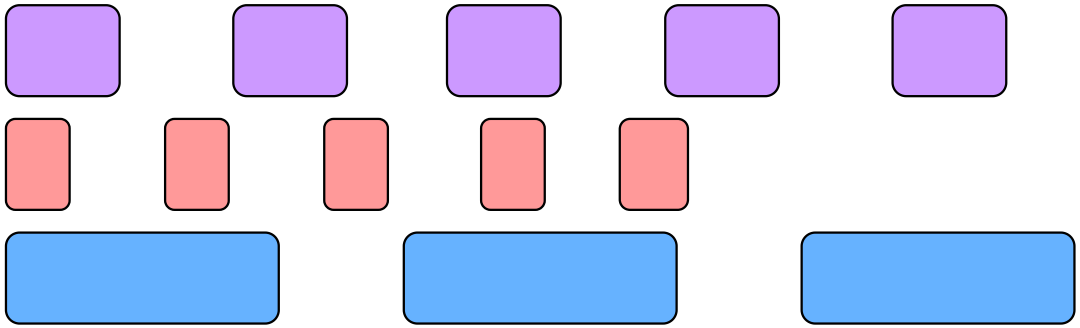
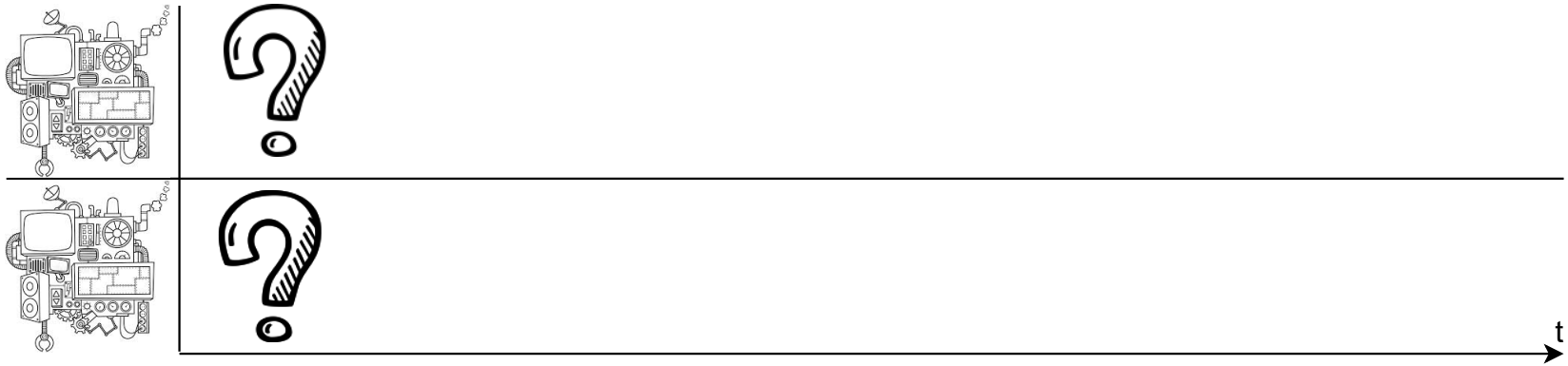
TU Delft

Adjustable Robust Machine Scheduling

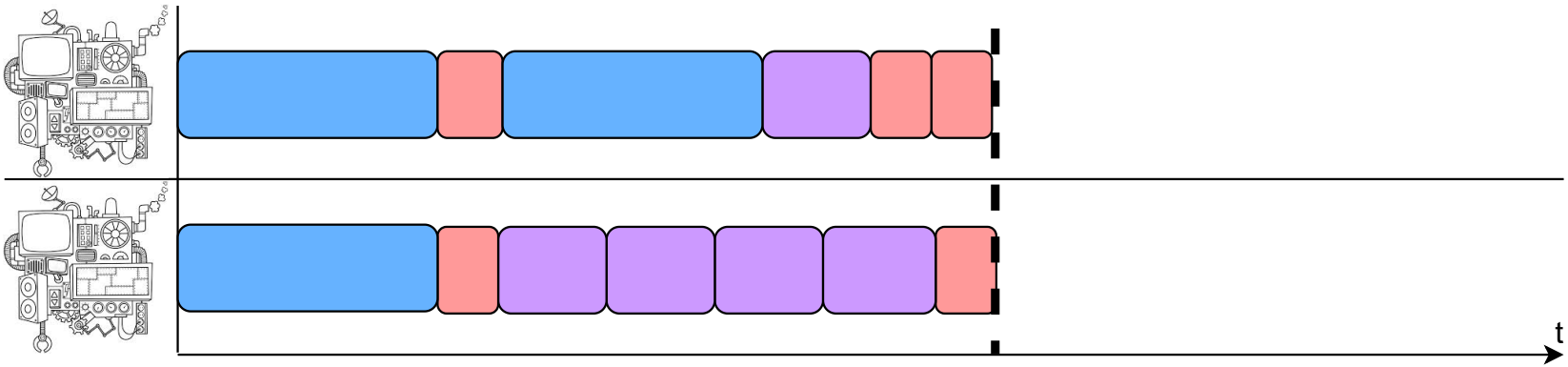
Krzysztof Postek
Izack Cohen (Bar Ilan University)
Shimrit Shtern (Technion)
Izak de Heer (TU Delft)

Motivation



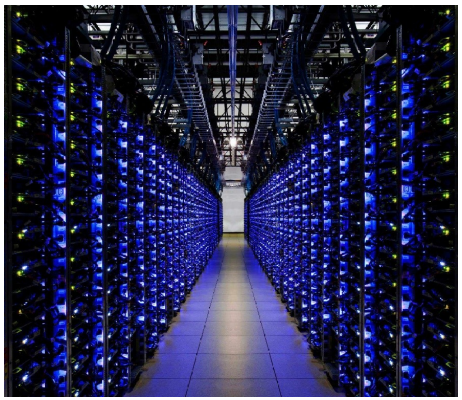


Motivation

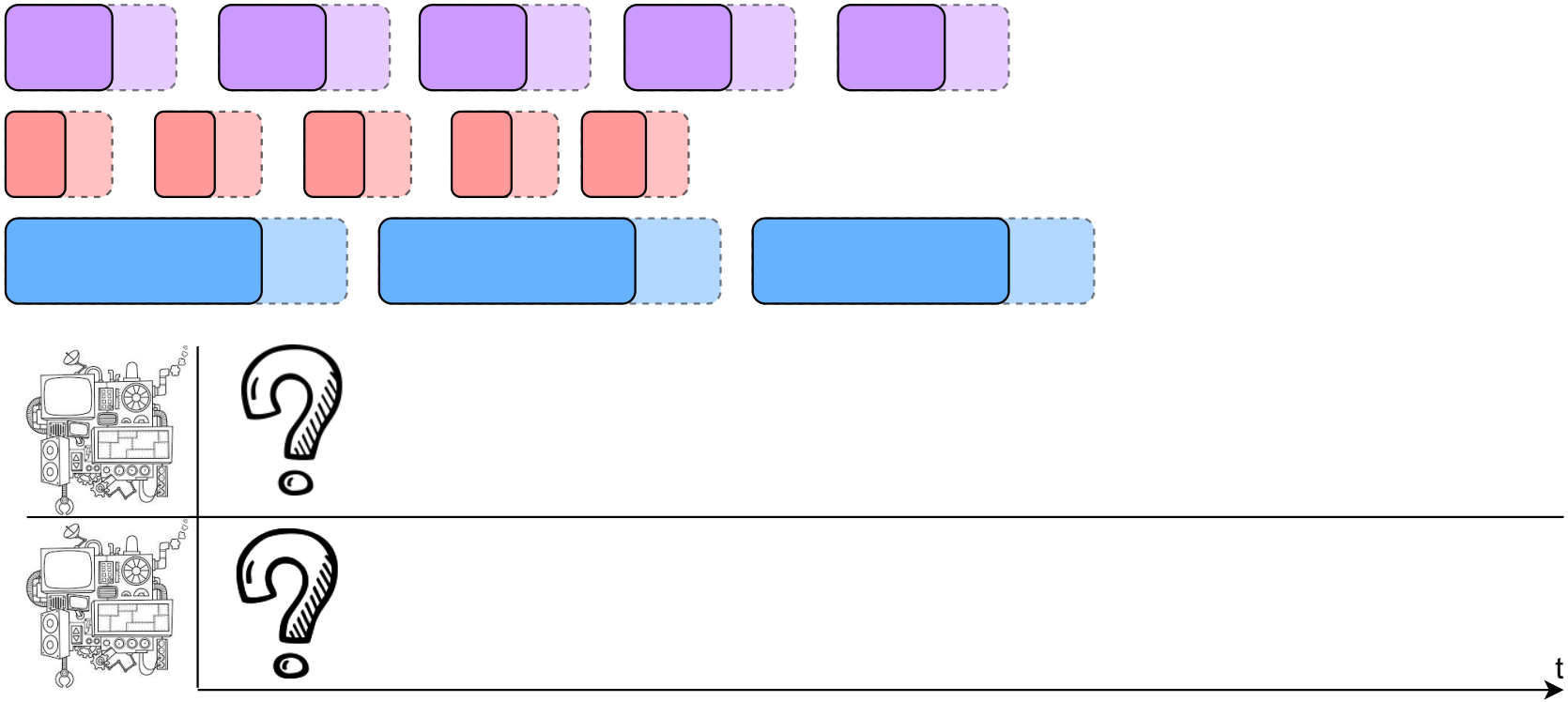


- equivalent to m-partition
- ordering per machine: doesn't matter
- solving: dynamic-programmming, heuristics (Mokotoff 2001)

Motivation

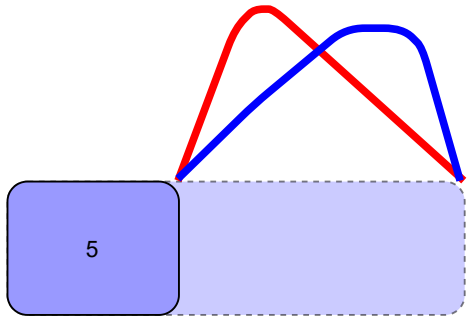



Motivation



Uncertainty

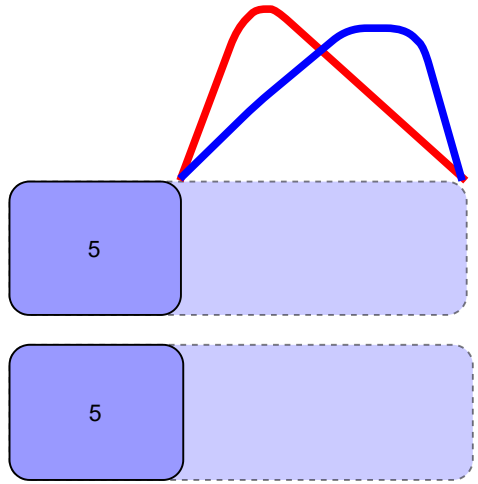
Stochastic



Min (expectation)

Uncertainty

Stochastic

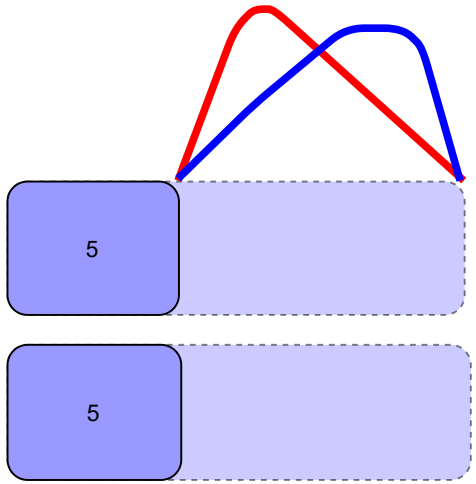


Min (expectation)

Robust

Min (worst case)

Uncertainty



Stochastic — Min (expectation)

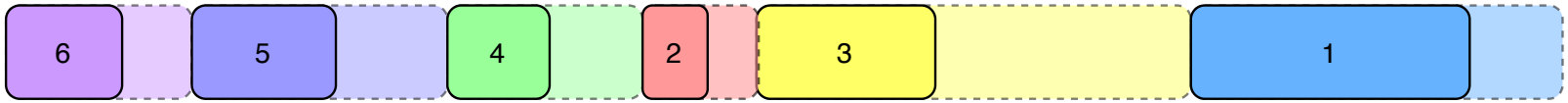
Robust — Min (worst case)

Static or adaptive?

Static Policies

- Static list: Fix the order of allocation

Static Policies



- Static list: Fix the order of allocation

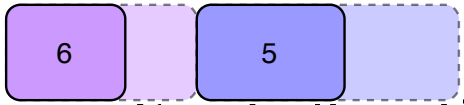
Static Policies

- Static list: Fix the order of allocation

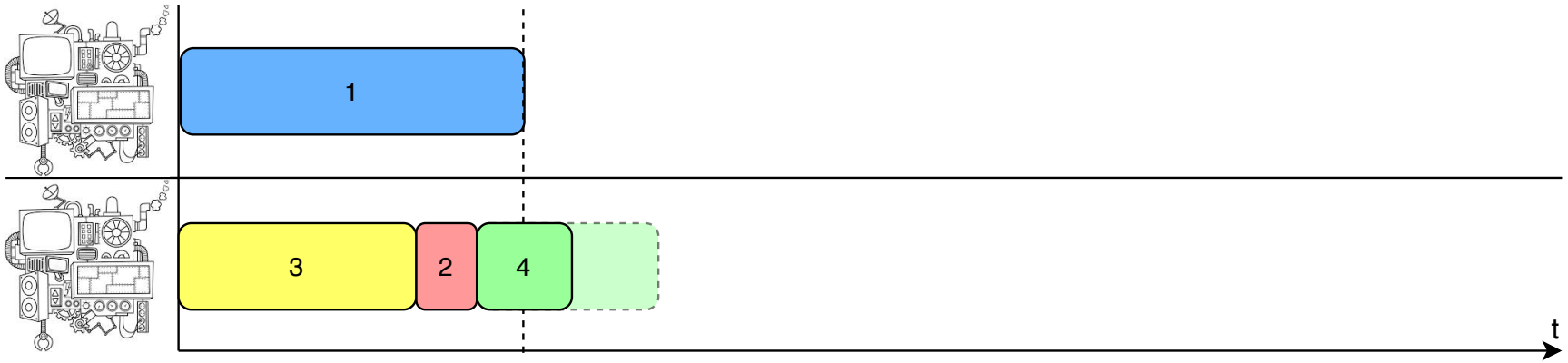
Static Policies

- Static list: Fix the order of allocation

Static Policies

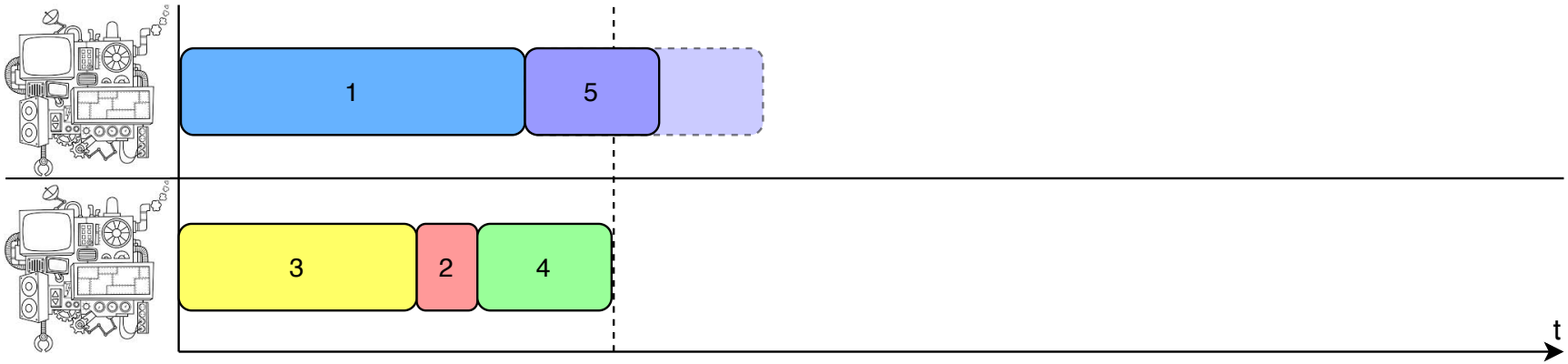


- Static list: Fix the order of allocation



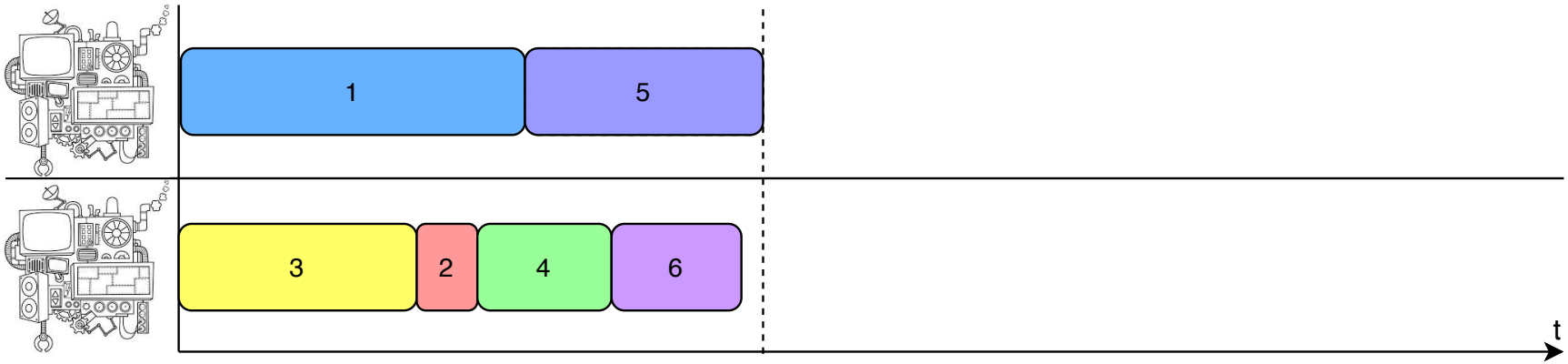
Static Policies

- Static list: Fix the order of allocation

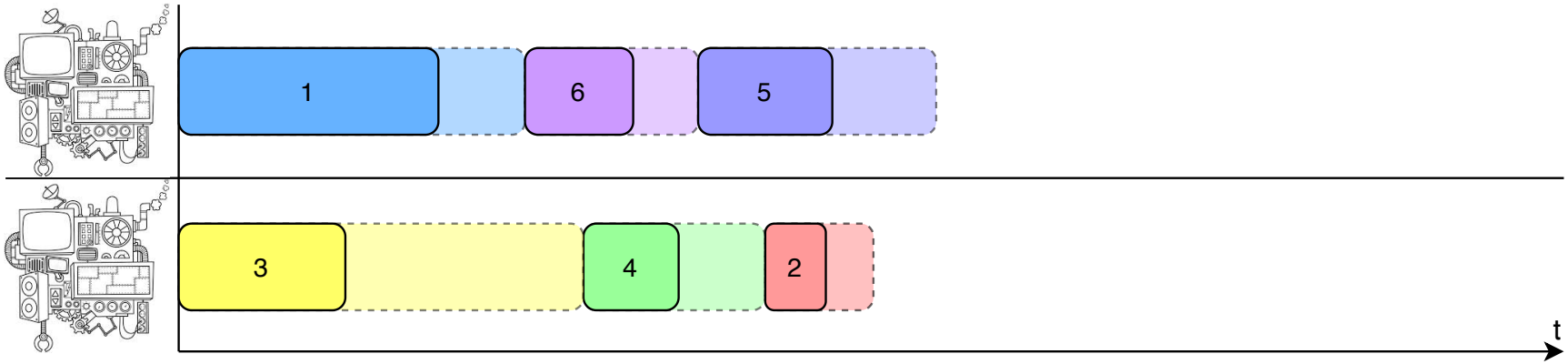


Static Policies

- Static list: Fix the order of allocation



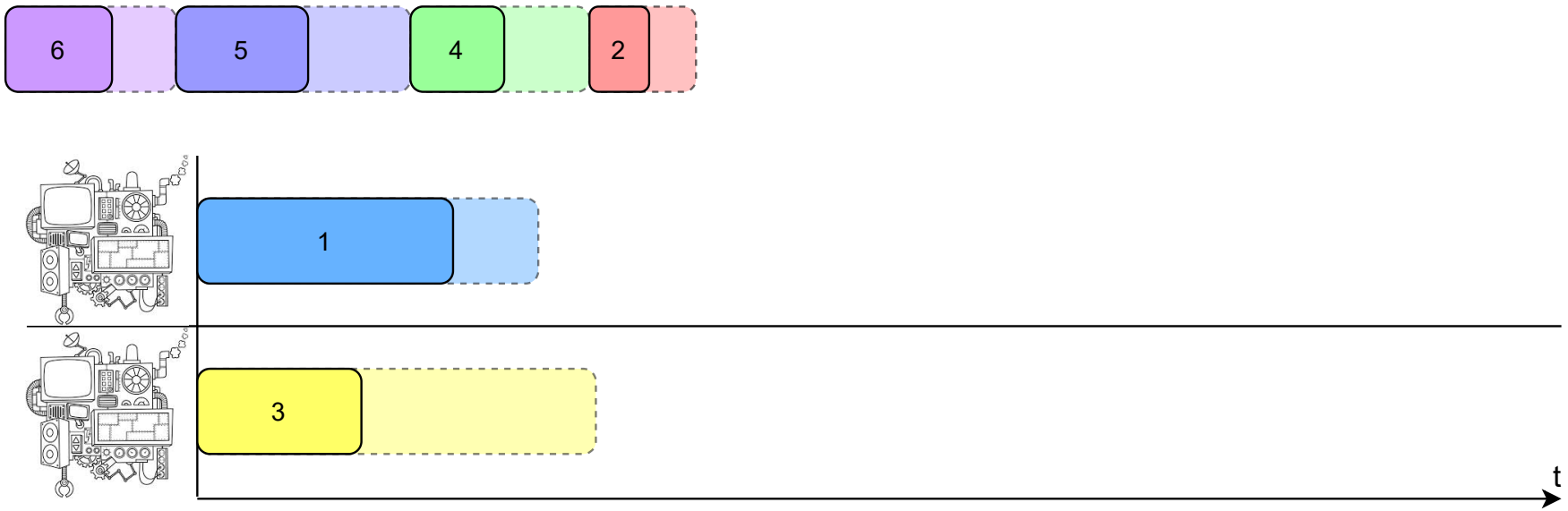
Static Policies



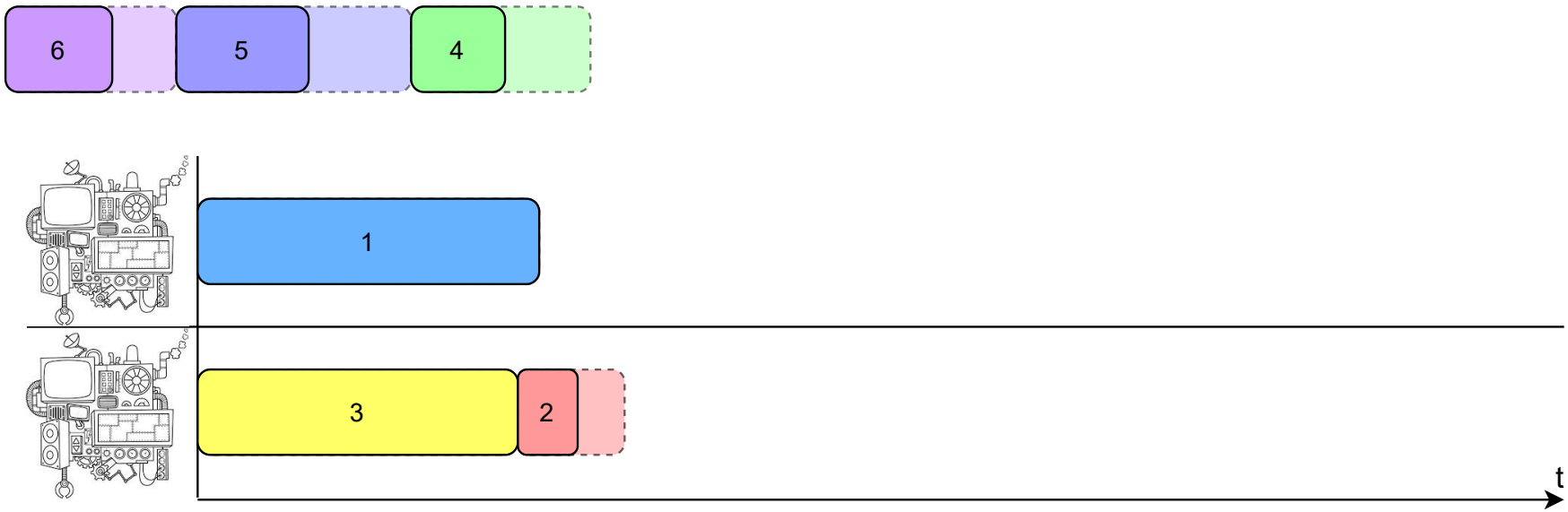
e.

Static Policies

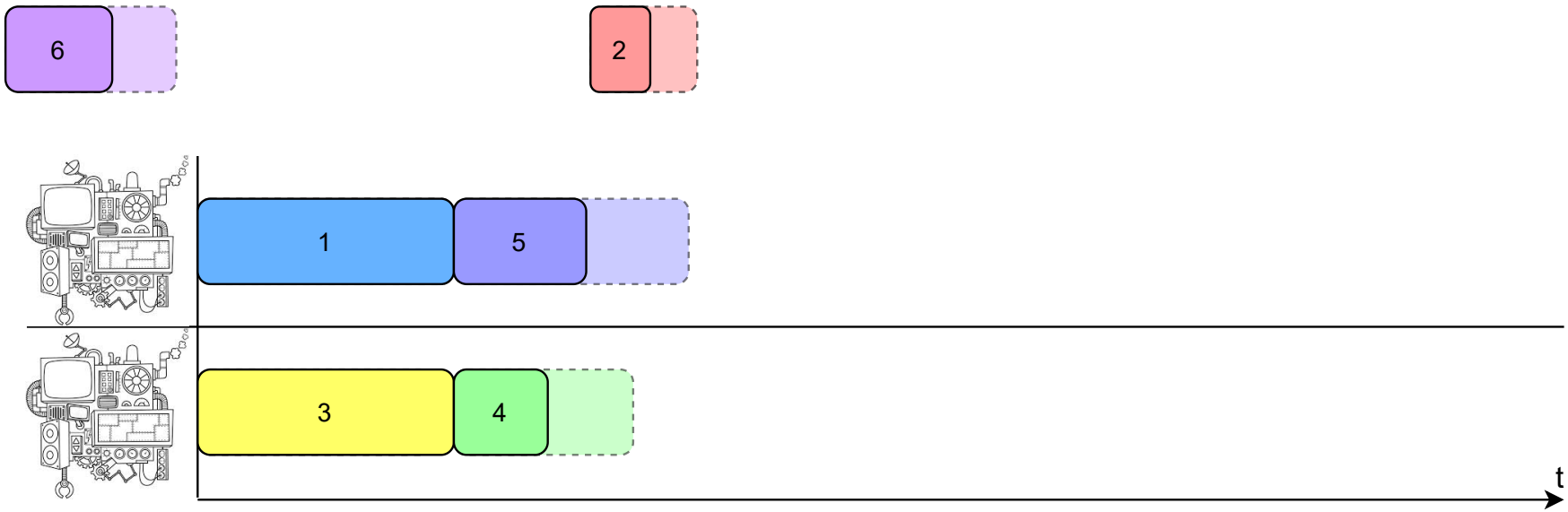
Adaptivity



Adaptivity



Adaptivity



Multi-stage optimization vs. adaptive scheduling

Multi-stage optimization vs. adaptive scheduling

Robust Optimization, Ben-Tal, El Ghaoui, and Nemirovski 2009, Example 14.2.1:

"...an attempt to fully utilize the possibilities to adjust the decisions to the actual values of the data results in an extremely complicated problem, where, not only the decisions themselves, but the very information base of the decisions become dependent on the uncertain data and our policy..."

Research questions

- Adaptive vs. static:
  - optimal makespans
  - here-and-now decisions

Research questions

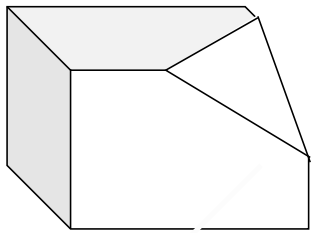
- Adaptive vs. static:
  - optimal makespans
  - here-and-now decisions
- How to solve?

Example

- 3 tasks, 2 machines.

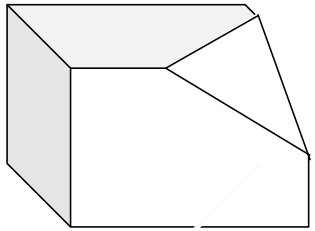
Example

- 3 tasks, 2 machines.
- Tasks' durations (d_1, d_2, d_3) belong to

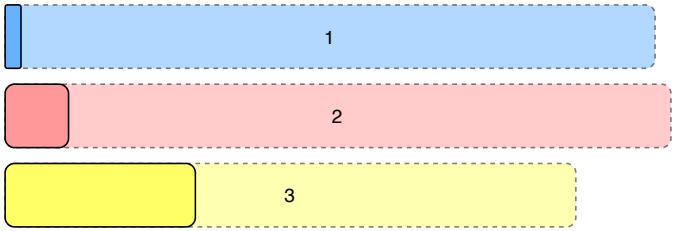


Example

- 3 tasks, 2 machines.
- Tasks' durations (d_1, d_2, d_3) belong to



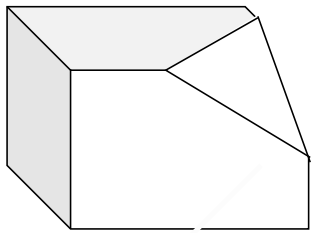
- Optimal static allocation:



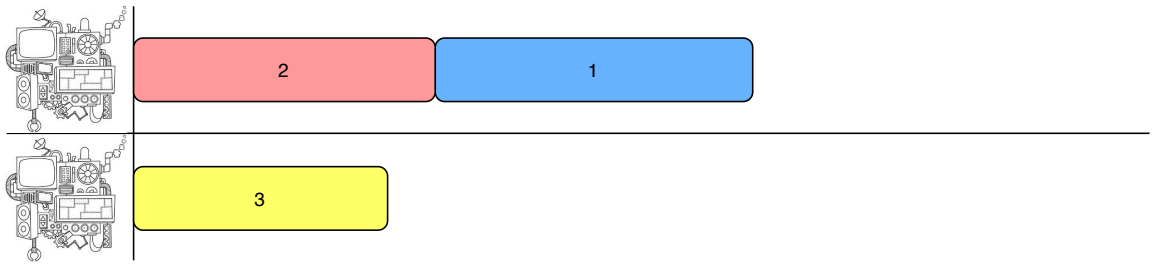
2 and 1 scheduled on one machine, 3 on the other.

Example

- 3 tasks, 2 machines.
- Tasks' durations (d_1, d_2, d_3) belong to



- Optimal static allocation:



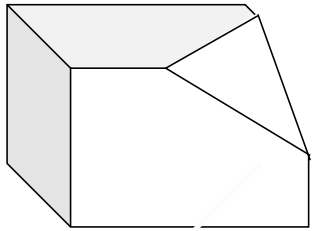
2 and 1 scheduled on one machine, 3 on the other.

Worst case duration 1.9525

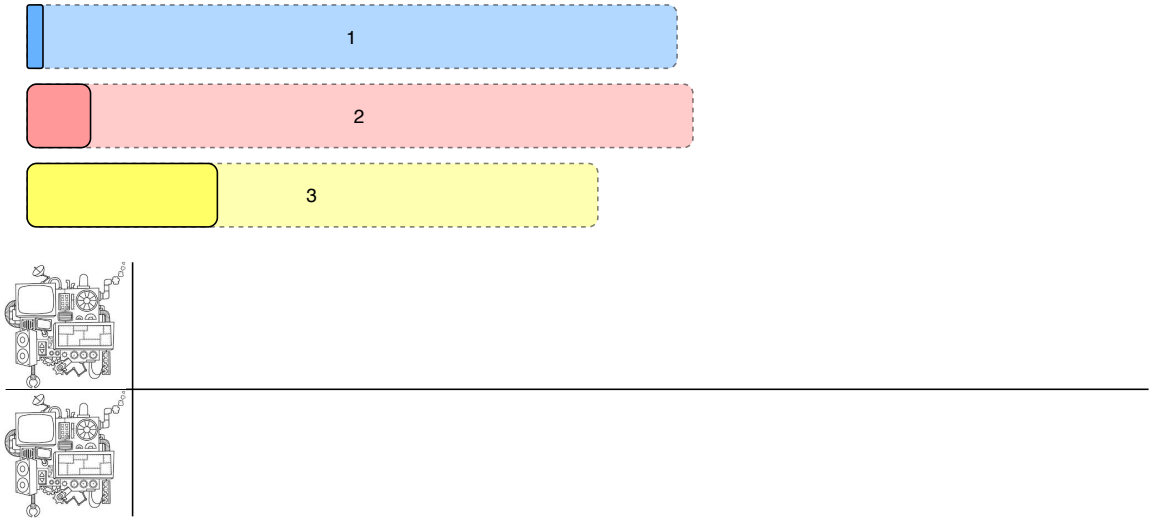
Not in practice!

Example

- 3 tasks, 2 machines.
- Tasks' durations (d_1, d_2, d_3) belong to



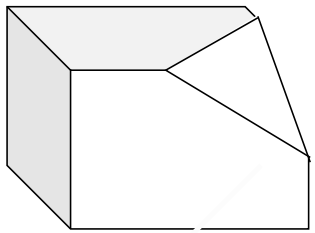
- Optimal static allocation:



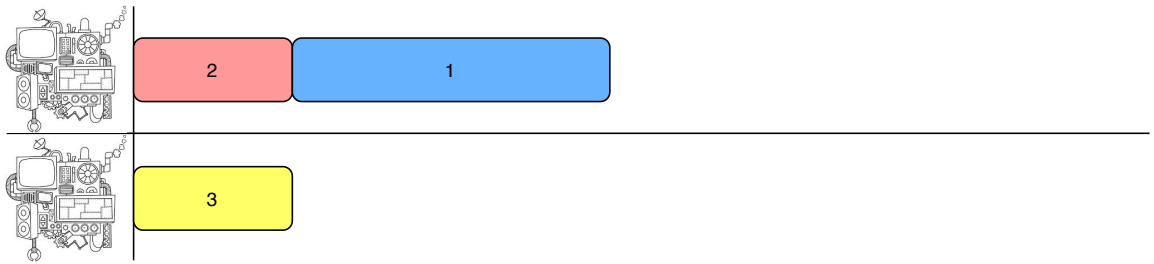
Since 2,3 go first, 1 starts as soon as either finishes.

Example

- 3 tasks, 2 machines.
- Tasks' durations (d_1, d_2, d_3) belong to



- Optimal static allocation:

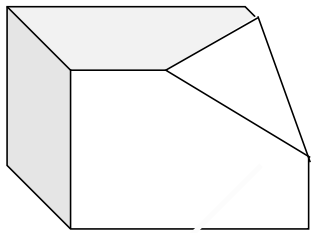


Since 2,3 go first, 1 starts as soon as either finishes.

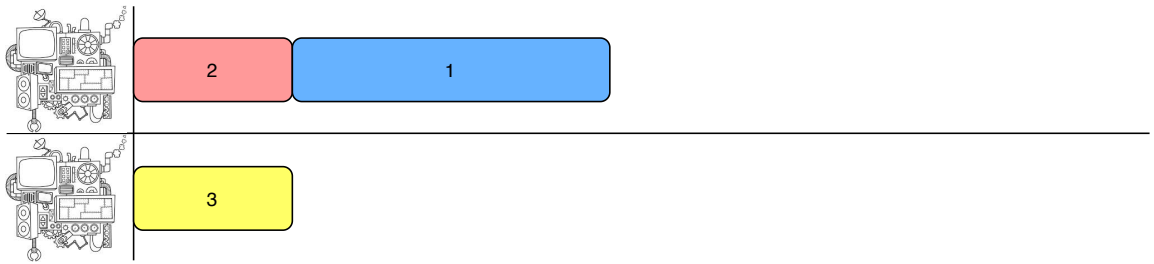
Worst case duration 1.8805

Example

- 3 tasks, 2 machines.
- Tasks' durations (d_1, d_2, d_3) belong to



- Optimal static allocation:



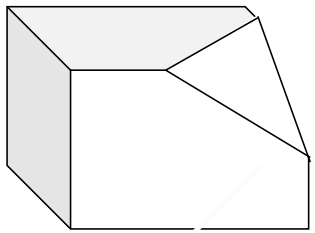
Since 2,3 go first, 1 starts as soon as either finishes.

Worst case duration 1.8805

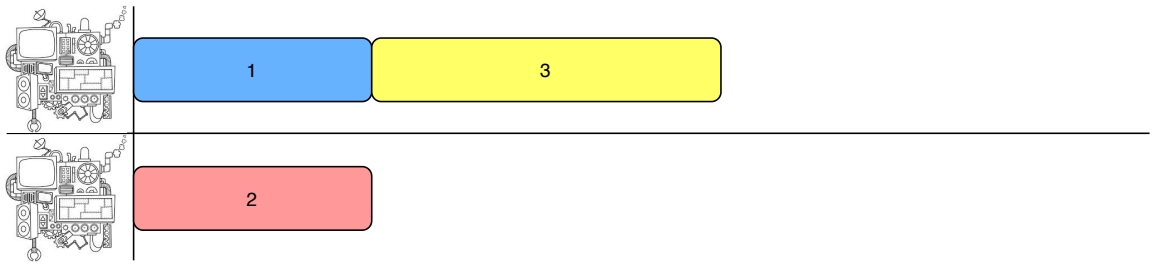
Is this solution optimal?

Example

- 3 tasks, 2 machines.
- Tasks' durations (d_1, d_2, d_3) belong to



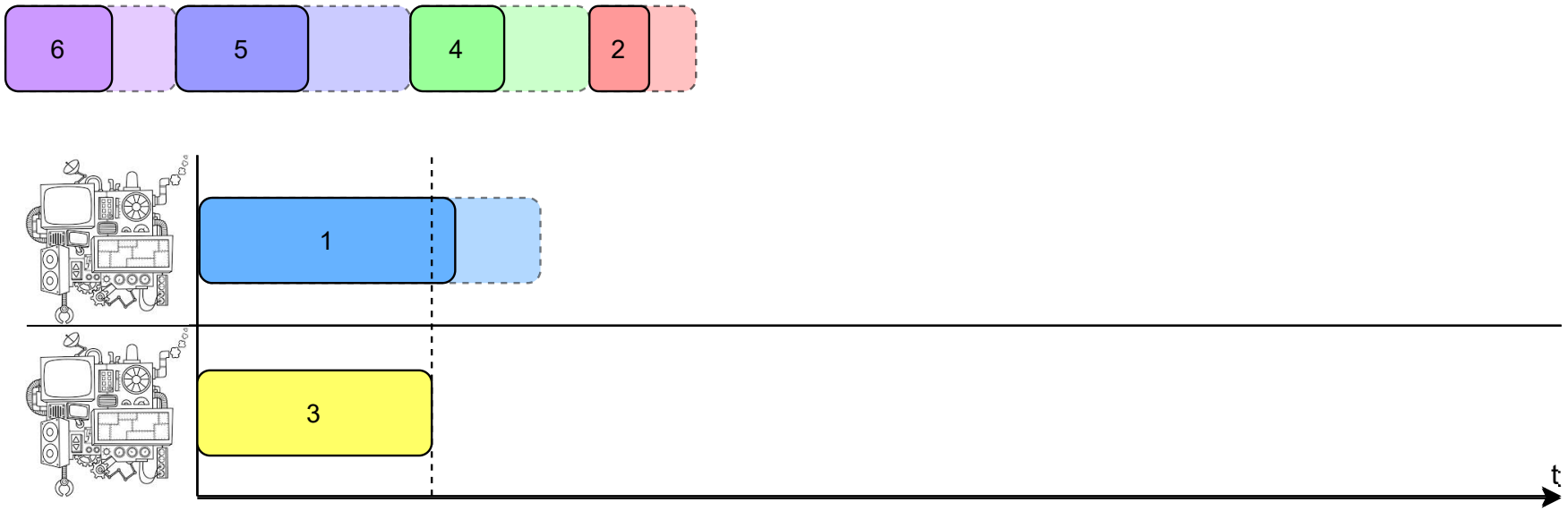
- Optimal adaptive schedule:



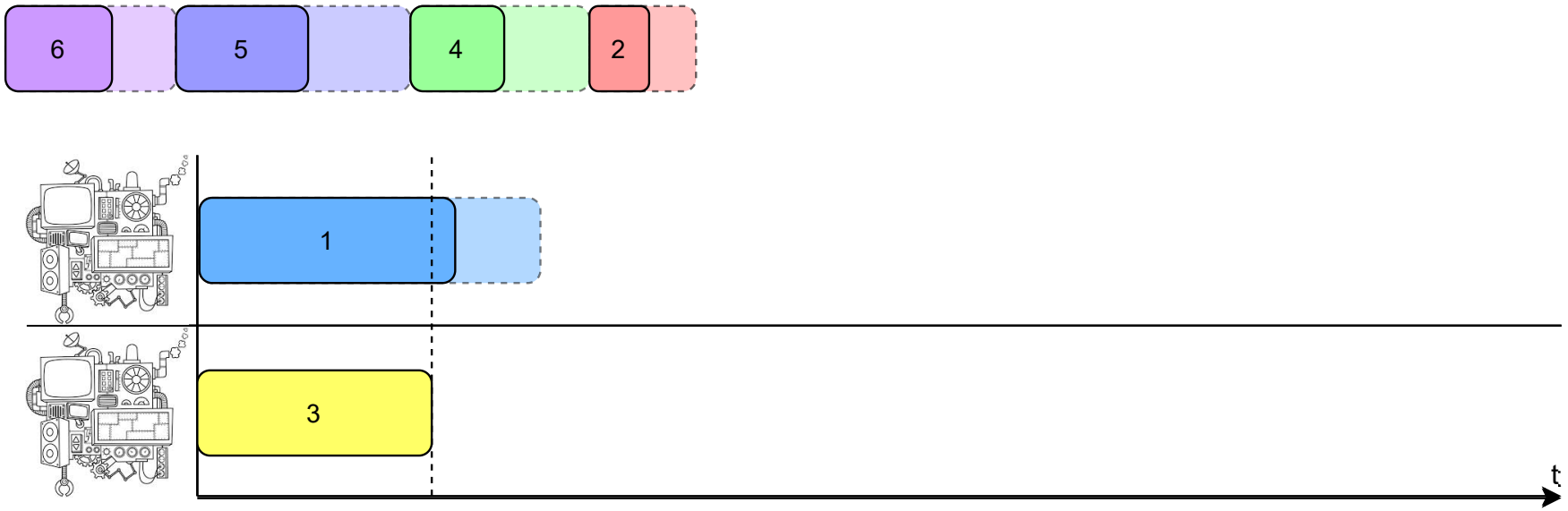
1 and 2 scheduled, 3 starts when either finishes.

Worst case duration 1.8295

Solution approach: state space



Solution approach: state space



$$T \begin{pmatrix} \text{Started} \\ \text{Finished} \\ \text{Finished durations} \\ \text{Running} \\ \text{Running durations} \end{pmatrix}$$

Scheduler's recursion

$$
\begin{aligned}
& T(S, F, D, i, \bar{D}_i) \\
= & \min_{k \notin F \cup \{i\}} \max \Bigg\{ \max_{\substack{d_k : d \in U_{[S,k],F,D,i,\bar{D}_i}, \\ d_k \leq d_i - \bar{D}_i}} d_k + T([S,k],[F,k],[D,d_k],i,\bar{D}_i + d_k), \\
& \qquad \max_{\substack{d_i : d \in U_{[S,k],F,D,i,\bar{D}_i}, \\ d_k > d_i - \bar{D}_i}} d_i - \bar{D}_i + T([S,k],[F,i],[D,d_i],k,d_i - \bar{D}_i) \Bigg\}.
\end{aligned}
$$

Scheduler's recursion

$$T(S, F, D, i, \bar{D}_i)$$

$$= \min_{k \notin F \cup \{i\}} \max \Big(\qquad k], i, \bar{D}_i + d_k),$$

$$d_i : d \in \qquad k, d_i - \bar{D}_i) \Big\}.$$

Adversary's recursion

$$T'([S,k],F,D,i,\bar{D}_i) =$$

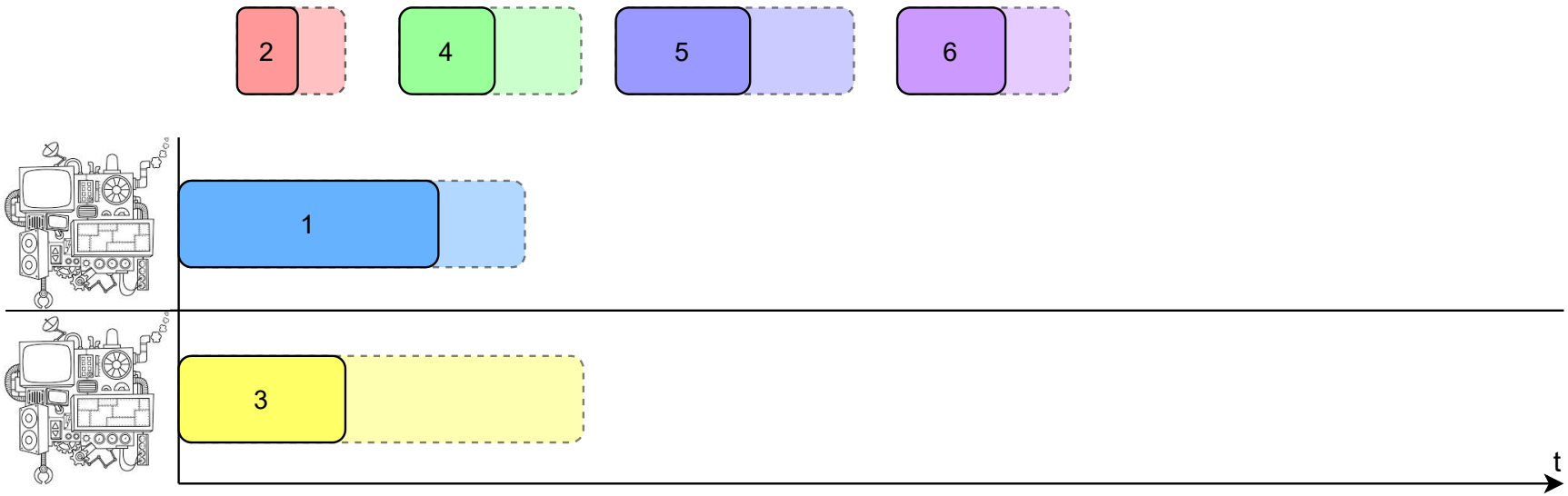
$$\max\Bigg\{ \max_{\substack{d_k : d \in U_{[S,k],F,D,i,\bar{D}_i}, \\ d_k \le d_i - \bar{D}_i}} d_k + \min_{l \notin S \cup \{k\}} T'([S,k,l],[F,k],[D,d_k],i,\bar{D}_i + d_k),$$

$$\max_{\substack{d_i : d \in U_{[S,k],F,D,i,\bar{D}_i}, \\ d_k > d_i - \bar{D}_i}} d_i - \bar{D}_i + \min_{l \notin S \cup \{k\}} T'([S,k,l],[F,i],[D,d_i],k,d_i - \bar{D}_i) \Bigg\},$$

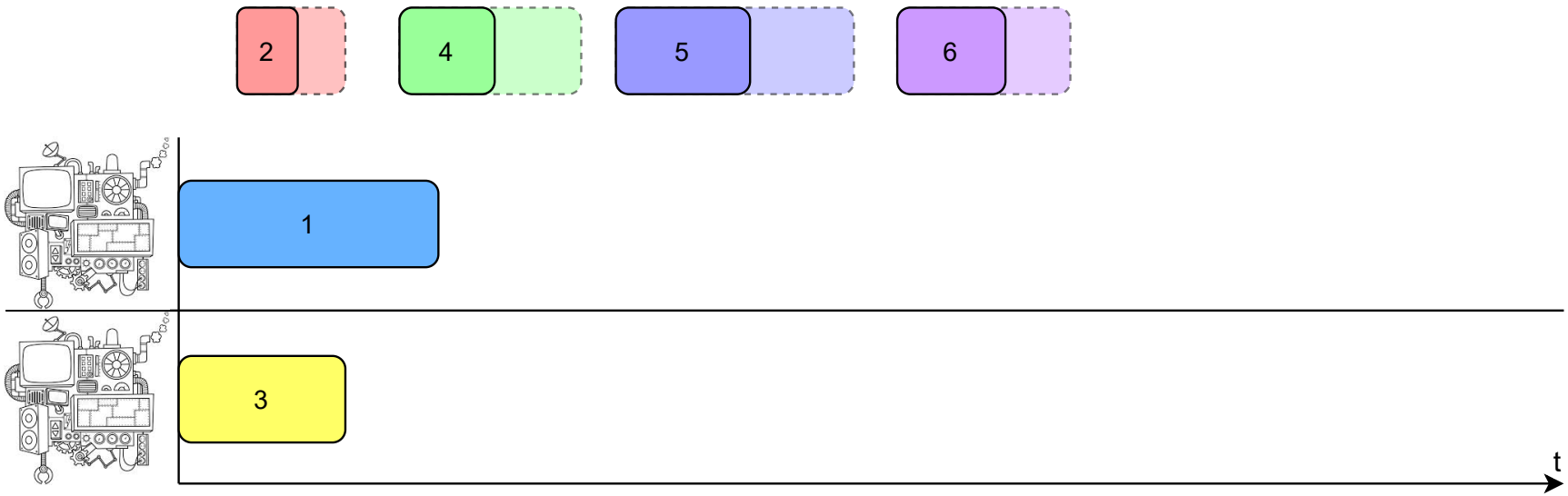
Adversary's recursion

$$
\begin{aligned}
\max_{d_\sigma, t_\sigma, z_\sigma} \quad & t_0 \\
\text{s.t.} \quad & t_\sigma = \min_{\delta \in \text{Children}(\sigma)} t_\delta && \forall \sigma \in \mathcal{D} \\
& t_\sigma = \max_{\delta \in \text{Children}(\sigma)} t_\delta && \forall \sigma \in \mathcal{N} \setminus \mathcal{L} \\
& d_\sigma(i) = d_\delta(i) && \forall i \in F, (S,F) = \mathcal{A}(\sigma,\delta), \sigma, \delta \in \mathcal{L} d_{\sigma,\sigma(i)} = d_{\delta,\delta(i)} \quad \forall i \\
& t_\sigma = e_\sigma^\top d_\sigma - z_\sigma M && \forall \sigma \in \mathcal{L} \\
& E_\sigma d_\sigma \leq z_\sigma M && \forall \sigma \in \mathcal{L} \\
& z_\sigma \in \{0,1\},\ d_\sigma \in U && \forall \sigma \in \mathcal{L},
\end{aligned}
$$

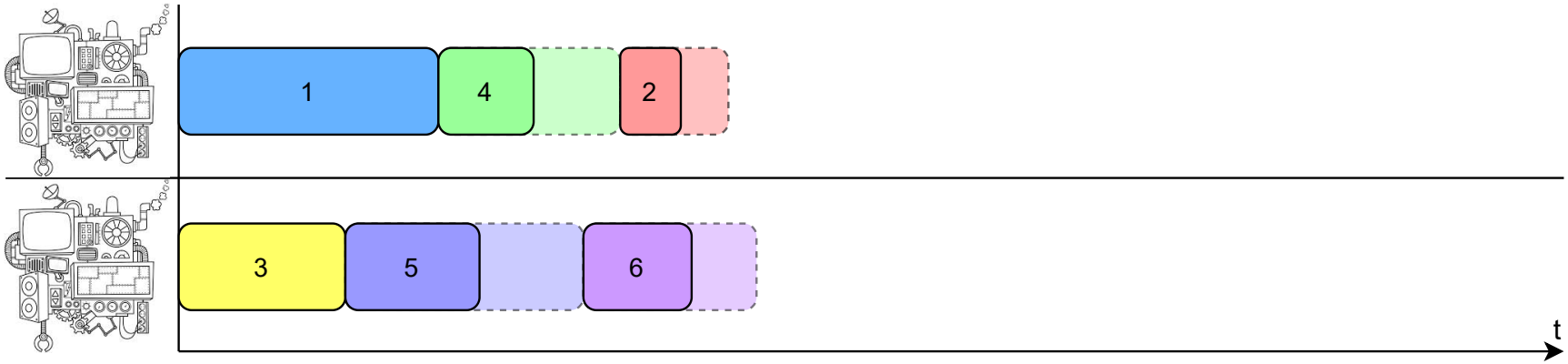
Adaptive heuristic: two-stage static allocation



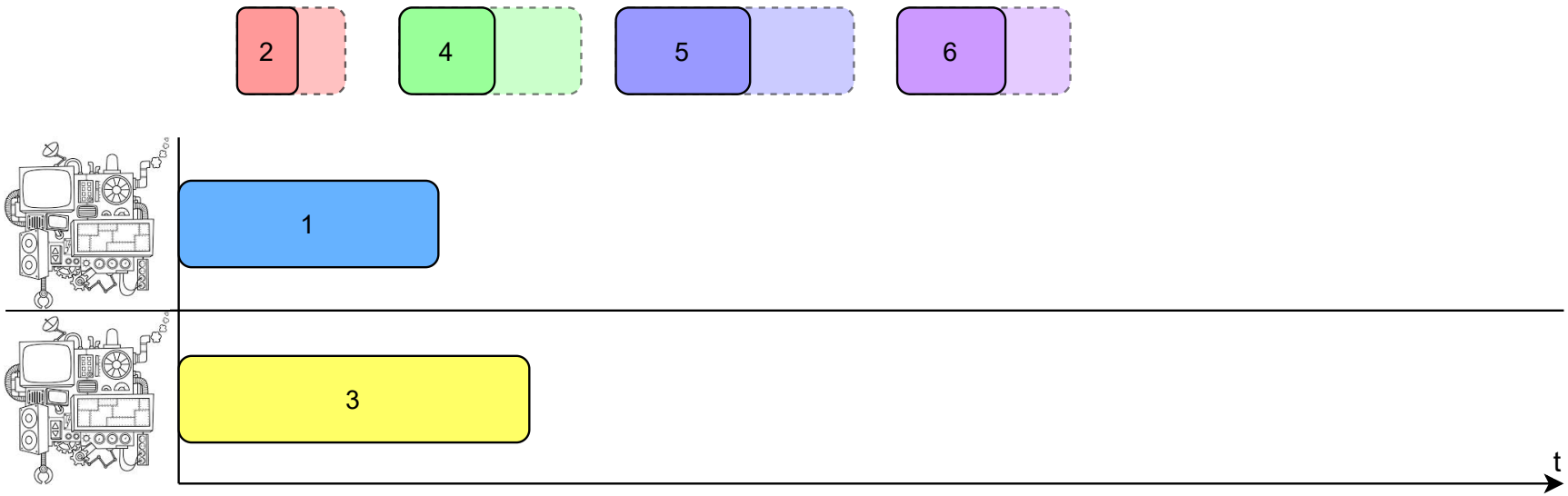
Adaptive heuristic: two-stage static allocation



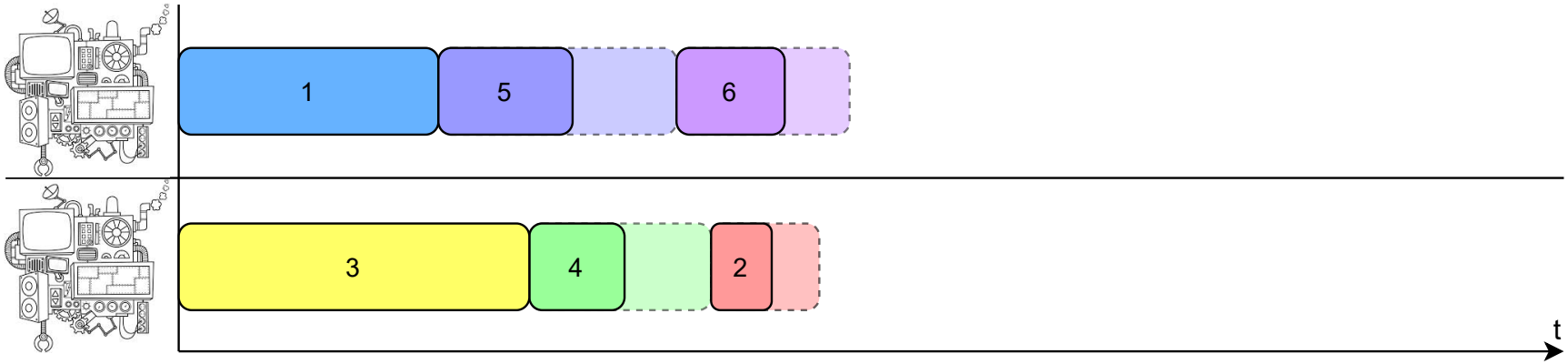
Adaptive heuristic: two-stage static allocation



Adaptive heuristic: two-stage static allocation



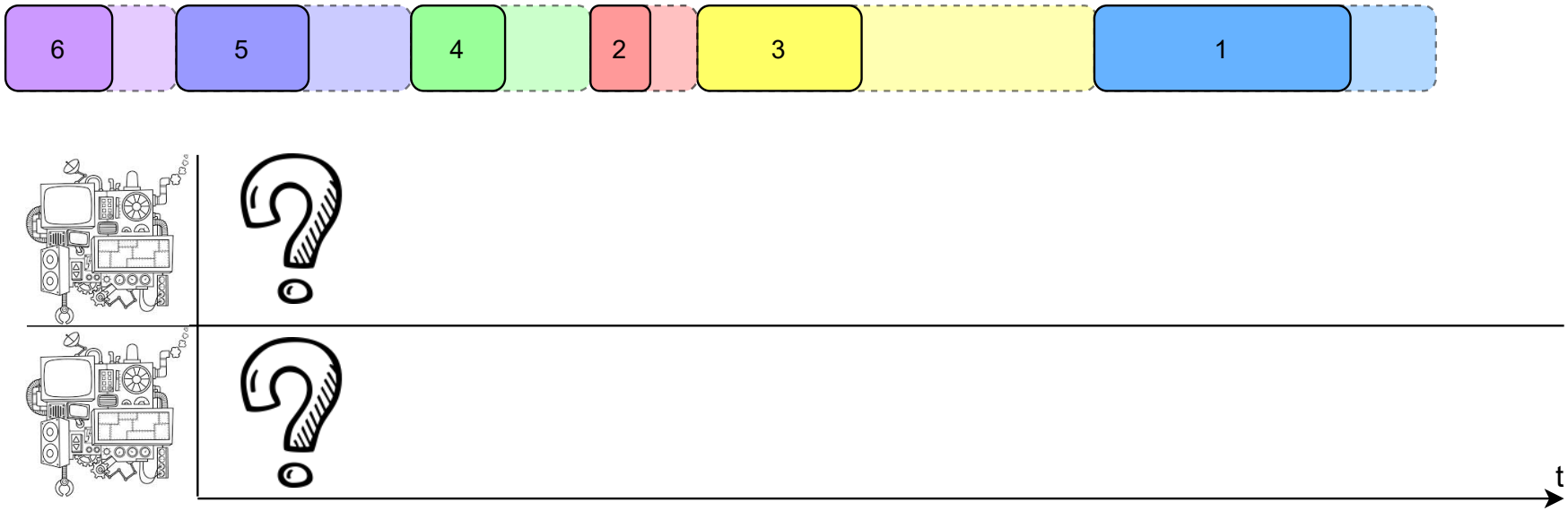
Adaptive heuristic: two-stage static allocation



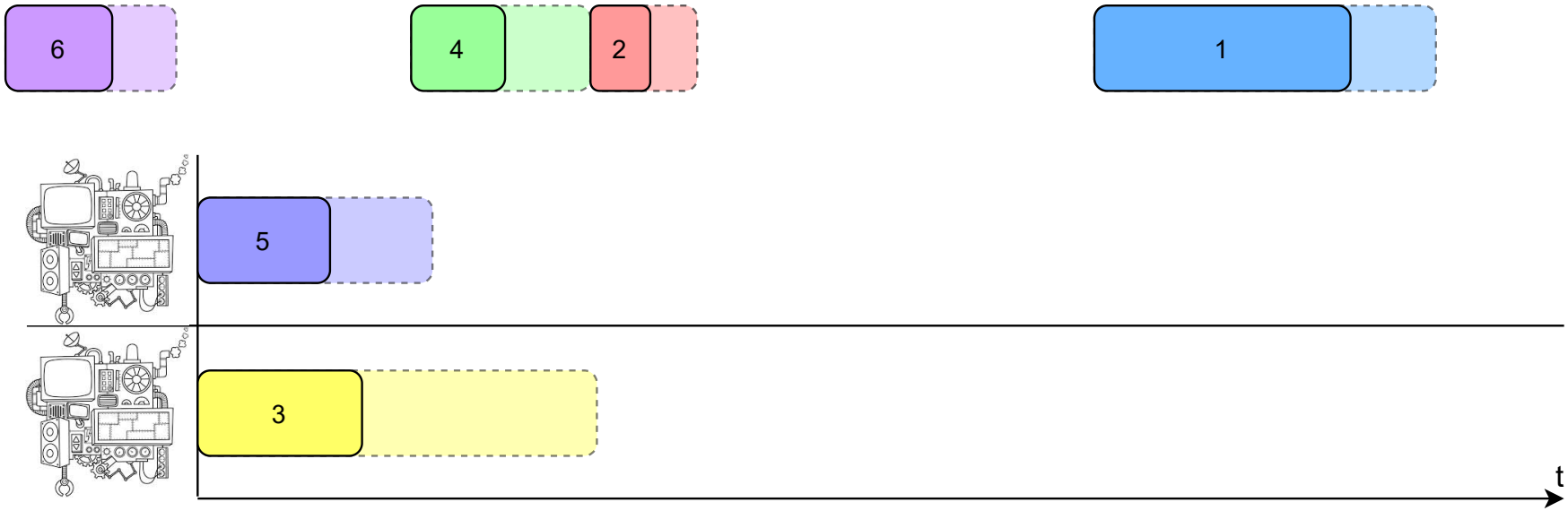
Tested methods

	Adaptive?	Scalable?
SA: Static allocation	✗	✓
SL: Static list	✓ / ✗	✗
AR: Adaptive	✓	✗
2SSA: Two-stage static	✓ / ✗	✓
PH: Perfect hindsight	-	-

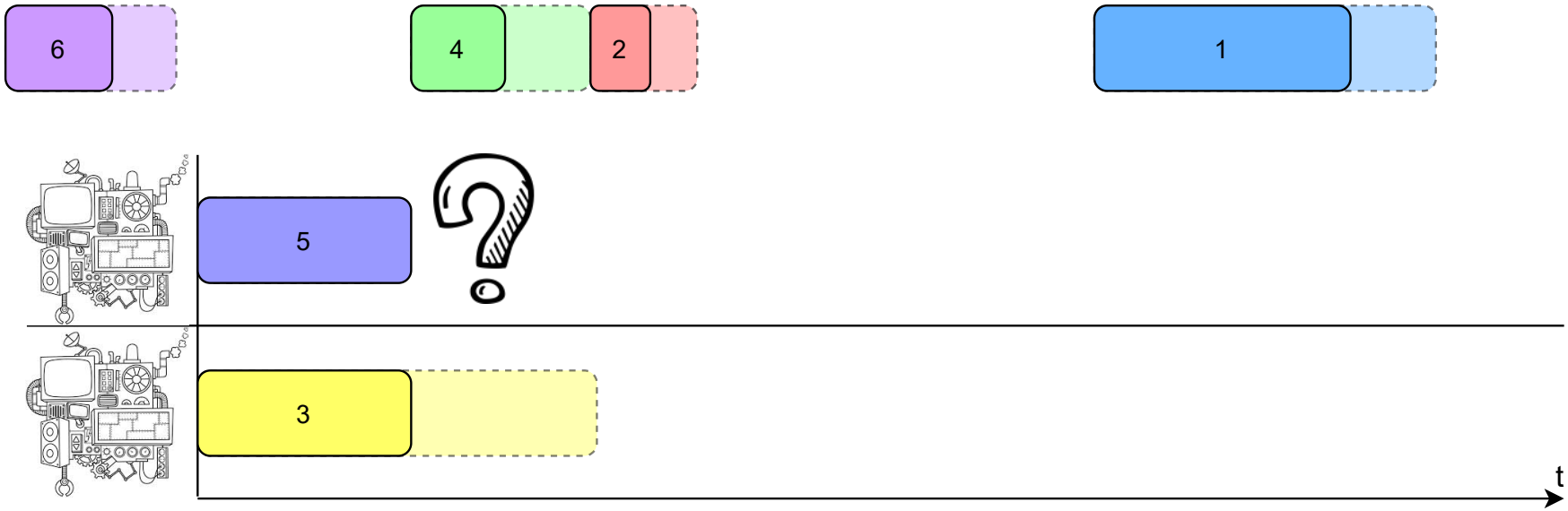
Rolling horizon



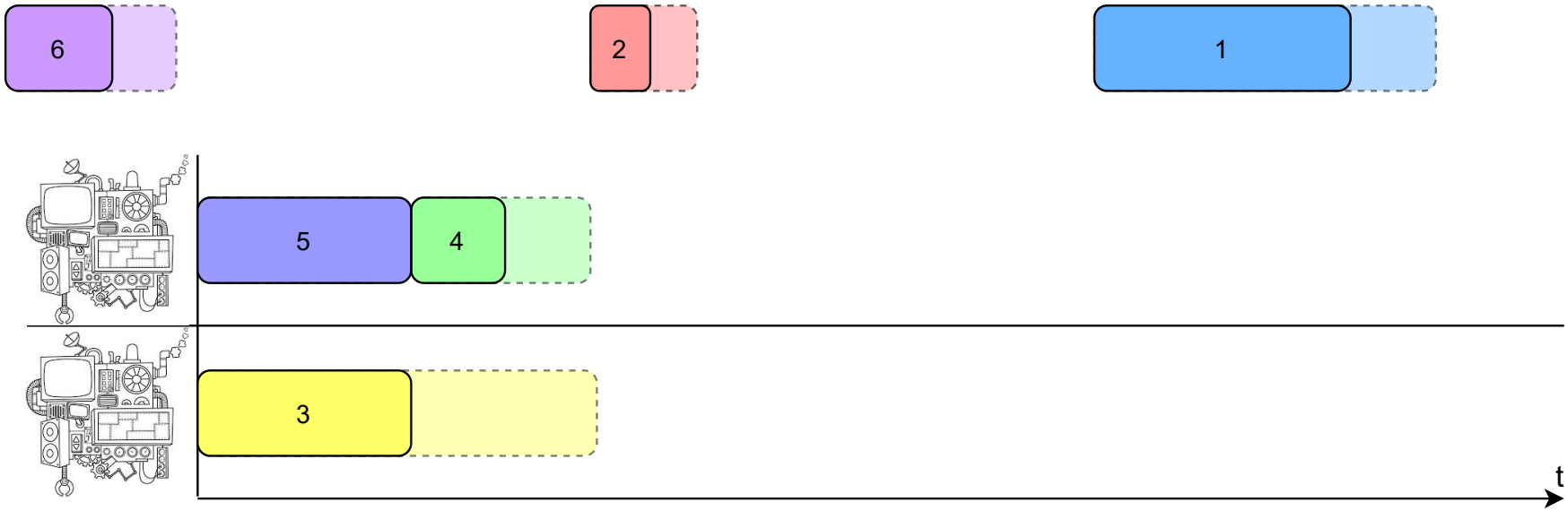
Rolling horizon



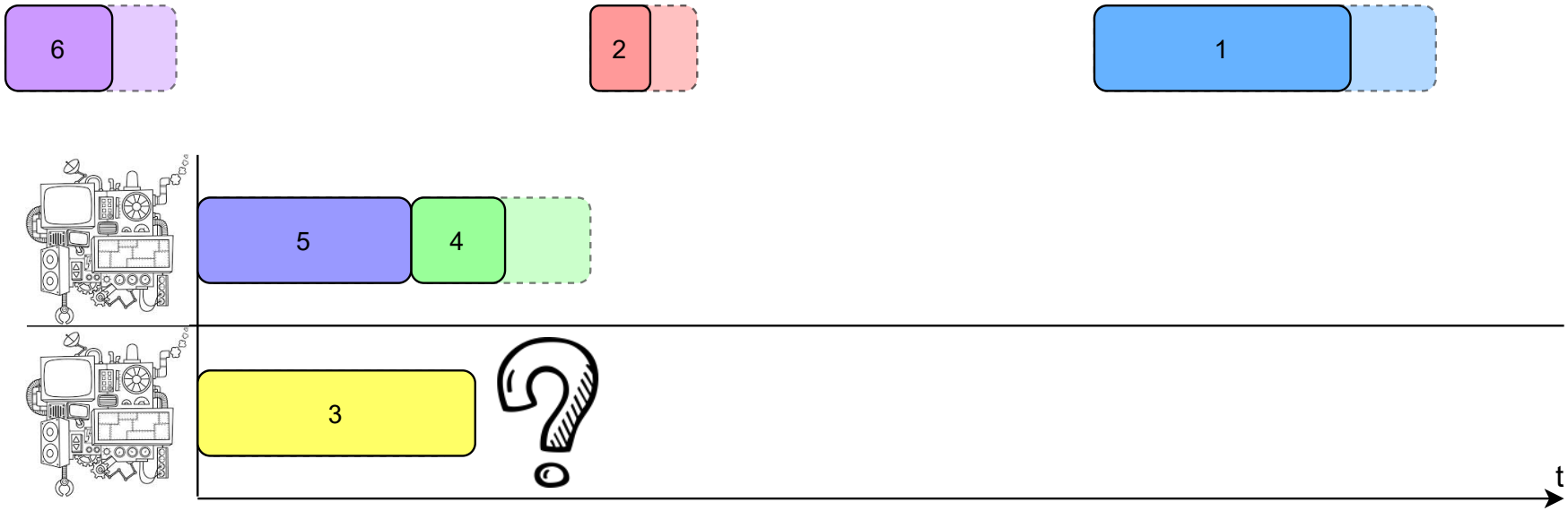
Rolling horizon



Rolling horizon



Rolling horizon



Small instances

- 5 tasks
- 2 machines
- 15 scenarios

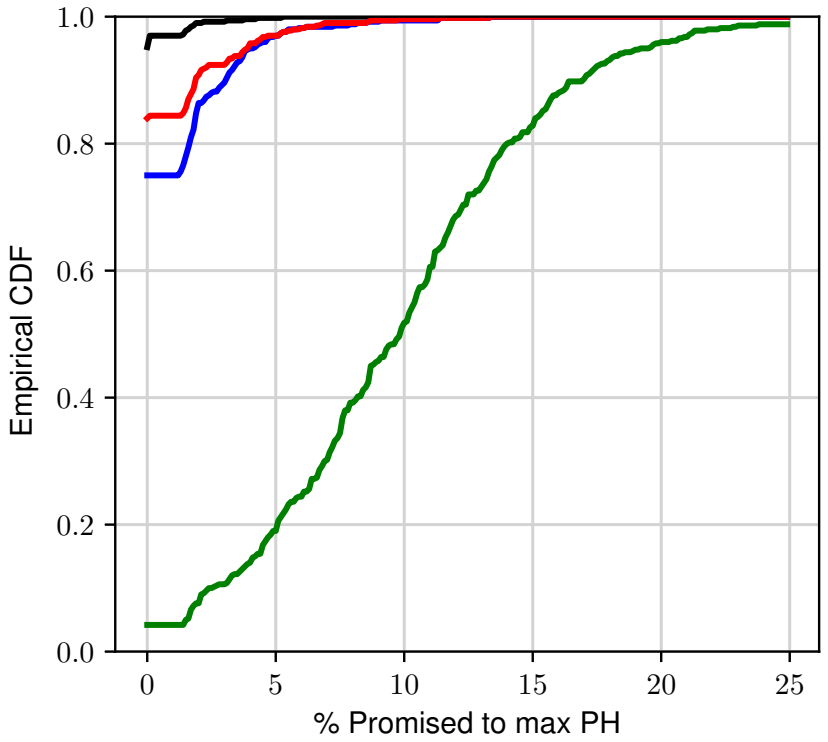
Small instances

- 5 tasks
- 2 machines
- 15 scenarios

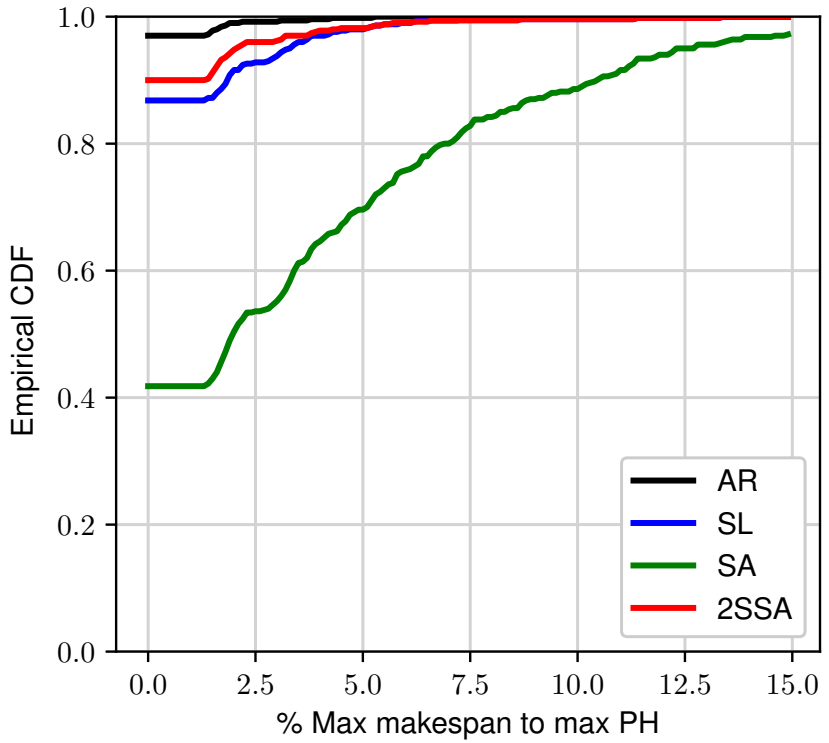
	Non-optimal here-and-now
SA	50%
SL	11%
2SSA	7%

Small instances

- 5 tasks
- 2 machines
- 15 scenarios



(a)



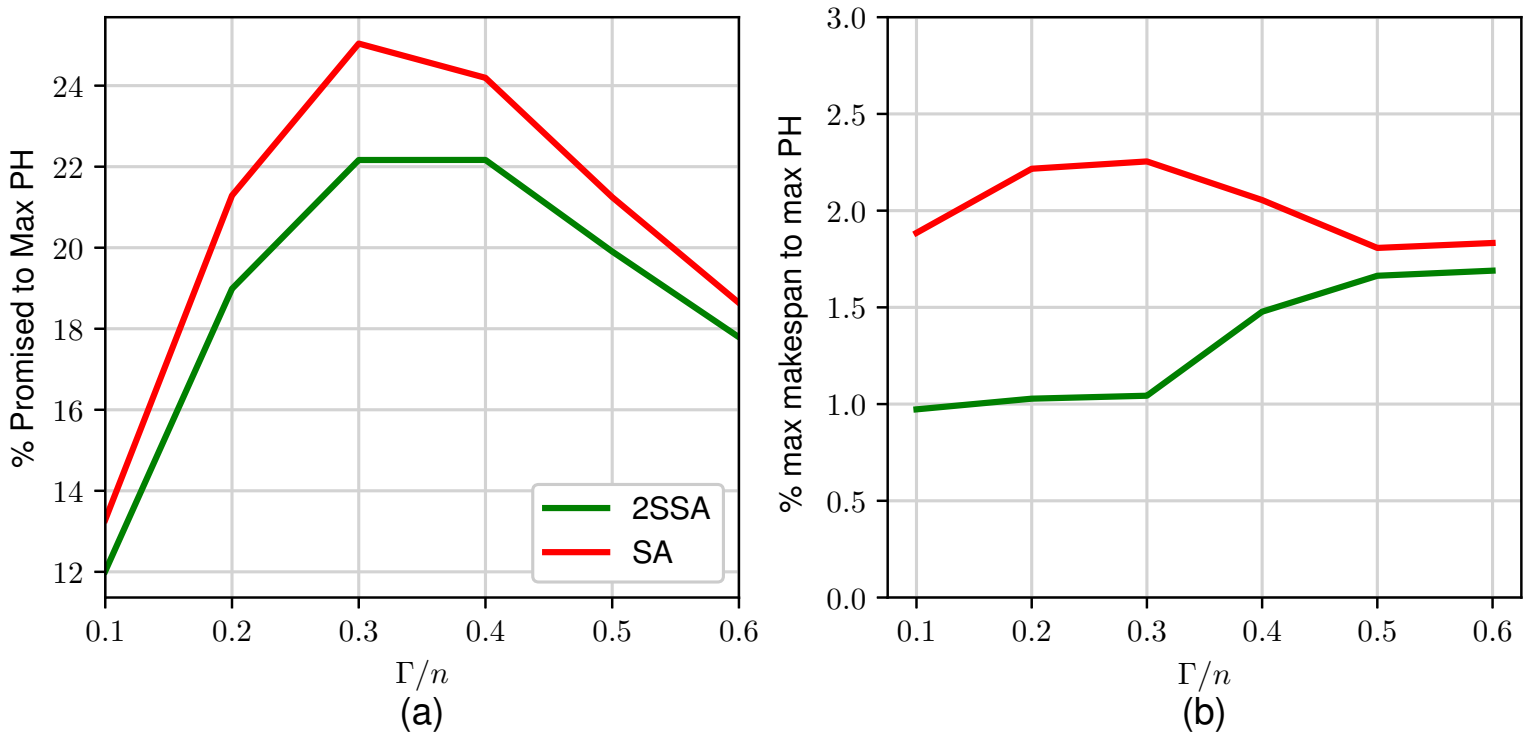
(b)

Big instances

- $n = 20$ tasks
- budgeted uncertainty: Γ

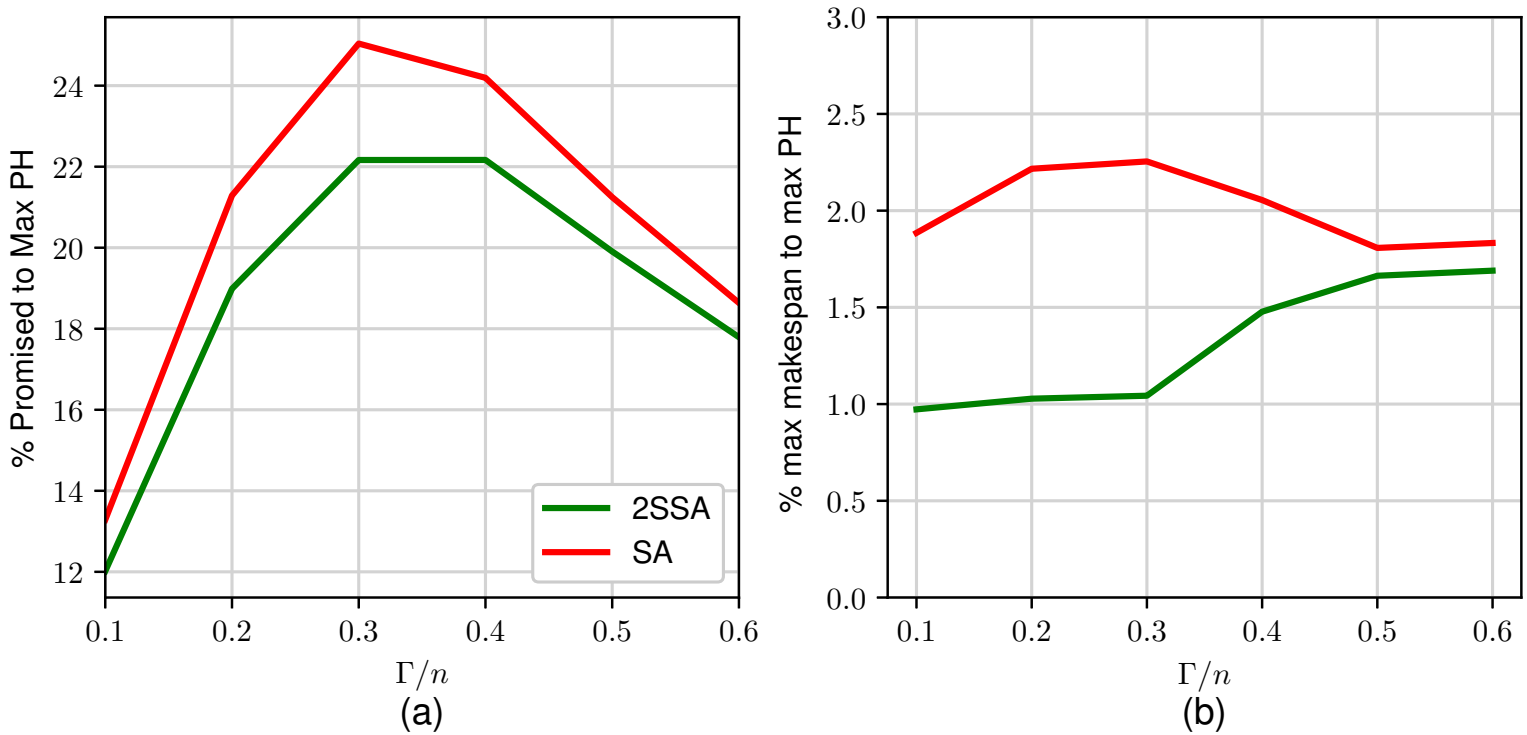
Big instances

- $n = 20$ tasks
- budgeted uncertainty: Γ



Big instances

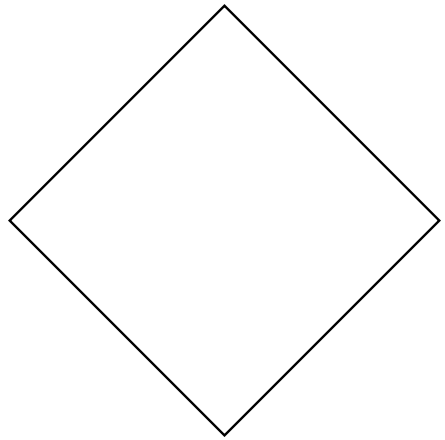
- $n = 20$ tasks
- budgeted uncertainty: Γ



Same problem - different look

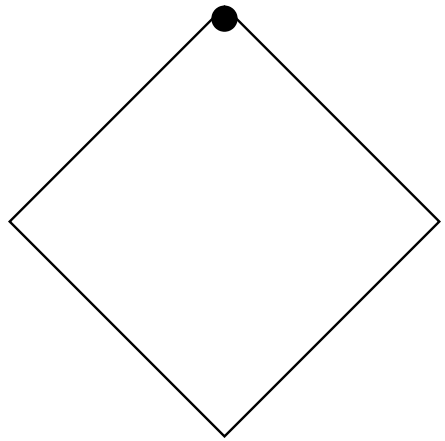
- So far – trying to consider all possible scenarios in one go
- Probably only a few scenarios out of U 'really matter'
- Proposal: a different procedure
- Loop between
  - Creating a schedule for a 'bag' of scenarios
  - Finding a scenario that is makes this schedule have a longer duration + and adding that scenario to the bag of scenarios

The new idea graphically



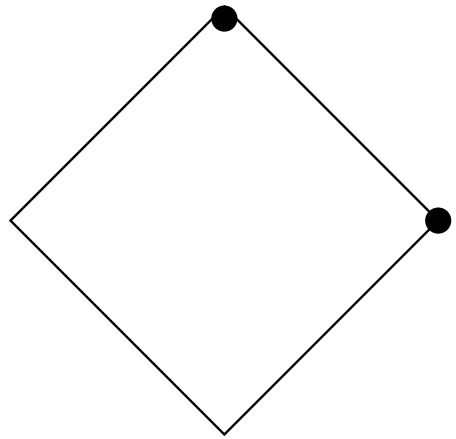
A schedule for a subset of scenarios gives a lower bound

The new idea graphically



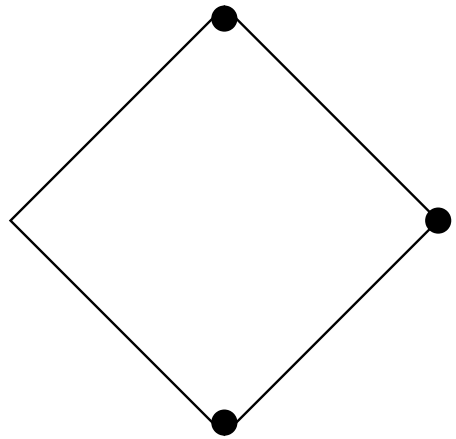
A schedule for a subset of scenarios gives a lower bound

The new idea graphically



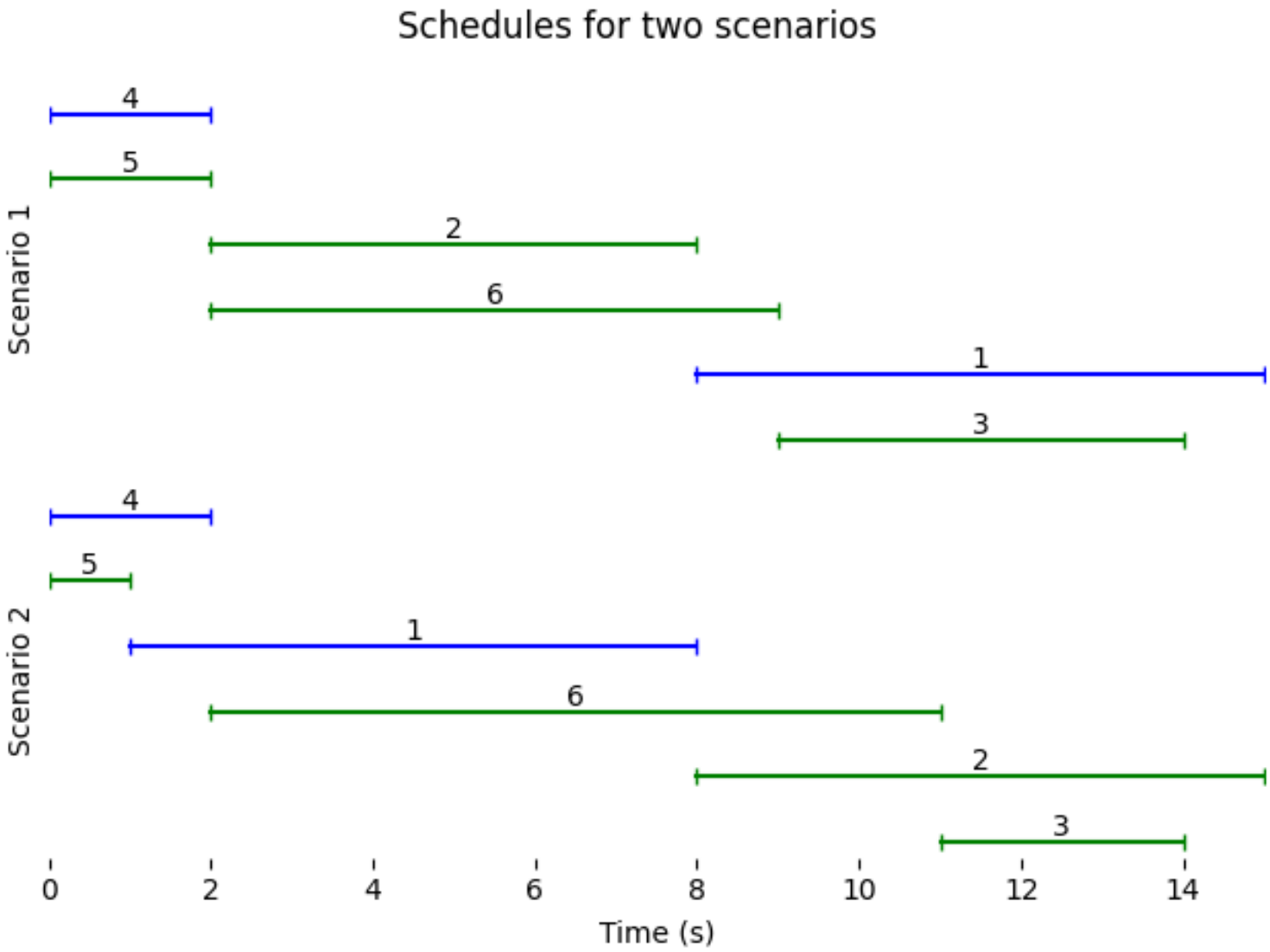
A schedule for a subset of scenarios gives a lower bound

The new idea graphically



A schedule for a subset of scenarios gives a lower bound

The new idea graphically



Summary

- Ignoring adaptivity bad on
  - contract
  - actual execution

Summary

- Ignoring adaptivity bad on
  - contract
  - actual execution
- Our results can improve:
  - Dynamic programming
  - Approximation techniques

Summary

- Ignoring adaptivity bad on
  - contract
  - actual execution
- Our results can improve:
  - Dynamic programming
  - Approximation techniques
- An almost untouched theoretical territory

Thank you!