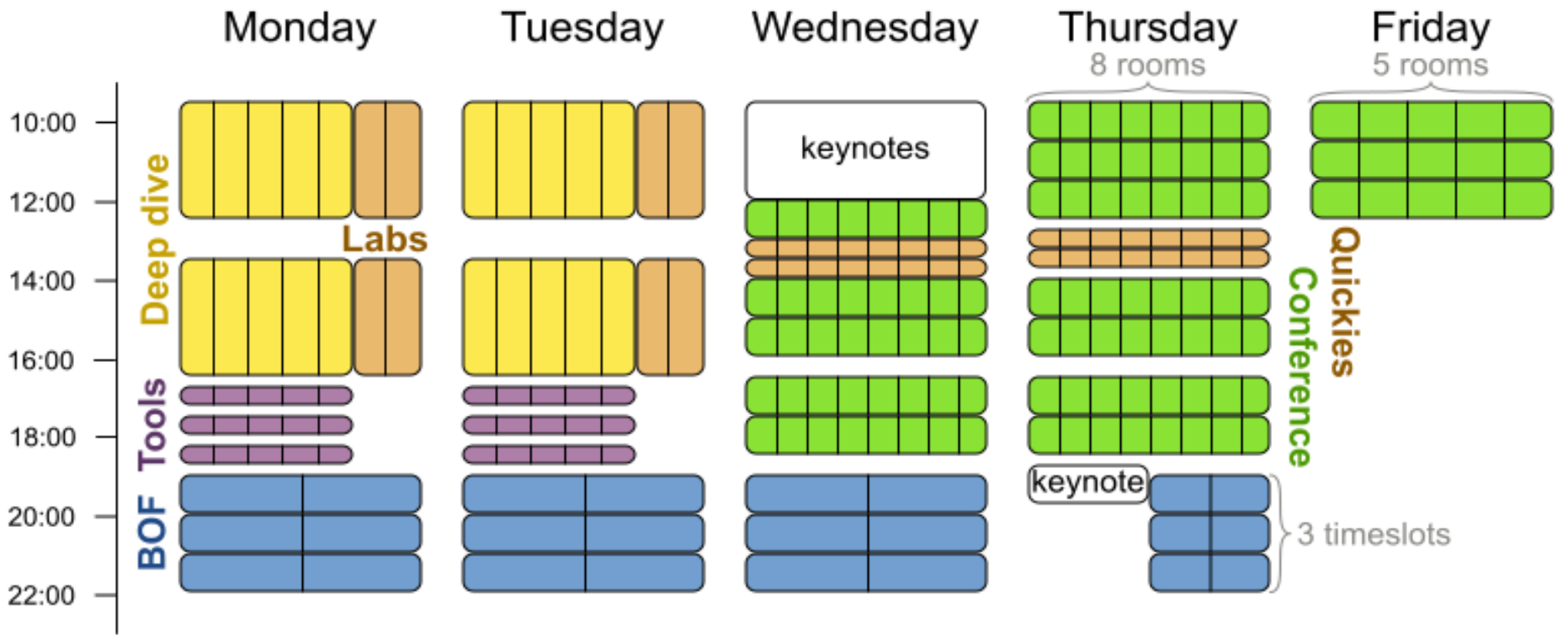
Assign each talk to a timeslot and a room.



Devoxx Belgium assigns 214 talks to 40 timeslots and 10 rooms for 3500 attendees.

Conference scheduling problem

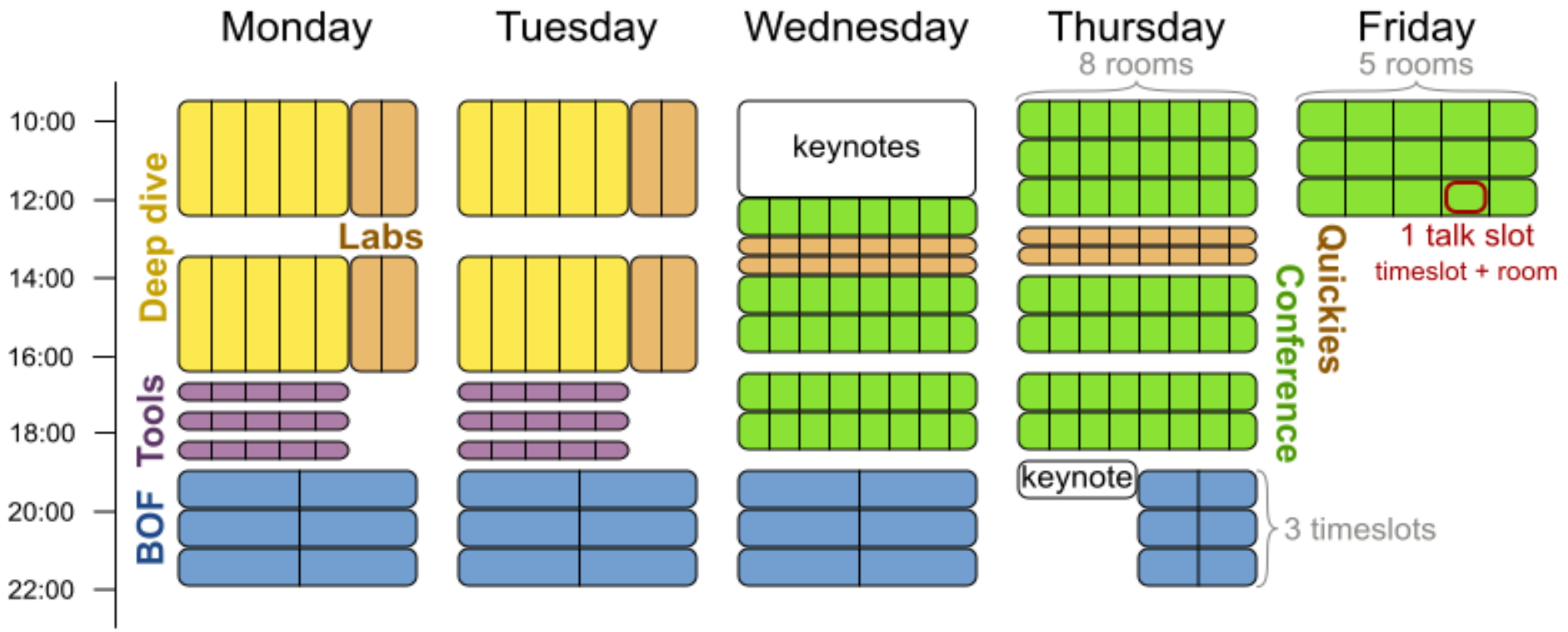
Assign each talk to a timeslot and a room.



Devoxx Belgium assigns 214 talks to 40 timeslots and 10 rooms for 3500 attendees.

Conference scheduling problem

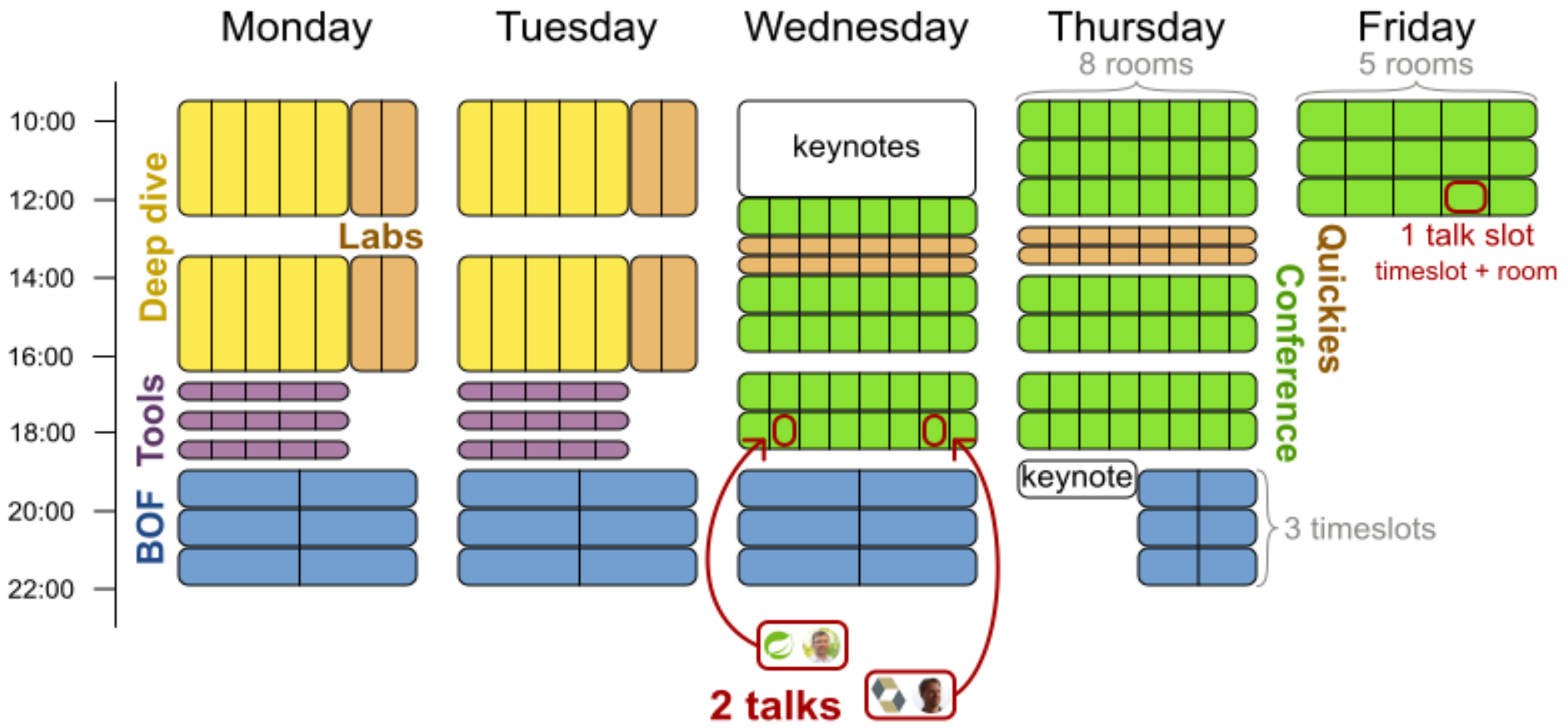
Assign each talk to a timeslot and a room.



Devoxx Belgium assigns 214 talks to 40 timeslots and 10 rooms for 3500 attendees.

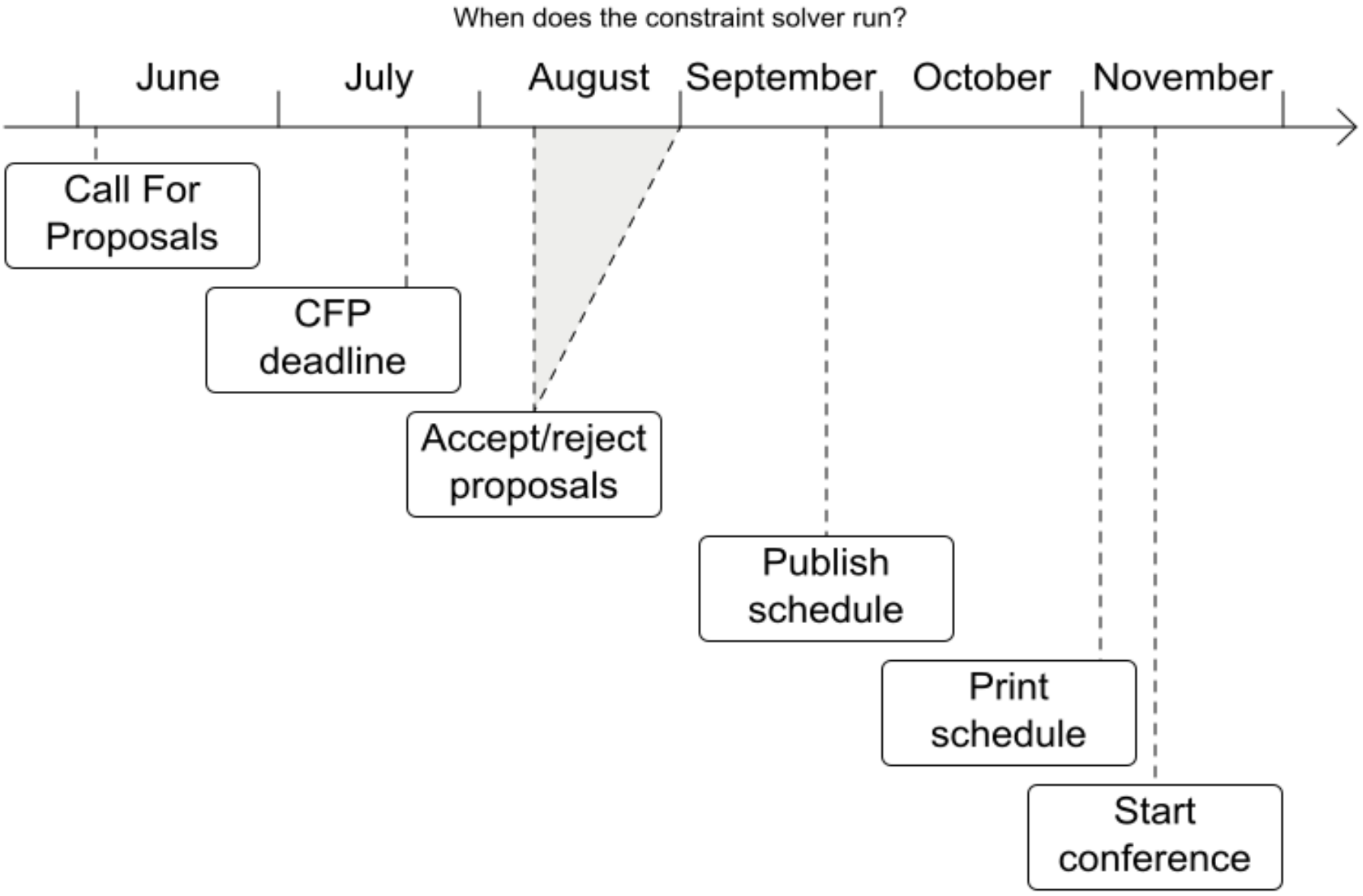
Conference scheduling problem

Assign each talk to a timeslot and a room.

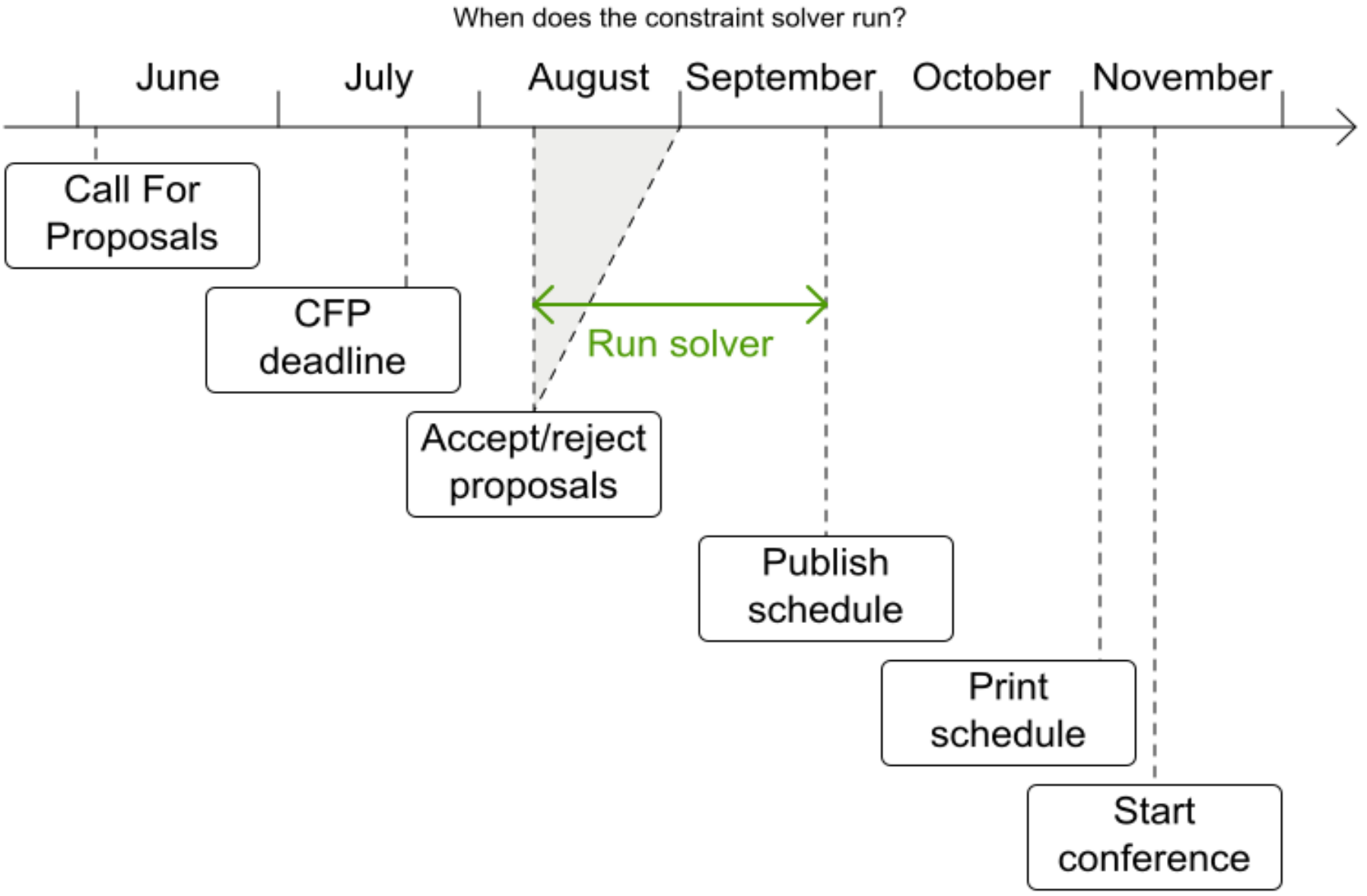


Devoxx Belgium assigns 214 talks to 40 timeslots and 10 rooms for 3500 attendees.

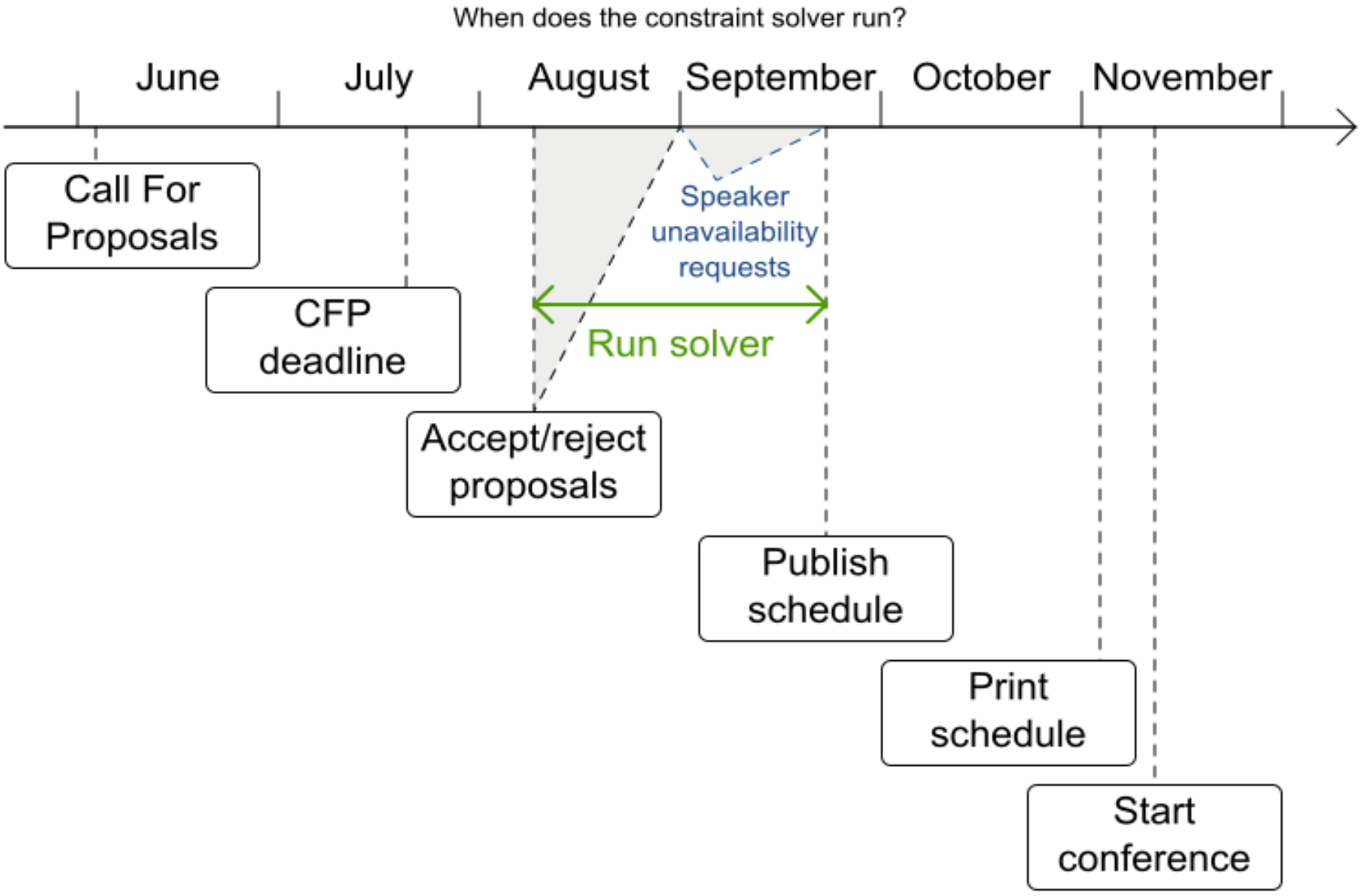
Conference scheduling milestones timeline



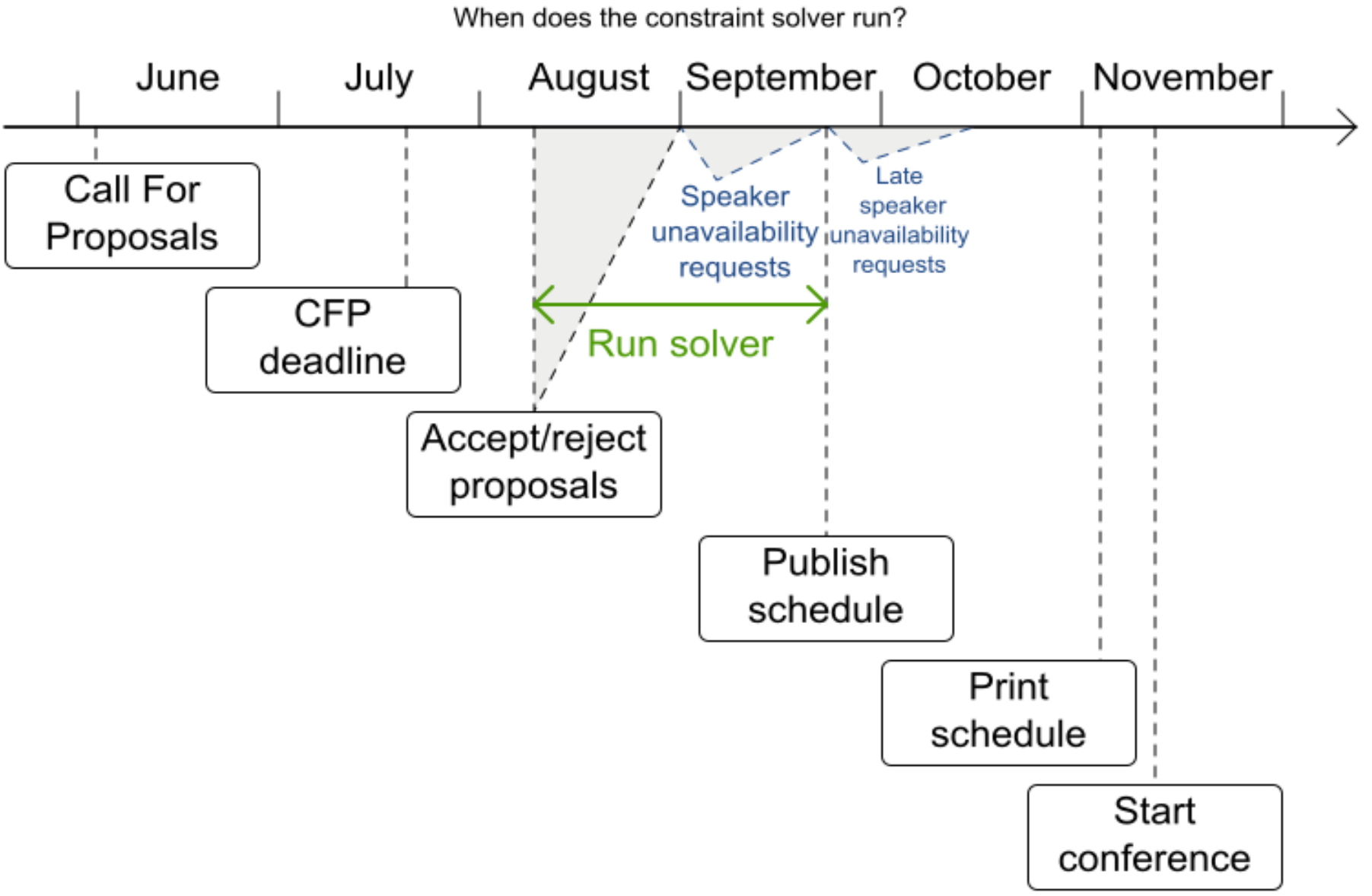
Conference scheduling milestones timeline



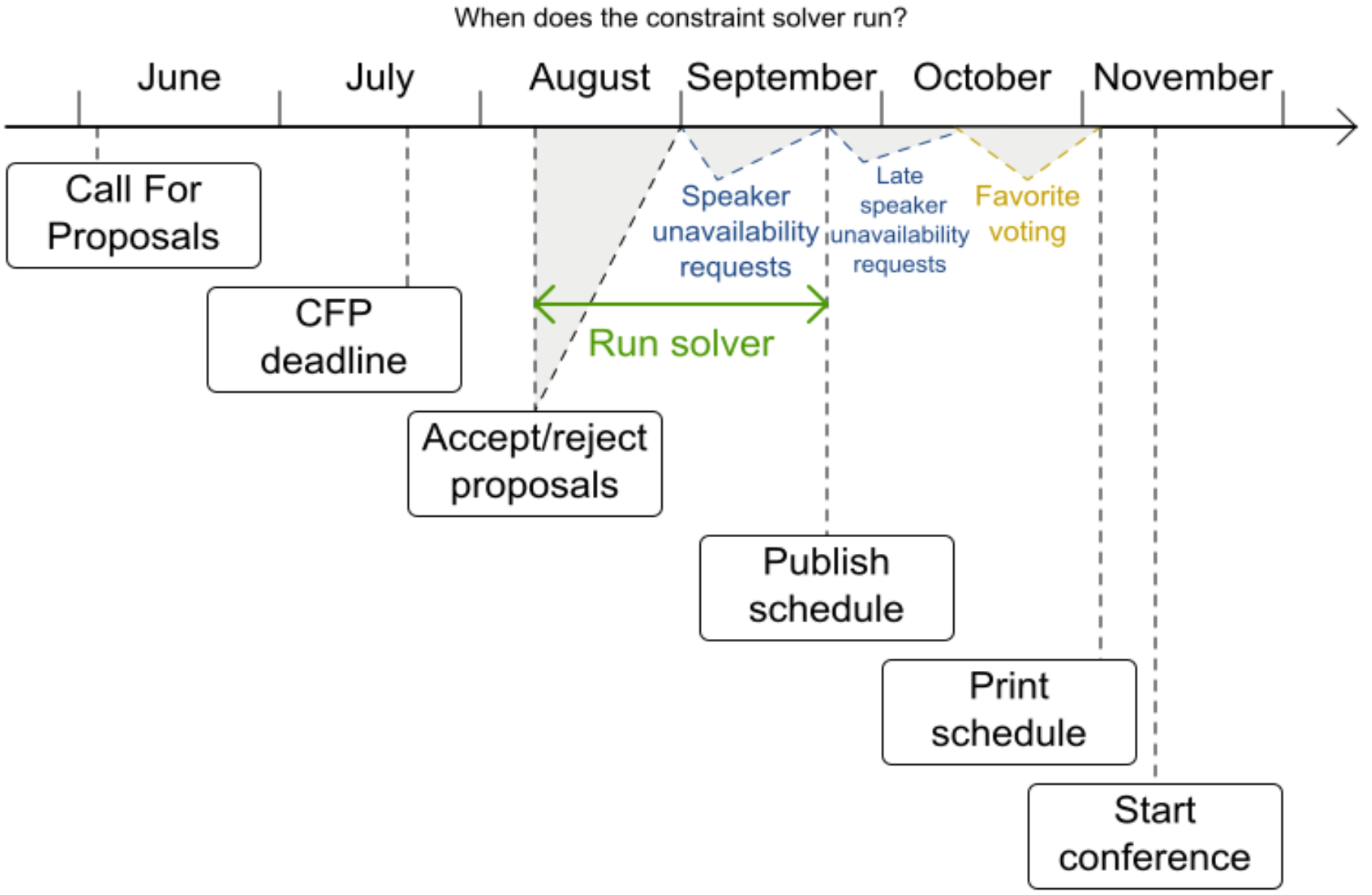
Conference scheduling milestones timeline



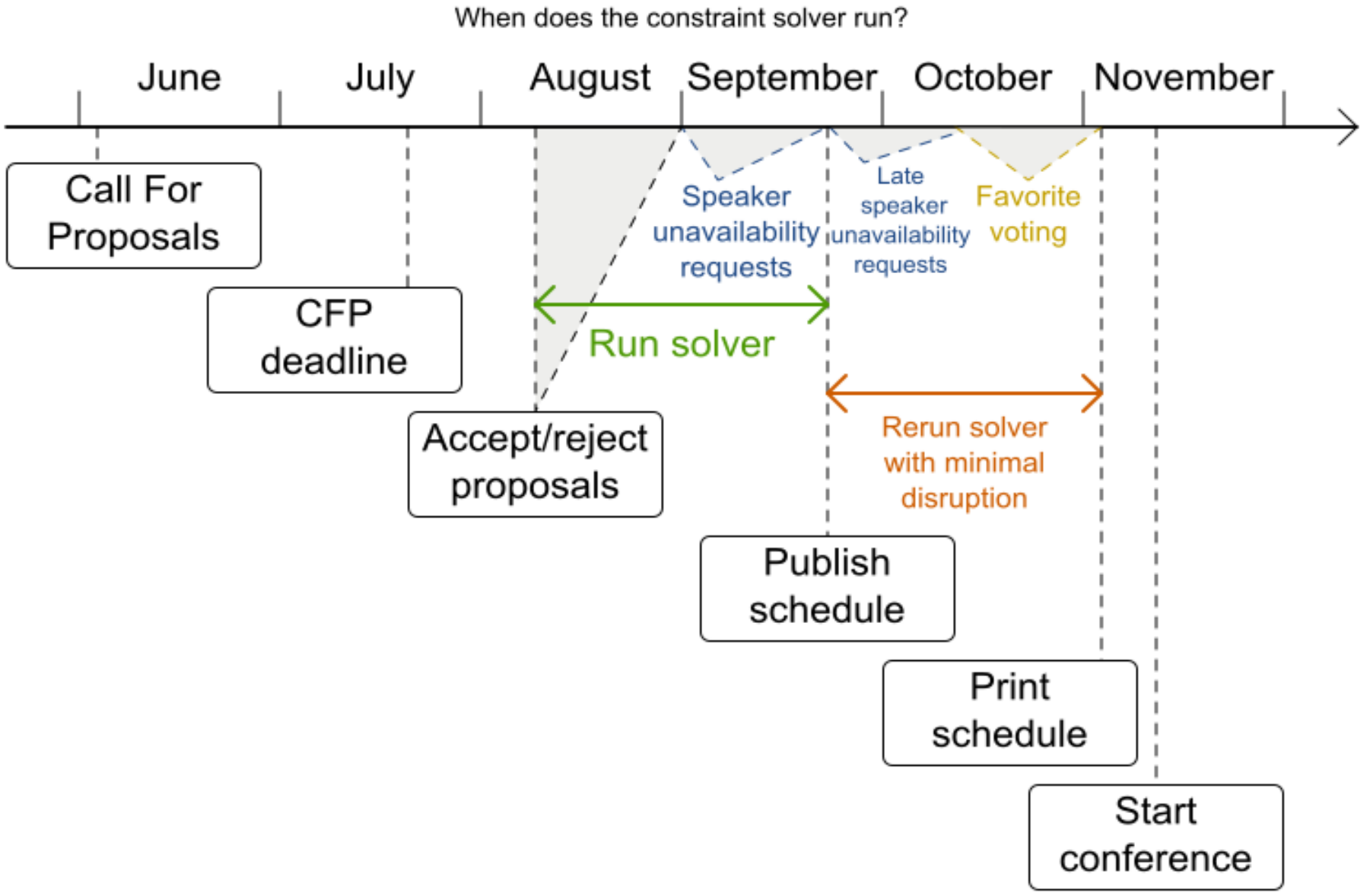
Conference scheduling milestones timeline



Conference scheduling milestones timeline



Conference scheduling milestones timeline



Implementation

```
@PlanningEntity
public class Talk {

    private Timeslot publishedTimeslot;
    ...

    @PlanningVariable(...)
    private Timeslot timeslot;

}
```

```
  Constraint publishedTimeslot(ConstraintFactory f) {
    return f.from(Talk.class)
        .filter(talk -> talk.getPublishedTimeslot() != null
            && talk.getTimeslot()
                != talk.getPublishedTimeslot())
        .penalize(PUBLISHED_TIMESLOT,
          HardMediumSoft.ONE_MEDIUM);
  }
```

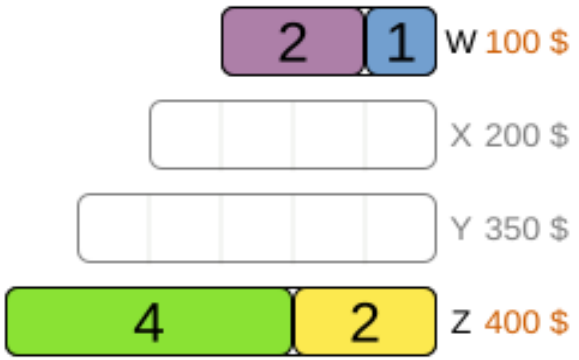
Non-disruptive Replanning

- Medium penalty: disrupt only to become feasible
  - Timeslot in conference scheduling
- Soft penalty: disrupt if gain is higher than a threshold
  - Room in conference scheduling

Non disruptive replanning

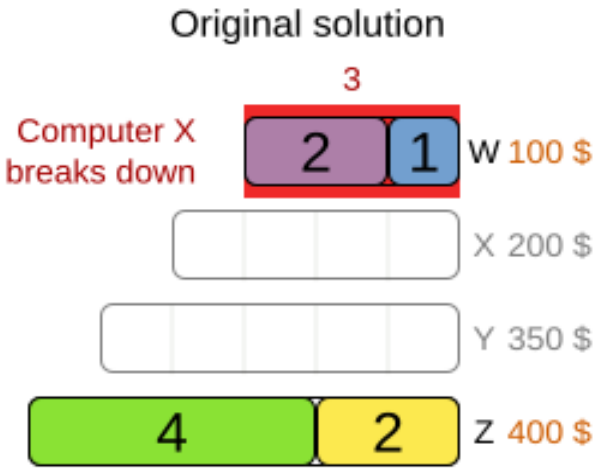
Real-time planning must not distort the entire plan to deal with a real-time change.

Original solution



Non disruptive replanning

Real-time planning must not distort the entire plan to deal with a real-time change.



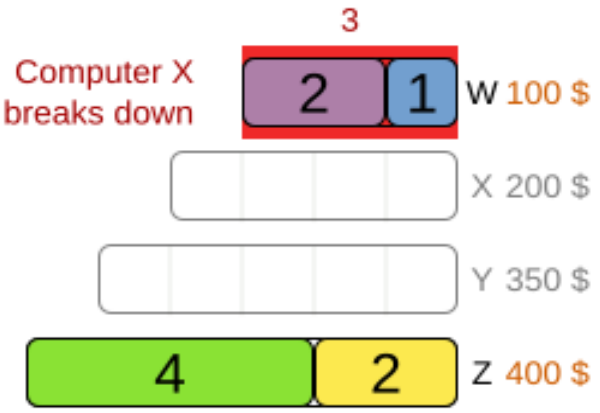
Normal score

-3hard / -500soft

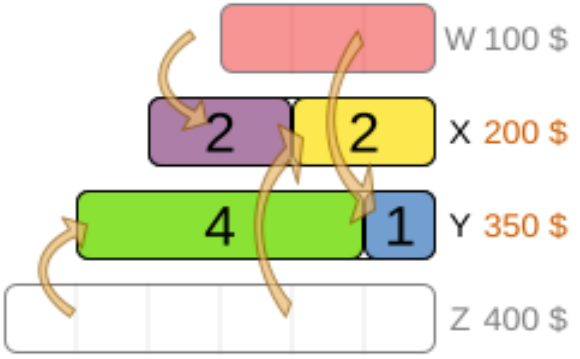
Non disruptive replanning

Real-time planning must not distort the entire plan to deal with a real-time change.

Original solution



Disruptive solution



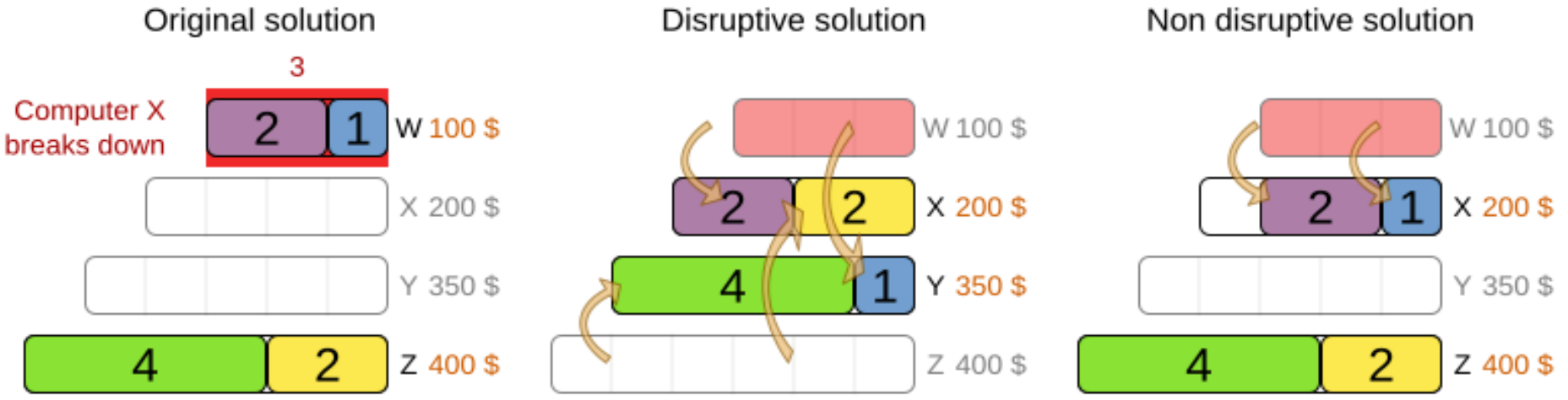
Normal score

-3hard / -500soft

0hard / -550soft

Non disruptive replanning

Real-time planning must not distort the entire plan to deal with a real-time change.



Normal score

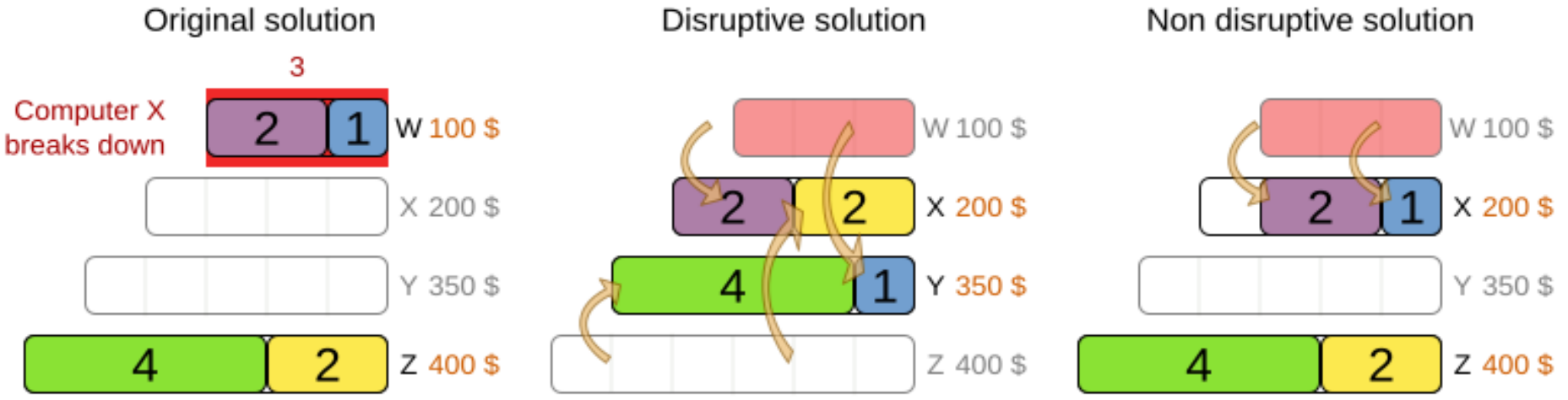
-3hard / -500soft

0hard / -550soft

0hard / -600soft

Non disruptive replanning

Real-time planning must not distort the entire plan to deal with a real-time change.



Normal score

-3hard / -500soft

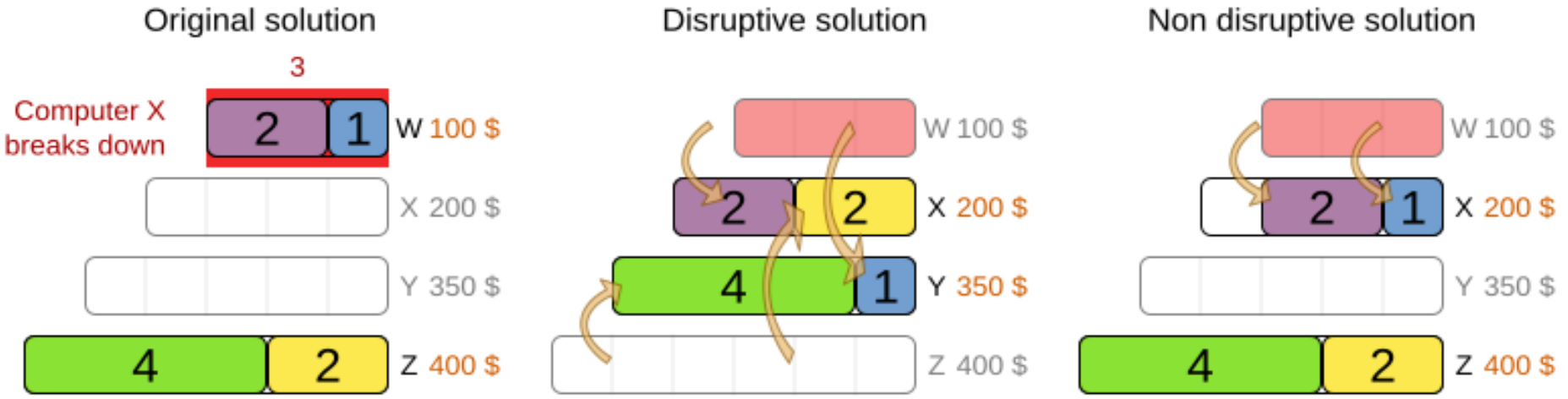
0hard / -550soft

Highest score

0hard / -600soft

Non disruptive replanning

Real-time planning must not distort the entire plan to deal with a real-time change.



Normal score

-3hard / -500soft

0hard / -550soft

Highest score

0hard / -600soft

Adjusted score (-100 per moved process)

no moved processes: 0soft

-3hard / -500soft

4 moved processes: -400soft

0hard / -950soft

2 moved processes: -200soft

0hard / -800soft

Highest score

5) Does your problem change every few (milli)seconds?

Real-time planning

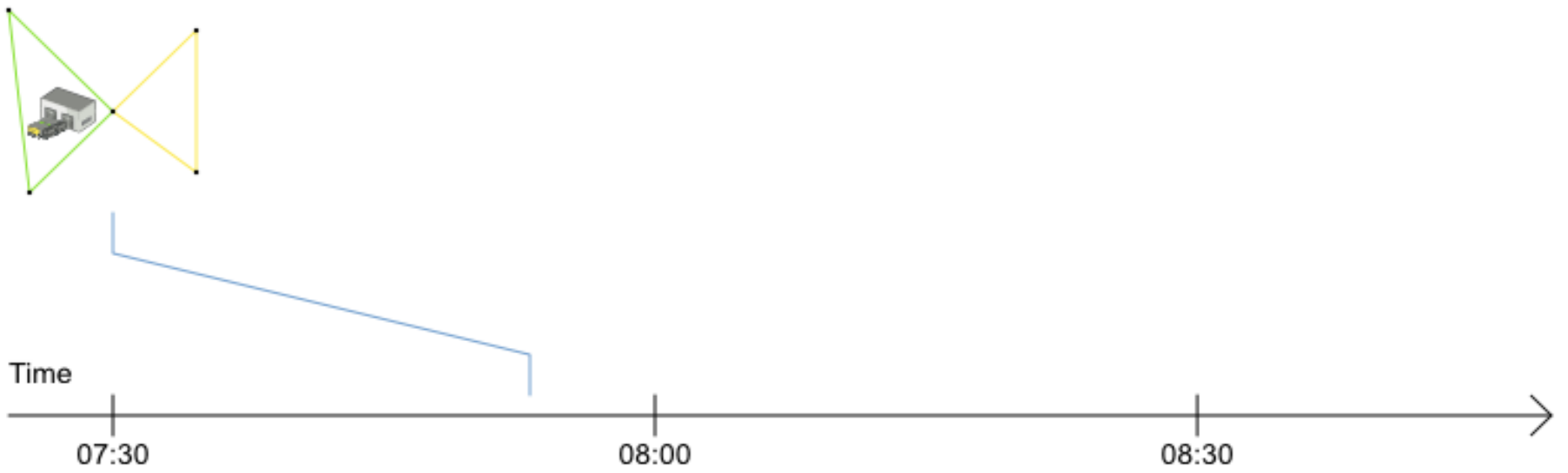
Real-time planning

DEMO

Real-time planning

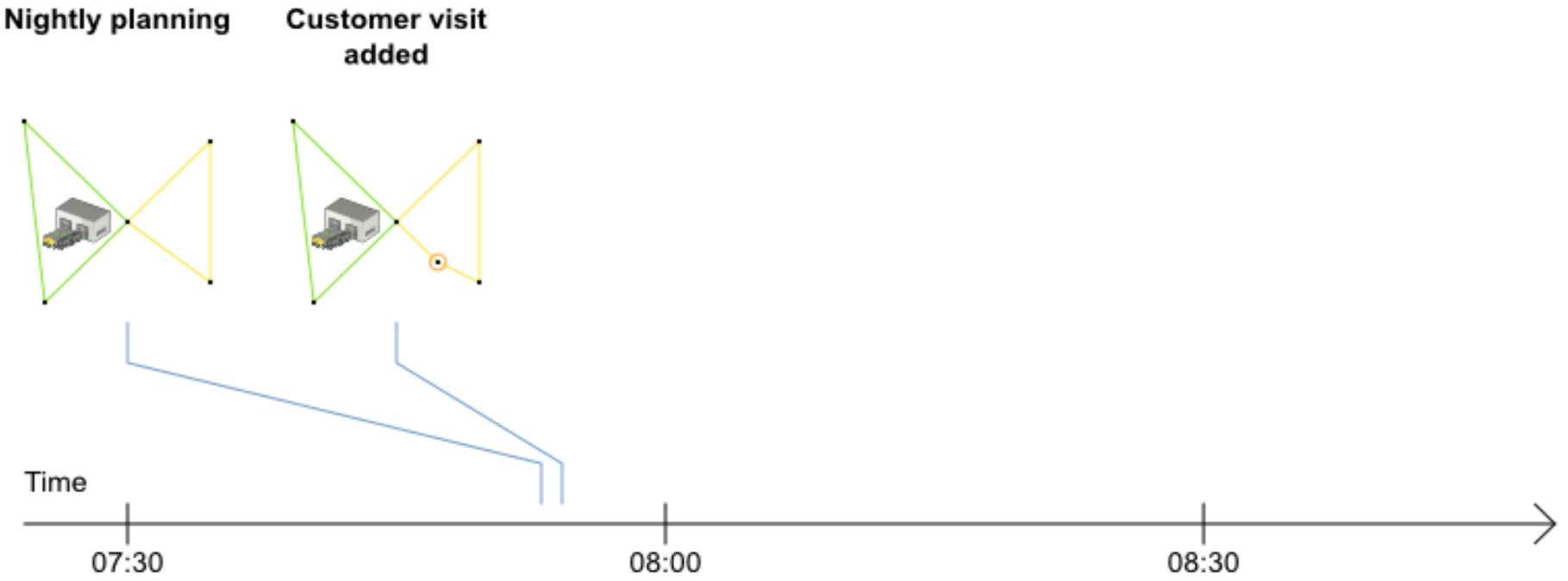
When the problem changes in real-time, the plan is adjusted in real-time.

Nightly planning



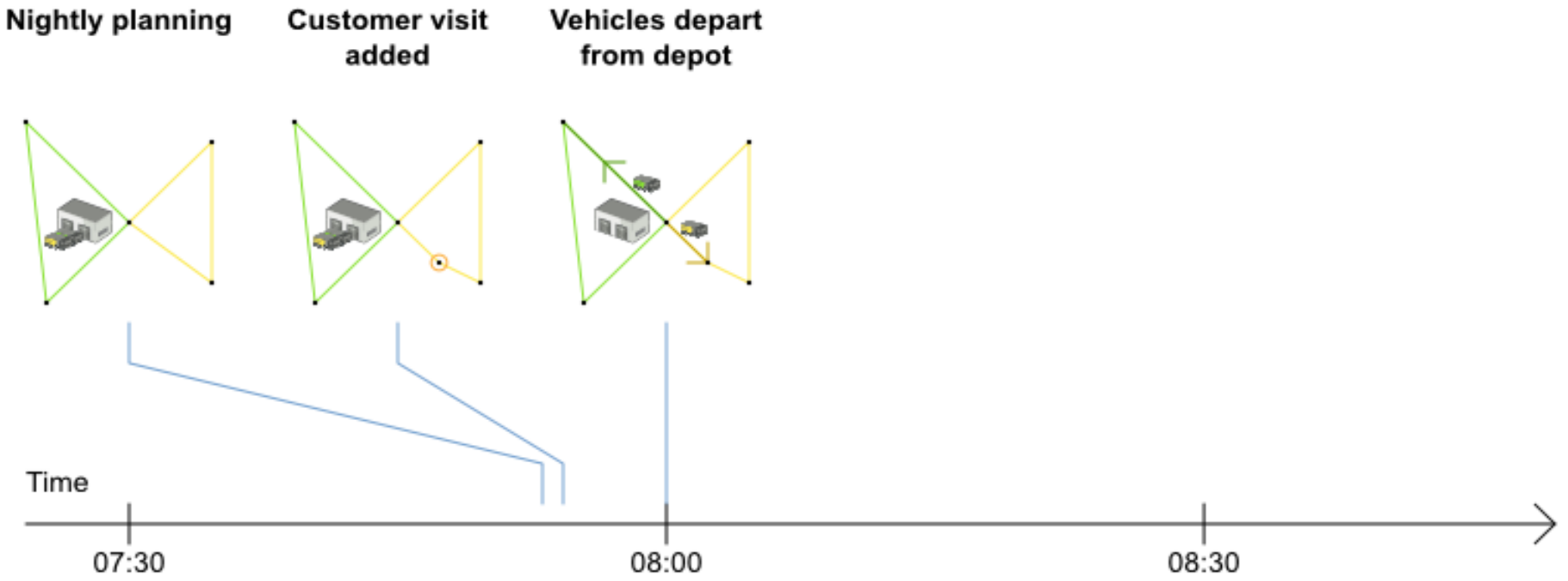
Real-time planning

When the problem changes in real-time, the plan is adjusted in real-time.



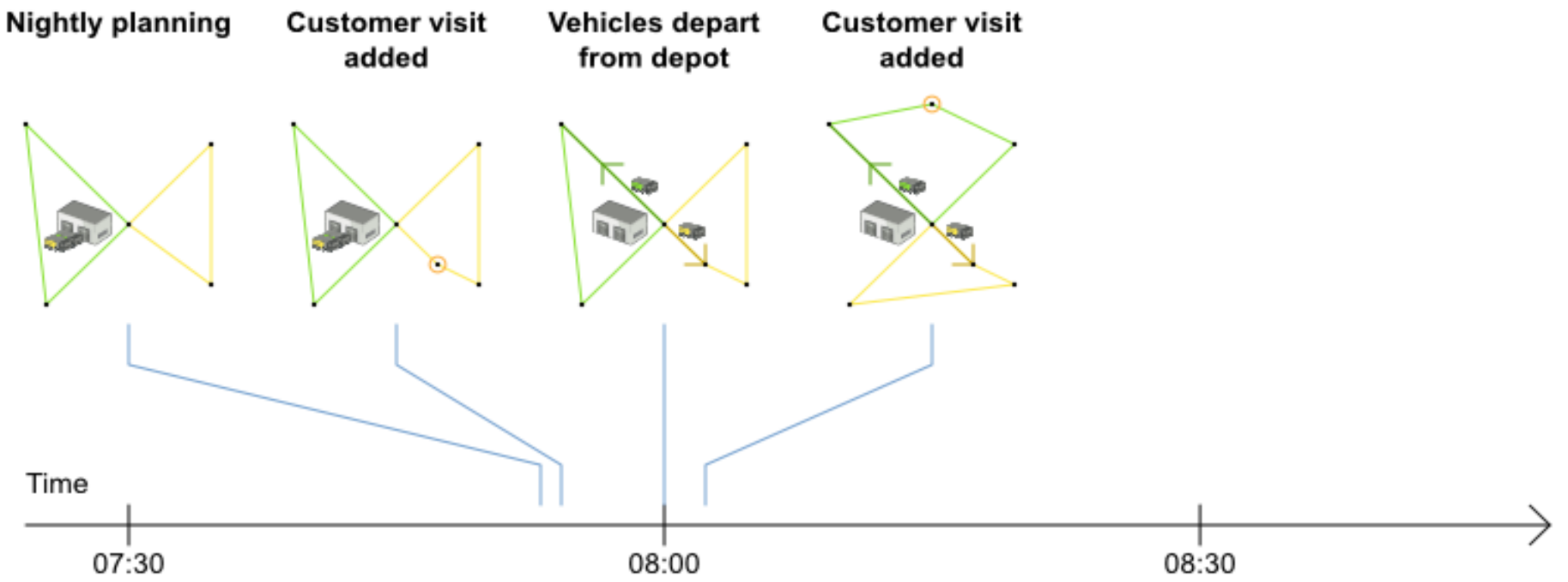
Real-time planning

When the problem changes in real-time, the plan is adjusted in real-time.



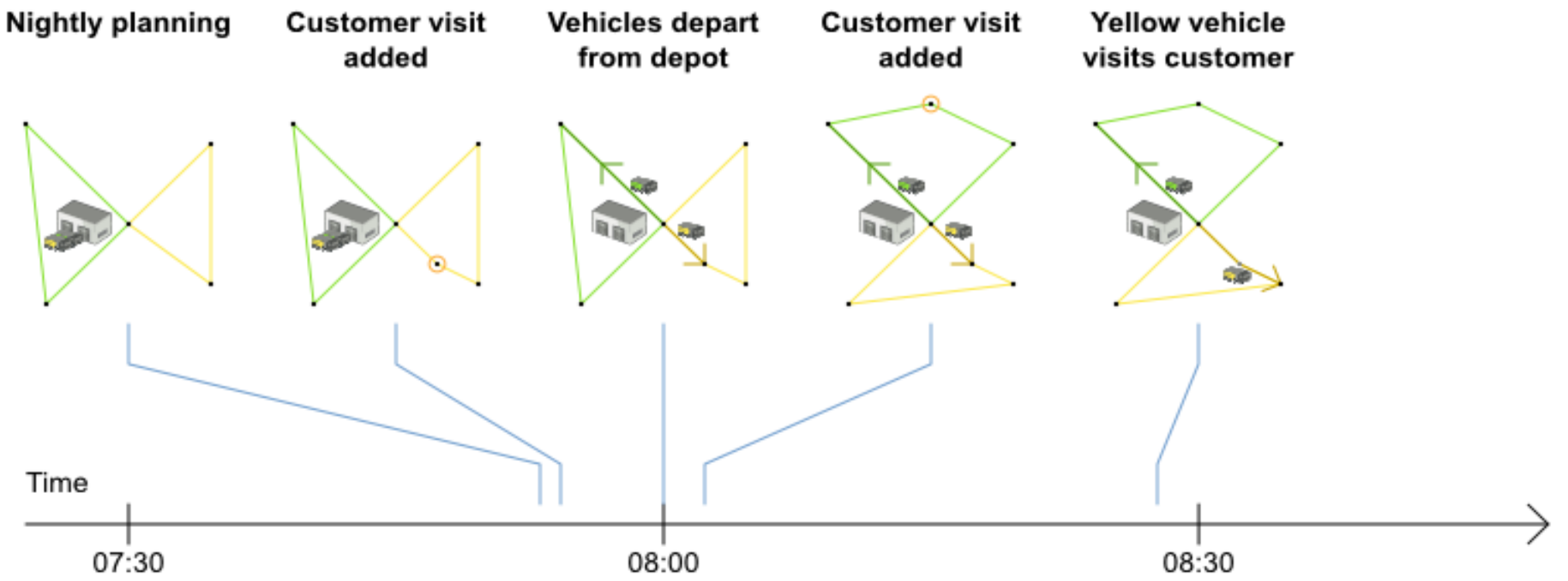
Real-time planning

When the problem changes in real-time, the plan is adjusted in real-time.



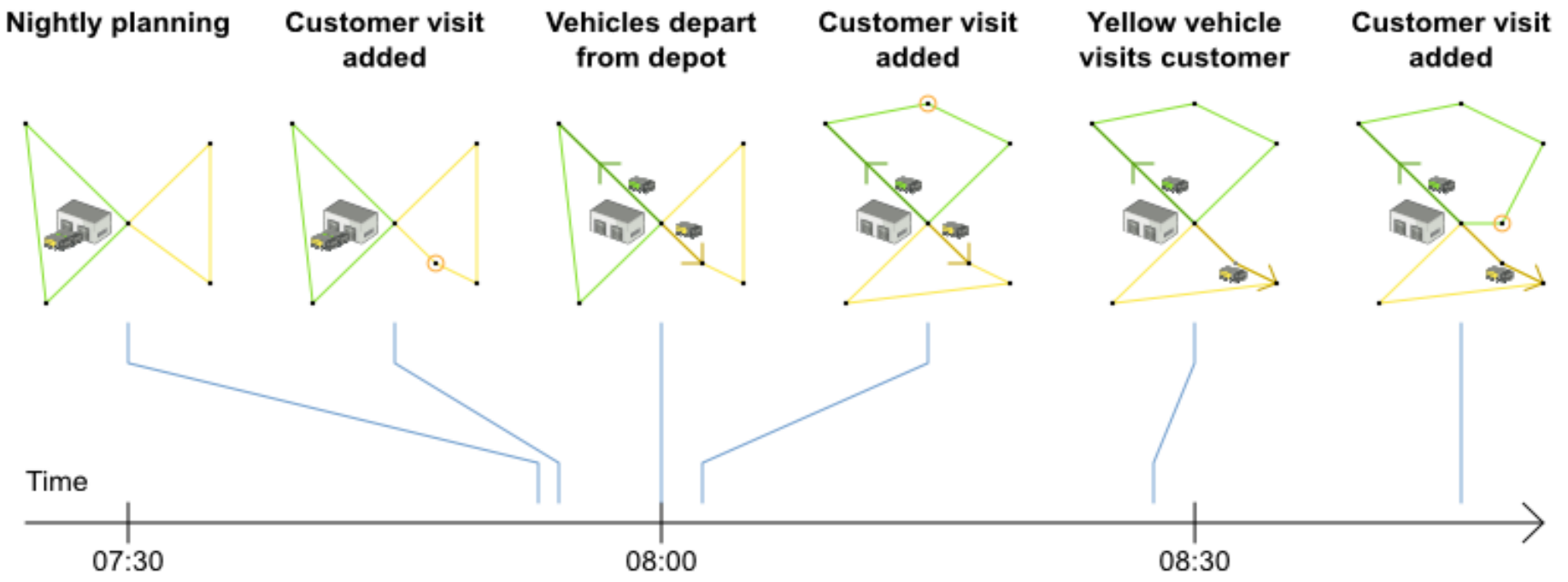
Real-time planning

When the problem changes in real-time, the plan is adjusted in real-time.



Real-time planning

When the problem changes in real-time, the plan is adjusted in real-time.

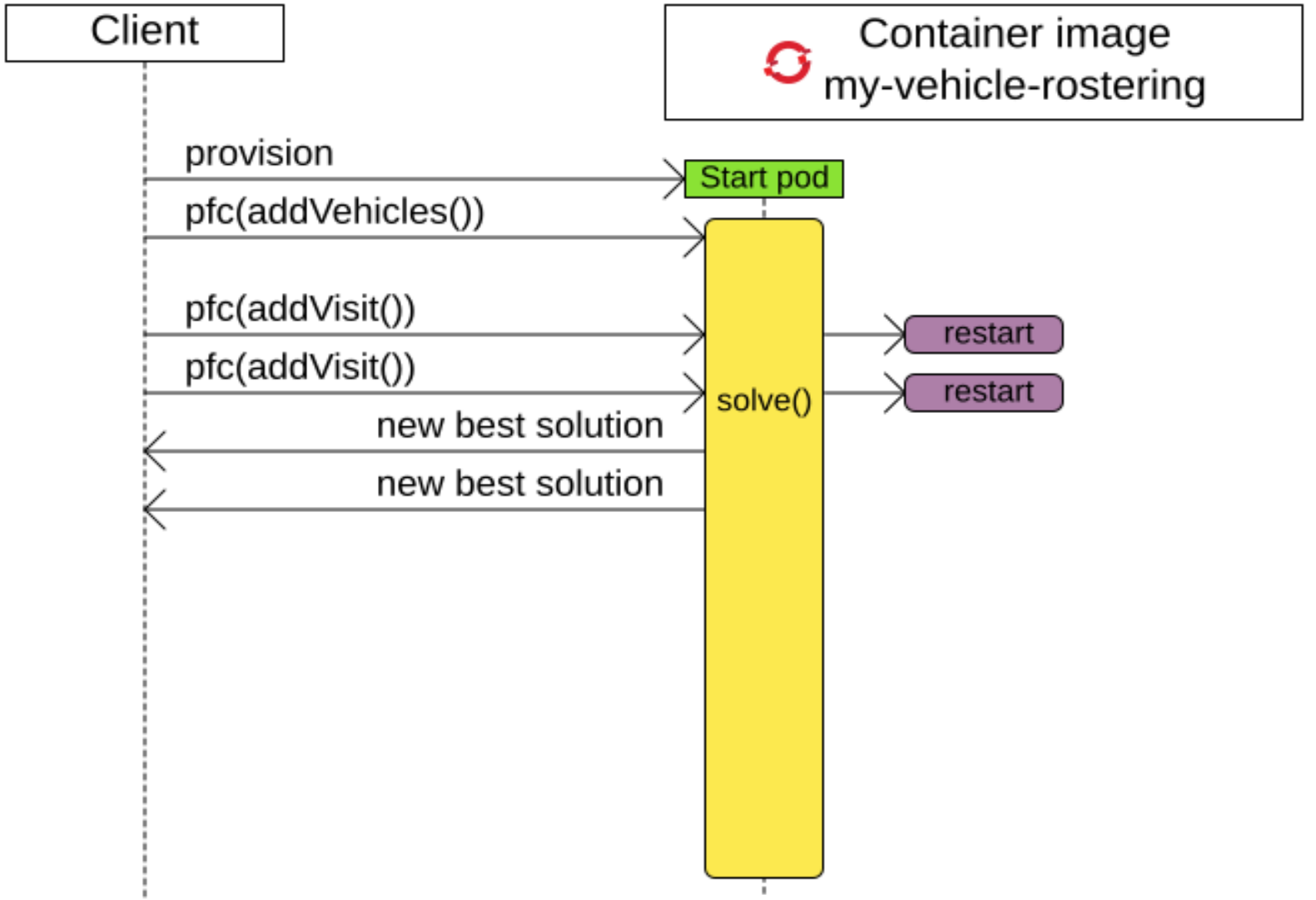


Real-time planning

- `solver.addProblemFactChange(...)`
- Daemon mode
  - Don't waste CPU time at night and during breaks

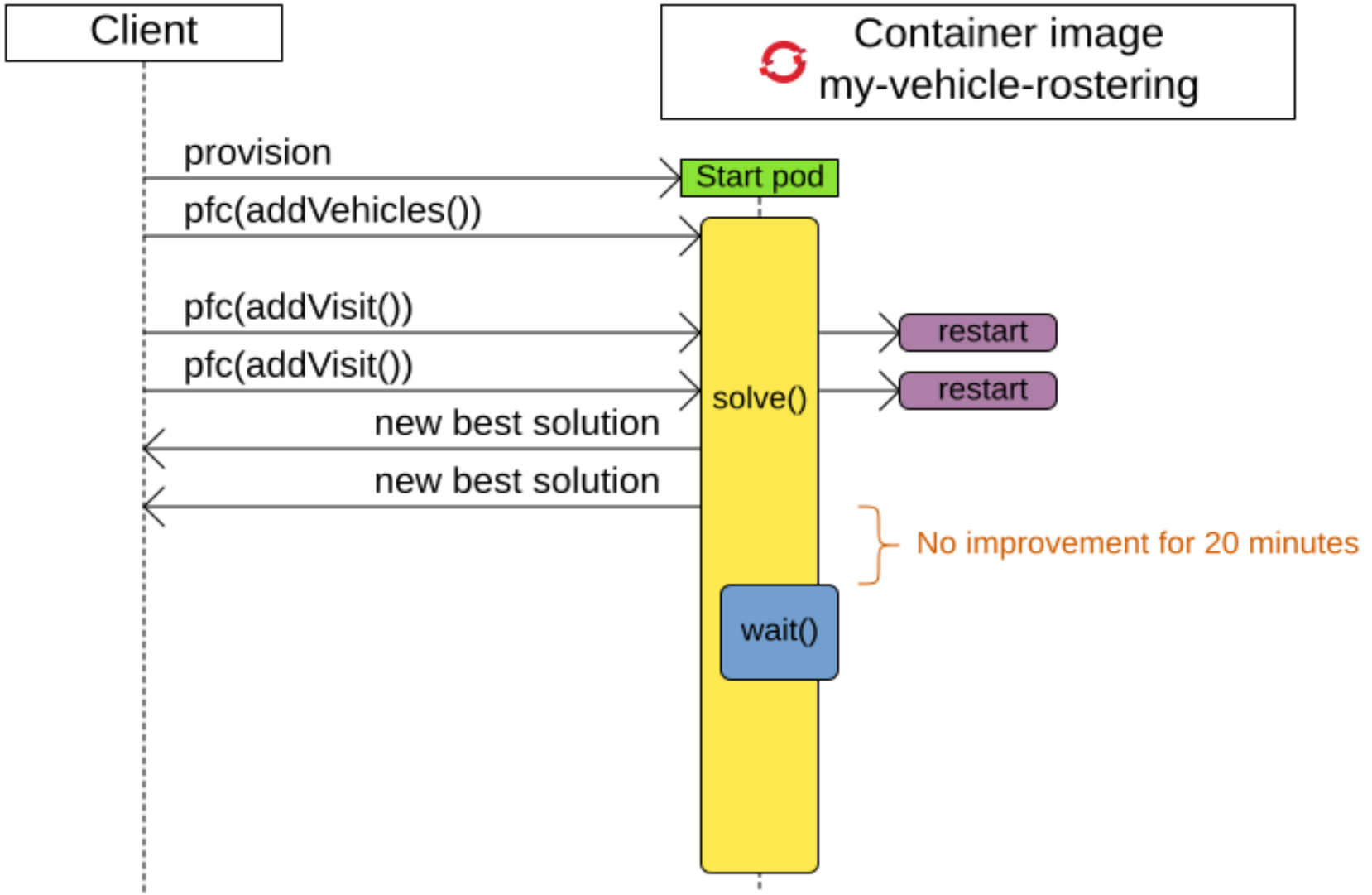
Real time planning cloud architecture

A statefull architecture that runs the solver in deamon mode.



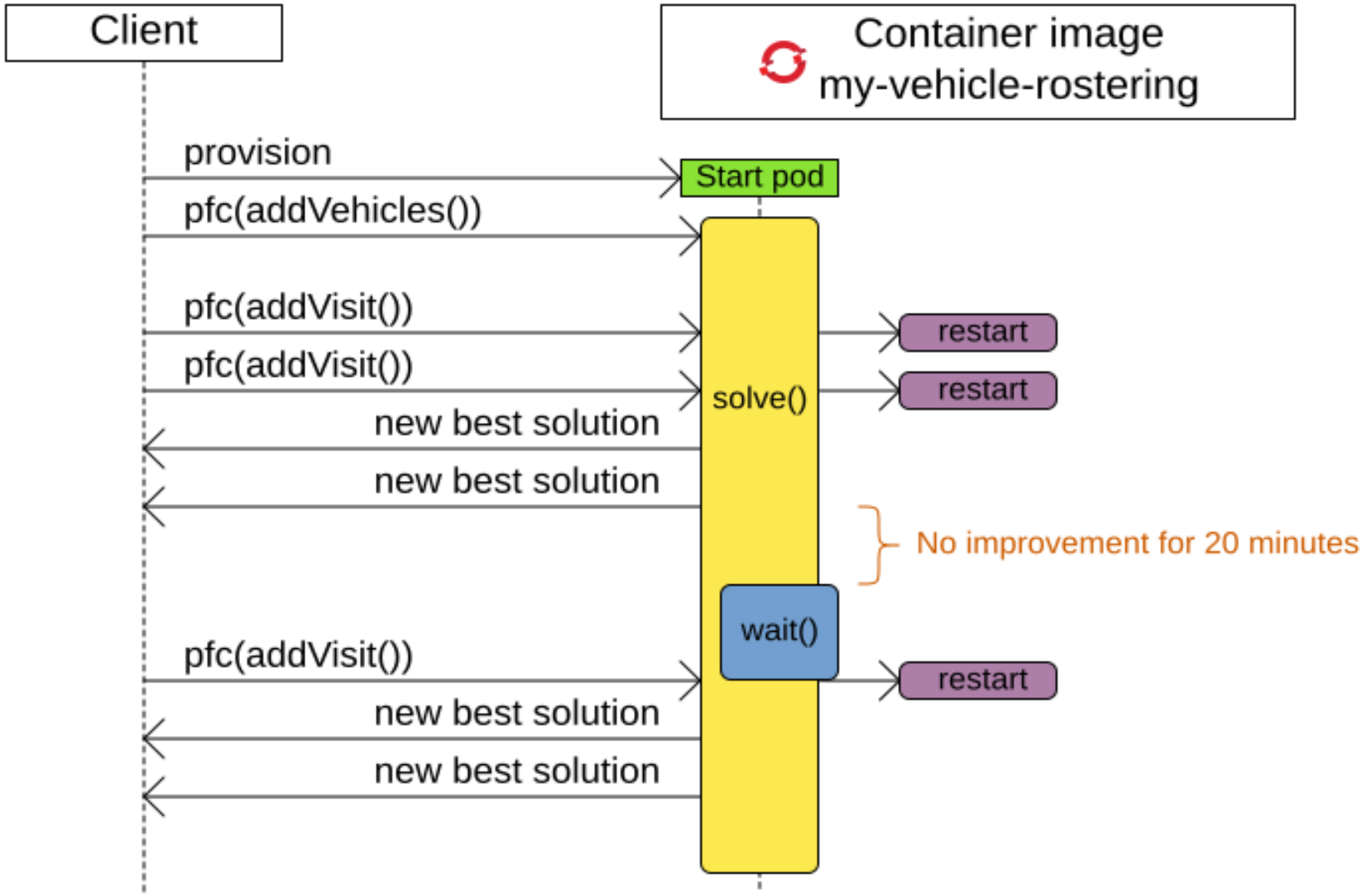
Real time planning cloud architecture

A statefull architecture that runs the solver in deamon mode.



Real time planning cloud architecture

A statefull architecture that runs the solver in deamon mode.

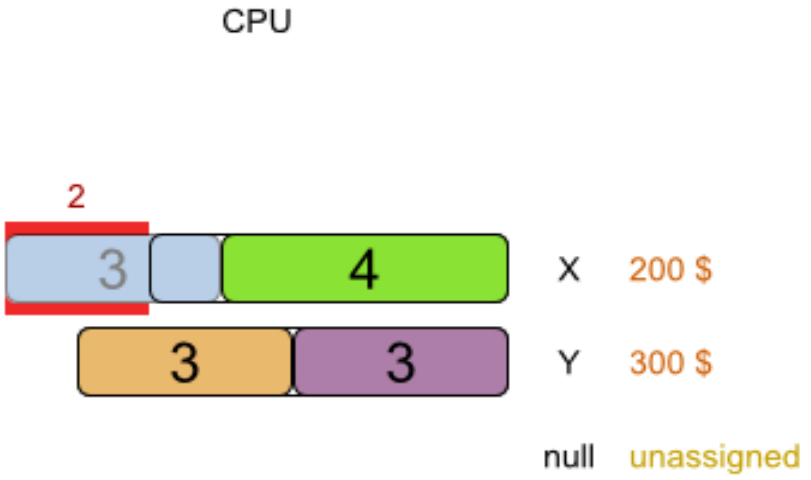


7) Do you have a resource shortage?

Overconstrained planning

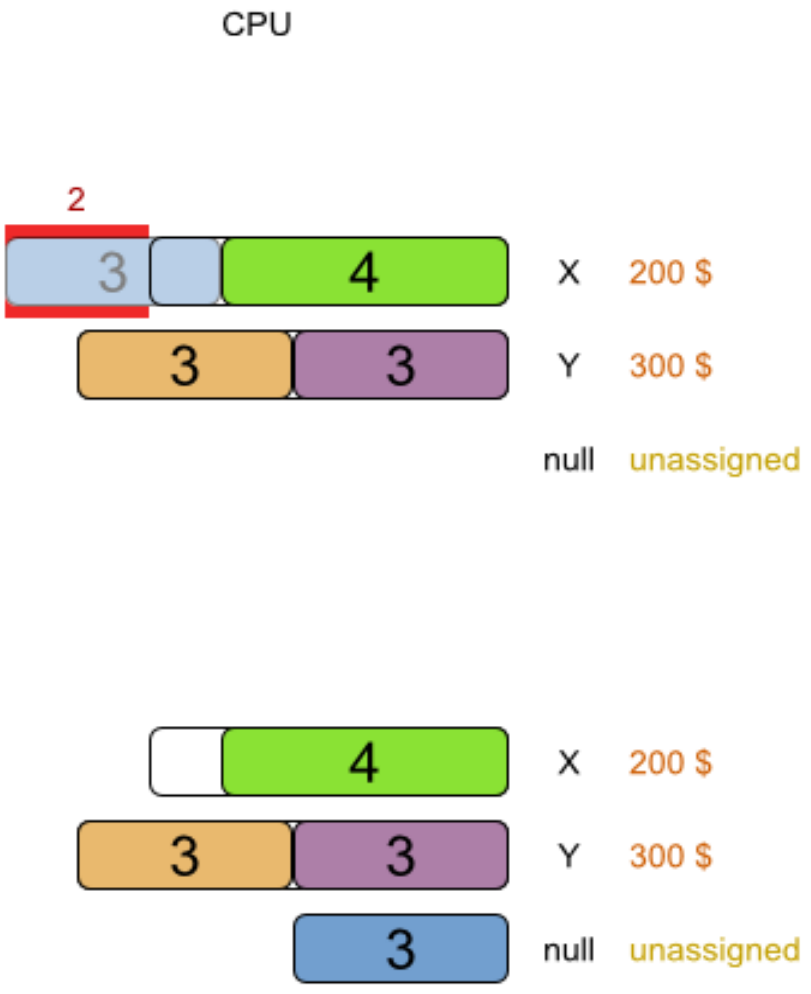
Overconstrained planning

If there is no feasible solution that assigns everything, then assign as many as possible.



Overconstrained planning

If there is no feasible solution that assigns everything, then assign as many as possible.



Overconstrained planning

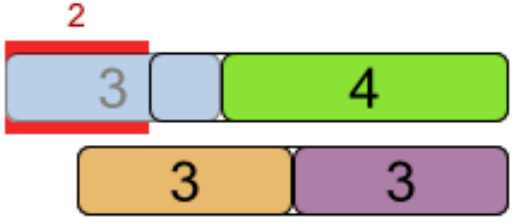
If there is no feasible solution that assigns everything, then assign as many as possible.

CPU

HardSoftScore

@PVariable()

Assign all entities

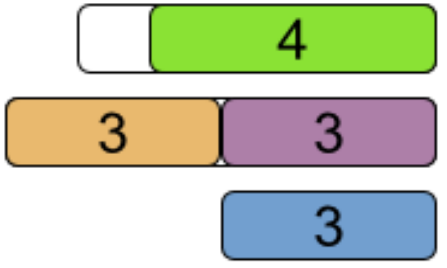


X 200 $

Y 300 $

-2hard / -500soft

null unassigned



X 200 $

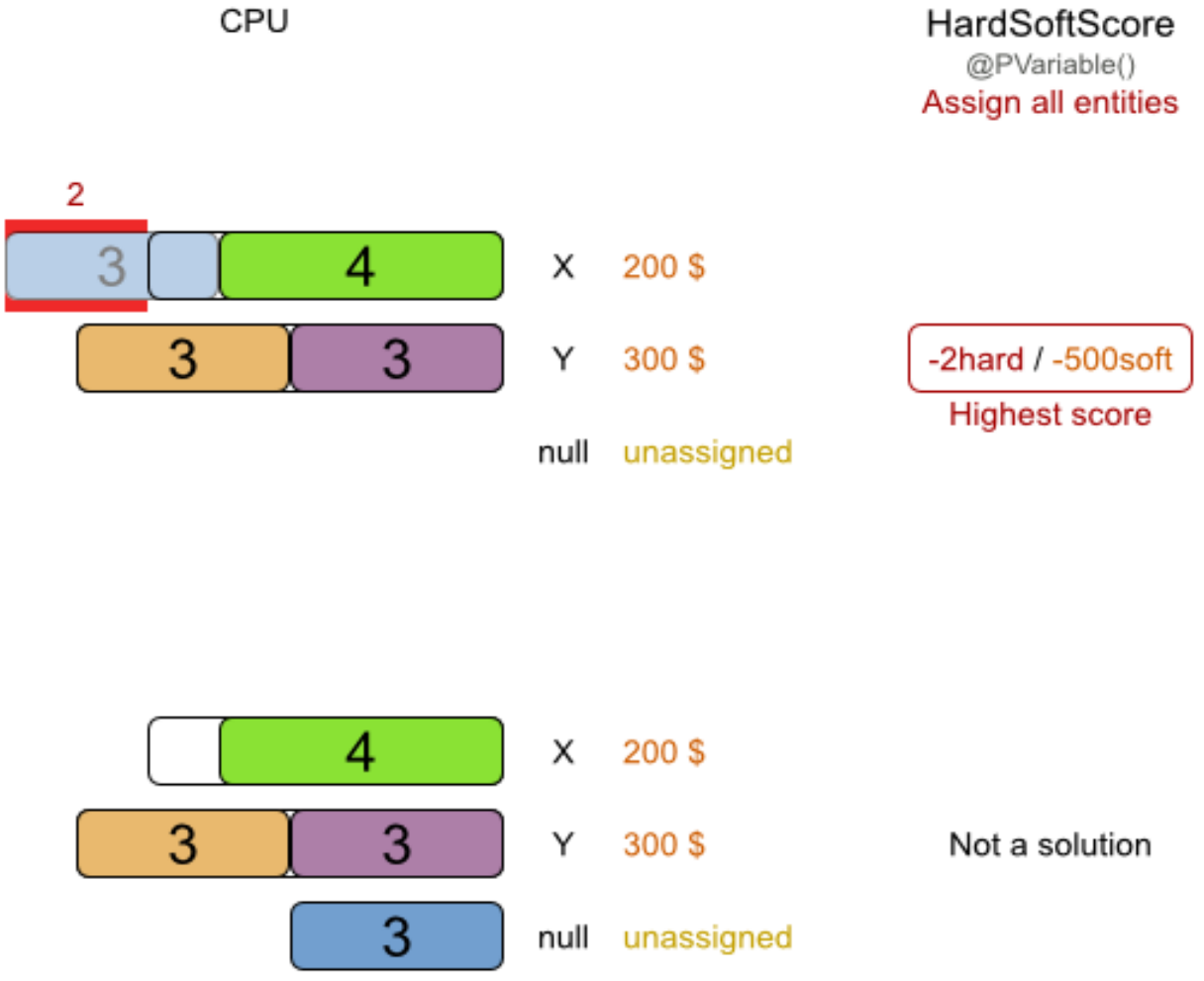
Y 300 $

Not a solution

null unassigned

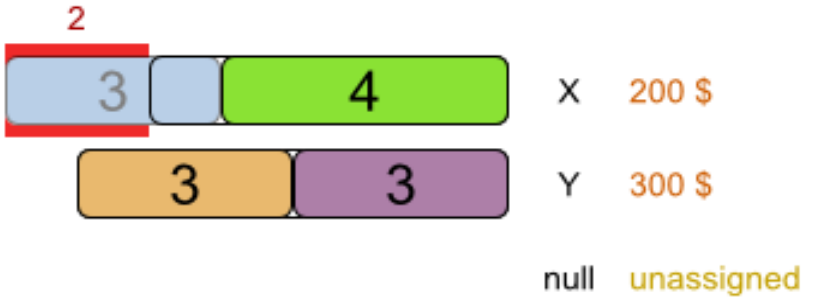
Overconstrained planning

If there is no feasible solution that assigns everything, then assign as many as possible.



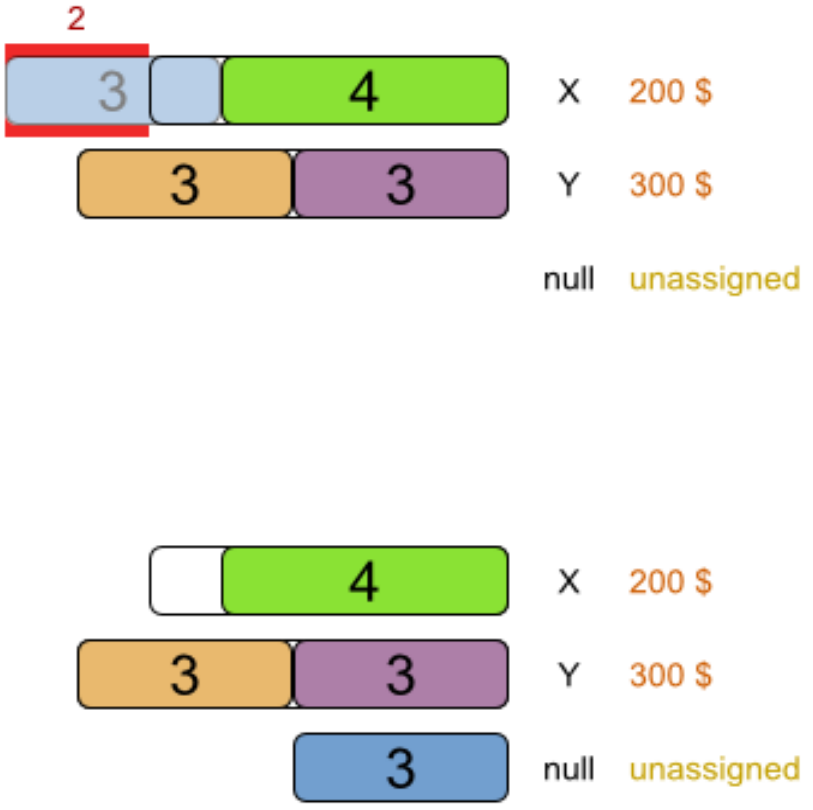
Overconstrained planning

If there is no feasible solution that assigns everything, then assign as many as possible.



Overconstrained planning

If there is no feasible solution that assigns everything, then assign as many as possible.



Overconstrained planning

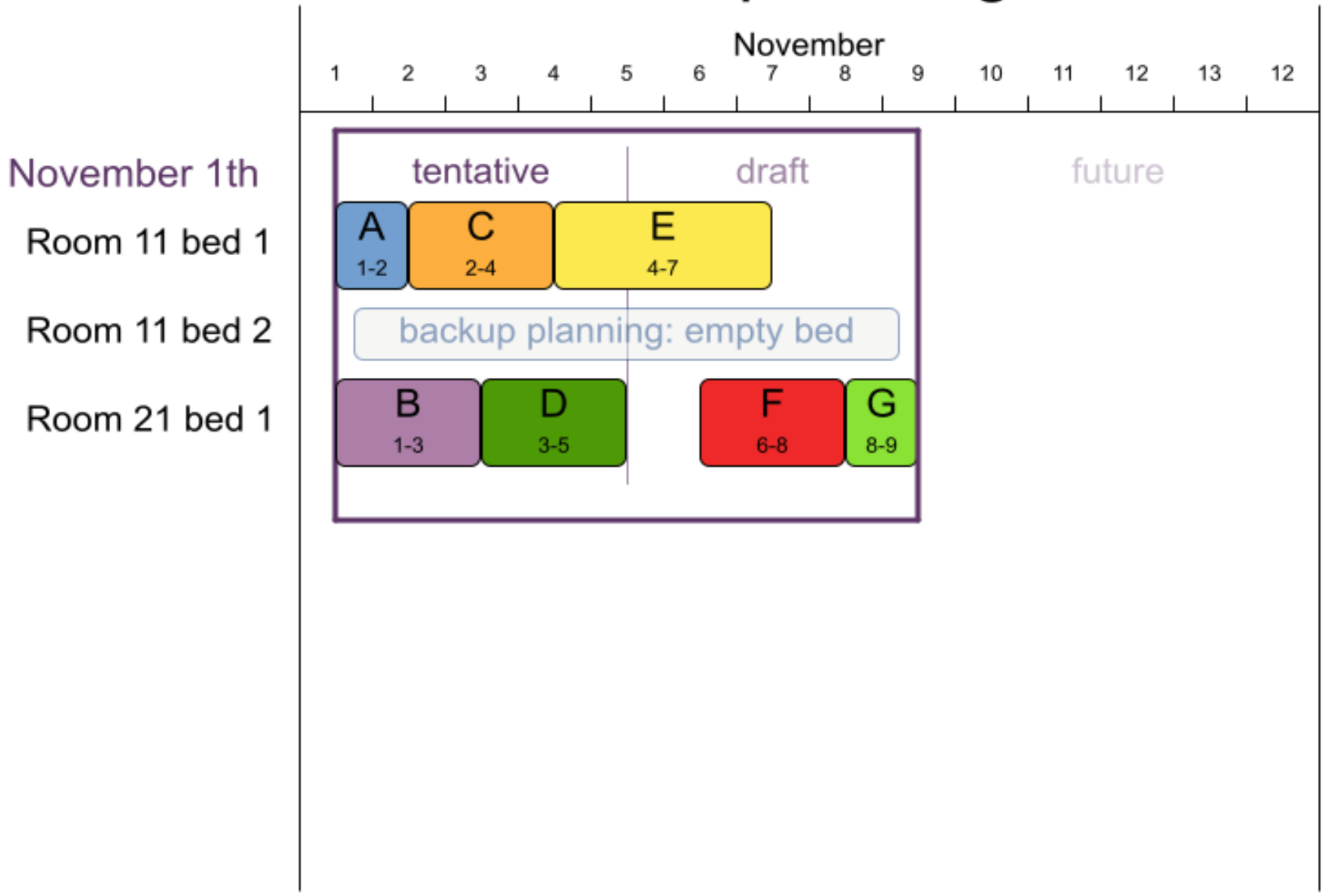
- Nullable
  - `@PlanningVariable(nullable = true)`
  - medium constraint penalizes unassigned entities
- Virtual resources
  - Add virtual resources to the input problem

```
class Employee {
  boolean virtual;
}
```

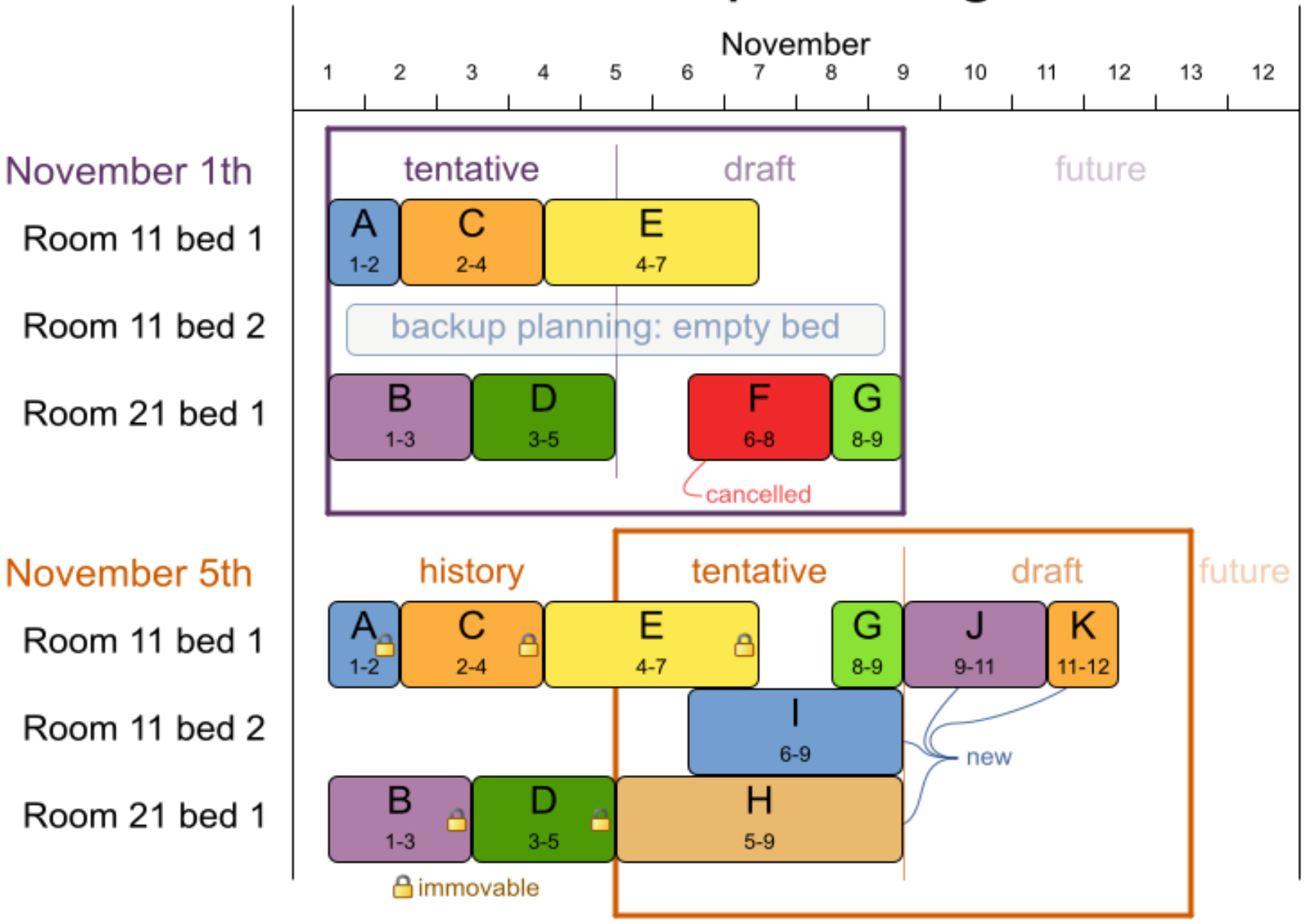
  - Soft constraint penalizes cost of virtual resources

8) Do you fear a worst case scenario?

Continuous planning



Continuous planning



Backup planning

- Don't make your plan a house of cards
- Add constraints to penalize such plans

Summary

Know your toolbox

- Multi-stage planning
- Continuous planning
- Non-disruptive replanning
- Real-time planning
- Overconstrained planning
- Backup planning

Know your toolbox

- Multi-stage planning
- Continuous planning
- Non-disruptive replanning
- Real-time planning
- Overconstrained planning
- Backup planning

Combine them as needed.

Q & A

Homepage	www.optaplanner.org (https://www.optap
Slides	www.optaplanner.org/learn/slides.html (https://www.optaplanner.org/learn/slides
User guide	www.optaplanner.org/learn/documentatic (https://www.optaplanner.org/learn/docu
Feedback	 @GeoffreyDeSmet (https://twitter.com/GeoffreyDeSmet)