STRANDS

Task scheduling and execution for long-term autonomy

Nick Hawes, University of Birmingham

n.a.hawes@cs.bham.ac.uk
http://www.cs.bham.ac.uk/~nah
@hawesie

http://strands-project.eu

SEVENTH FRAMEWORK PROGRAMME

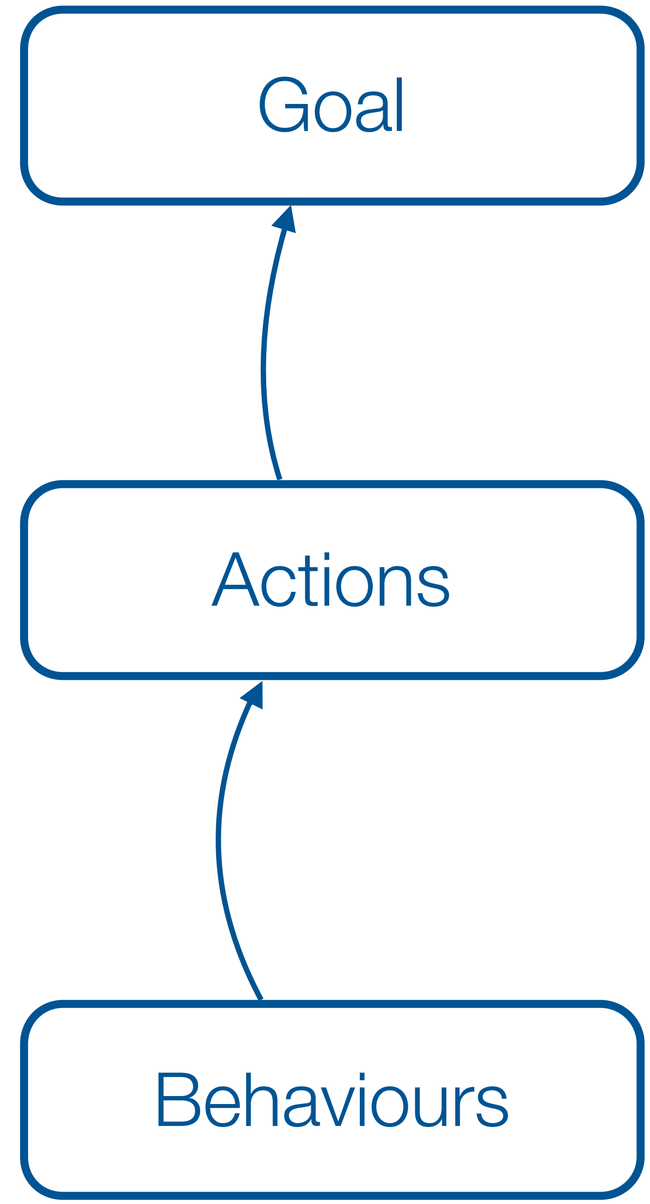
Long-Term Autonomy in Everyday Environments

Goal-Oriented Behaviour

We assume our robot has *goals* which are either provided by a user or generated by an internal system

Some system will provide a sequence of *actions* which achieve a given goal

Each *action* maps to an underlying *behaviour* which is the implementation of the *action* on the robot



Planning gives us an **ordering** of actions to achieve a goal

Real-world problems also need **time** and **resources**

Scheduling assigns time and resources to **jobs**

A **job** is a collection of **actions** with ordering constraints. Each action has a **duration**.

The aim is to make an **assignment of times to actions** (a **schedule**) in order to achieve some criterion, e.g. **makespan**.

Jobs({*AddEngine1* < *AddWheels1* < *Inspect1*},
{*AddEngine2* < *AddWheels2* < *Inspect2*})

Action(*AddEngine1*, DURATION:30)

Action(*AddEngine2*, DURATION:60)

Action(*AddWheels1*, DURATION:30)

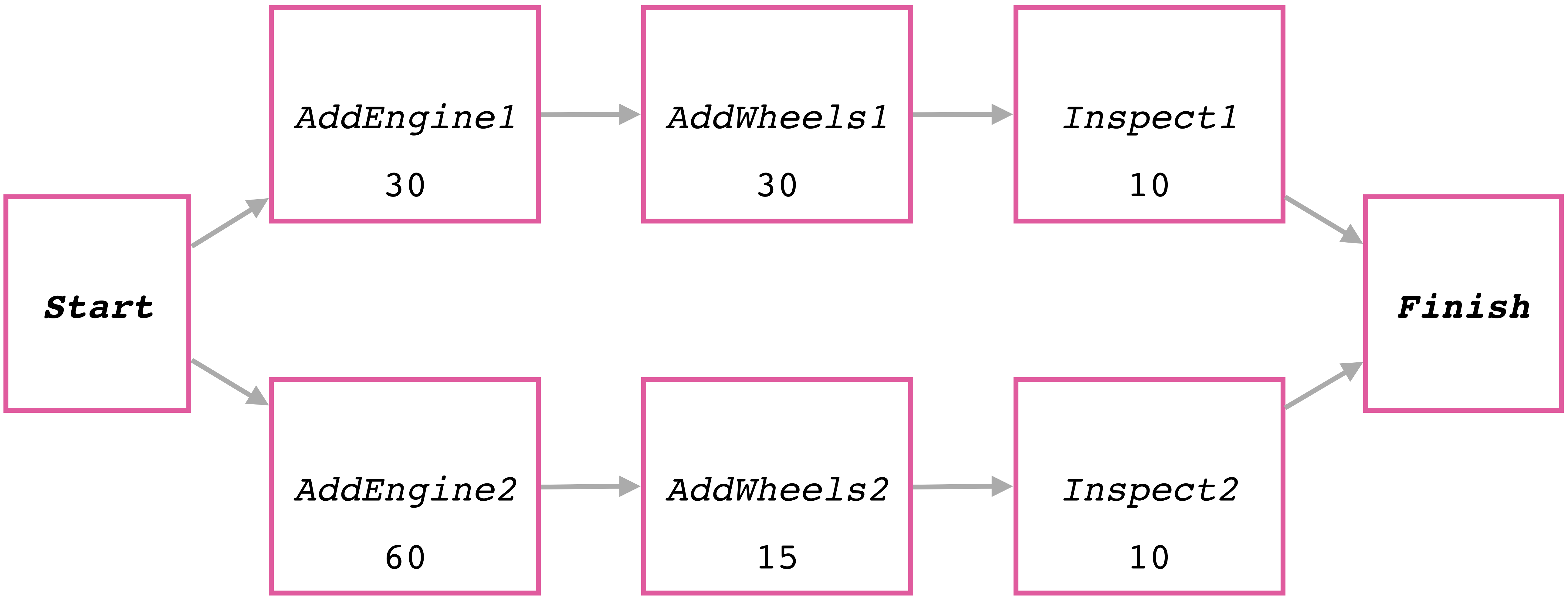
Action(*AddWheels1*, DURATION:15)

Action(*Inspect*, DURATION:10)

For each action assign

earliest start time ES and latest start time LS

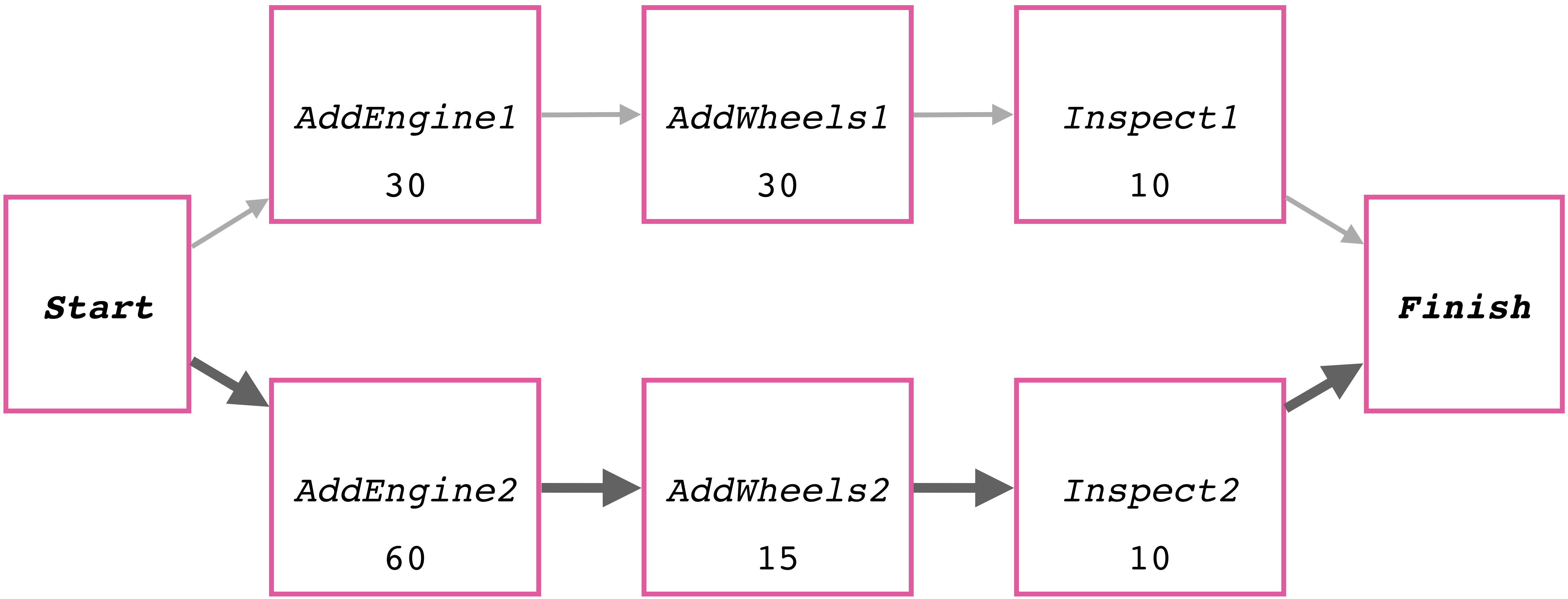
The **critical path** method: define the path through the action graph with longest duration



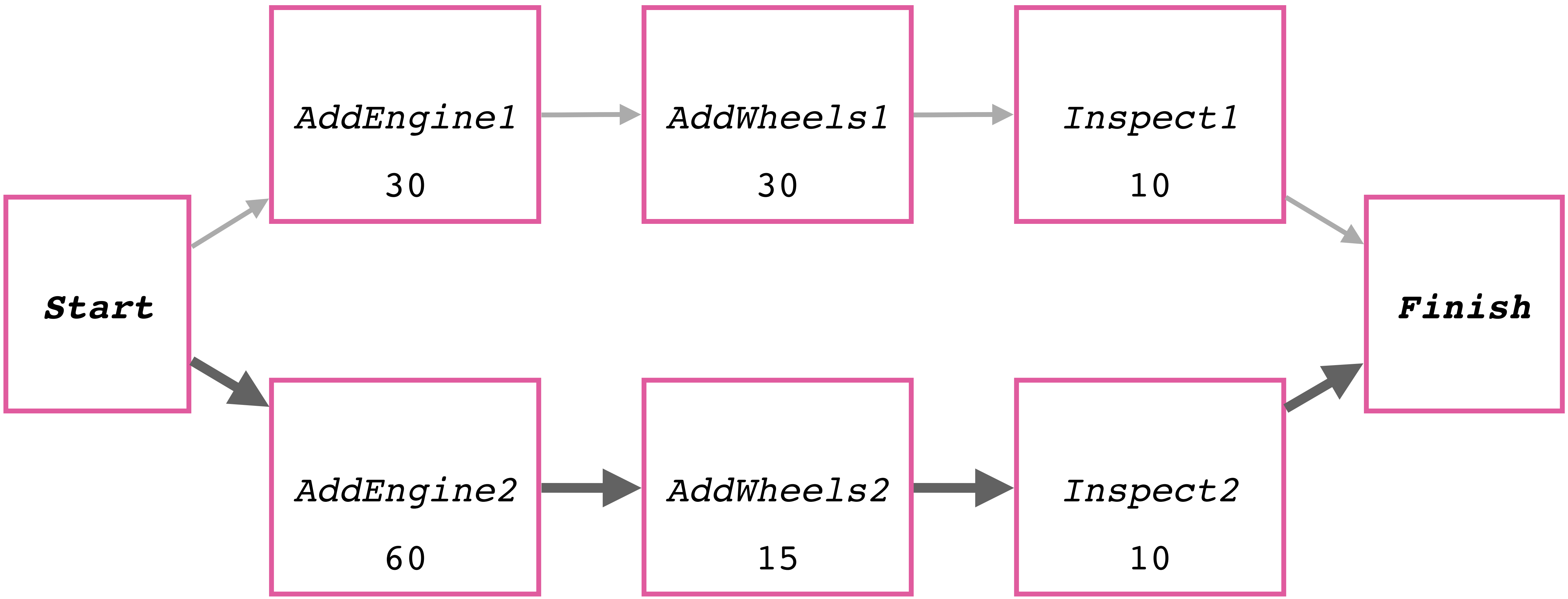
For each action assign

earliest start time ES and latest start time LS

The **critical path** method: define the path through the action graph with longest duration

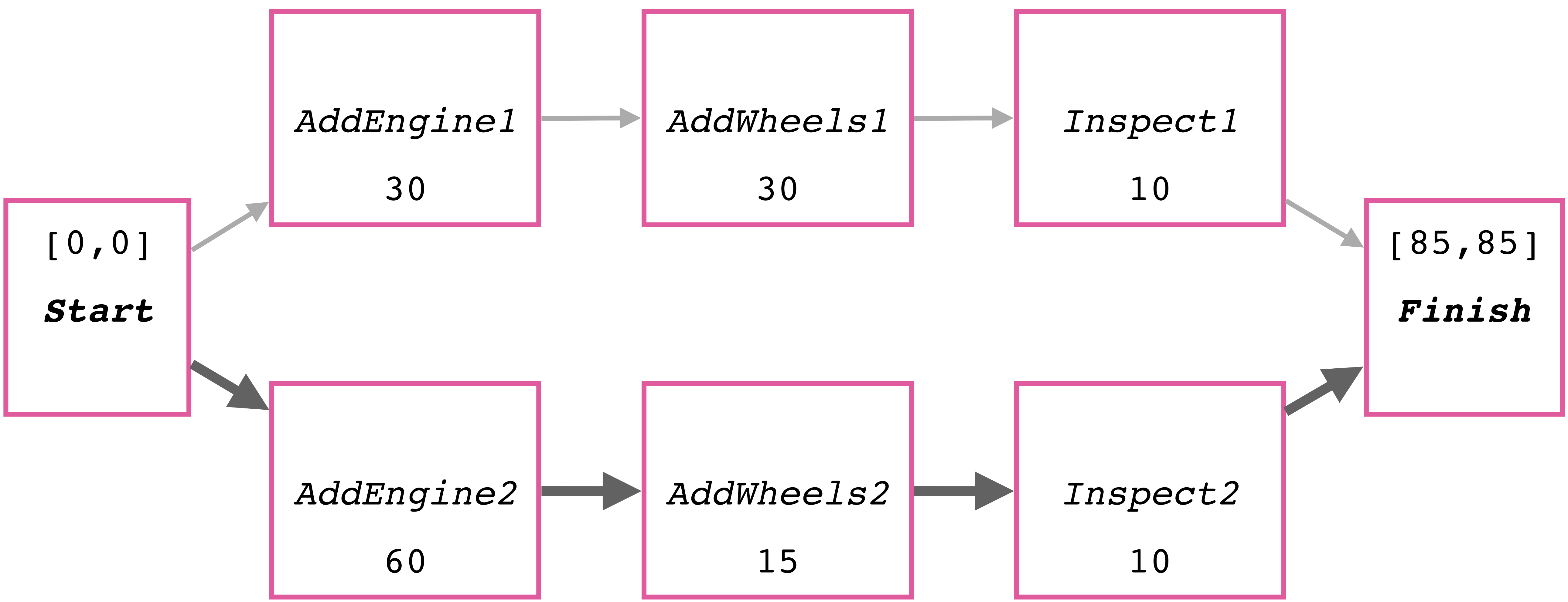


Use the **critical path** to define the overall length of the schedule, i.e. [ES, LS] for *Start* and *Finish*



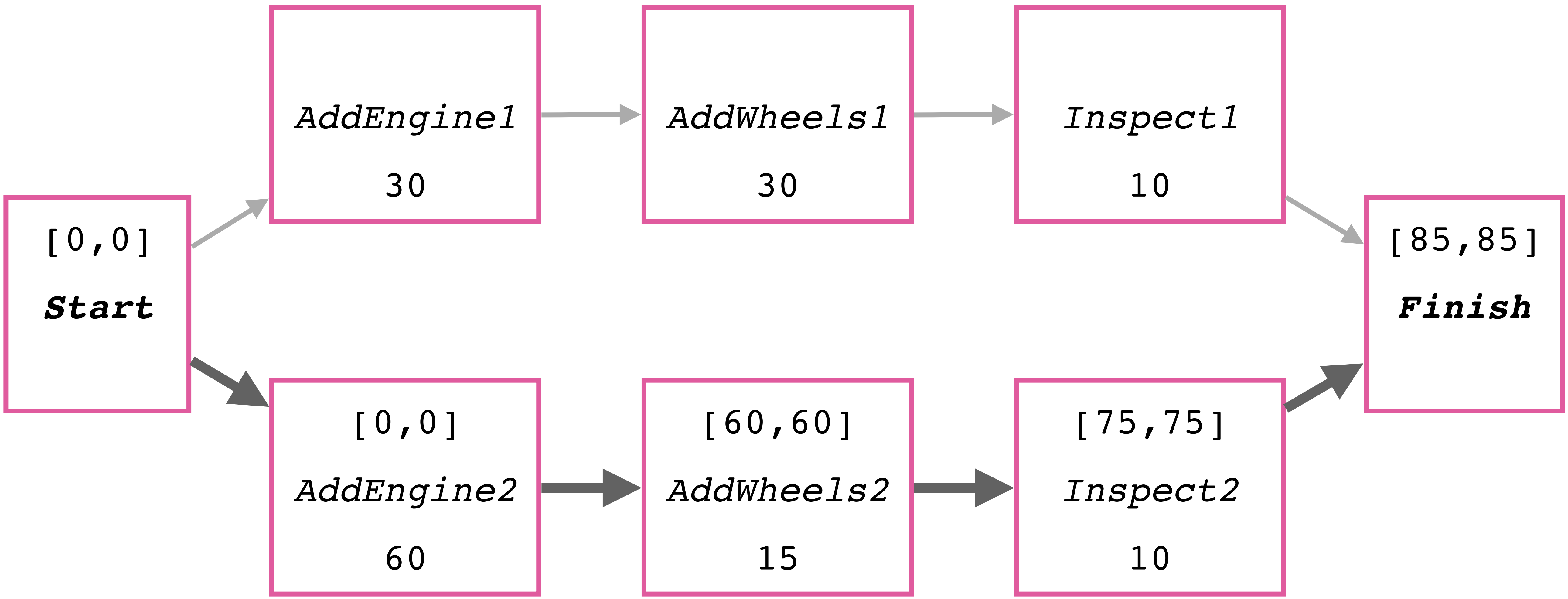
Use the **critical path** to define the overall length of the schedule, i.e. [ES, LS] for ***Start*** and ***Finish***

and for the actions on the critical path



Use the **critical path** to define the overall length of the schedule, i.e. [ES, LS] for ***Start*** and ***Finish***

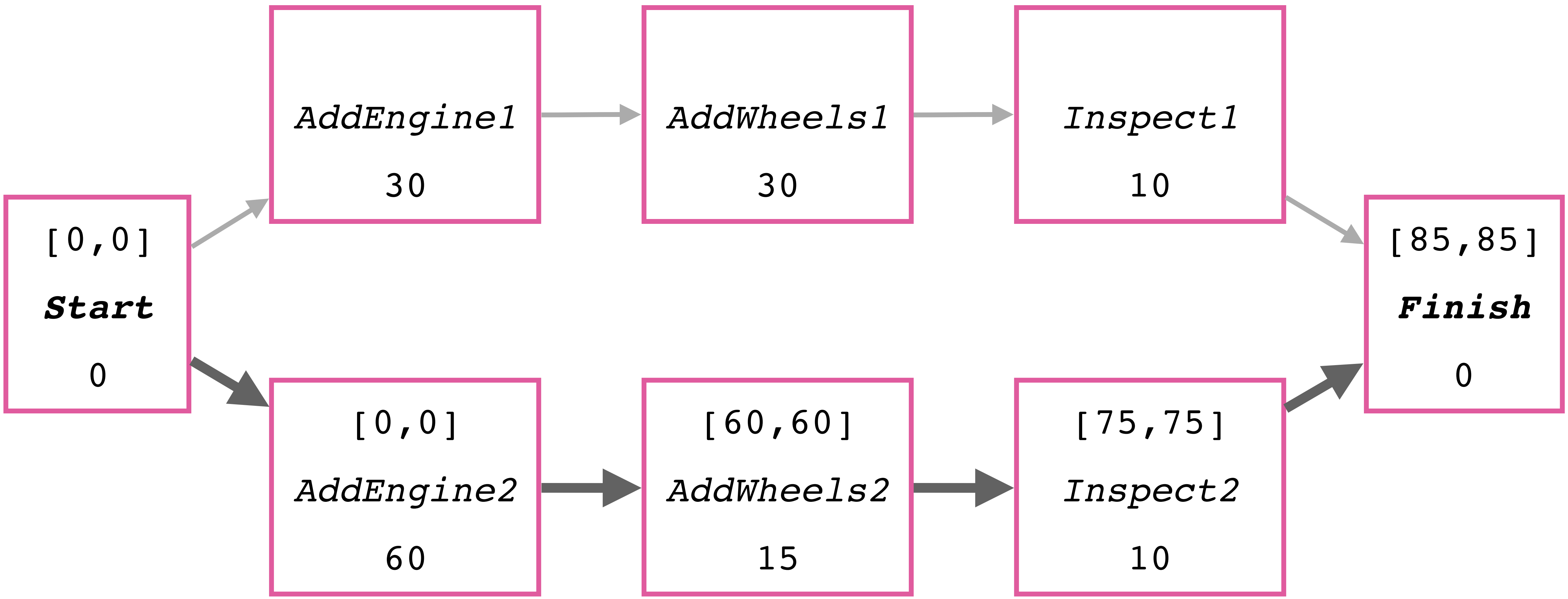
and for the actions on the critical path



Use these constraints to complete [ES, LS] for the remaining actions

$$ES(B) = \max_{A < B} ES(A) + Duration(A)$$

$$LS(A) = \min_{B > A} LS(B) - Duration(A)$$

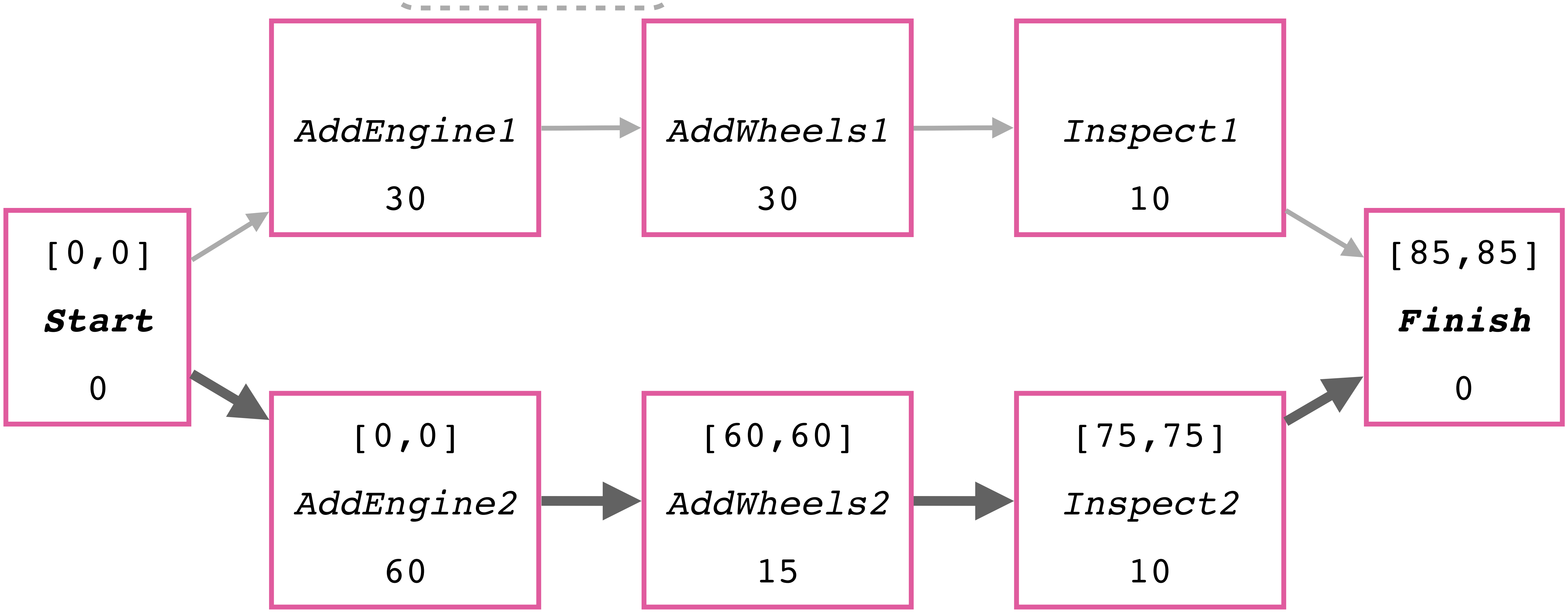


Action with *latest* earliest start preceding ***B***

Action with *earliest* latest start preceding ***A***

$$ES(B) = \max_{A < B} ES(A) + Duration(A)$$

$$LS(A) = \min_{B > A} LS(B) - Duration(A)$$

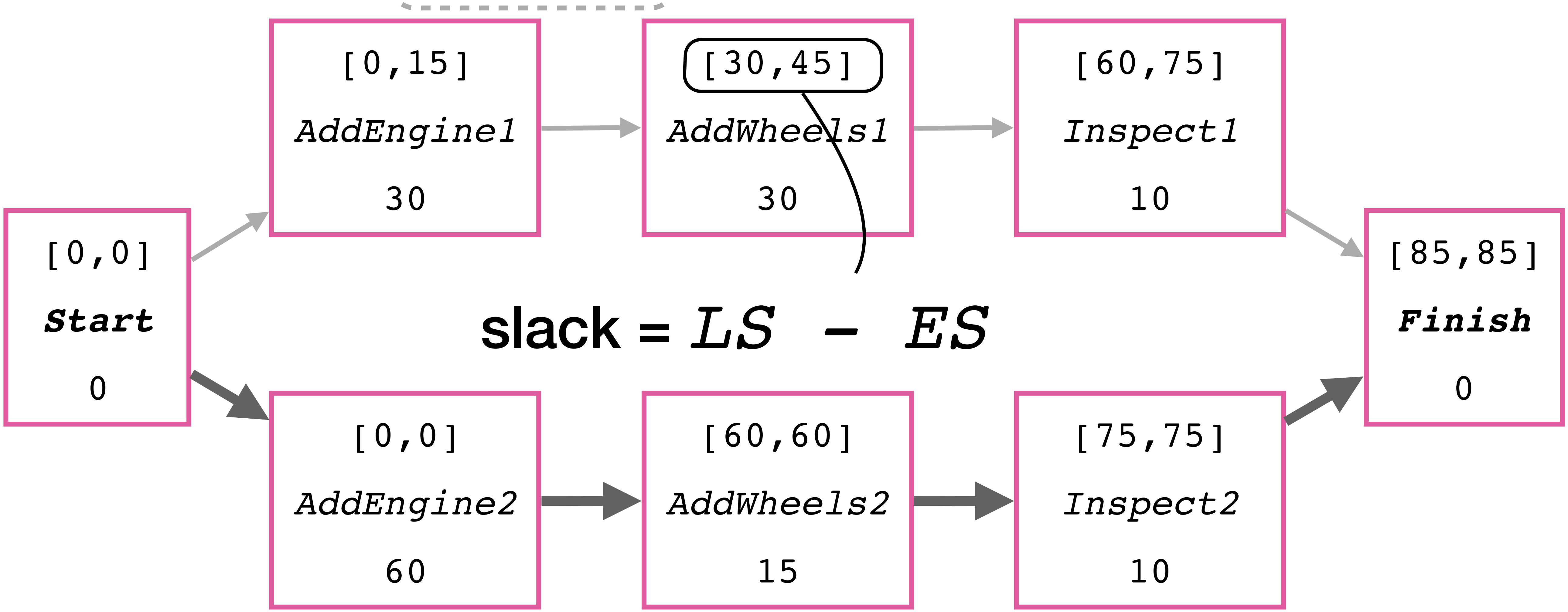


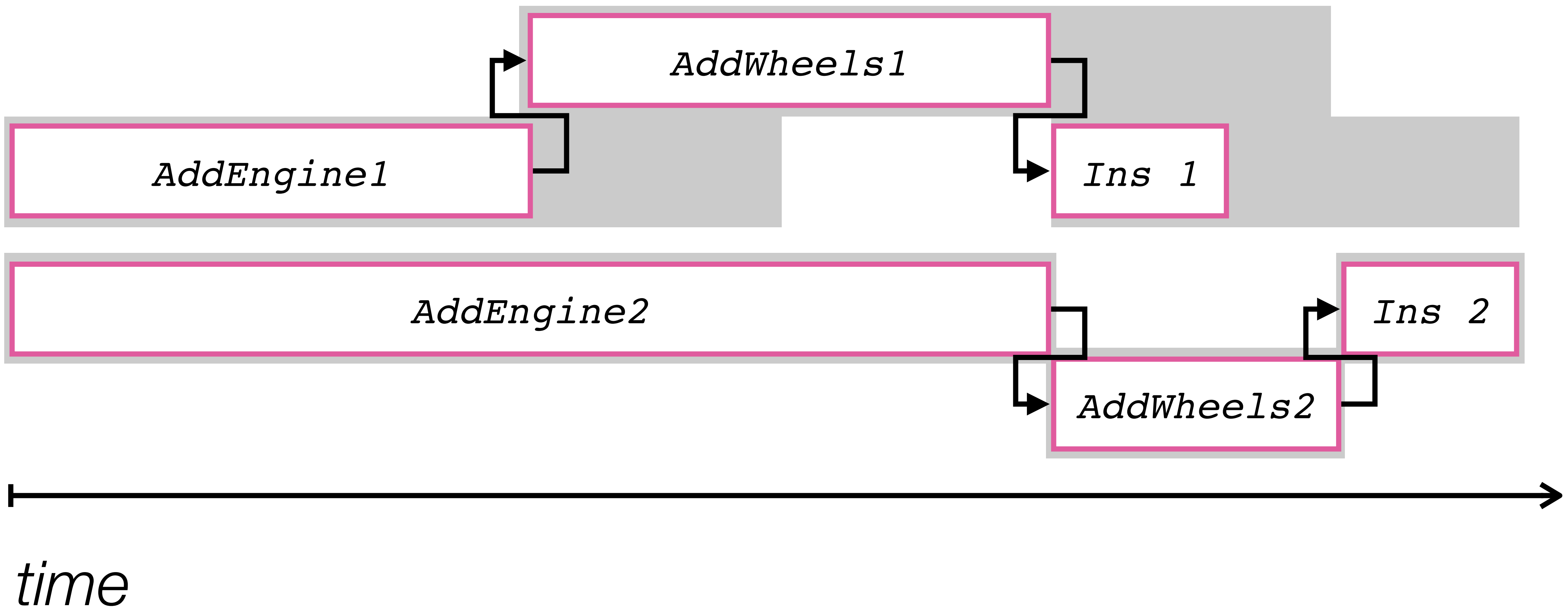
Action with *latest* earliest start preceding ***B***

Action with *earliest* latest start preceding ***A***

$$ES(B) = \boxed{\max_{A \prec B}} ES(A) + Duration(A)$$

$$LS(A) = \boxed{\min_{B \succ A}} LS(B) - Duration(A)$$




AddWheels1
AddEngine1
Ins 1
AddEngine2
Ins 2
AddWheels2
time

$$ES(Start) = 0$$

$$ES(B) = \max_{A < B} ES(A) + Duration(A)$$

$$LS(Finish) = ES(Finish)$$

$$LS(A) = \min_{B > A} LS(B) - Duration(A)$$

Scheduling ordered tasks with no additional constraints is pretty easy: a **conjunction** of **linear constraints**

Solve with dynamic programming, integer programming etc.

Jobs({*AddEngine1* < *AddWheels1* < *Inspect1*},
{*AddEngine2* < *AddWheels2* < *Inspect2*})

Action(*AddEngine1*, DURATION:30)

Action(*AddEngine2*, DURATION:60)

Action(*AddWheels1*, DURATION:30)

Action(*AddWheels1*, DURATION:15)

Action(*Inspect*, DURATION:10)

Jobs({*AddEngine1* < *AddWheels1* < *Inspect1*},
{*AddEngine2* < *AddWheels2* < *Inspect2*})

Resources(*EngineHoists*(1), *WheelStations*(1),
Inspectors(2), *LugNuts*(500))

Action(*AddEngine1*, DURATION:30)

Action(*AddEngine2*, DURATION:60)

Action(*AddWheels1*, DURATION:30)

Action(*AddWheels1*, DURATION:15)

Action(*Inspect*, DURATION:10)

```
Jobs( {AddEngine1 < AddWheels1 < Inspect1},
      {AddEngine2 < AddWheels2 < Inspect2})

Resources( EngineHoists(1), WheelStations(1),
           Inspectors(2), LugNuts(500))

Action( AddEngine1, DURATION:30,
        USE: EngineHoists(1))

Action( AddEngine2, DURATION:60,
        USE: EngineHoists(1)))

Action( AddWheels1, DURATION:30,
  CONSUME: LugNuts(20), USE: WheelStations(1)))

Action( AddWheels1, DURATION:15,
  CONSUME: LugNuts(20), USE: WheelStations(1)))

Action( Inspect, DURATION:10,
        USE: Inspectors(1)))
```

$$ES(Start) = 0$$

$$ES(B) = \max_{A < B} ES(A) + Duration(A)$$

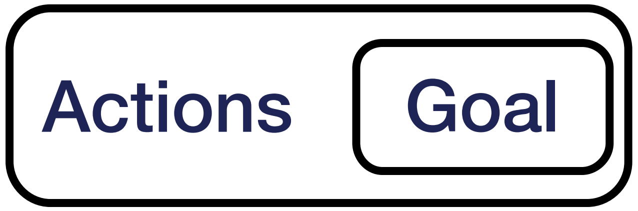
$$LS(Finish) = ES(Finish)$$

$$LS(A) = \min_{B > A} LS(B) - Duration(A)$$

Now we have to include **disjunctions** so we're back to an NP-hard problem.

Scheduling for a **mobile robot** introduces both **challenges** and **simplifications**.

A ***task*** is a single, indivisible unit of behaviour to achieve a goal (often implicit)



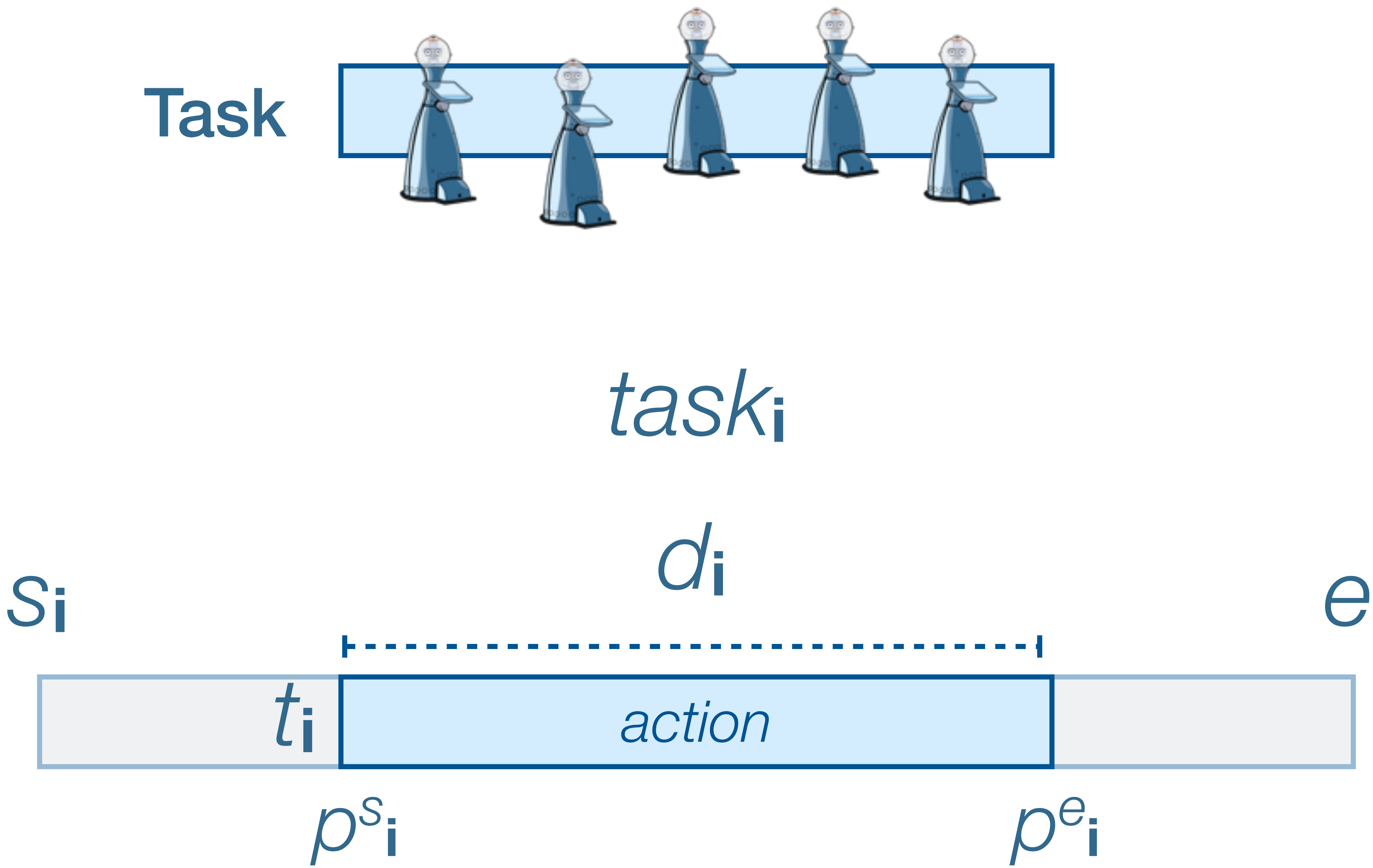
Task Scheduling for Mobile Robots Using Interval Algebra

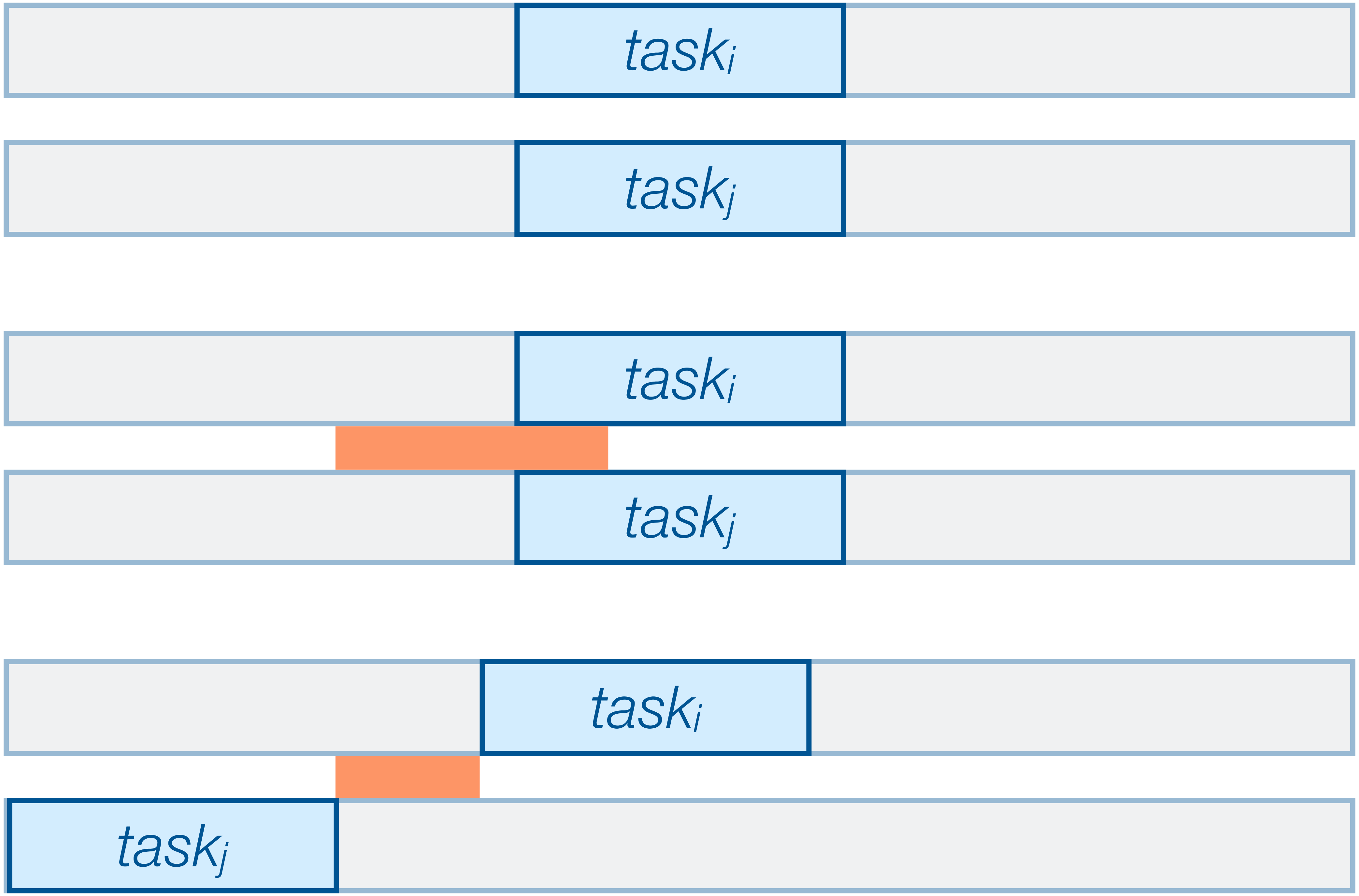
Mudrová and Hawes. In, *ICRA '15*.

How to tell a robot **what time** to do something?

Not just order, but precise starting times (e.g. 14:02)

Considering up to 100 tasks


Task
$task_i$
d_i
s_i
e_i
t_i
action
p^s_i
p^e_i



$$\forall \mathbf{i} : min \sum (t_{\mathbf{i}} - s_{\mathbf{i}})$$

*Coltin et al.**

Scheduling using mixed-integer programming

$$s_i \leq t_i \land (t_i + d_i) \leq e_i$$

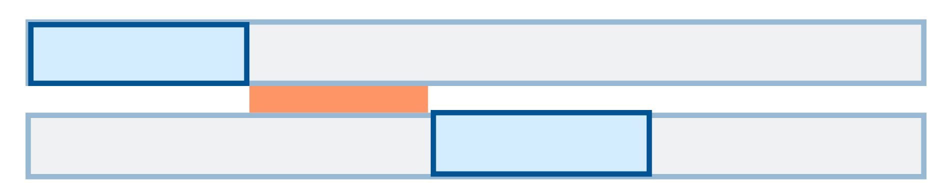
* e.g. Brian Coltin, Manuela Veloso, and Rodrigo Ventura. *Dynamic User Task Scheduling for Mobile Robots*. In Proceedings of the AAAI Workshop on Automated Action Planning for Autonomous Mobile Robots at AAAI. 2011.

*Coltin et al.** Scheduling using mixed-integer programming

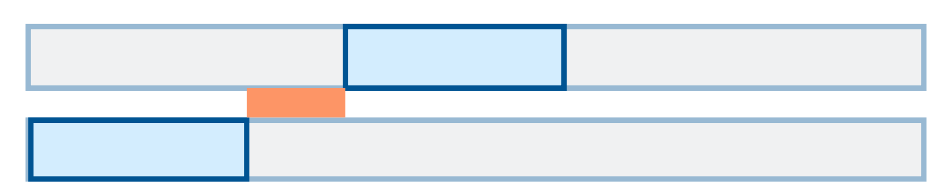
$$s_{\mathbf{i}} \leq t_{\mathbf{i}} \wedge (t_{\mathbf{i}} + d_{\mathbf{i}}) \leq e_{\mathbf{i}}$$

$$\forall_{\mathbf{i},\mathbf{j}} : t_{\mathbf{i}} + d_{\mathbf{i}} + time(p^{e}{}_{\mathbf{i}}, p^{s}{}_{\mathbf{j}}) \leq t_{\mathbf{j}}$$

or

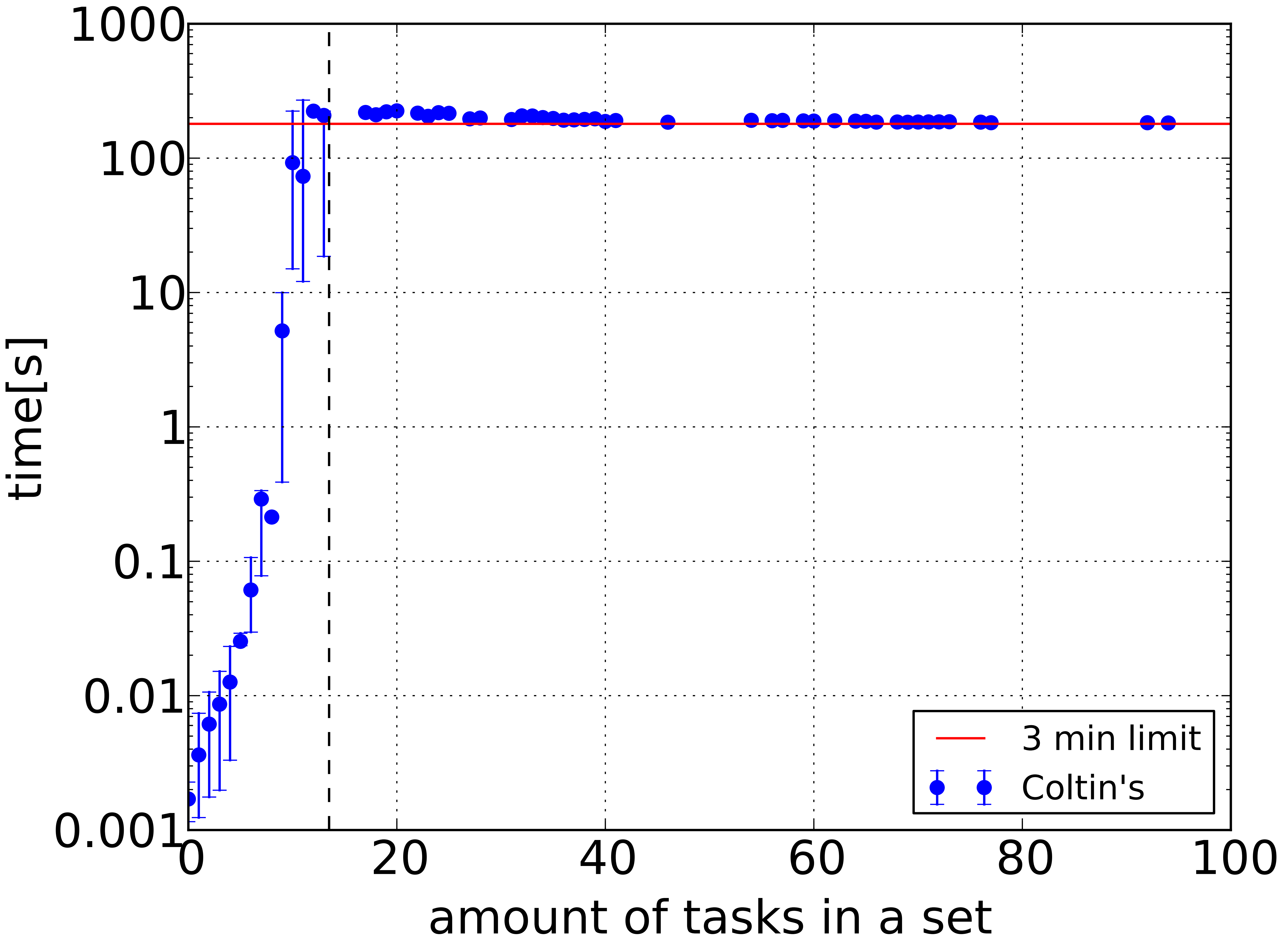


$$\forall_{\mathbf{i},\mathbf{j}} : t_{\mathbf{j}} + d_{\mathbf{j}} + time(p^{e}{}_{\mathbf{j}}, p^{s}{}_{\mathbf{i}}) \leq t_{\mathbf{i}}$$



$$\forall \mathbf{i} : min \sum (t_{\mathbf{i}} - s_{\mathbf{i}})$$

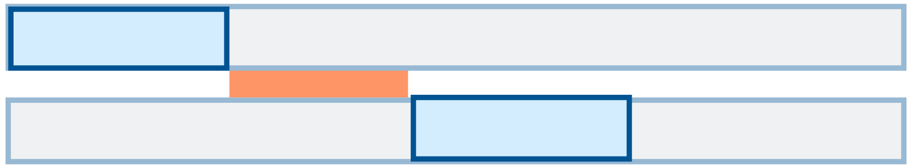
* e.g. Brian Coltin, Manuela Veloso, and Rodrigo Ventura. *Dynamic User Task Scheduling for Mobile Robots*. In Proceedings of the AAAI Workshop on Automated Action Planning for Autonomous Mobile Robots at AAAI. 2011.



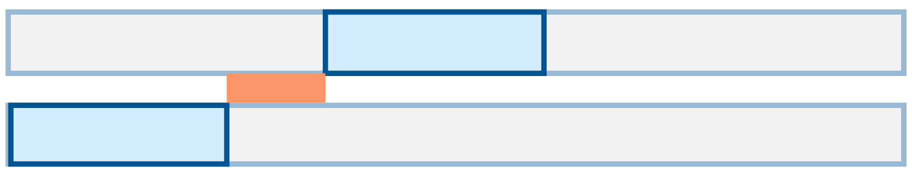
time[s]
1000
100
10
1
0.1
0.01
0.001
0
20
40
60
80
100
amount of tasks in a set
3 min limit
Coltin's

$$\forall_{\mathbf{i},\mathbf{j}} : t_{\mathbf{i}} + d_{\mathbf{i}} + time(p^{e}_{\mathbf{i}}, p^{s}_{\mathbf{j}}) \leq t_{\mathbf{j}}$$

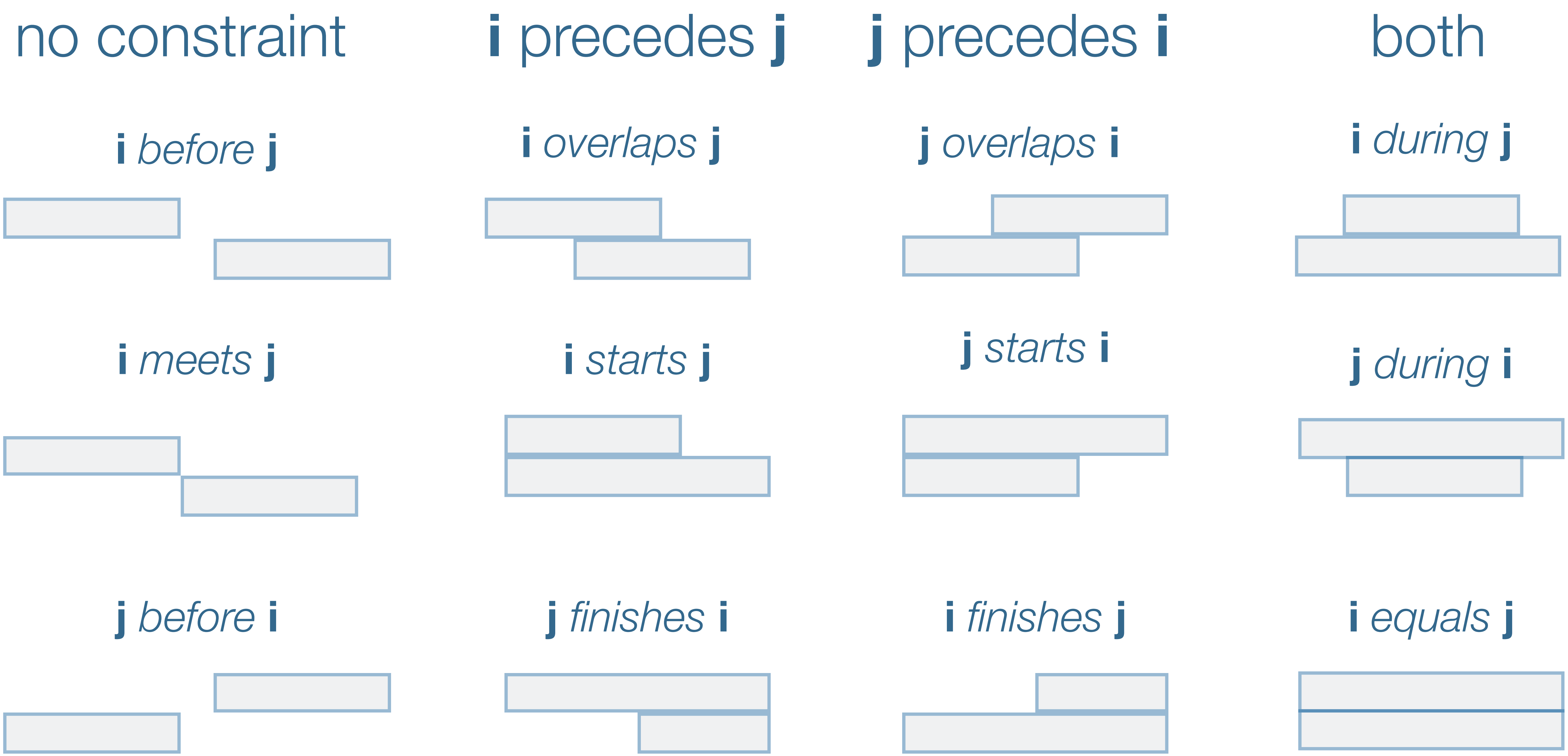
or

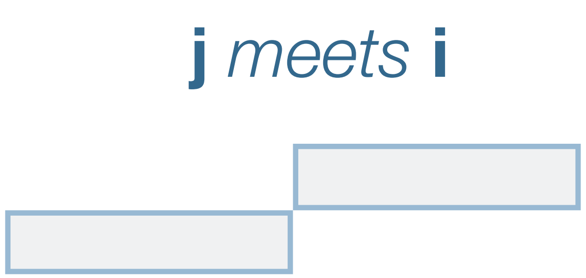


$$\forall_{\mathbf{i},\mathbf{j}} : t_{\mathbf{j}} + d_{\mathbf{j}} + time(p^{e}_{\mathbf{j}}, p^{s}_{\mathbf{i}}) \leq t_{\mathbf{i}}$$



Reasoning about **tasks' time windows**: $s_i \; e_i$ vs $s_j \; e_j$



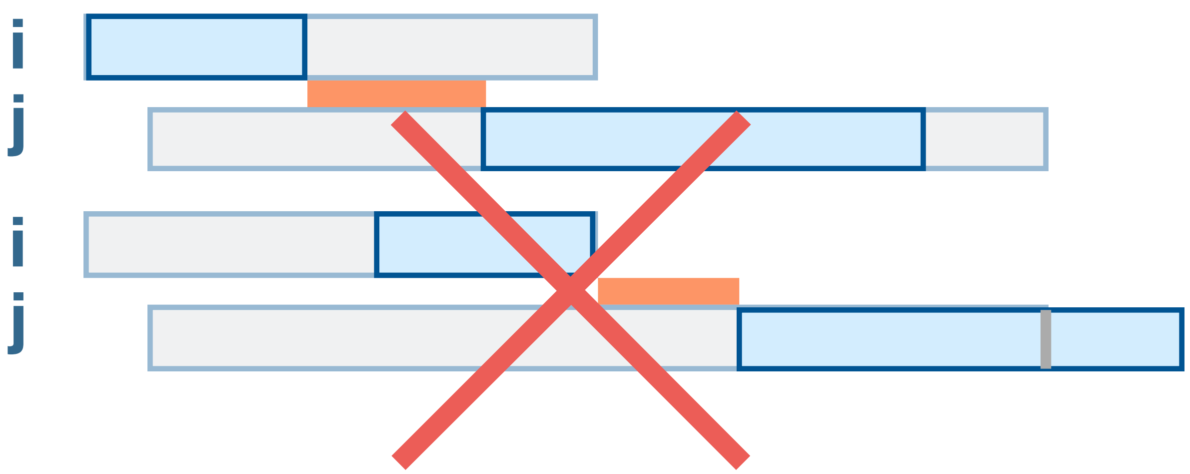


Allen's Interval Algebra

J. F. Allen. *Maintaining knowledge about temporal intervals. Communications of the ACM*, 26(11):832– 843, 1983.

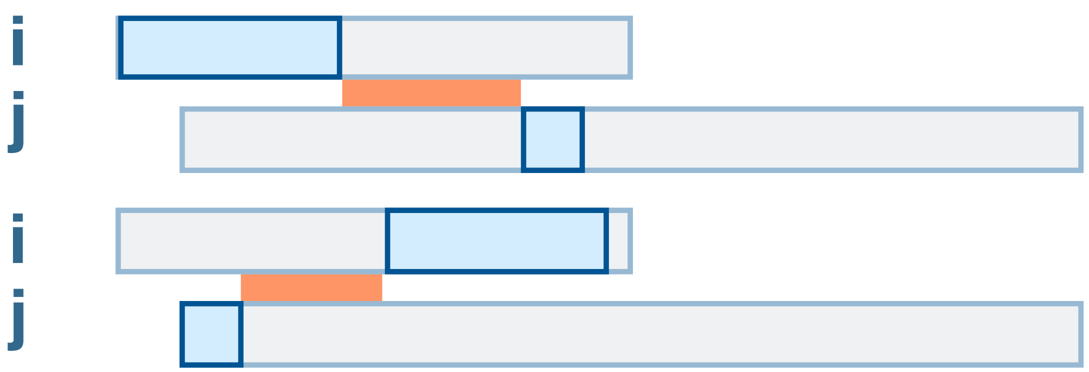
i *before* **j**

no order constraint



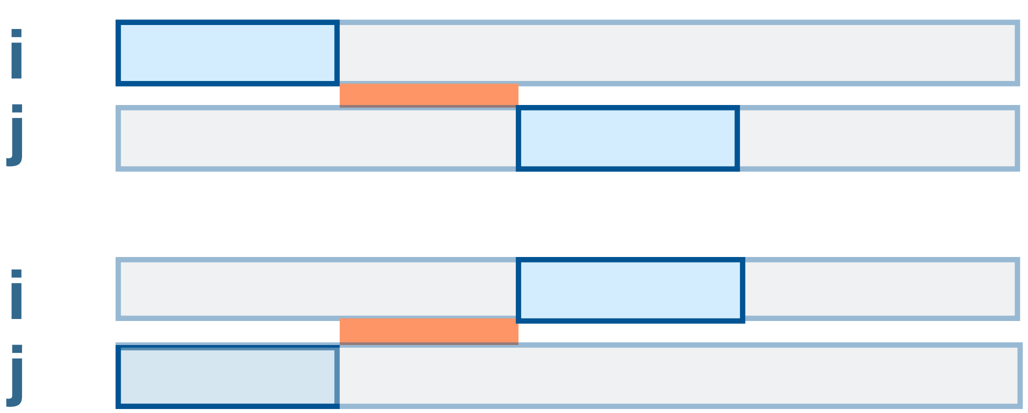
i *overlaps* **j**

choose only possible order constraint



i *overlaps* **j**

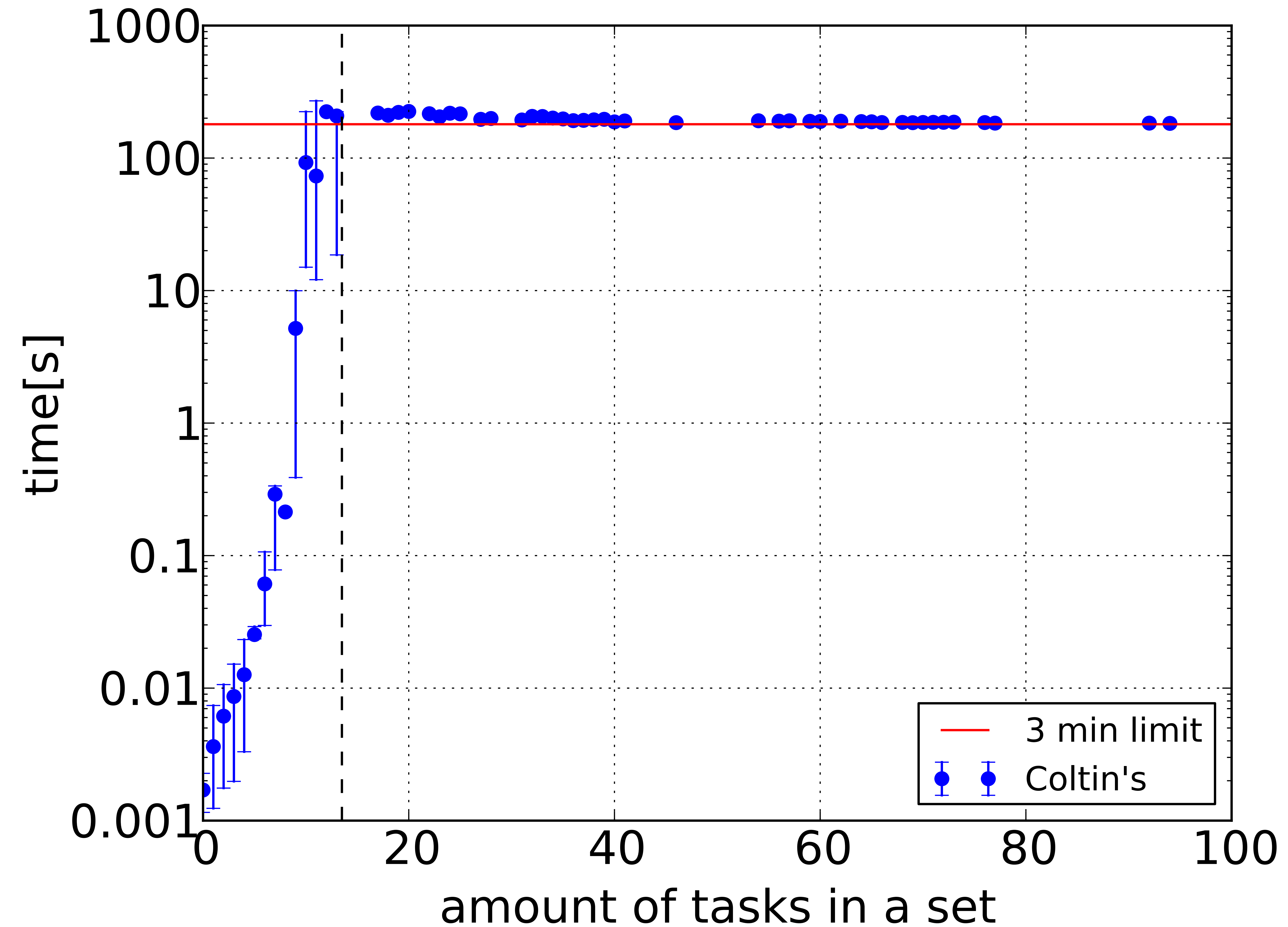
choose order to satisfy $\forall \mathbf{i} : min\sum(t_{\mathbf{i}} - s_{\mathbf{i}})$



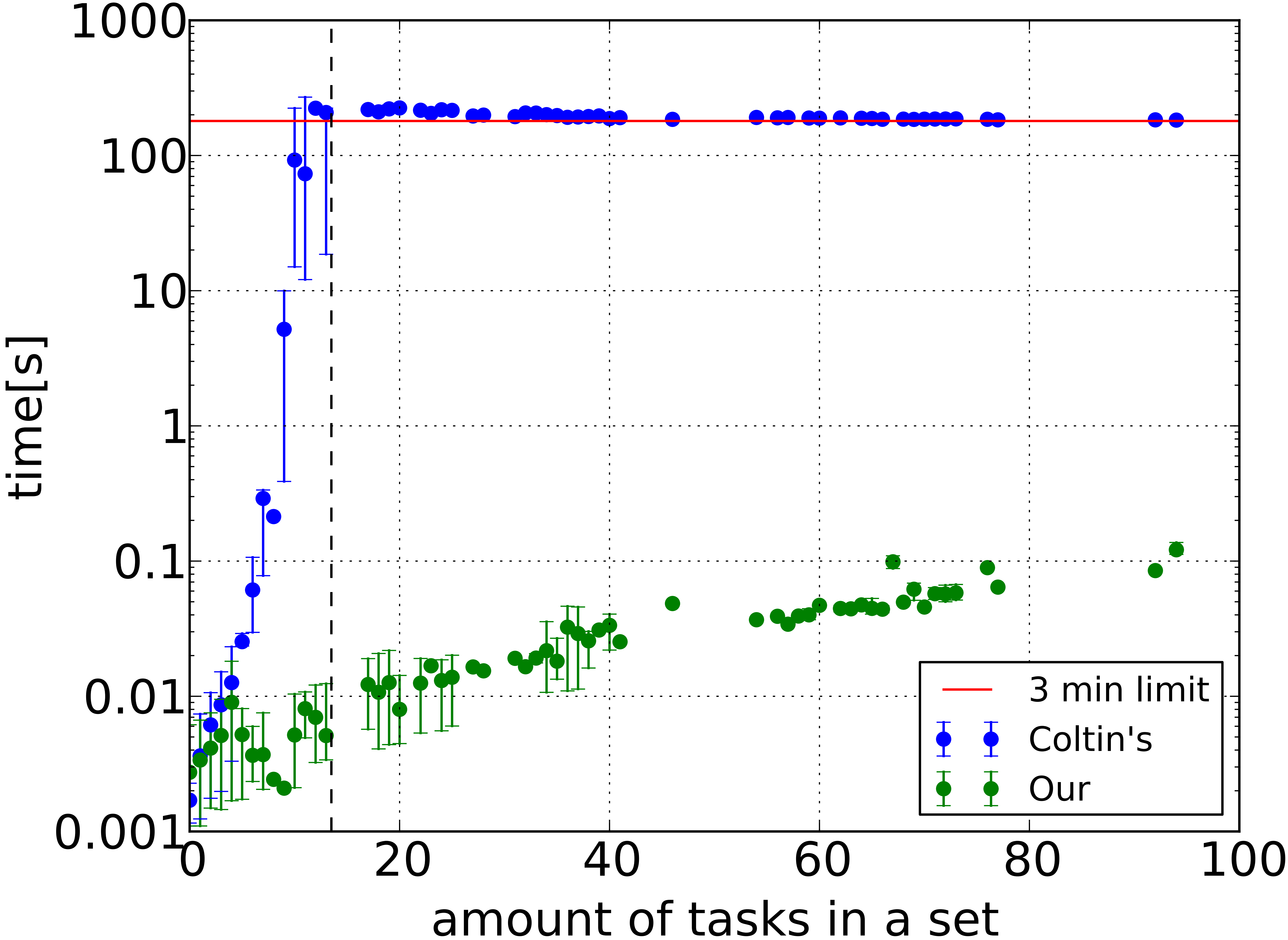
i *equals* **j**

pick first one seen

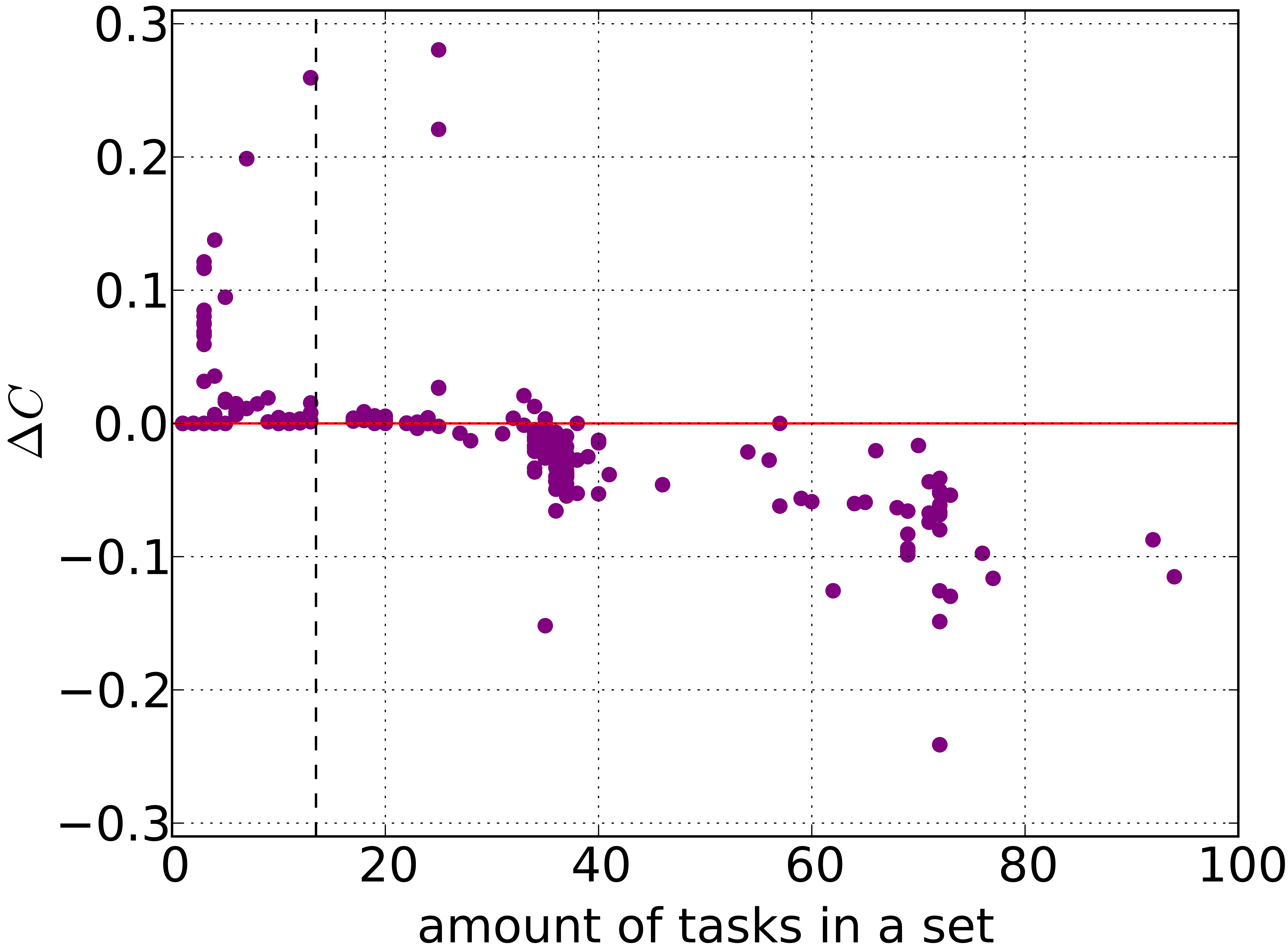
	Care	Security
# Problems	606	358
Smallest Problem	1	1
Largest Problem	135	71
Mean Problem Size	28.88 (σ 26.28)	9.59 (σ 12.97)
# Problems >15	349 (58%)	106 (30%)

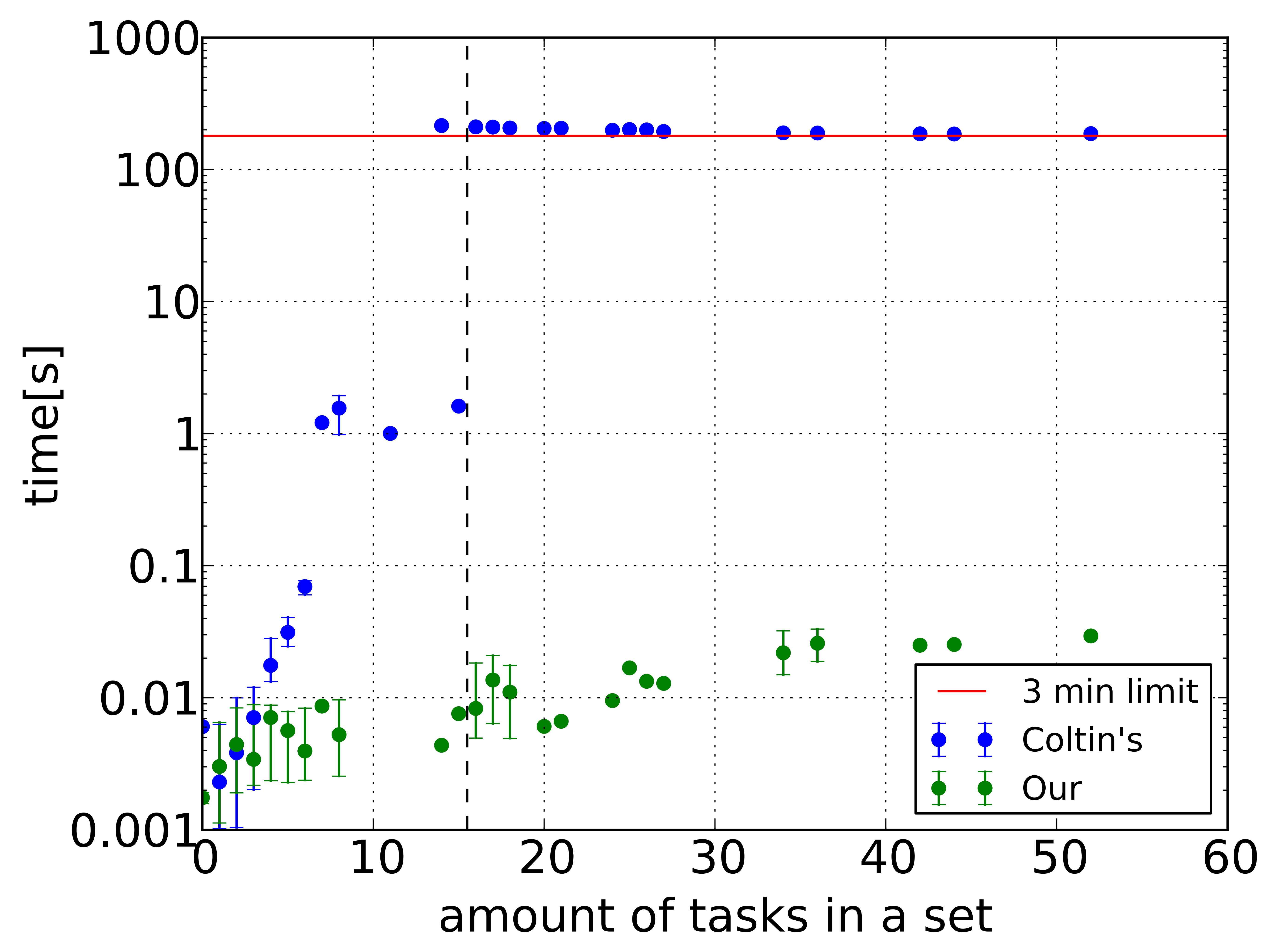


1000
100
10
1
0.1
0.01
0.001
0
20
40
60
80
100
time[s]
amount of tasks in a set
3 min limit
Coltin's

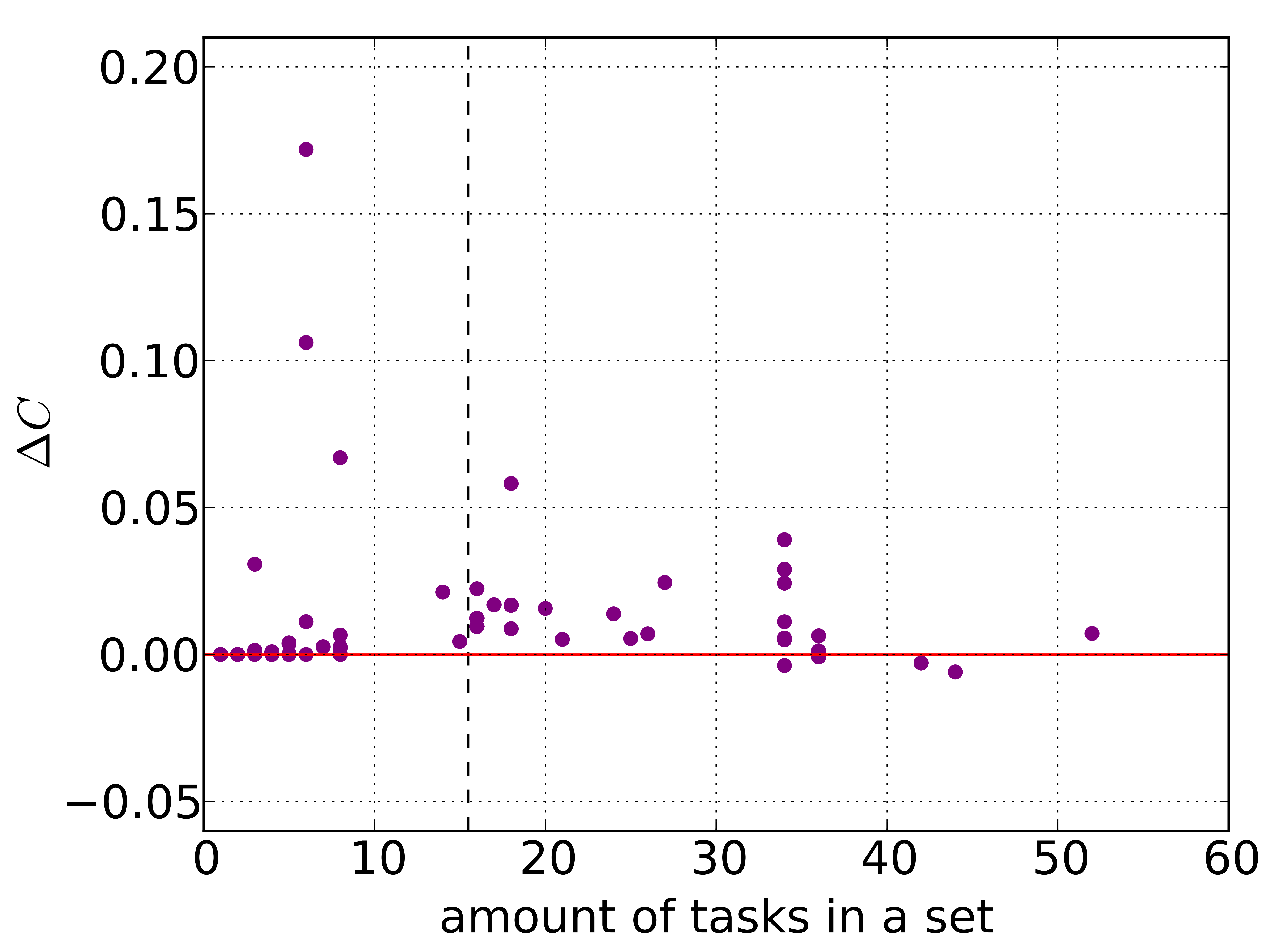


time[s]
1000
100
10
1
0.1
0.01
0.001
0
20
40
60
80
100
amount of tasks in a set
3 min limit
Coltin's
Our

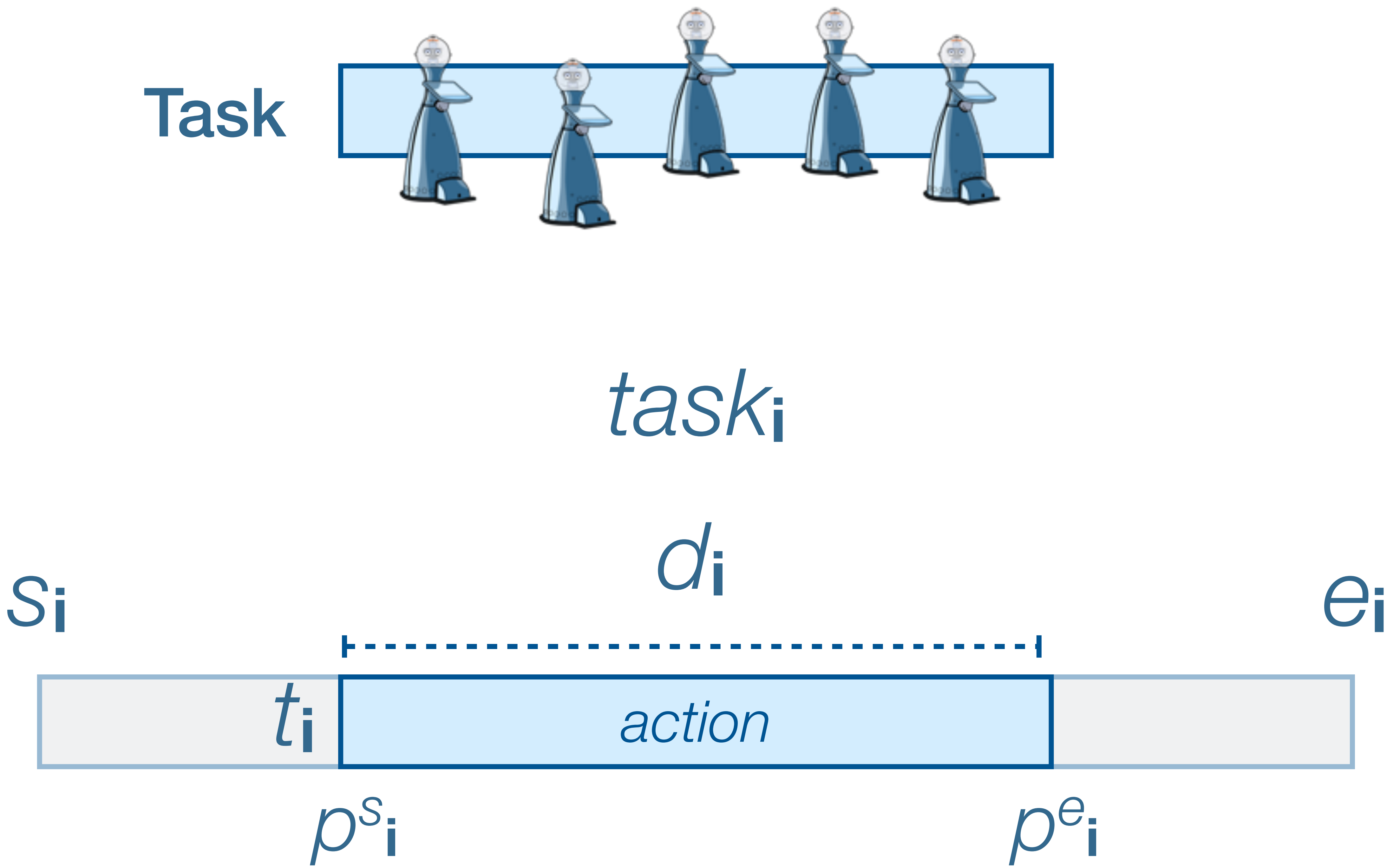

0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
ΔC
0
20
40
60
80
100
amount of tasks in a set



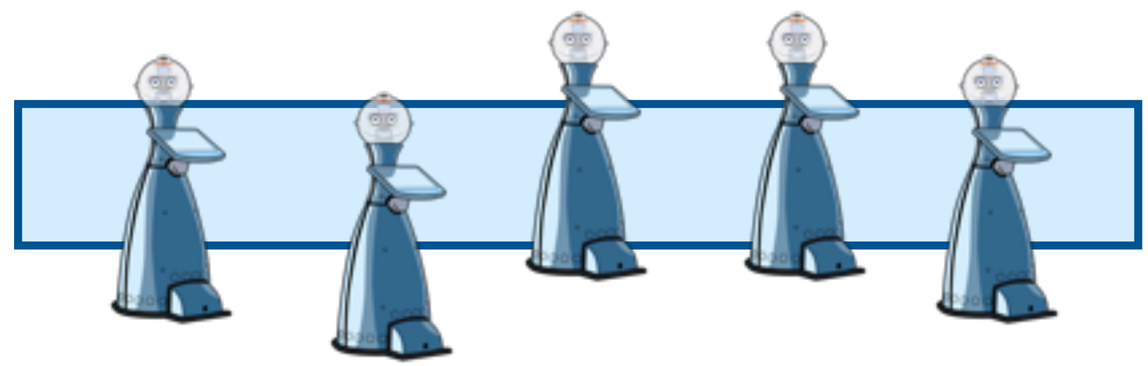
1000
100
10
1
0.1
0.01
0.001
time[s]
0
10
20
30
40
50
60
amount of tasks in a set
3 min limit
Coltin's
Our

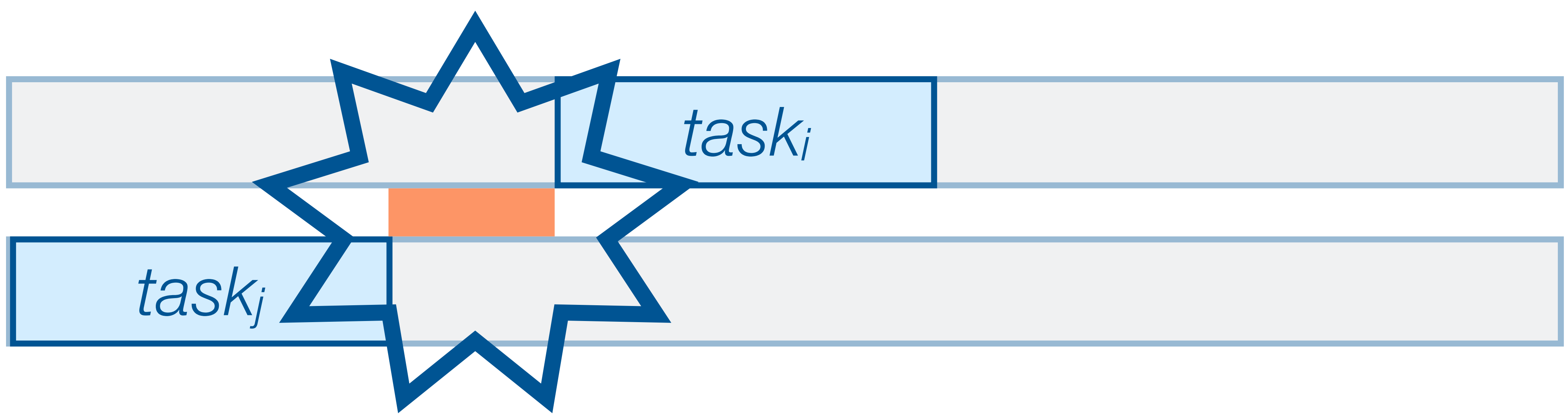


0.20
0.15
0.10
0.05
0.00
−0.05
ΔC
0
10
20
30
40
50
60
amount of tasks in a set


Task
$task_i$
d_i
s_i
e_i
t_i
action
p^s_i
p^e_i

Task

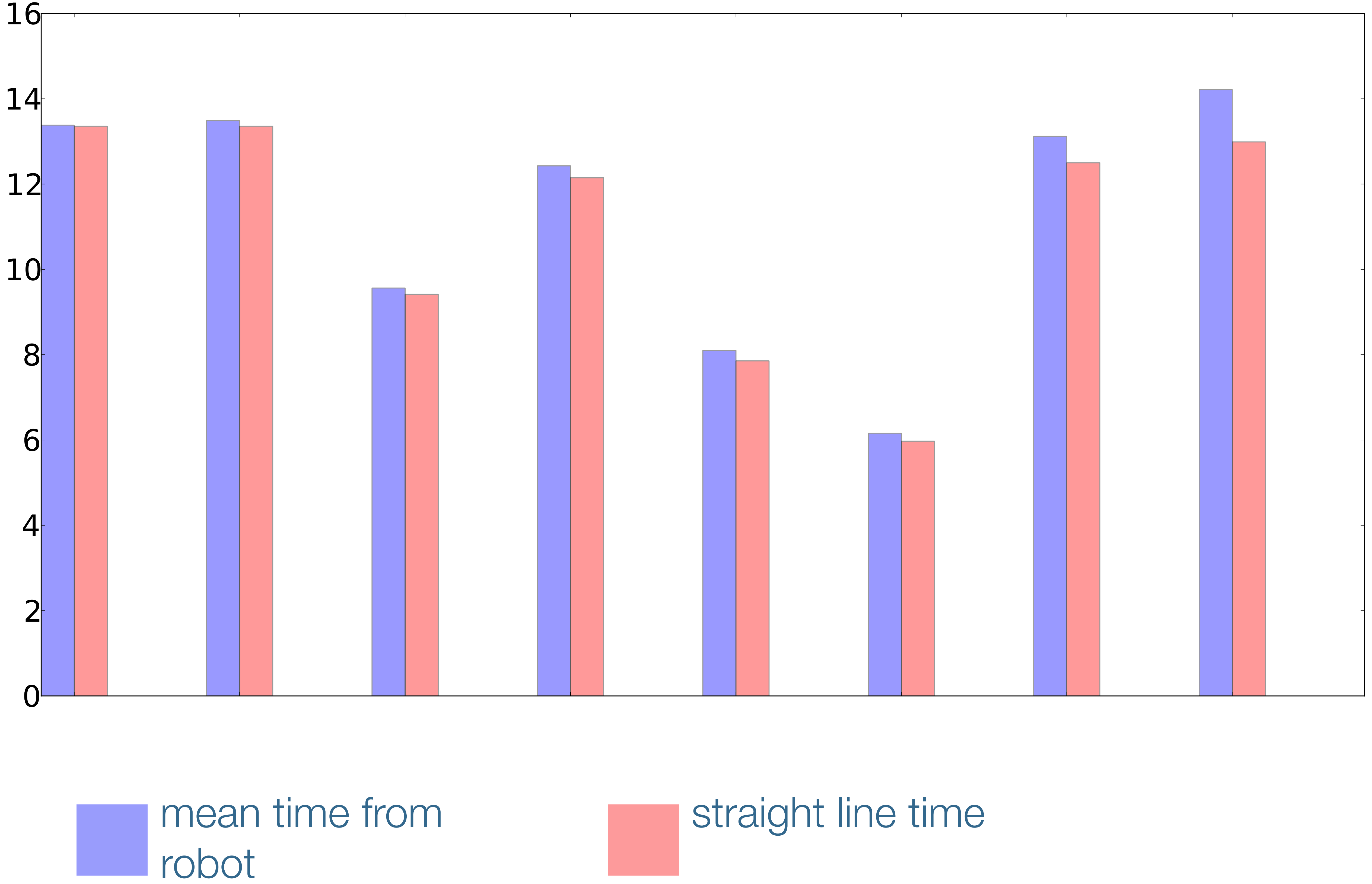




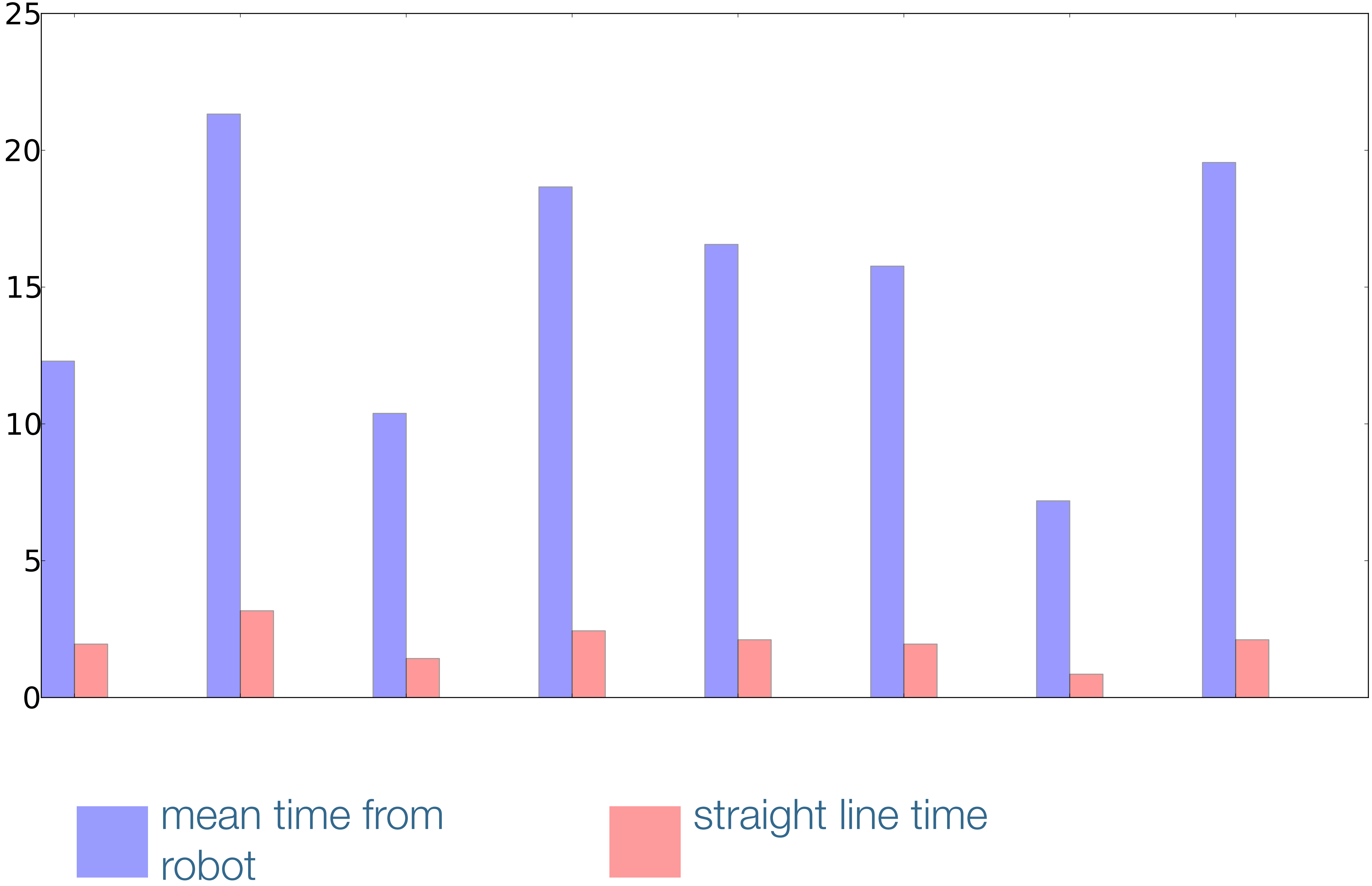
Optimal and Dynamic Planning for
Markov Decision Processes with Co-Safe LTL Specifications

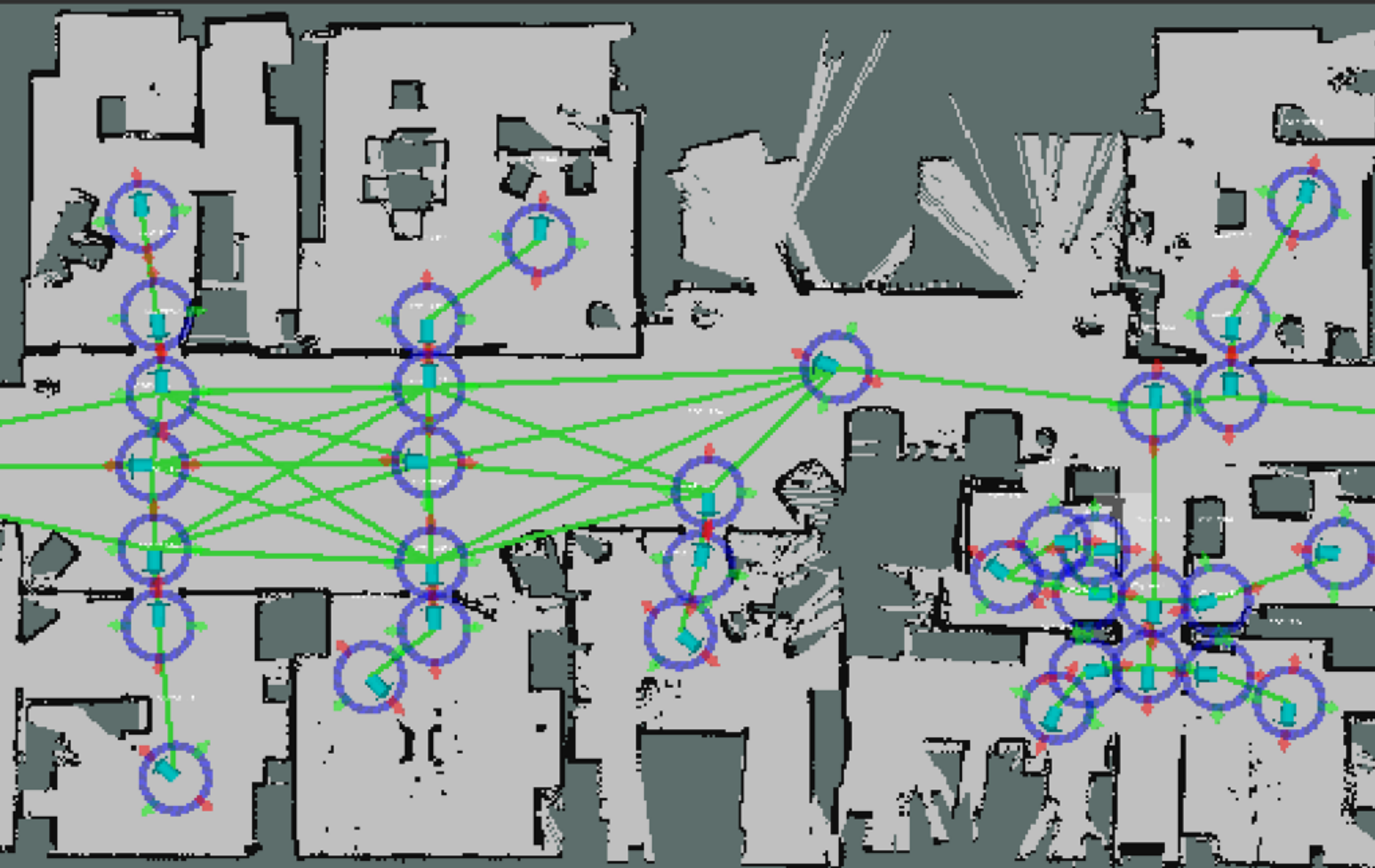
Lacerda, Parker and Hawes. In, *IROS'14*.

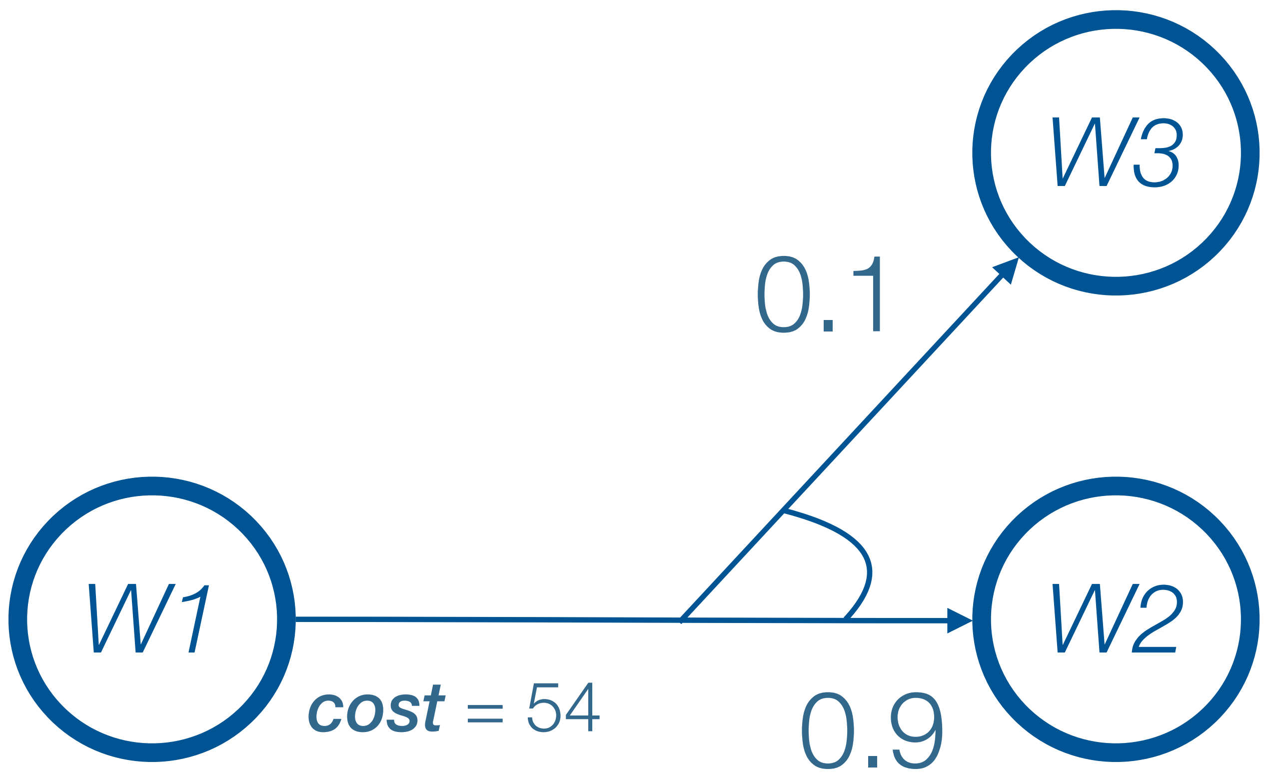
Best 8 matches between straight-line and recorded times



Worst 8 matches between straight-line and recorded times



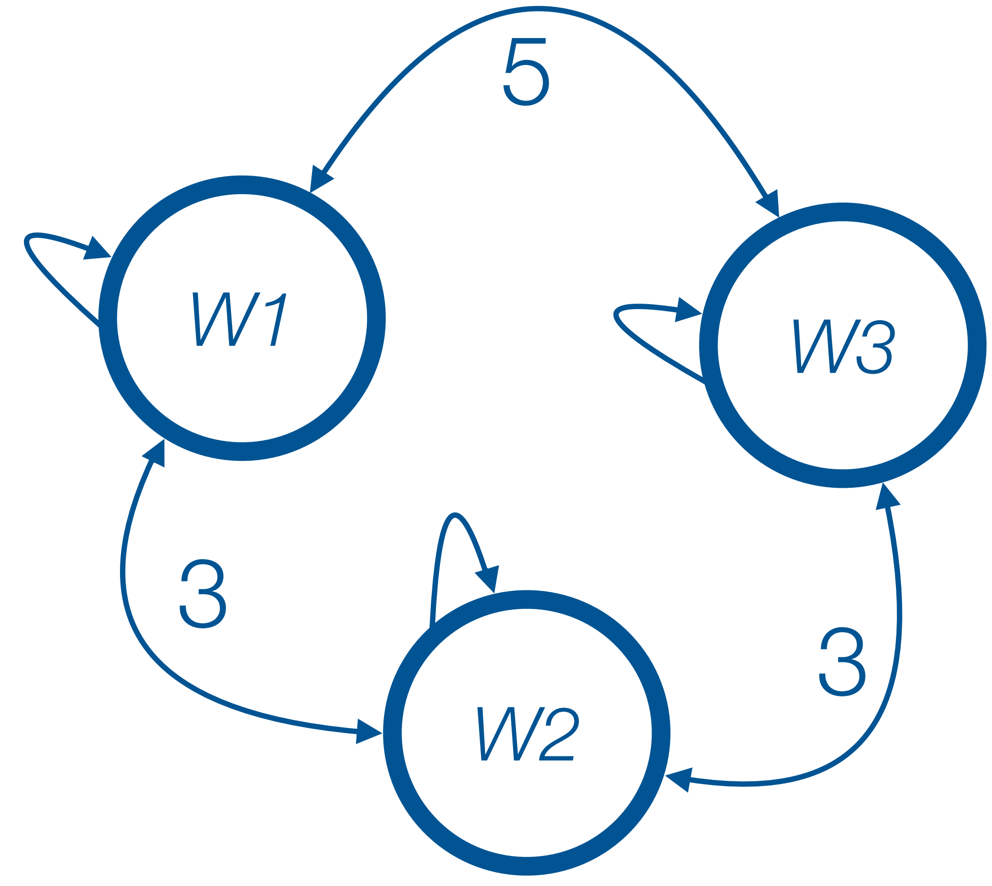




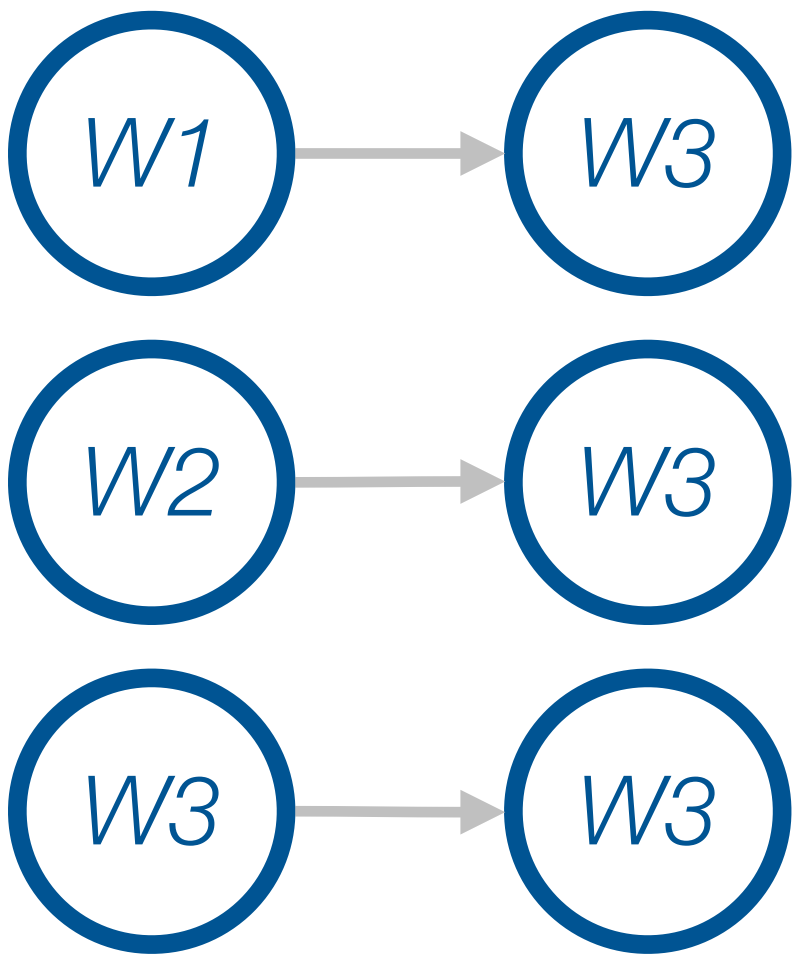
action goto *W2* from *W1*

Why use an MDP?

Goal is to be in state *W3*

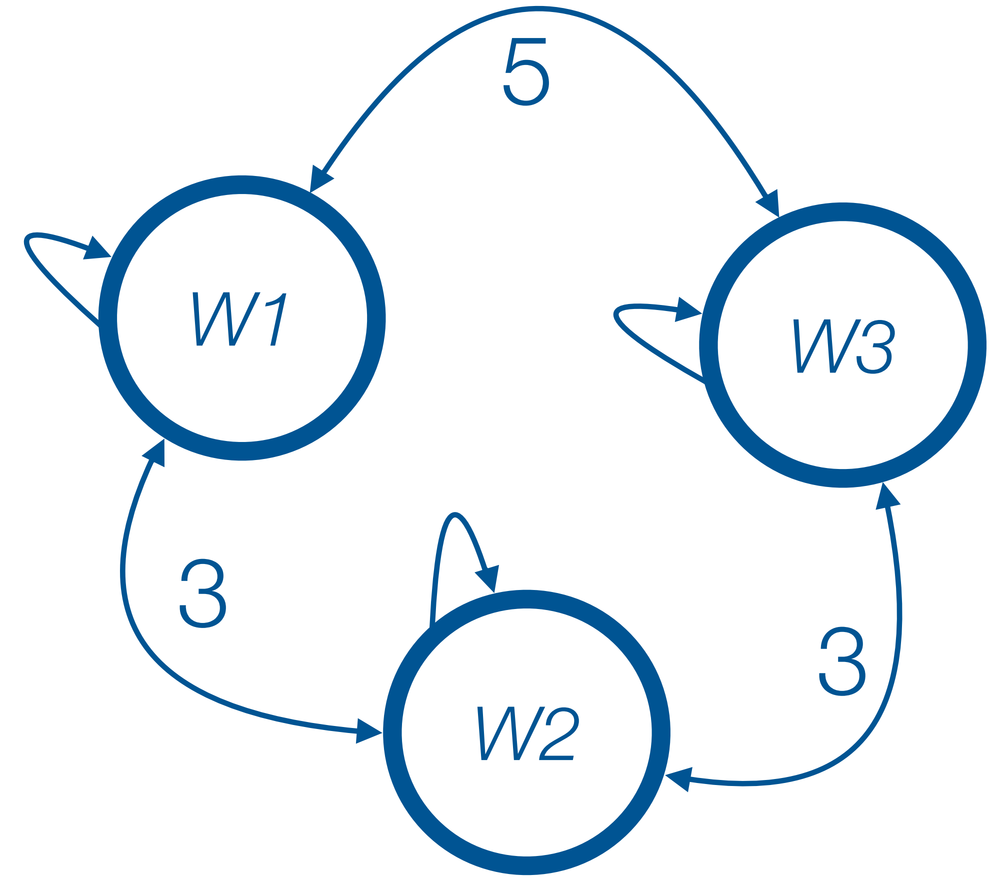


Policy:

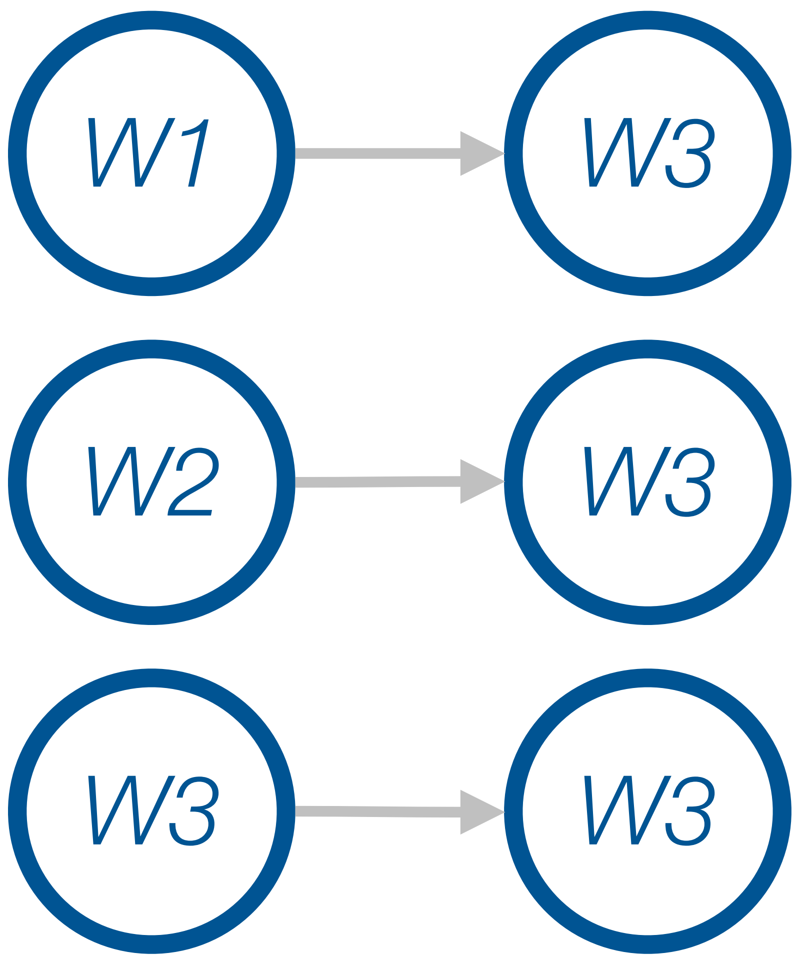


(***F*** *W2*)

eventually reach *W2*



Policy:

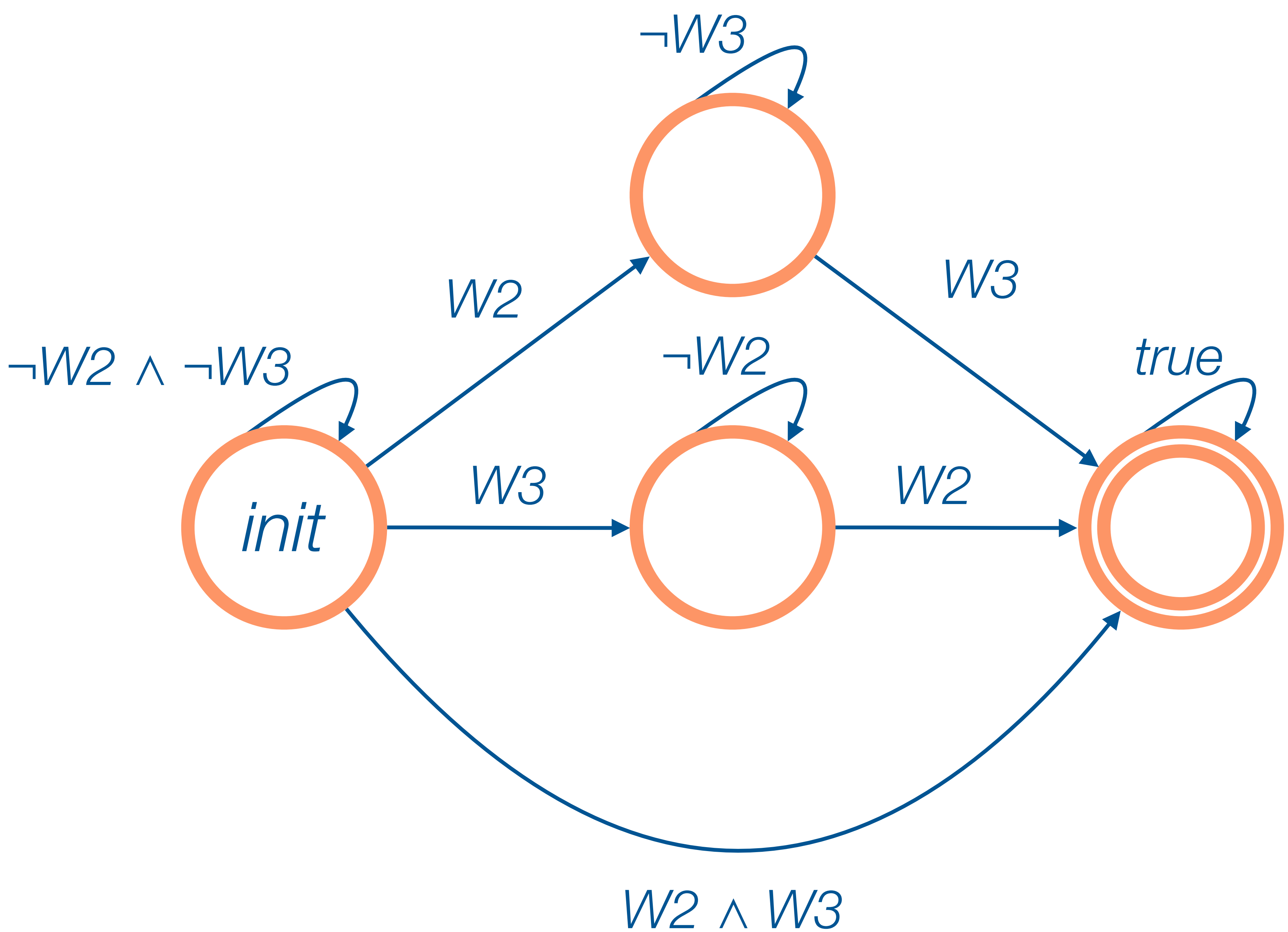


(***F*** *W2*) ∧ (***F*** *W3*)

eventually reach *W2* and *W3*

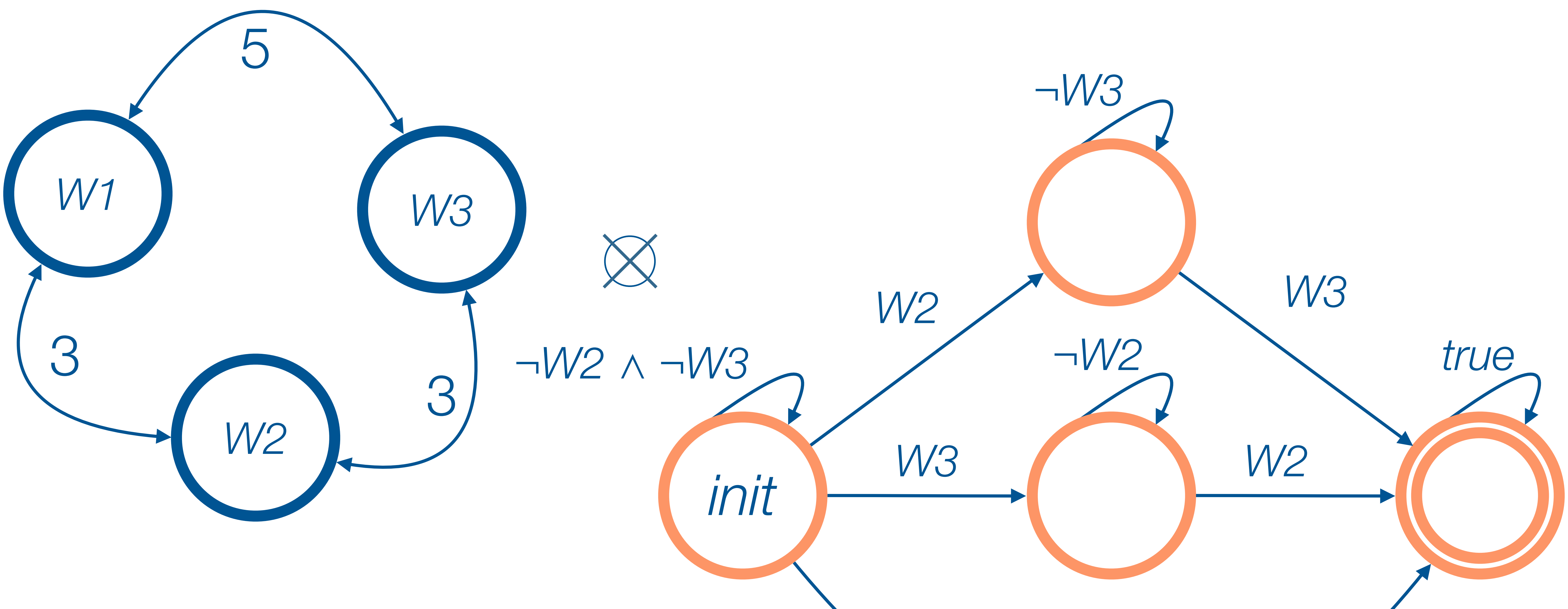
$(\boldsymbol{F}\ W2) \land (\boldsymbol{F}\ W3)$

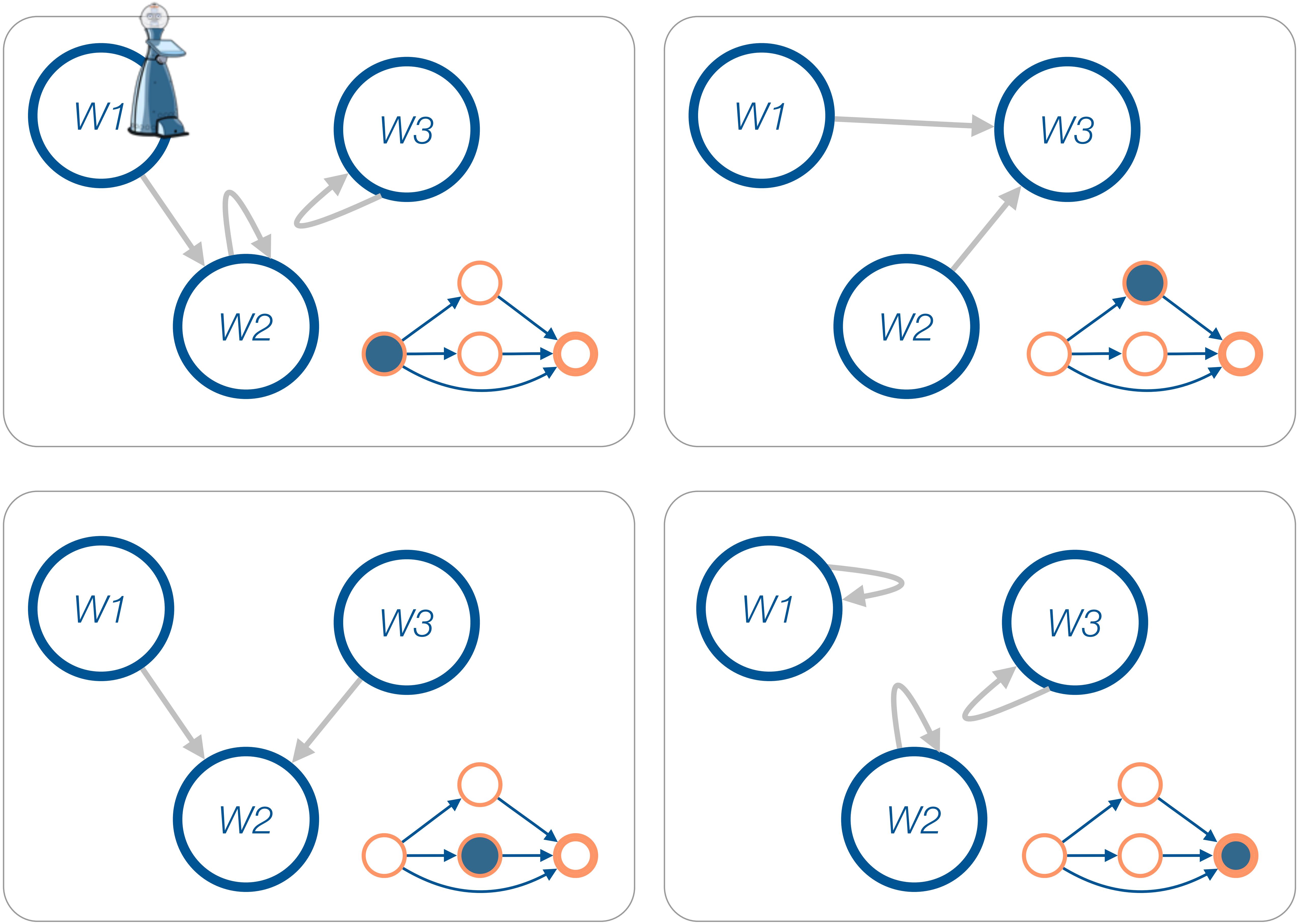
eventually reach *W2* and *W3*

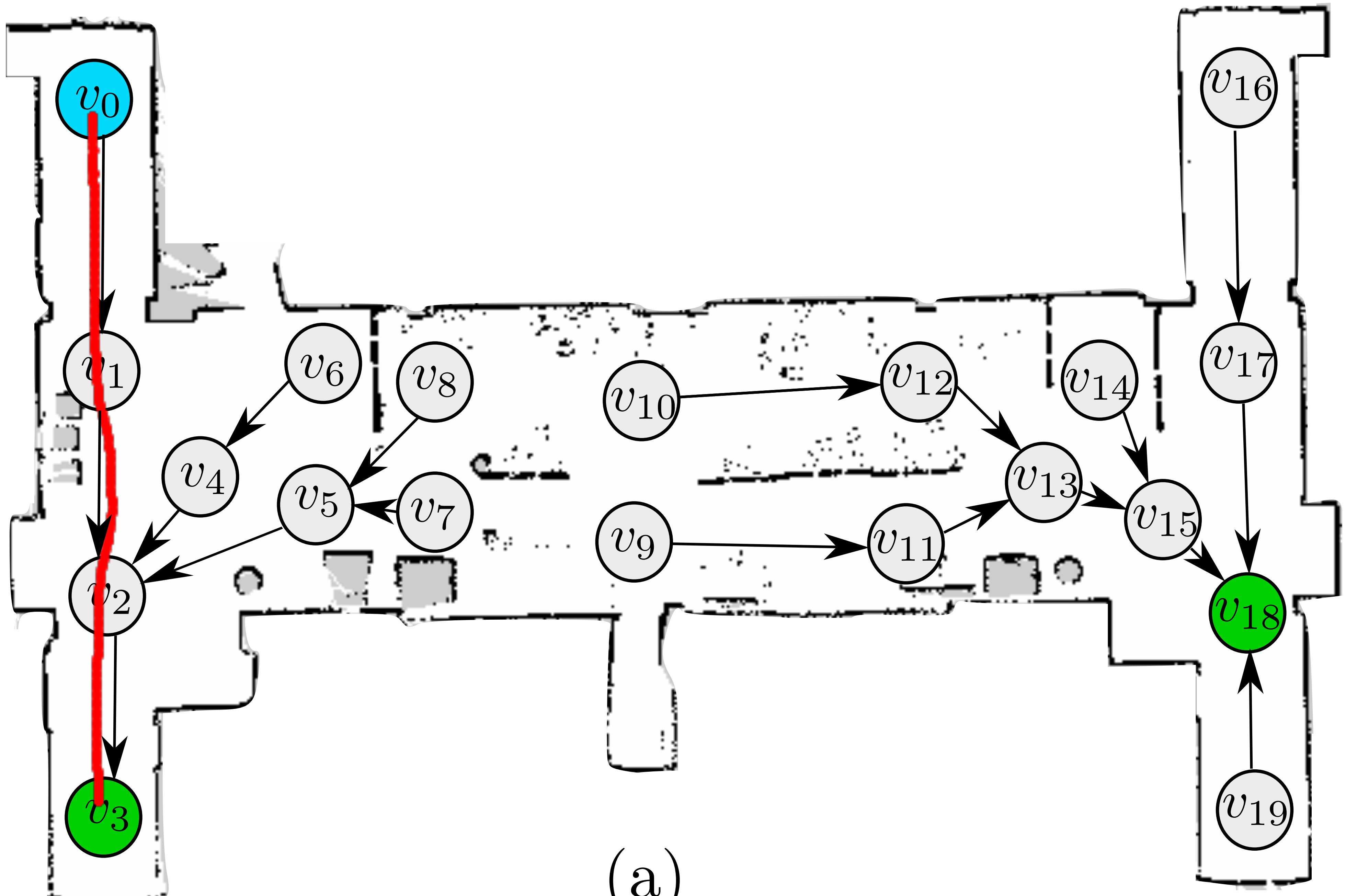


Cool tool: http://www.lsv.ens-cachan.fr/~gastin/ltl2ba


W1
W2
W1
W3
W1
W2
W1
W1

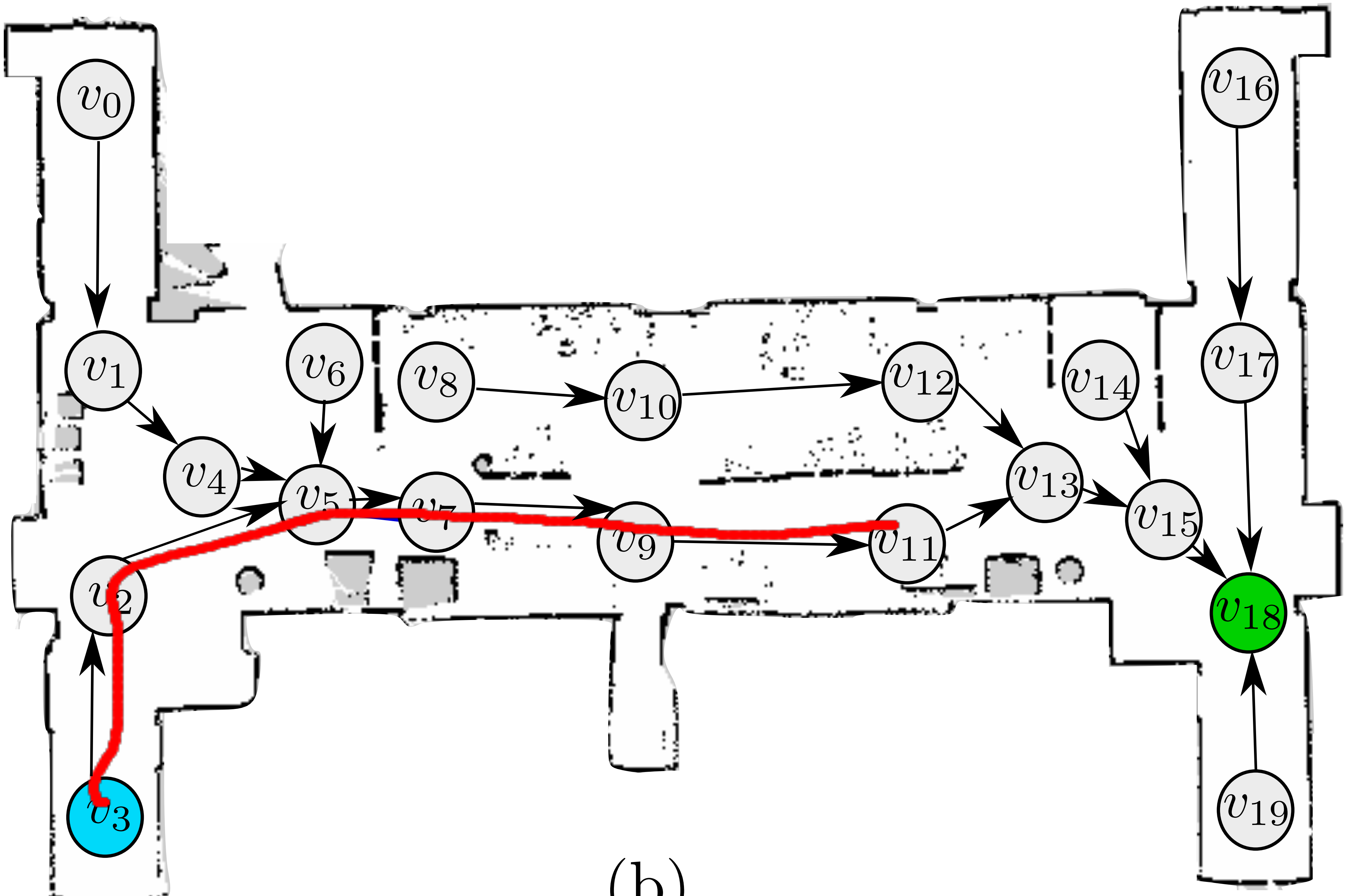

5
W1
W3
3
3
W2
⊗
¬W3
W2
W3
¬W2 ∧ ¬W3
¬W2
true
init
W3
W2


W1
W3
W2
W1
W3
W2
W1
W3
W2
W1
W3
W2



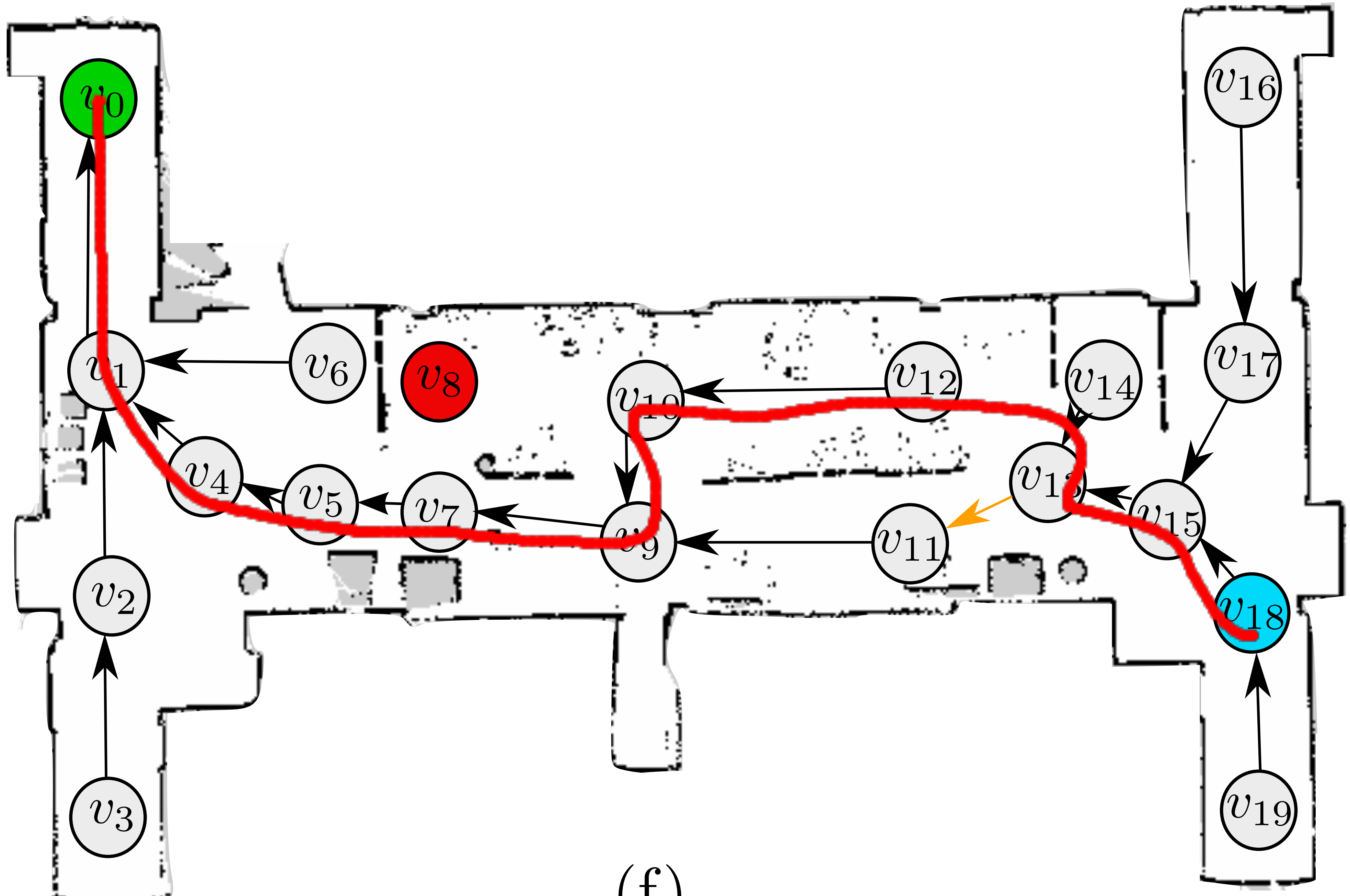
(a)

B. Lacerda, D. Parker, and N. Hawes. *Optimal and Dynamic Planning for Markov Decision Processes with Co-Safe LTL Specifications*. In: IROS 2014.



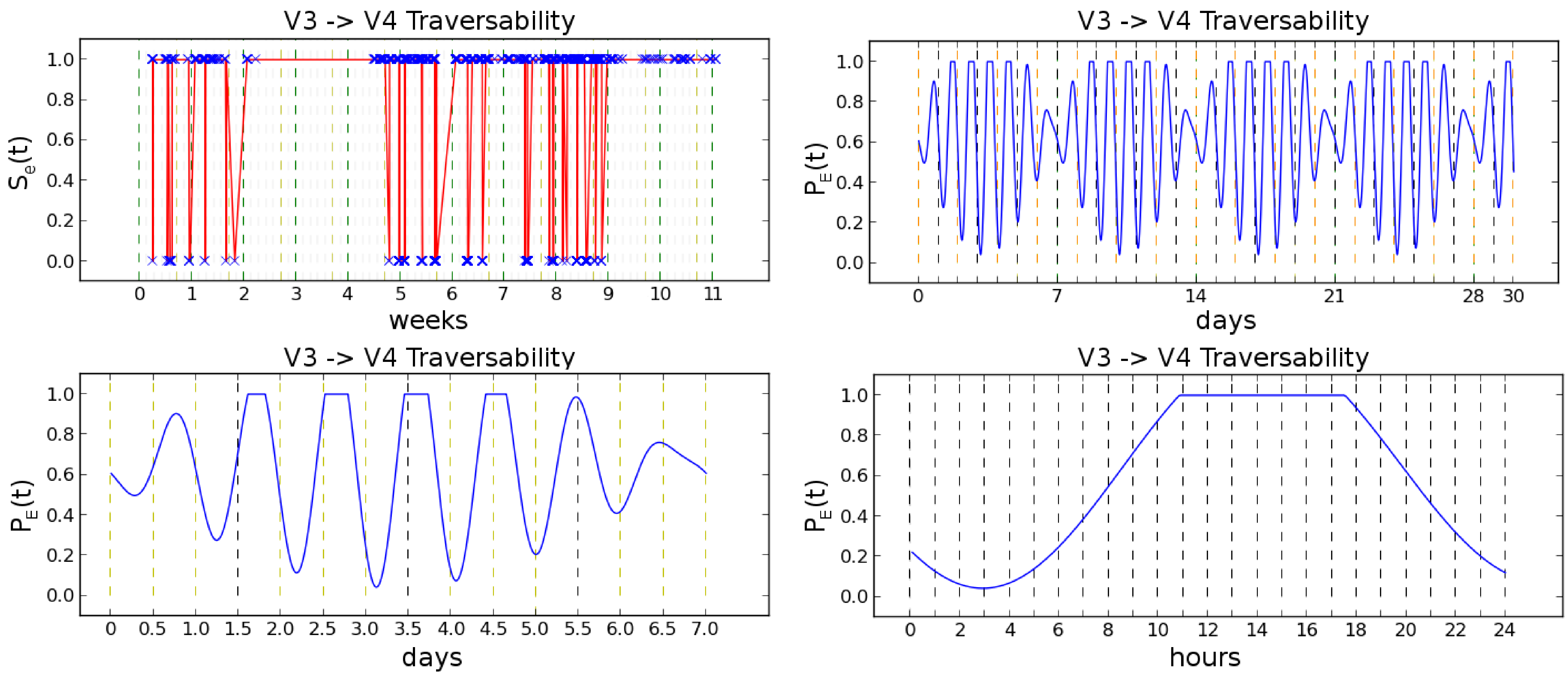
(b)

B. Lacerda, D. Parker, and N. Hawes. *Optimal and Dynamic Planning for Markov Decision Processes with Co-Safe LTL Specifications*. In: IROS 2014.

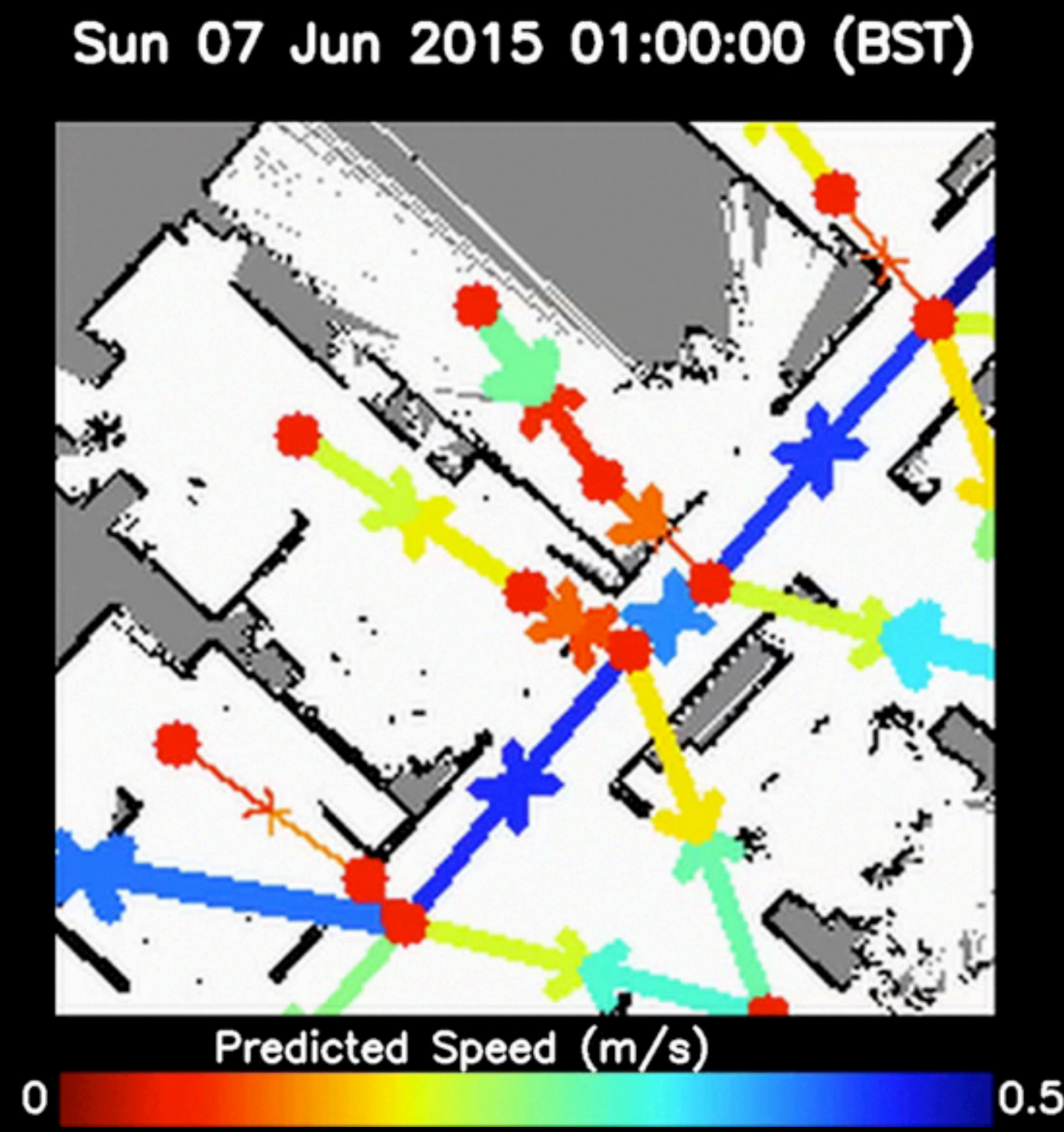


(f)

B. Lacerda, D. Parker, and N. Hawes. *Optimal and Dynamic Planning for Markov Decision Processes with Co-Safe LTL Specifications*. In: IROS 2014.



J. Pulido Fentanes, B. Lacerda, T. Krajník, N. Hawes, and M. Hanheide. Now or later? predicting and maximising success of navigation actions from long-term experience. In ICRA, 2015.


Sun 07 Jun 2015 01:00:00 (BST)
Predicted Speed (m/s)
0
0.5

Task framework