RTCSA'14

Power Minimization for Parallel Real-Time Systems with Malleable Jobs and Homogeneous Frequencies

Antonio Paolillo

Joël Goossens

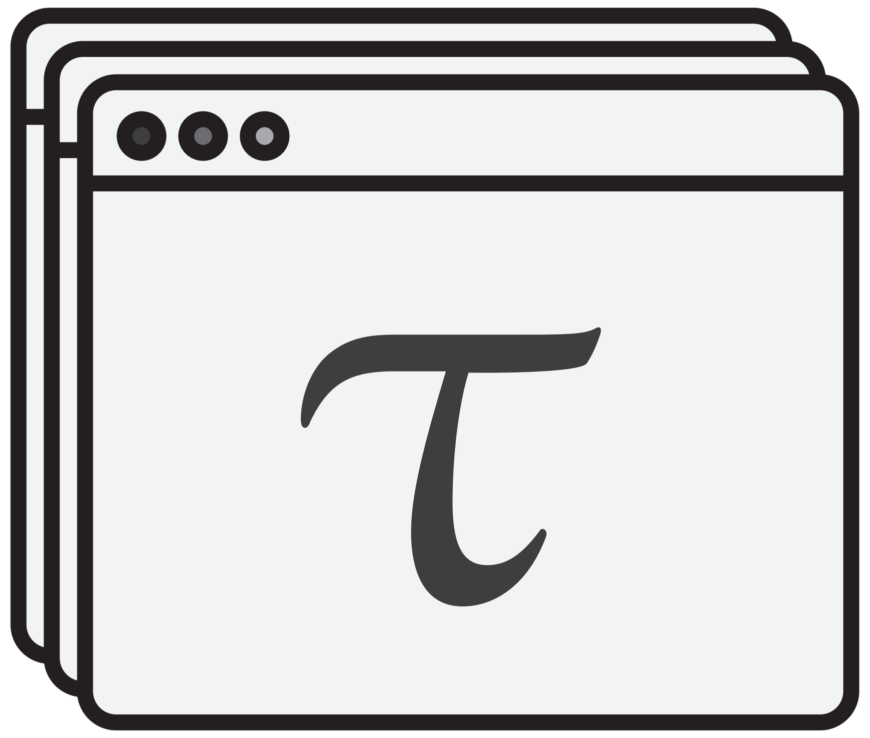
Pradeep M. Hettiarachchi

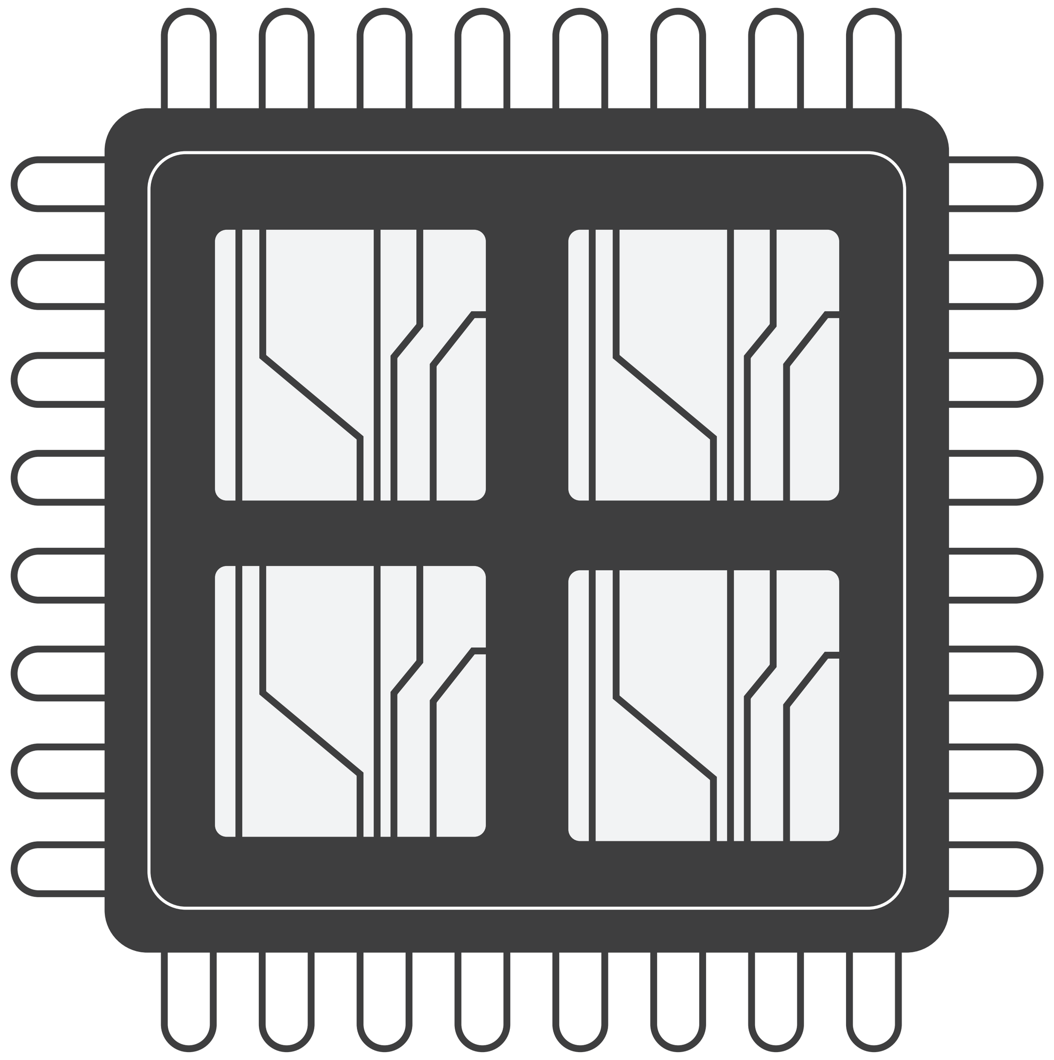
Nathan Fisher

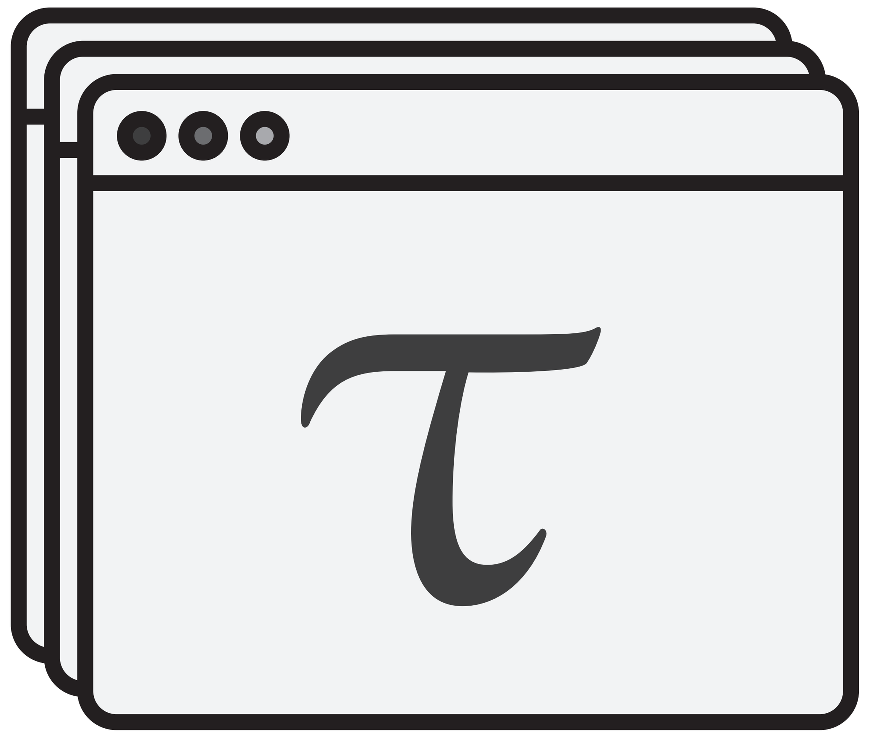
ULB

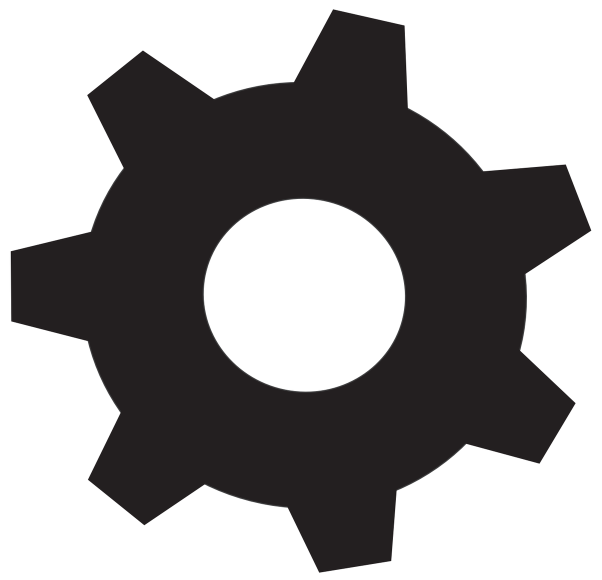
WAYNE STATE

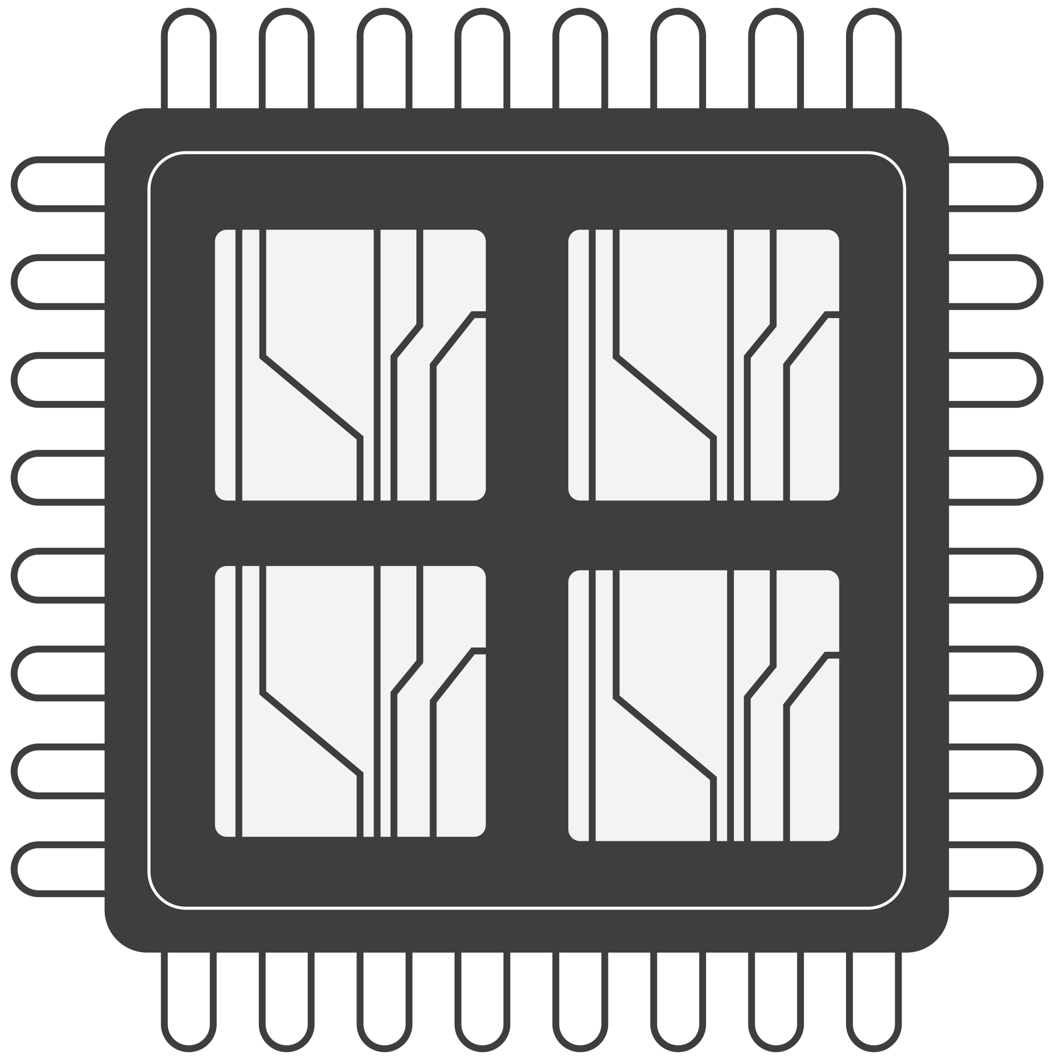
MangoGem


τ



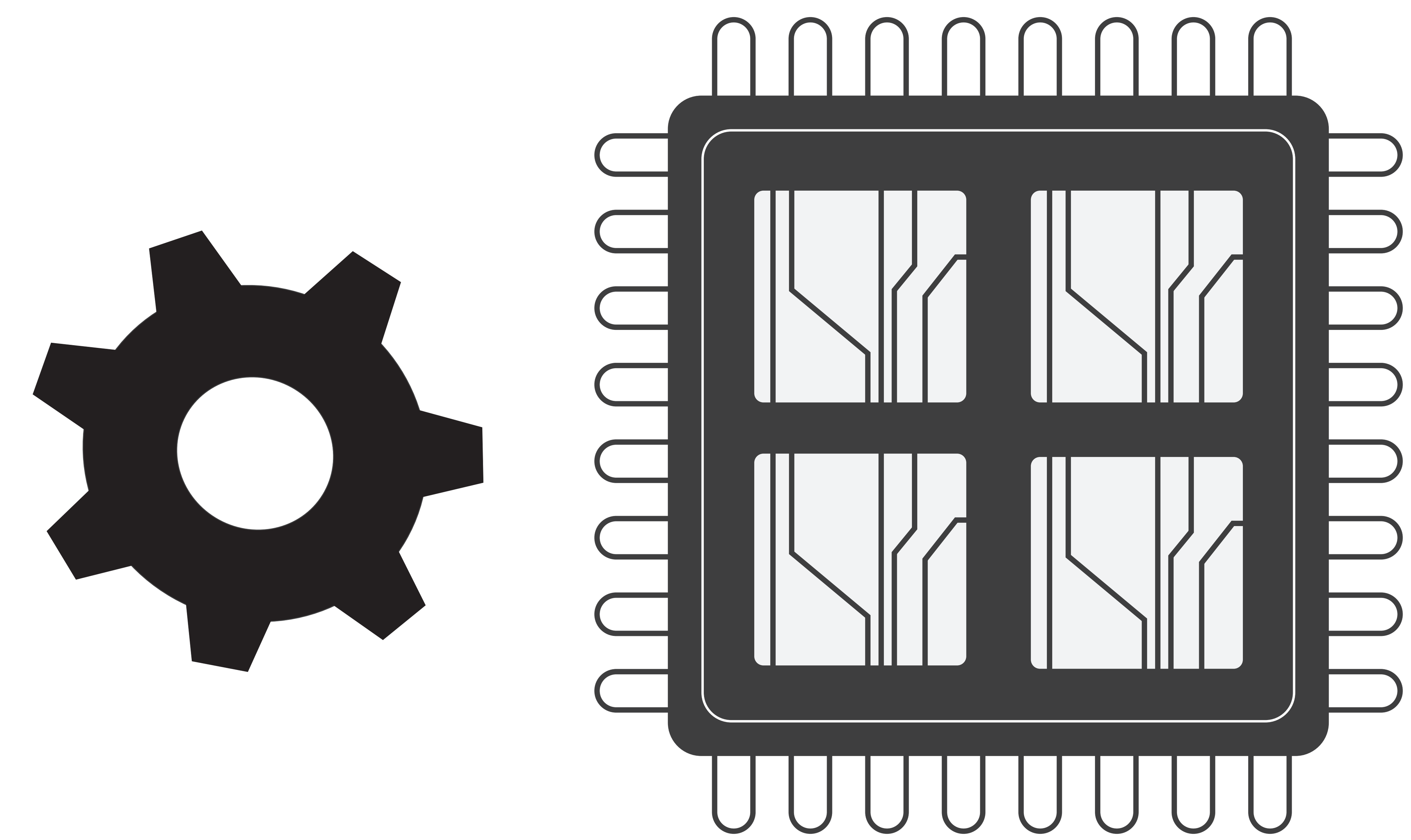

τ






τ


τ



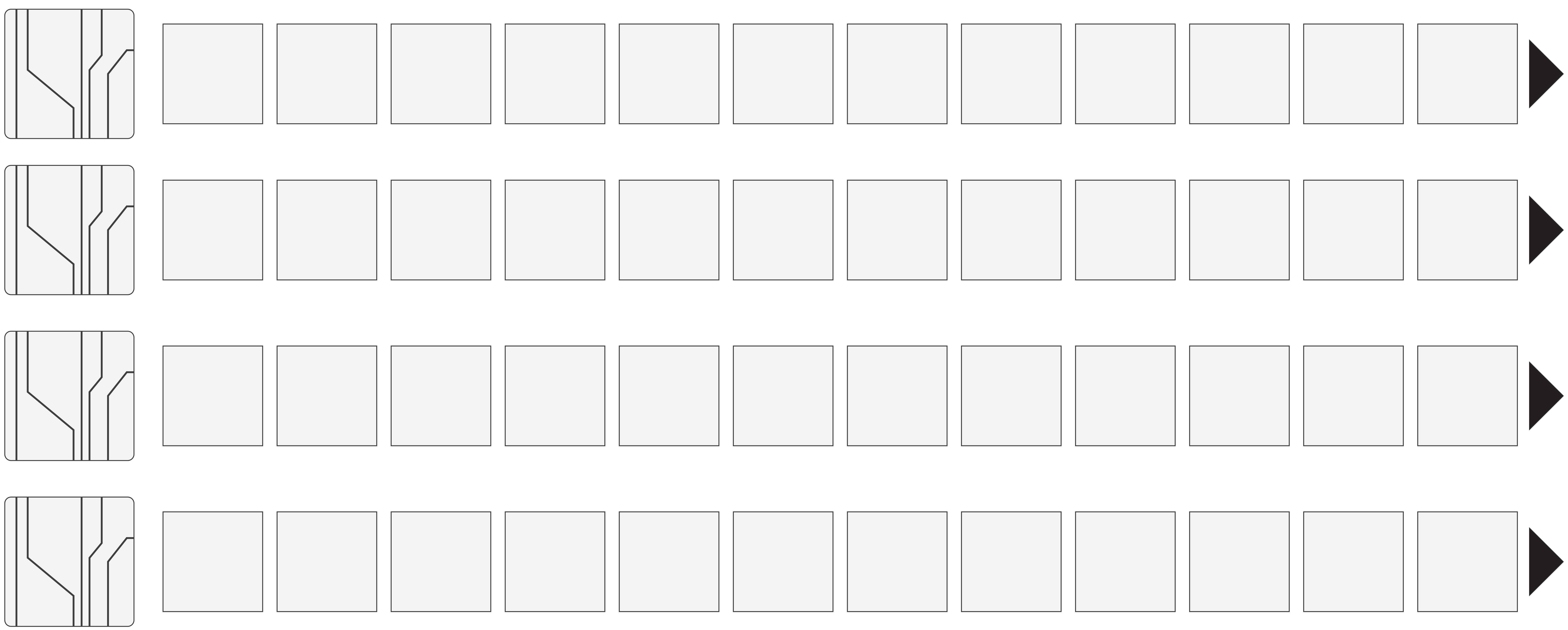

τ


τ

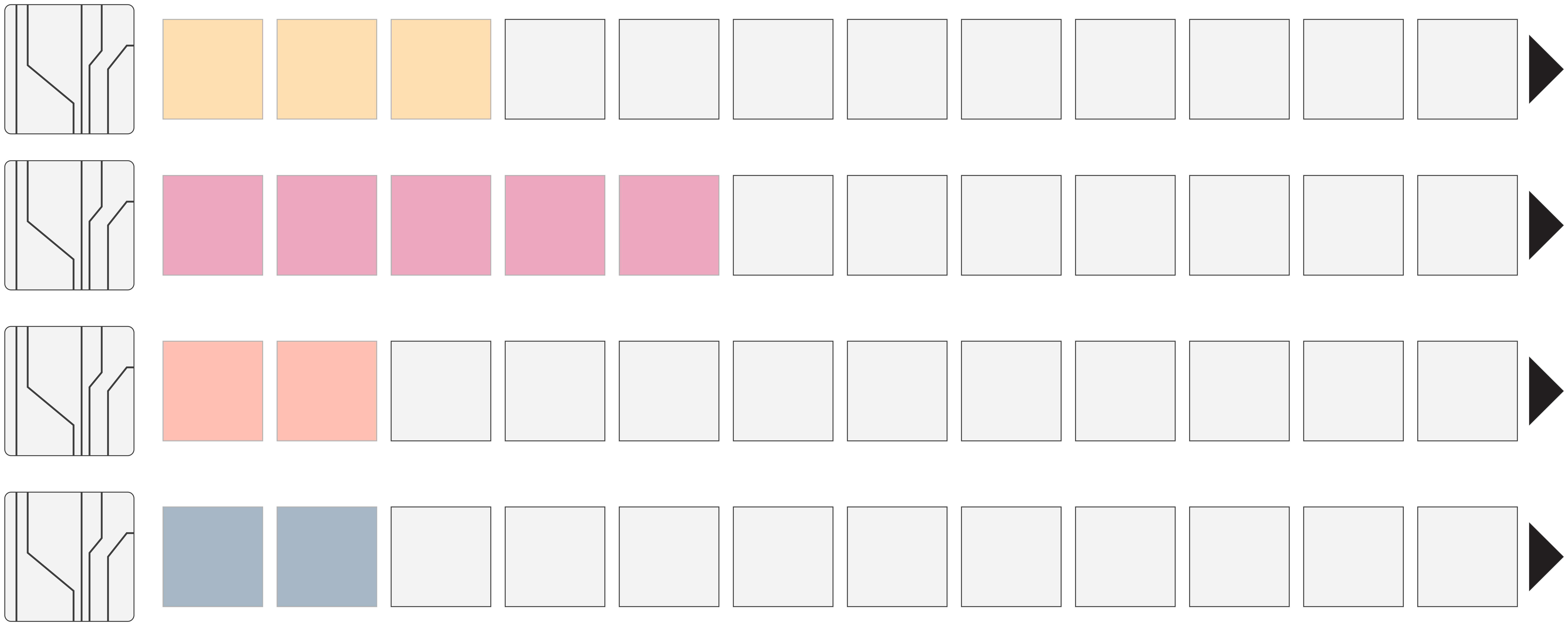

τ


τ

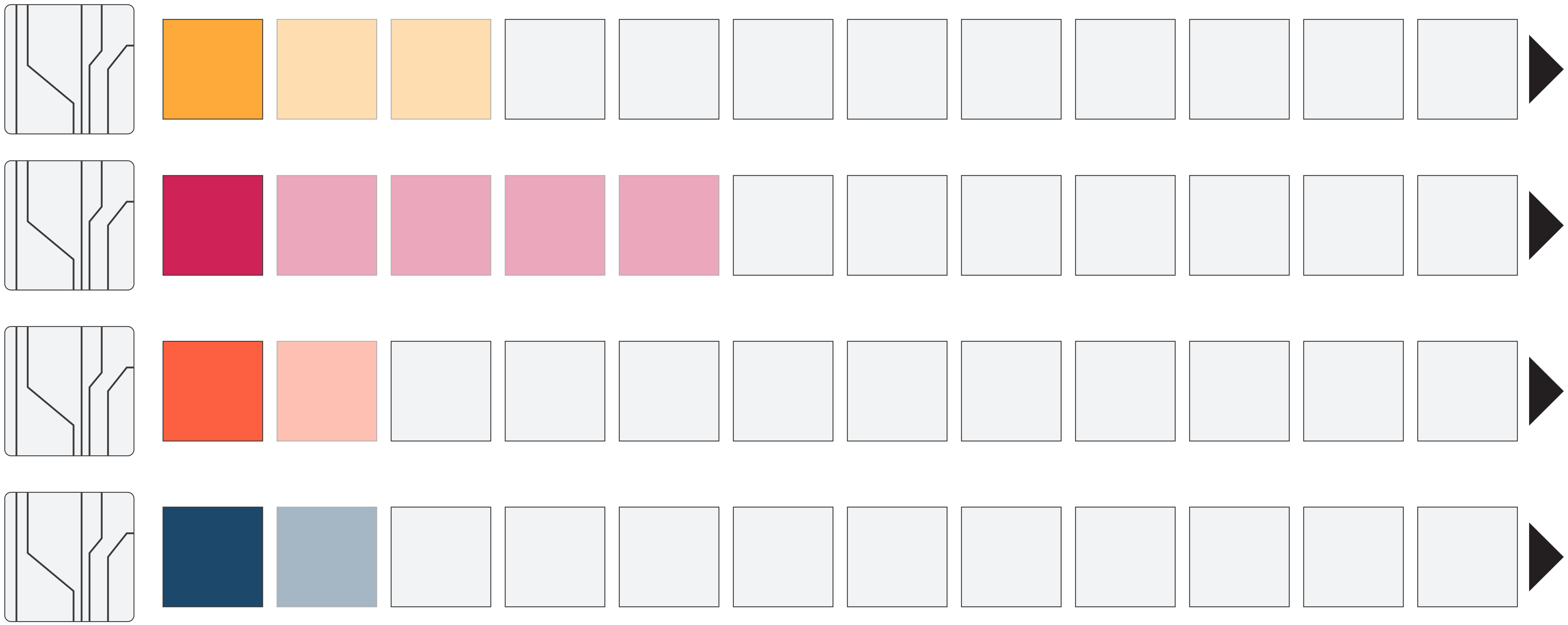
Sequential schedule



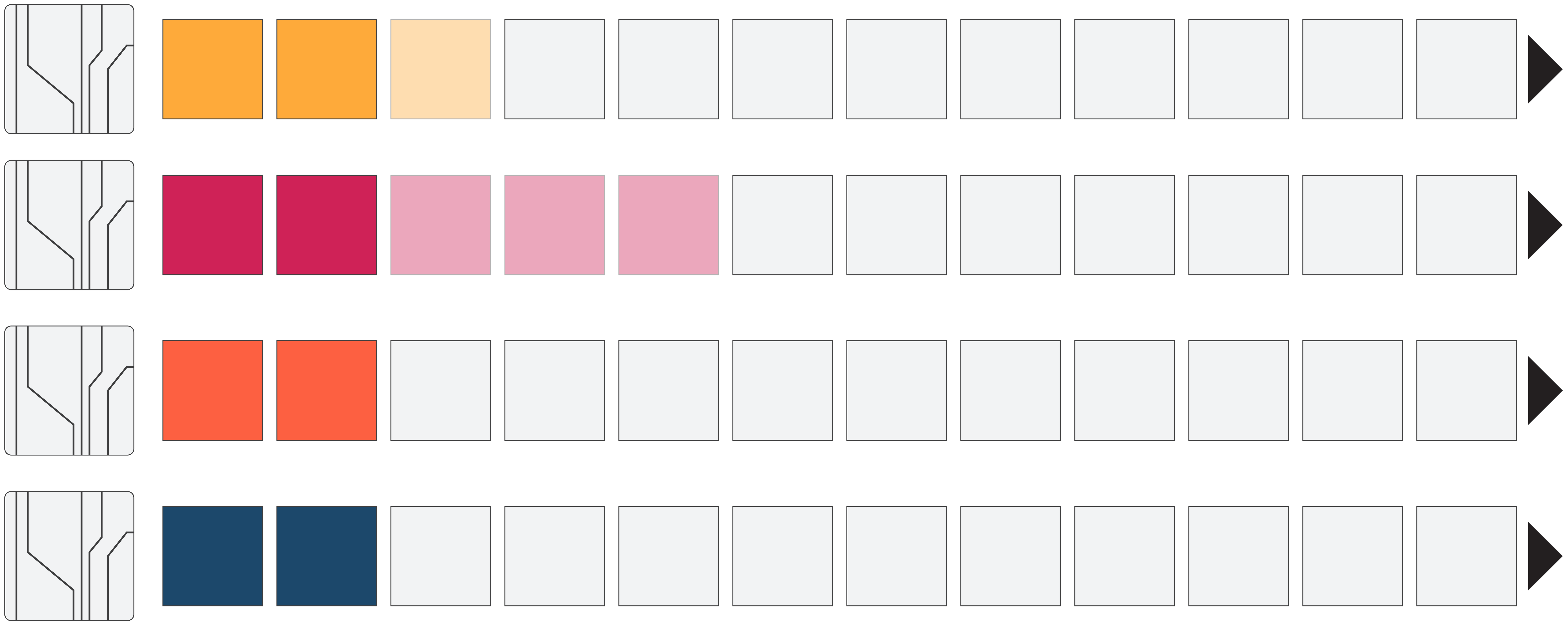
Sequential schedule



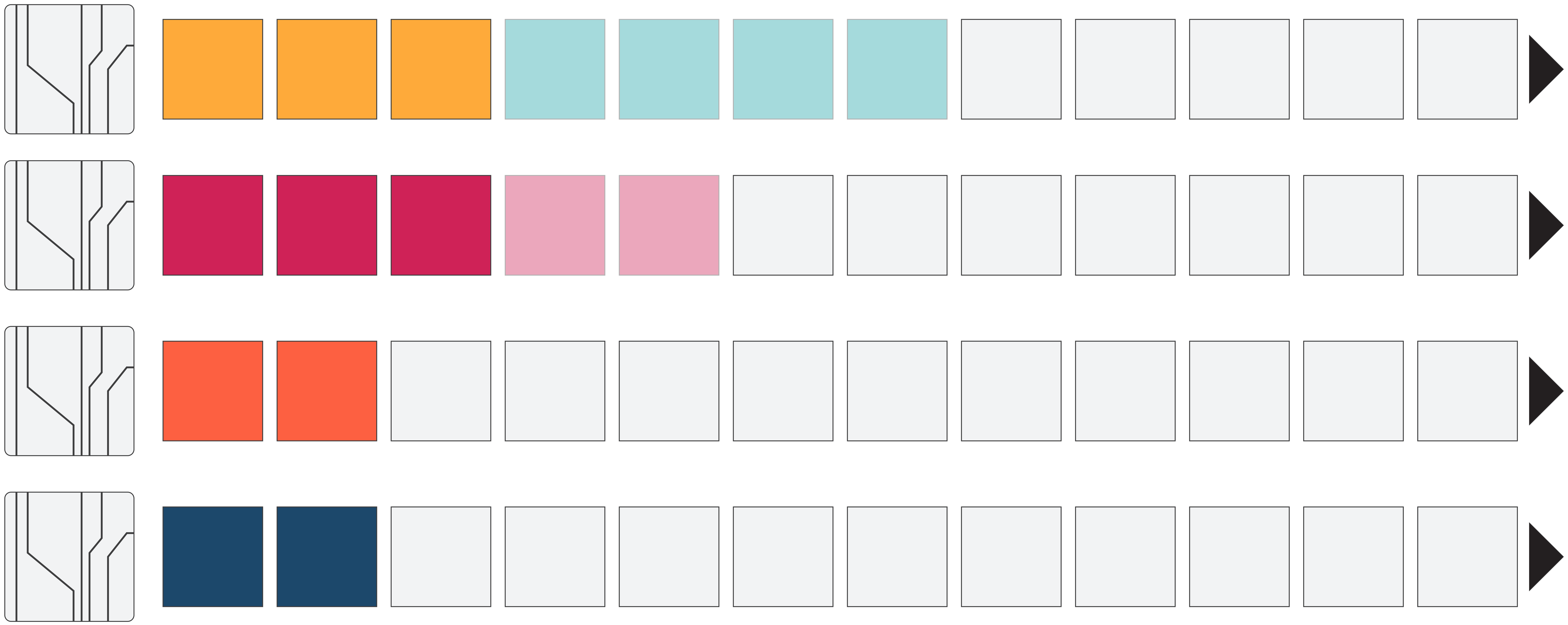
Sequential schedule



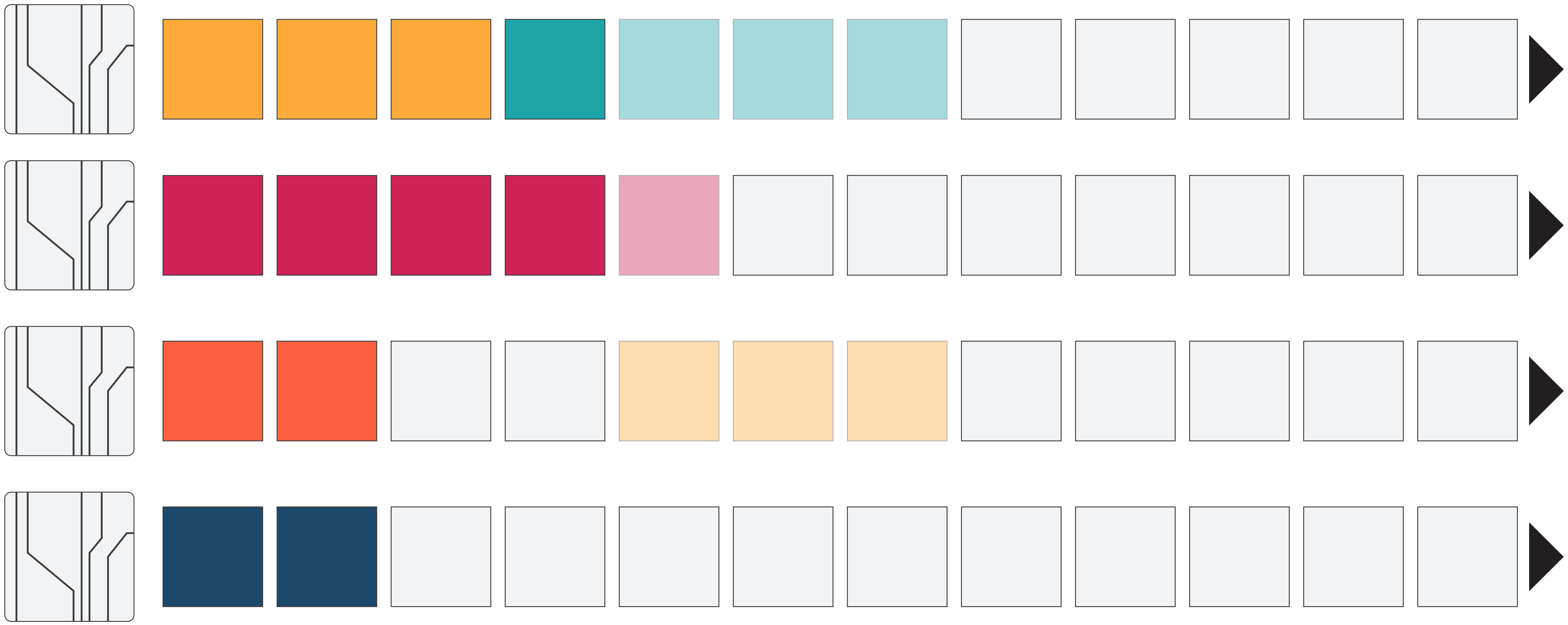
Sequential schedule



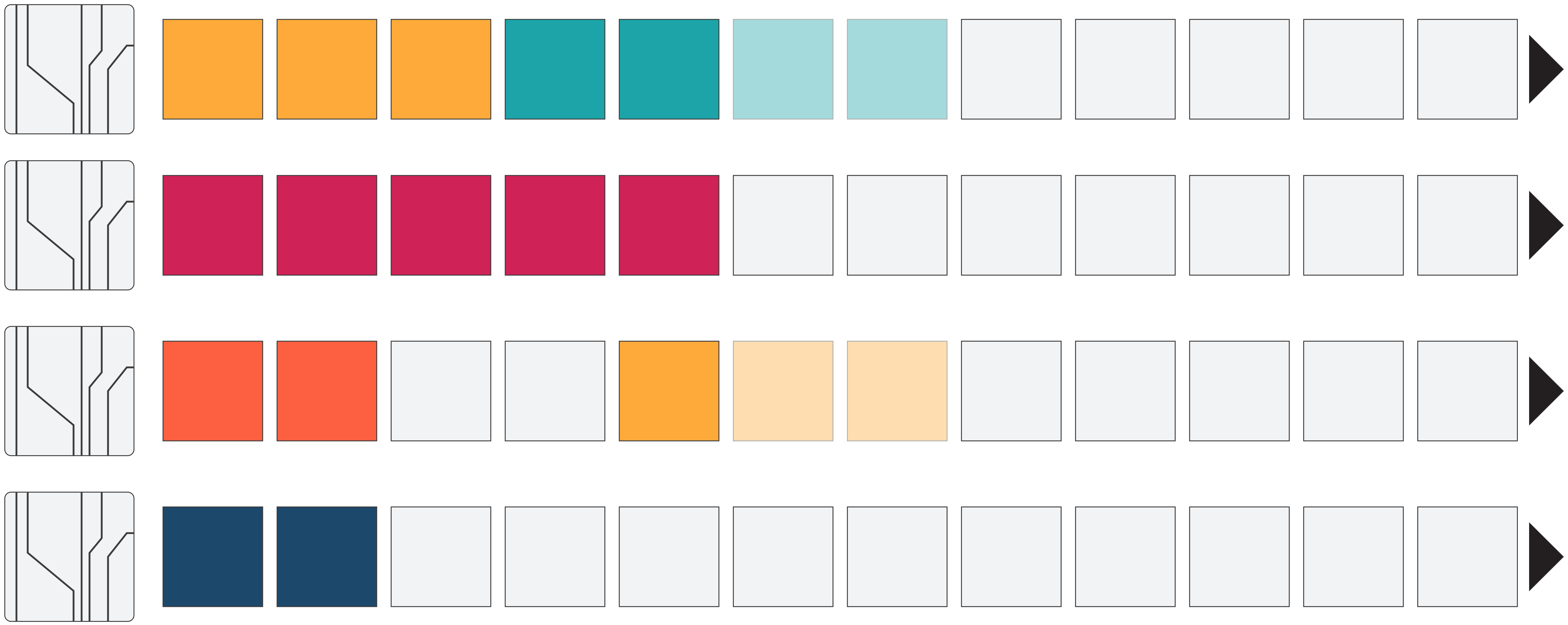
Sequential schedule



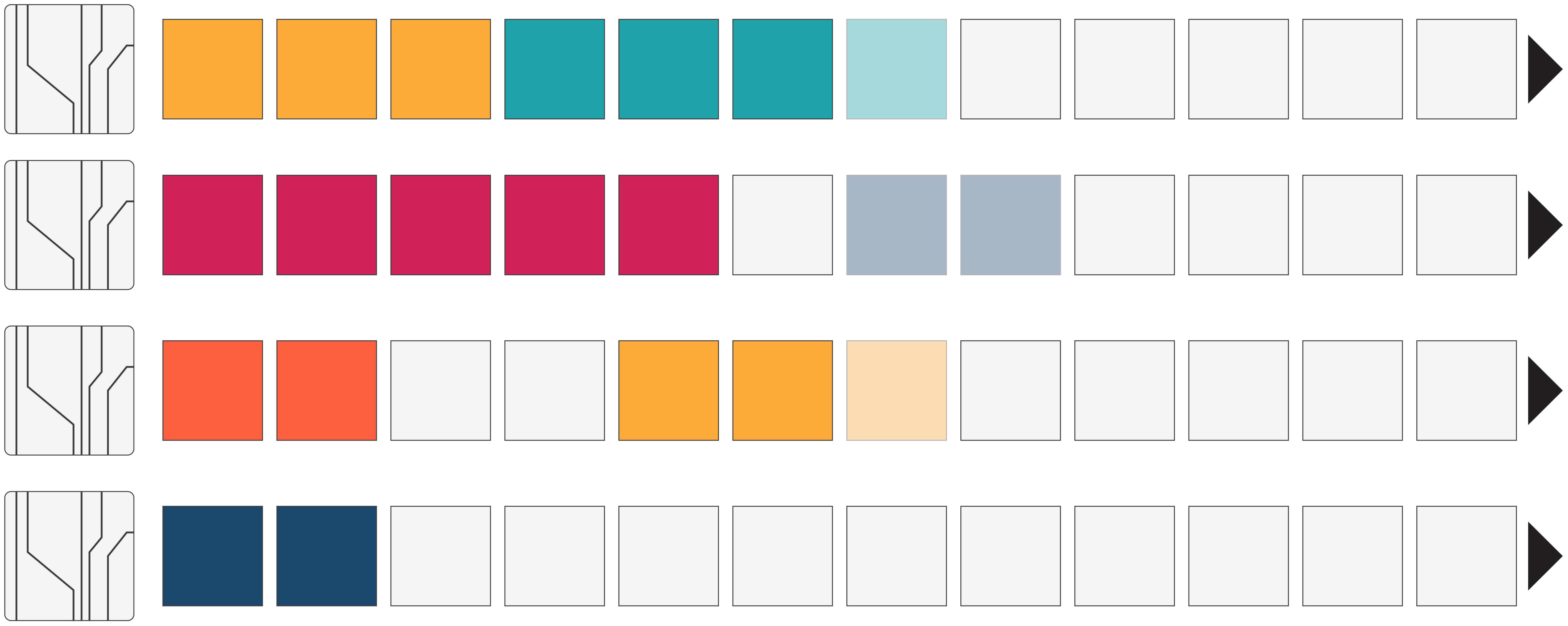
Sequential schedule



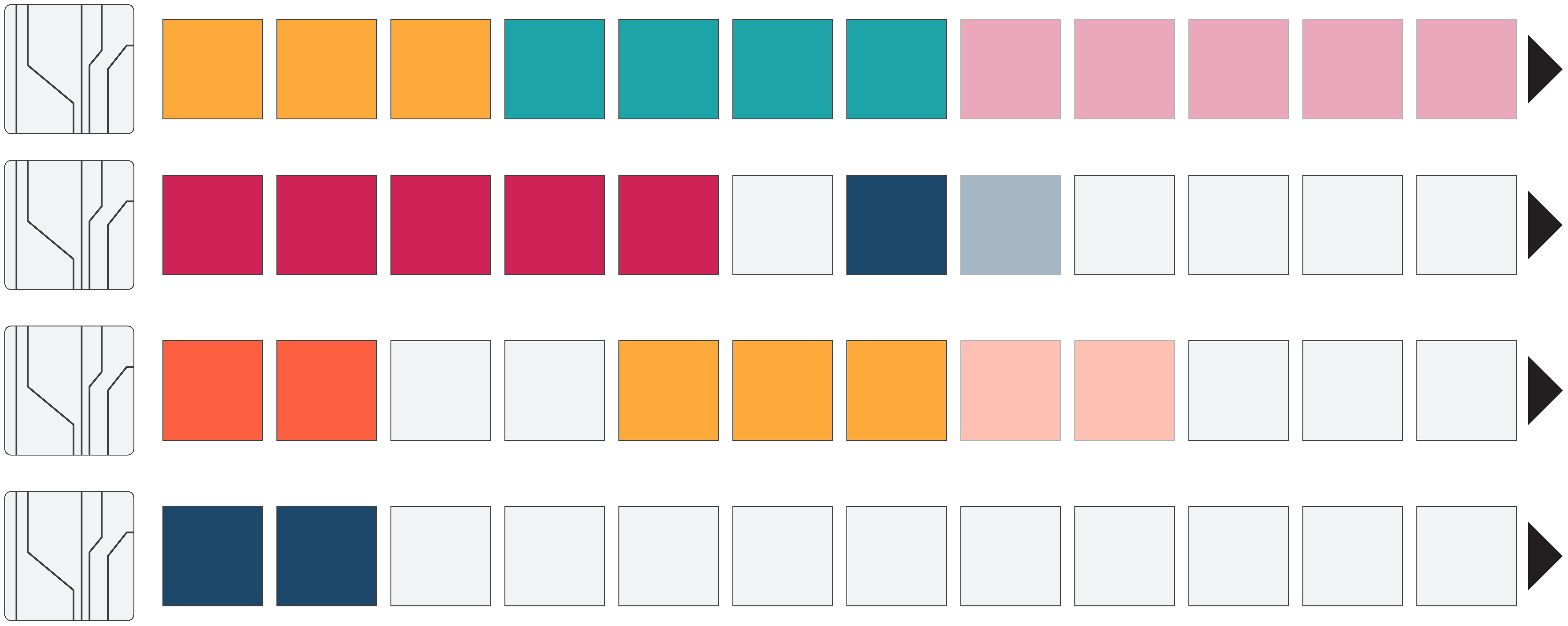
Sequential schedule



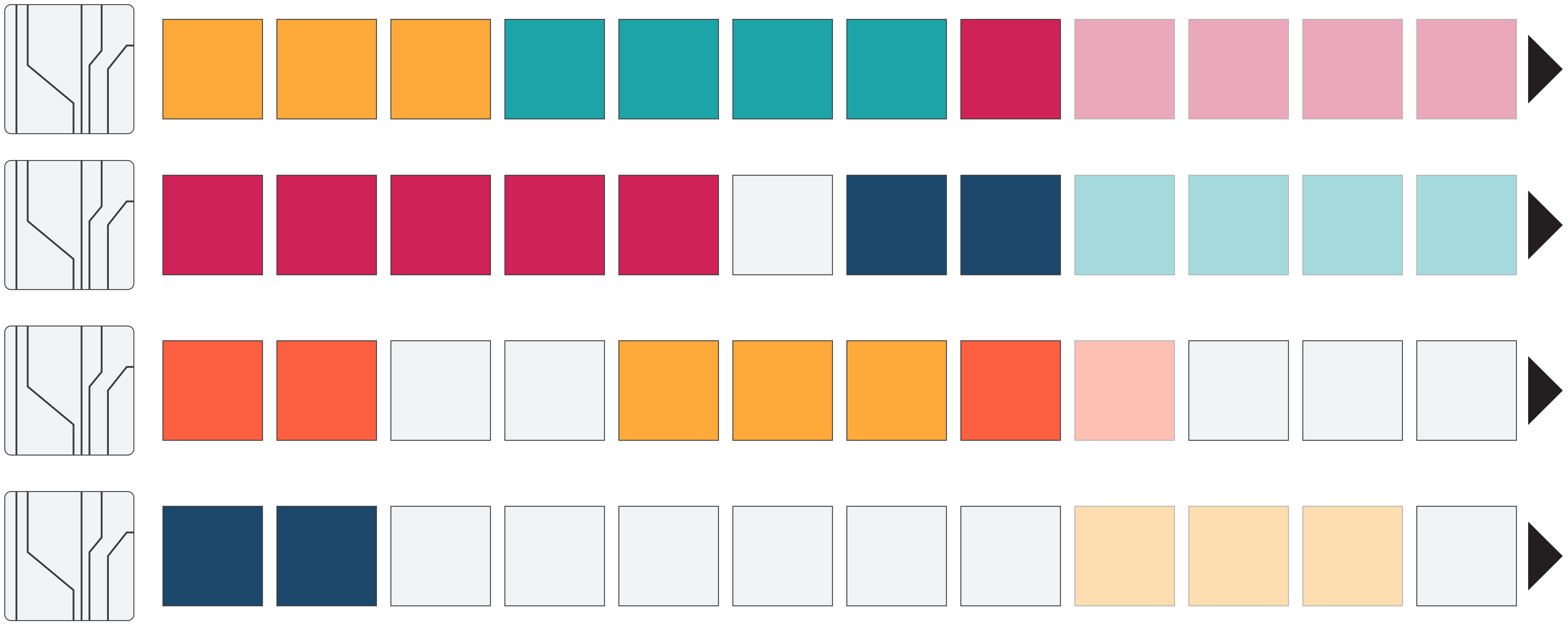
Sequential schedule



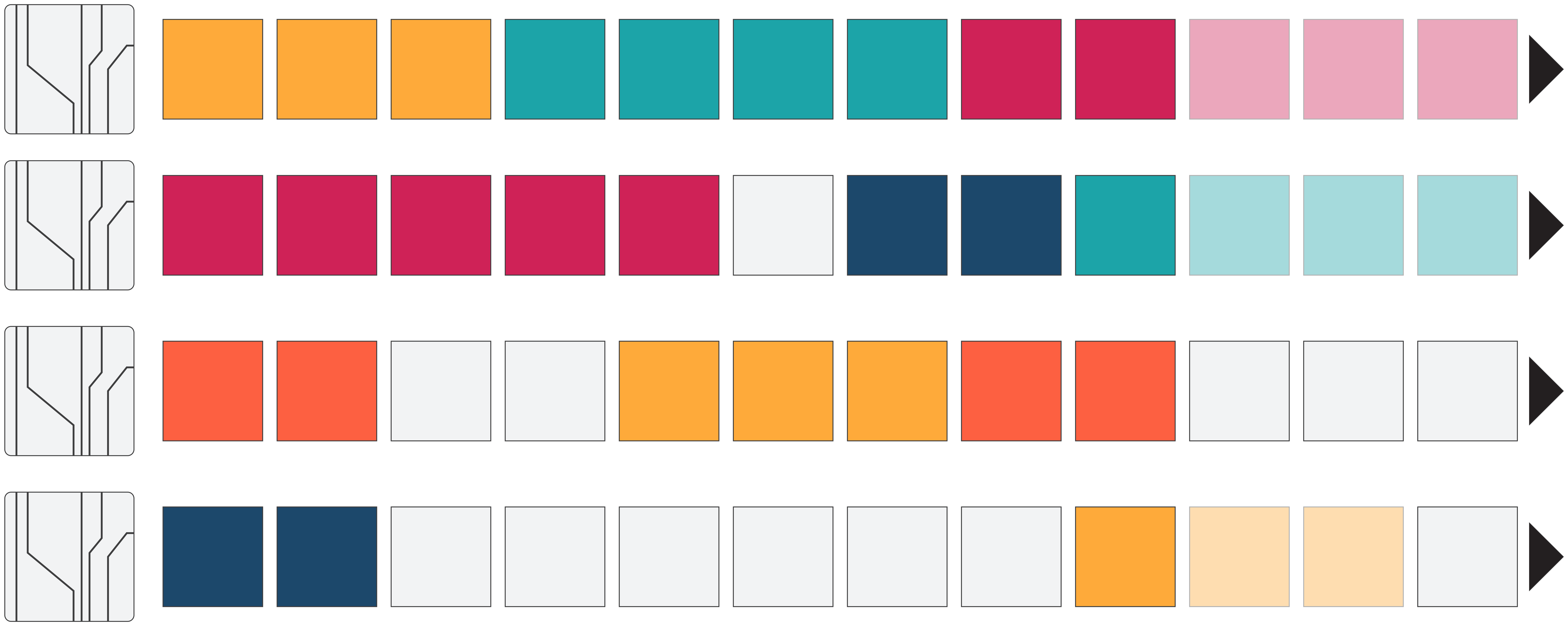
Sequential schedule



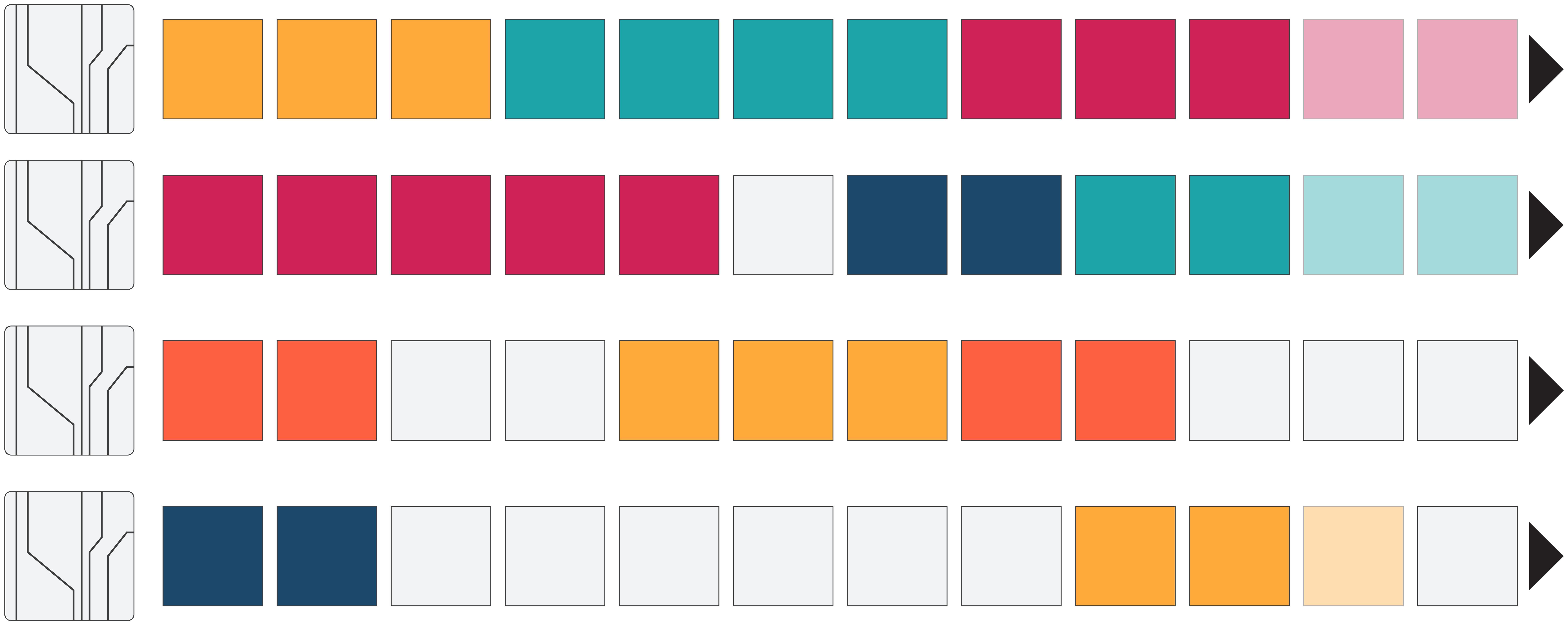
Sequential schedule



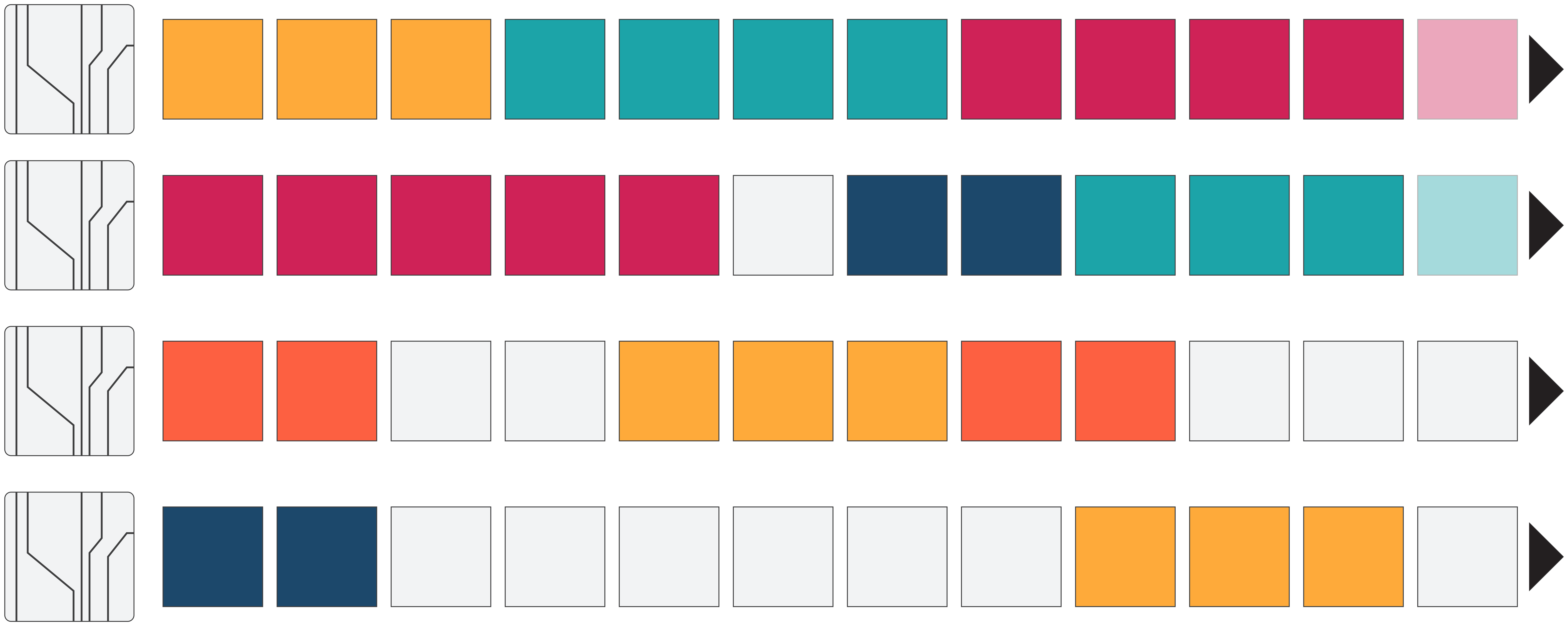
Sequential schedule



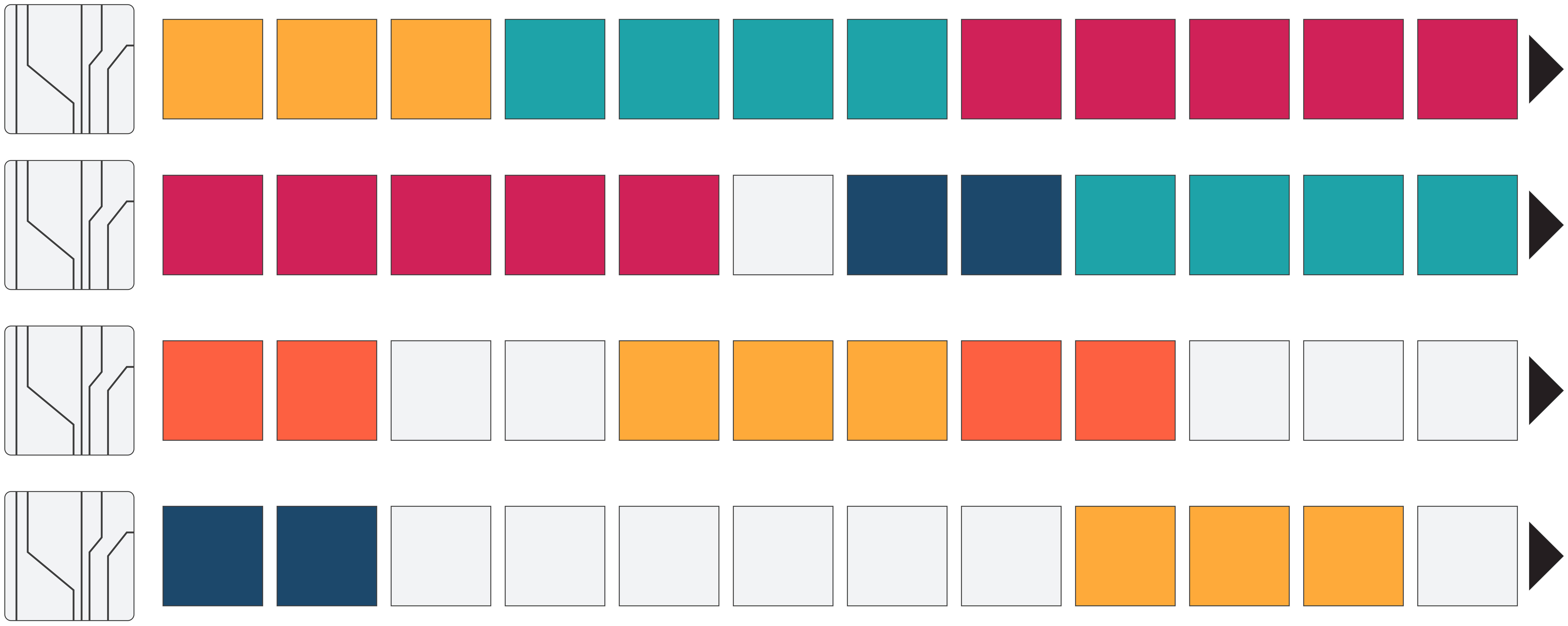
Sequential schedule



Sequential schedule

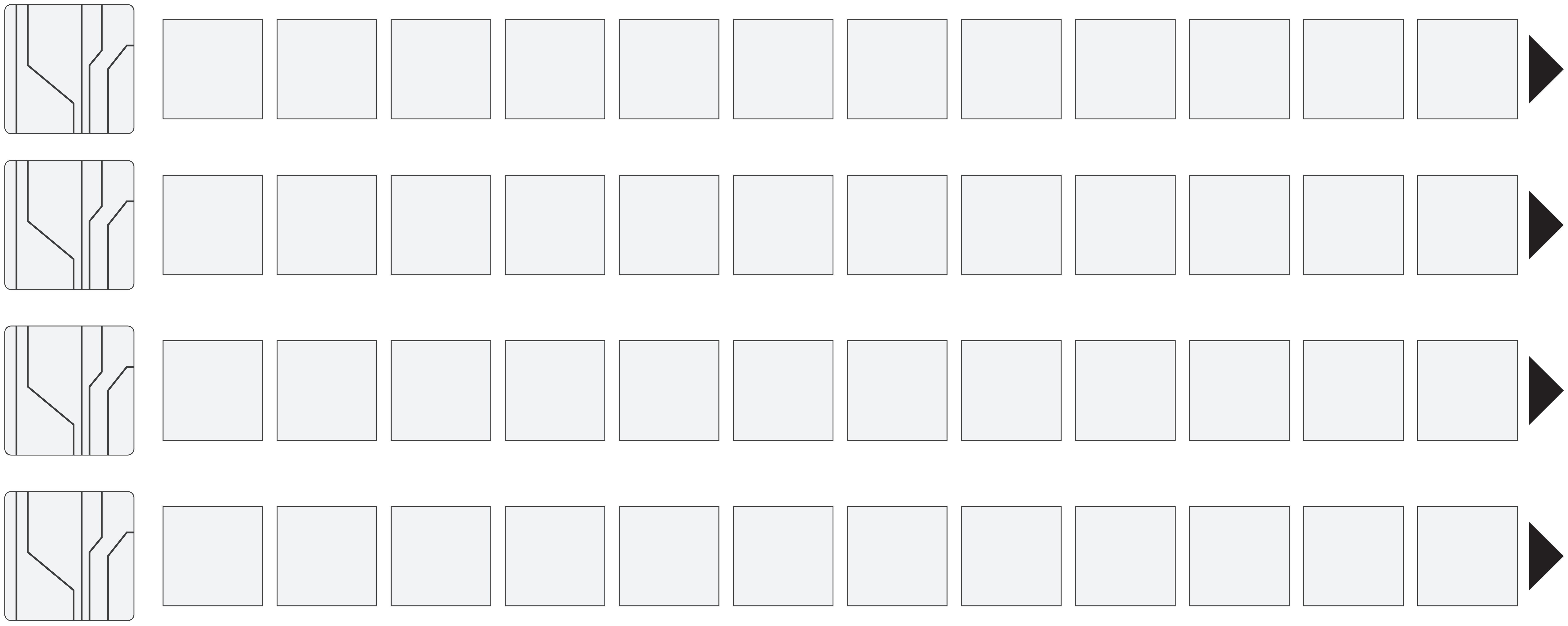


Sequential schedule

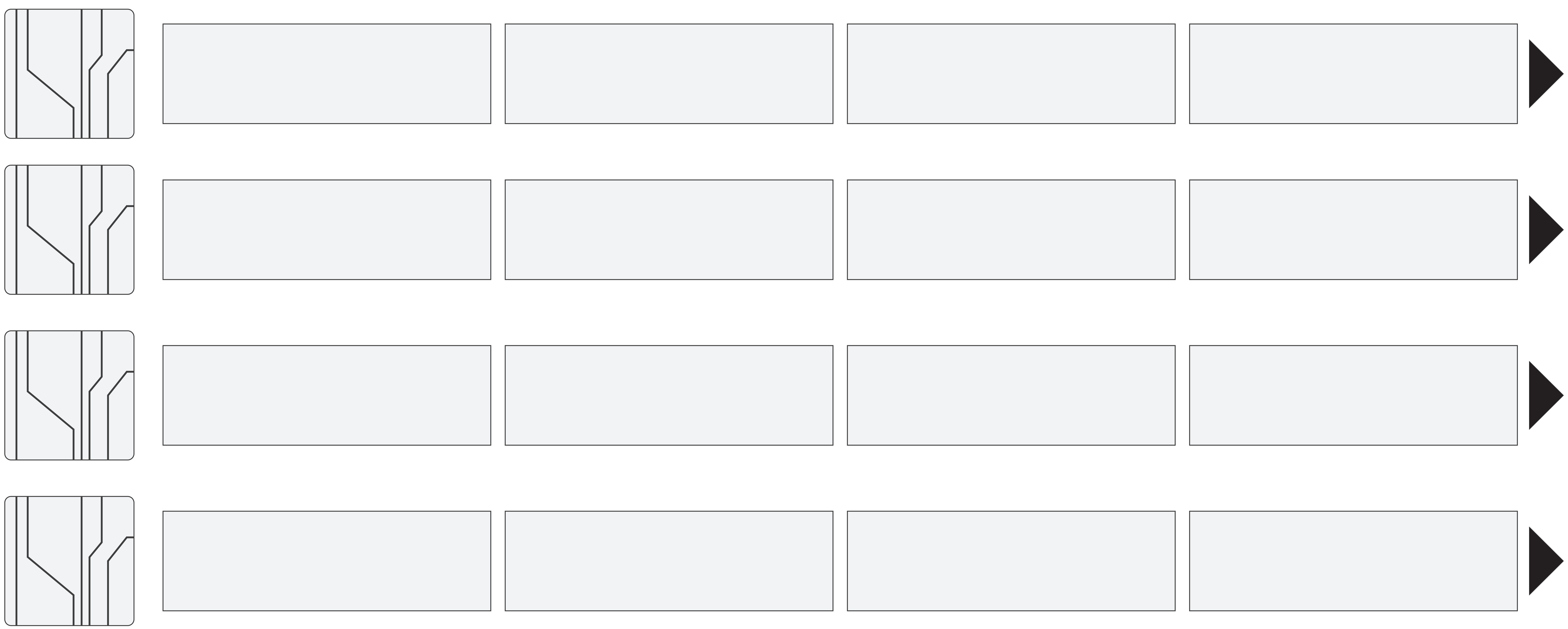


Goal = Save power

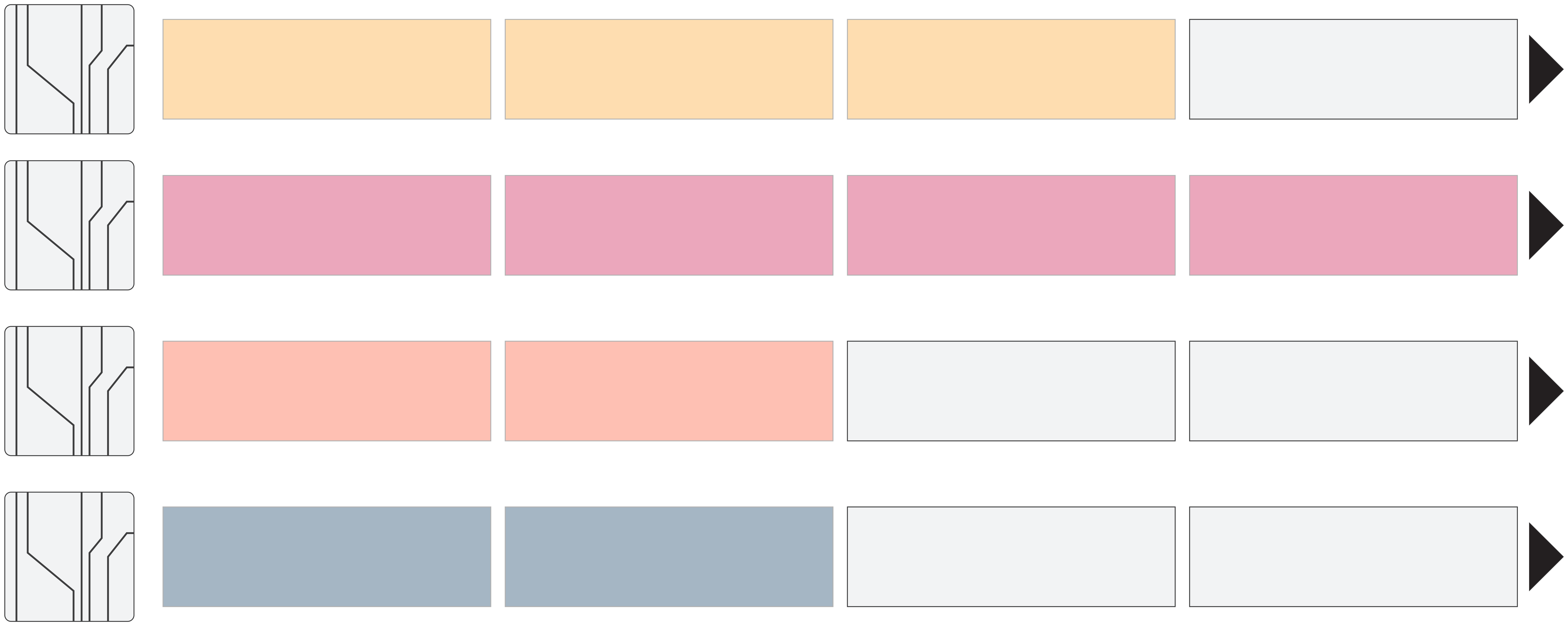
Power-aware schedule



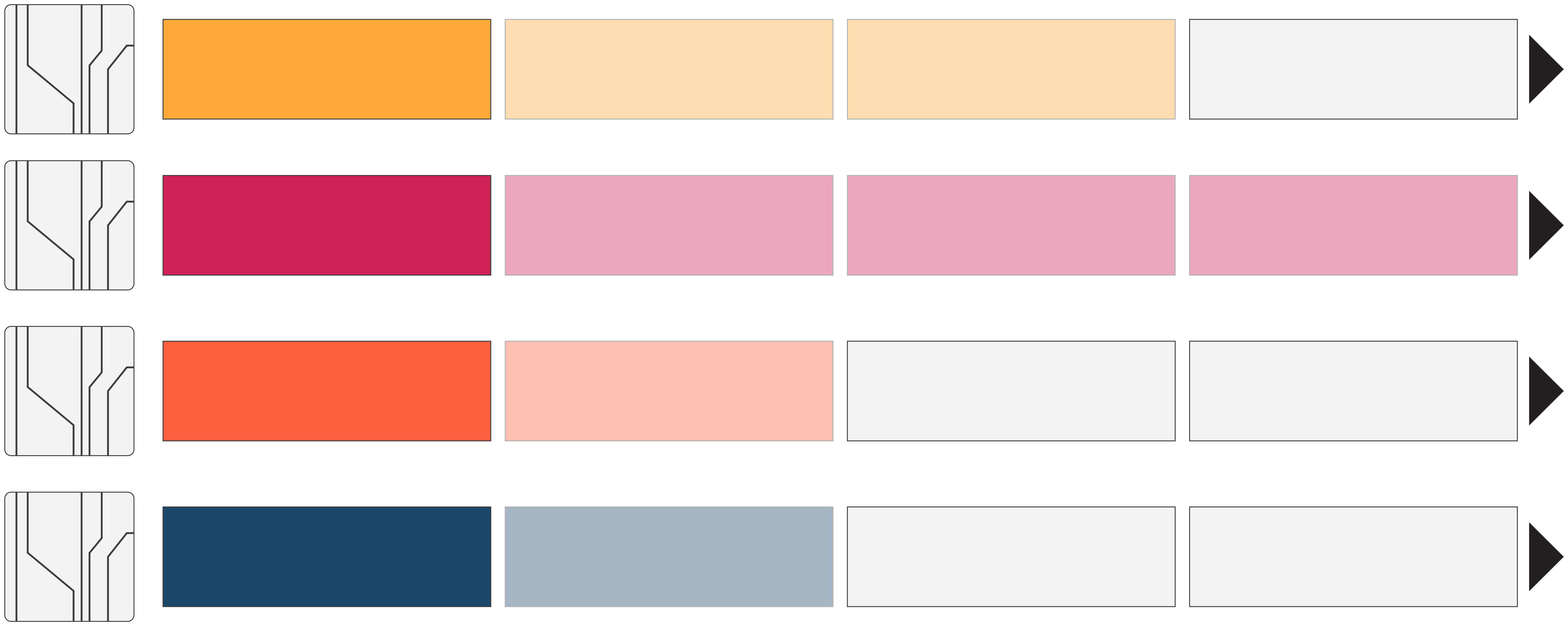
Power-aware schedule



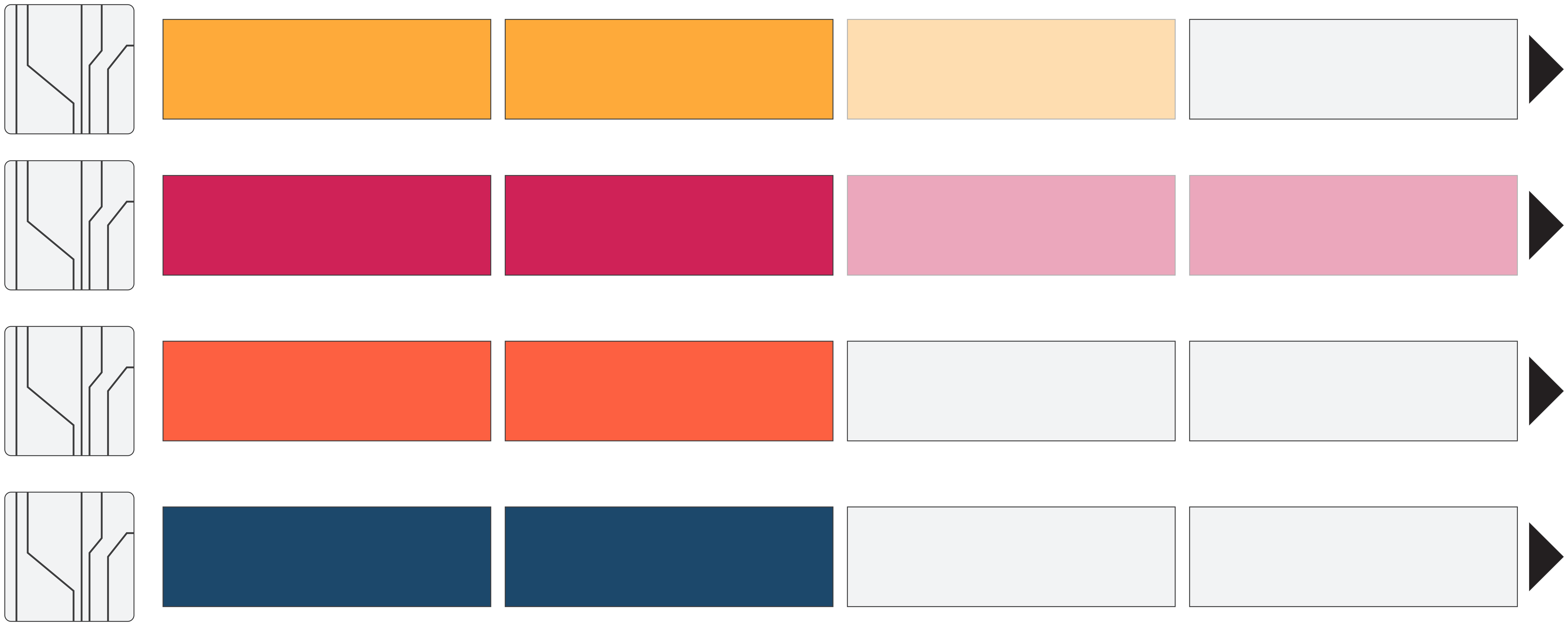
Power-aware schedule



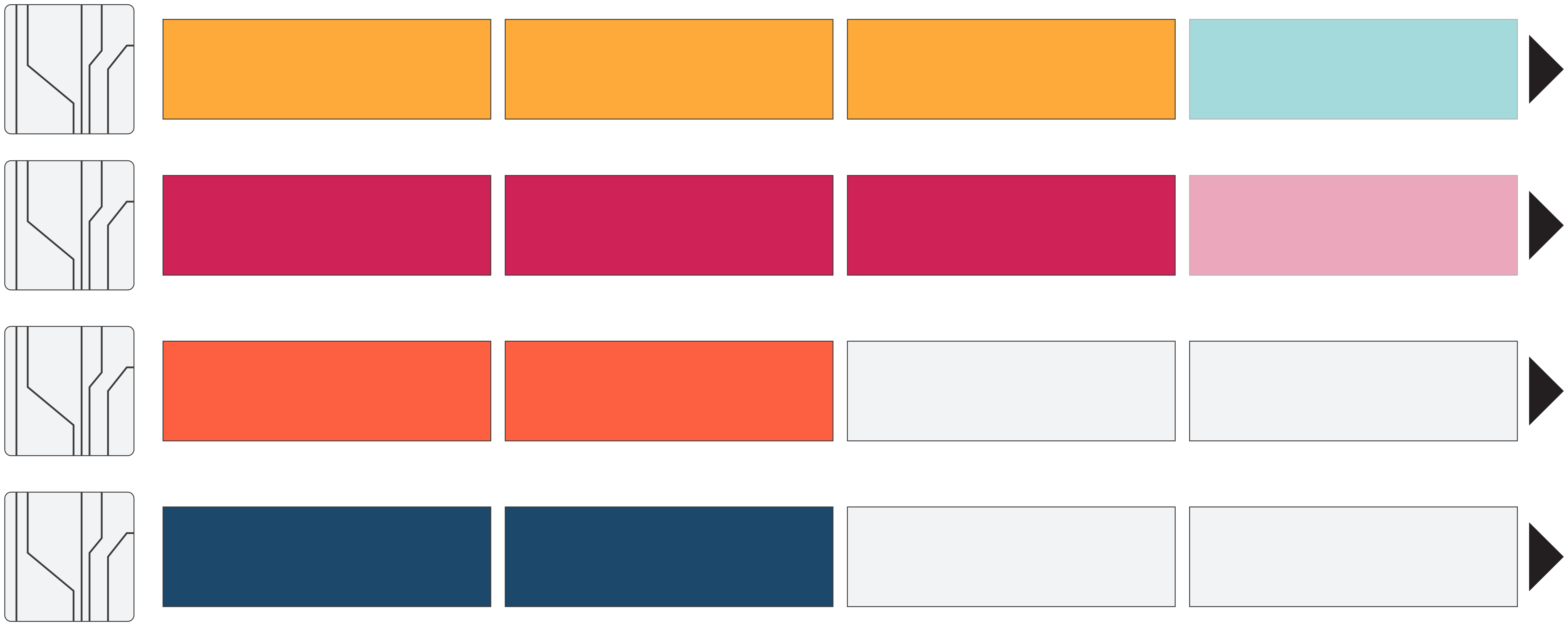
Power-aware schedule



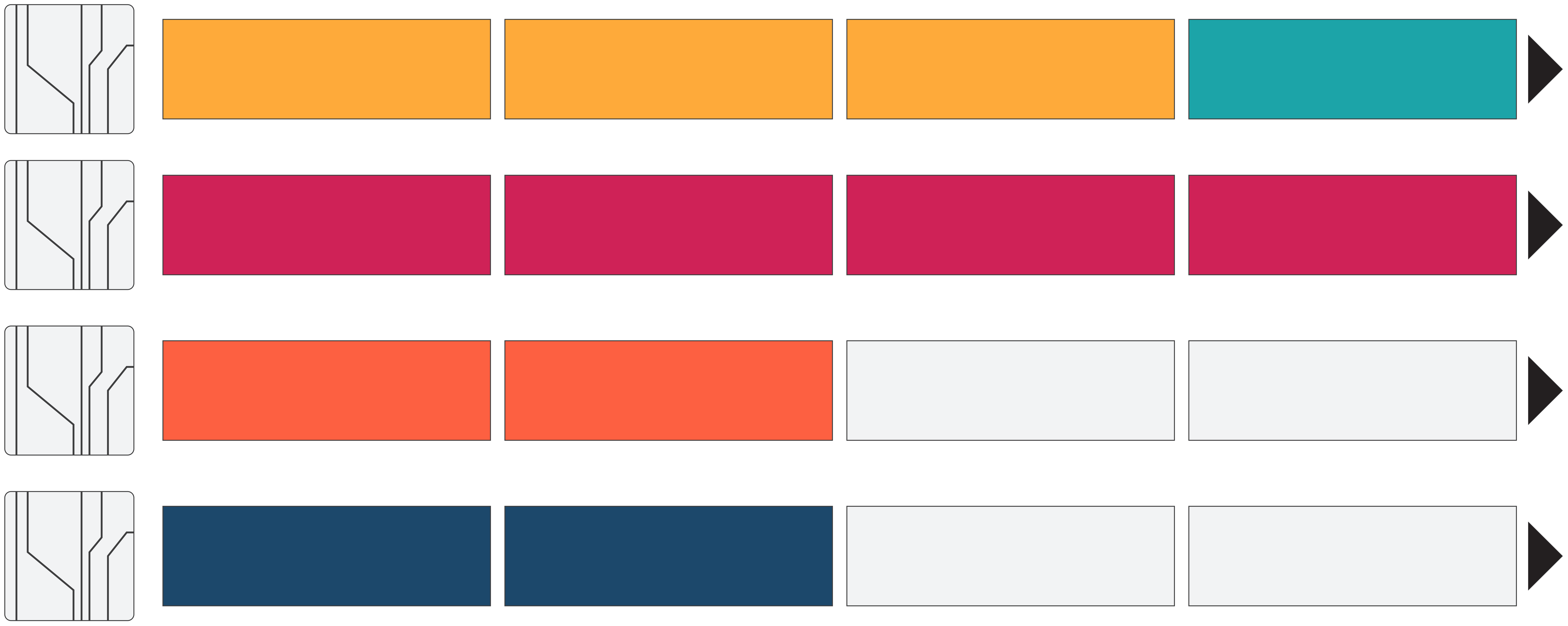
Power-aware schedule



Power-aware schedule



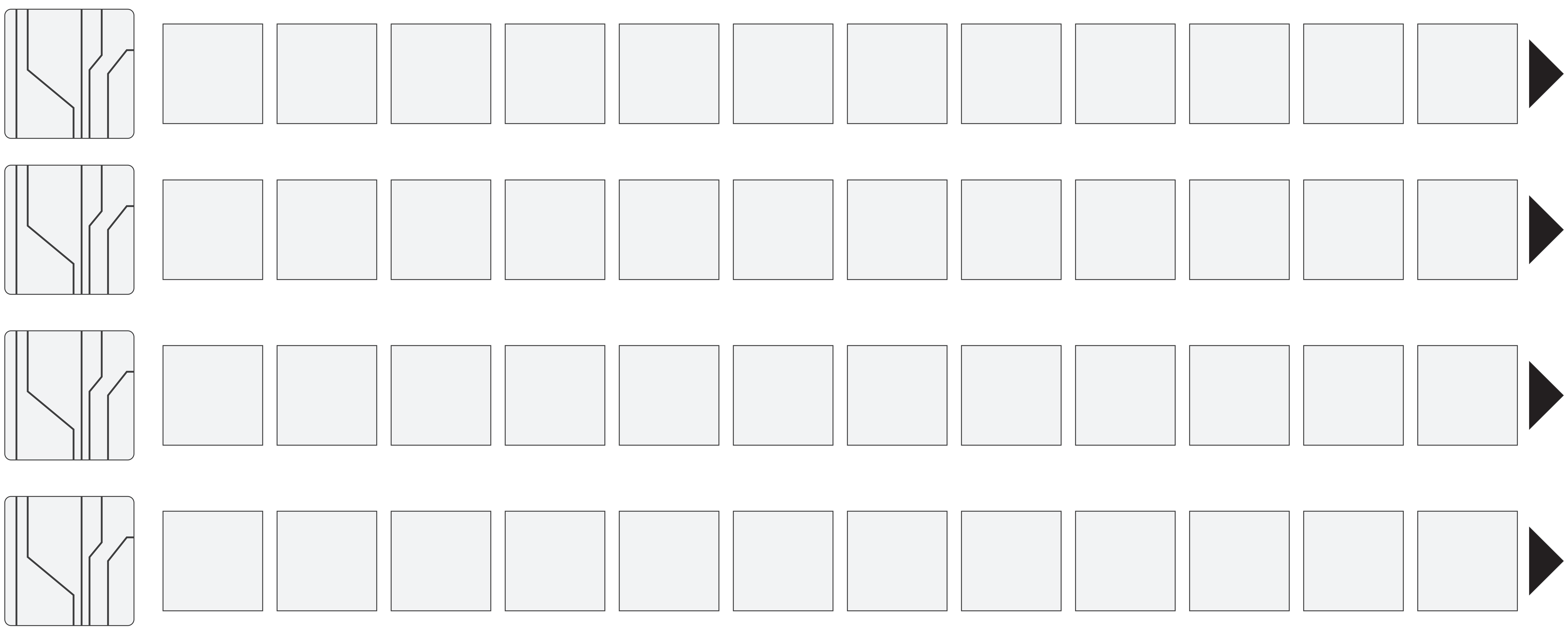
Power-aware schedule



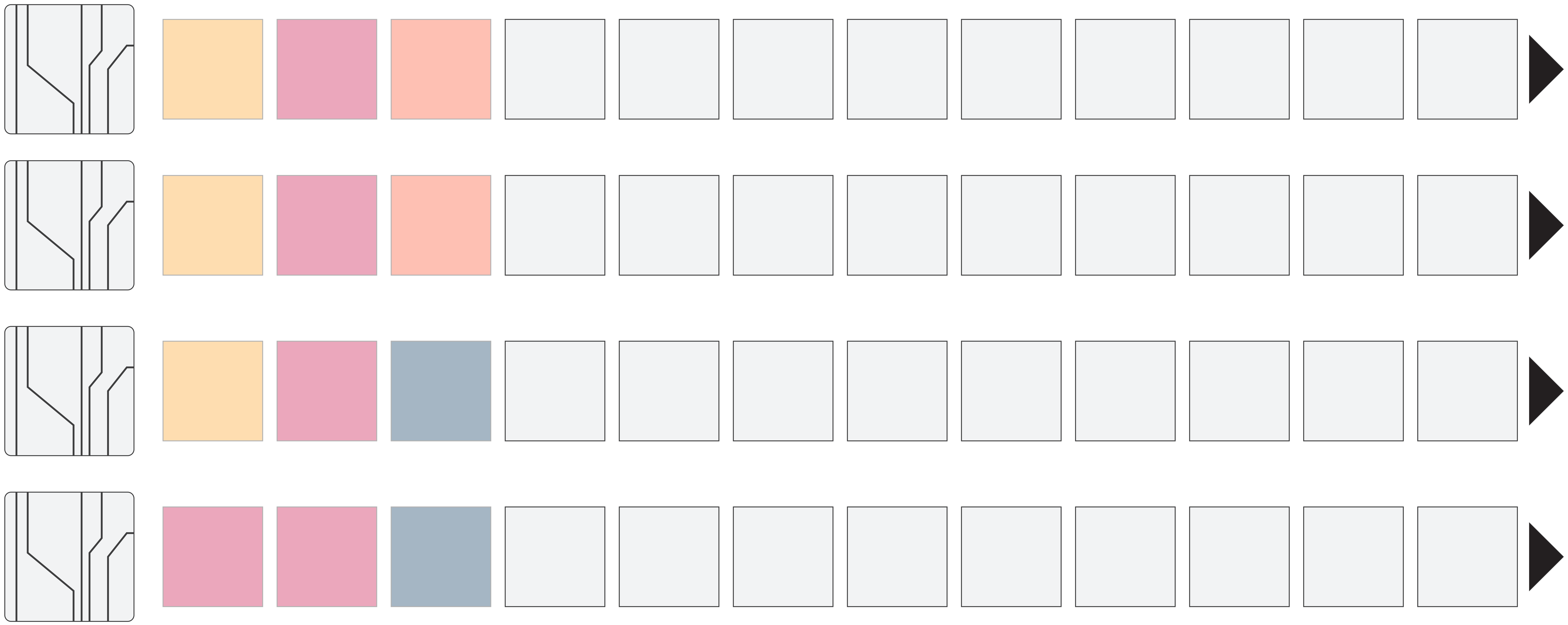
"The minimum **speed** is limited by the **sequential** job model"

- J. Anderson, S. Baruah

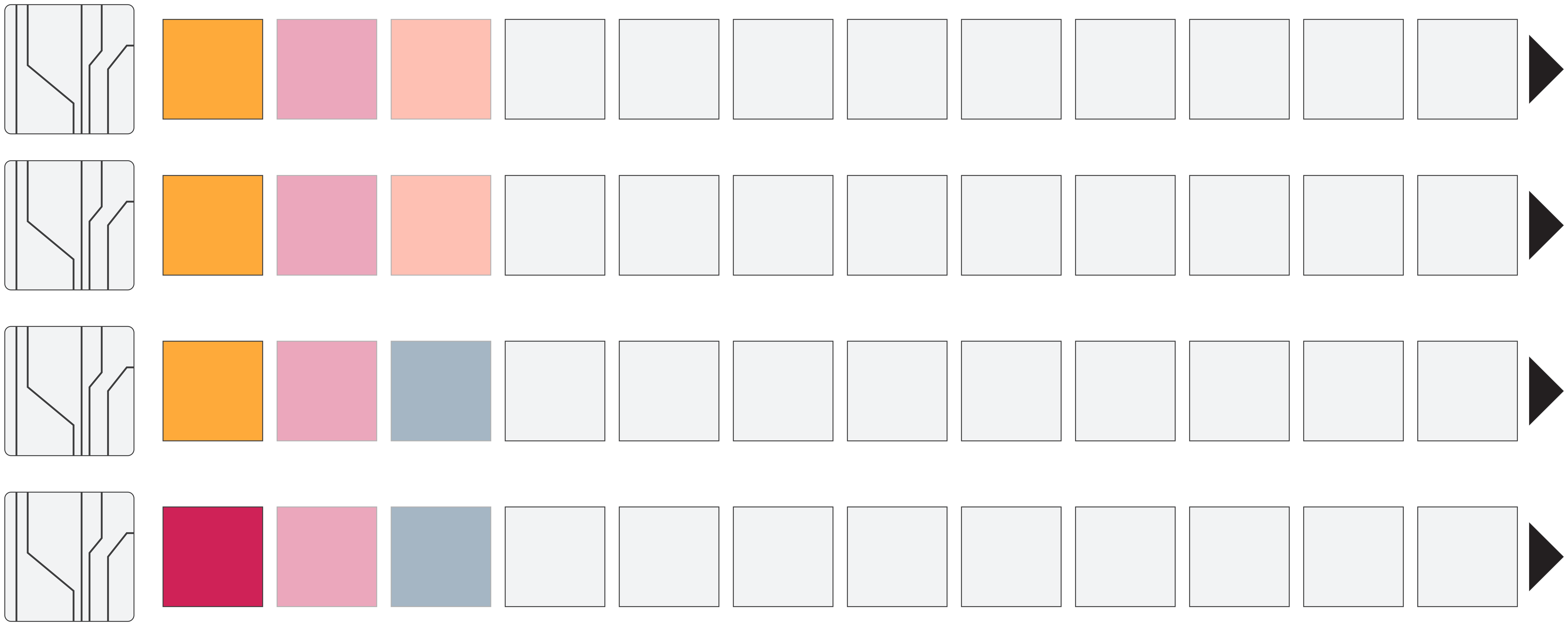
Parallel schedule



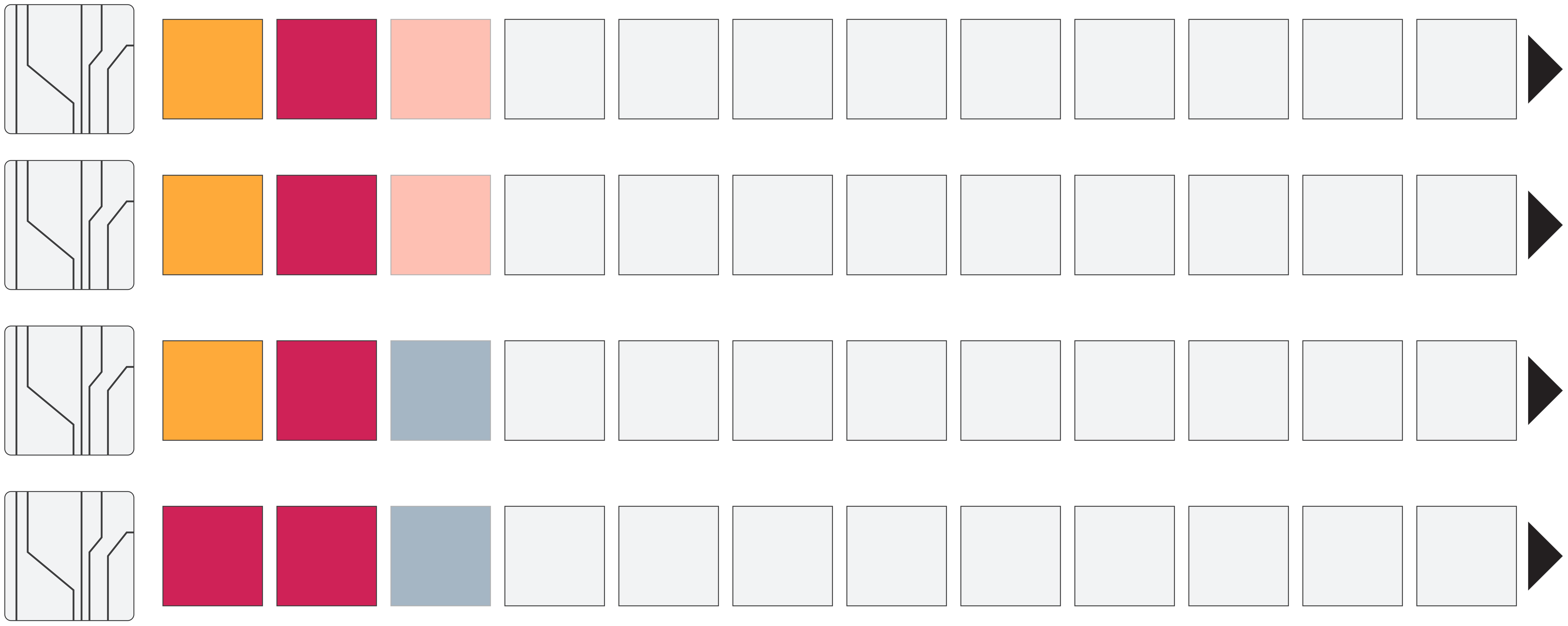
Parallel schedule



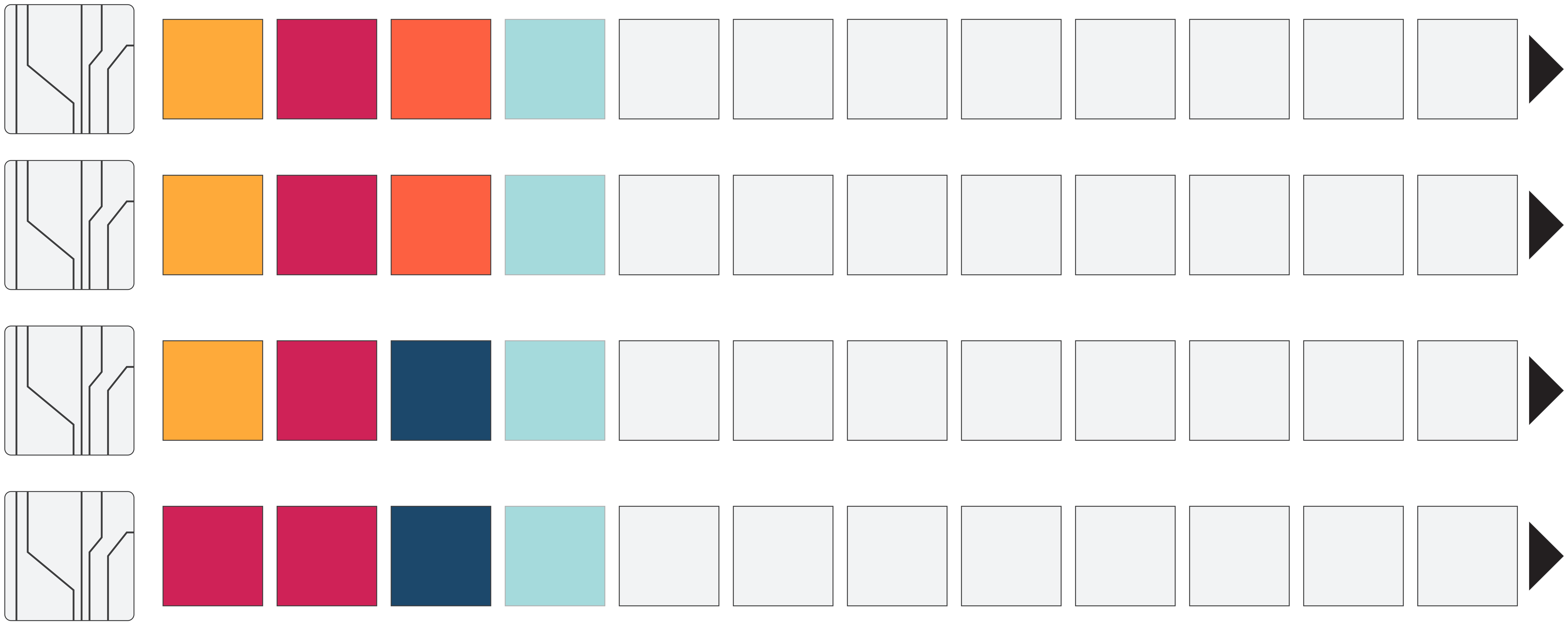
Parallel schedule



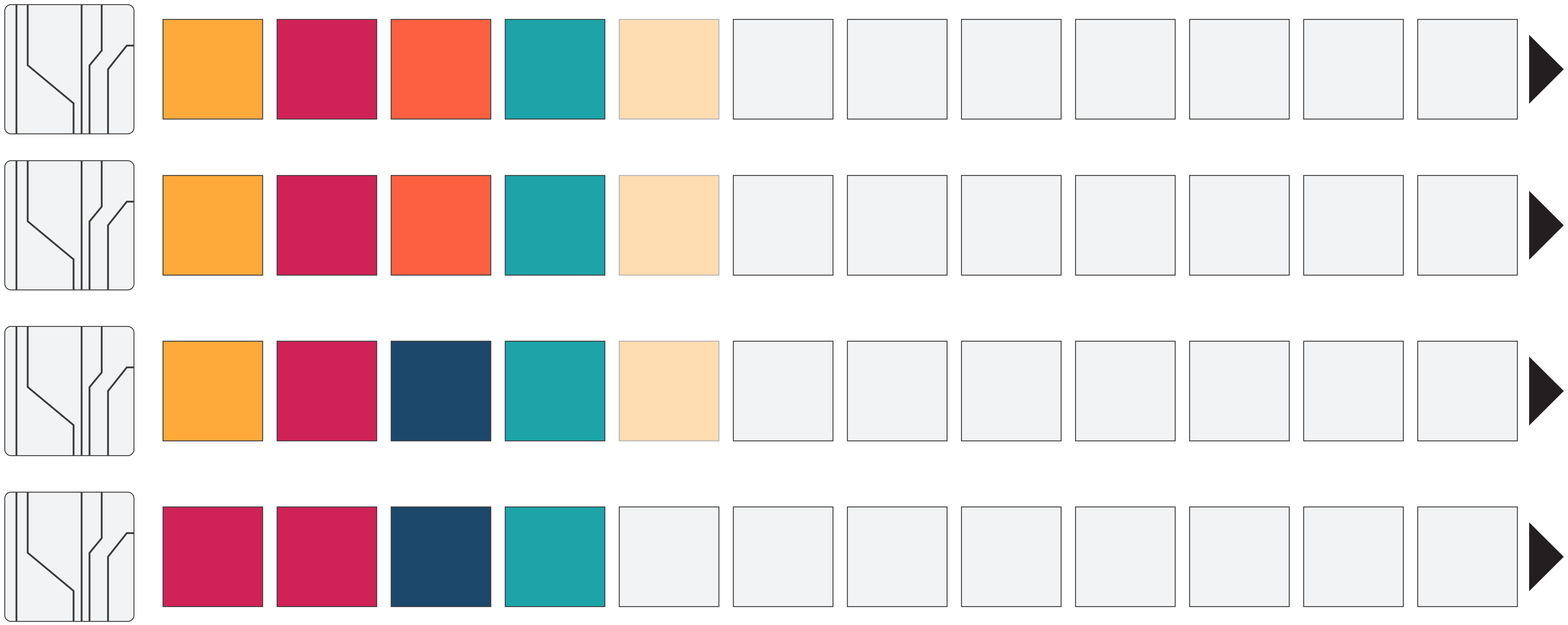
Parallel schedule



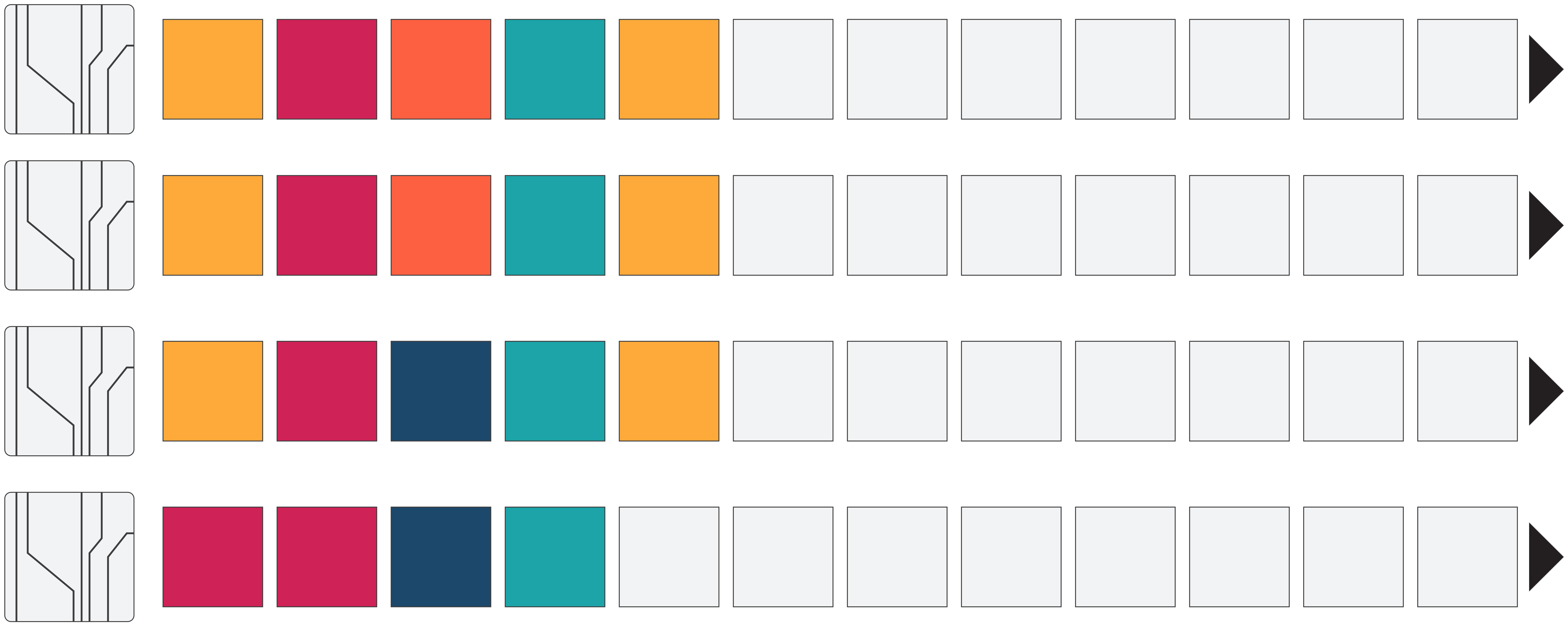
Parallel schedule



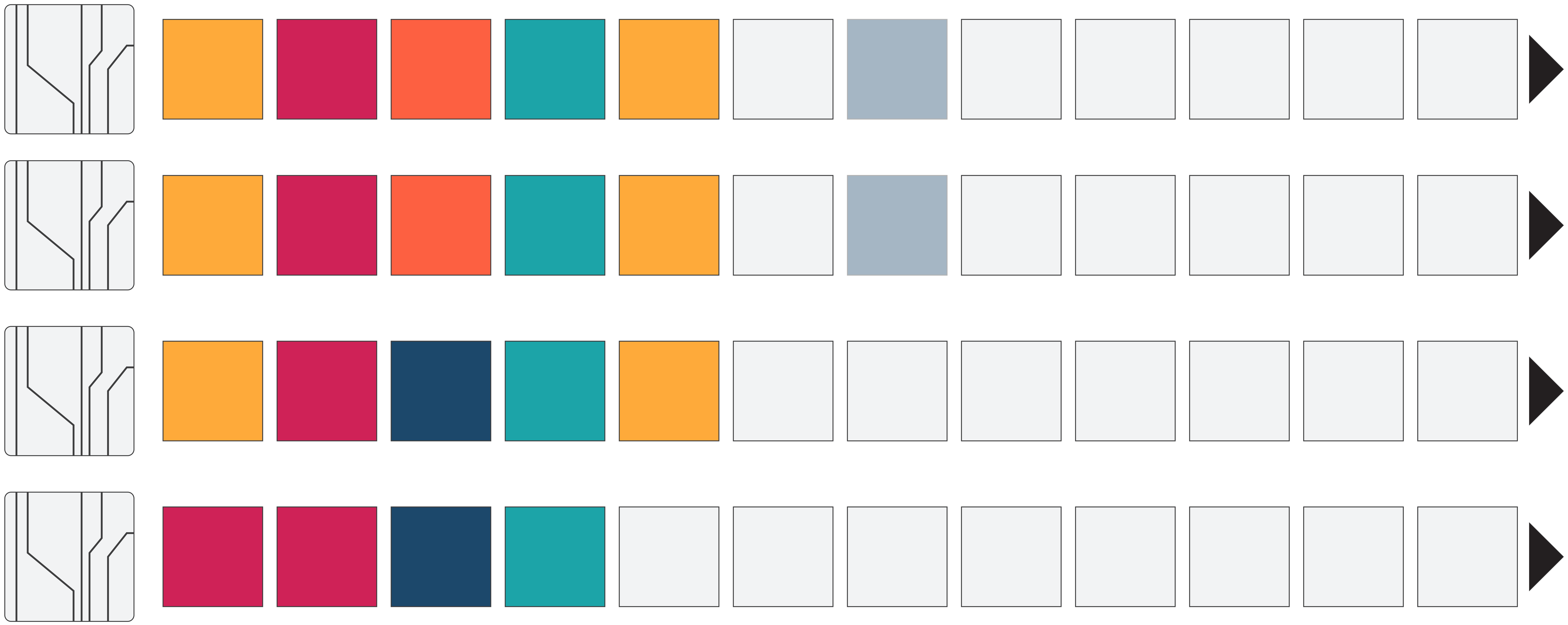
Parallel schedule



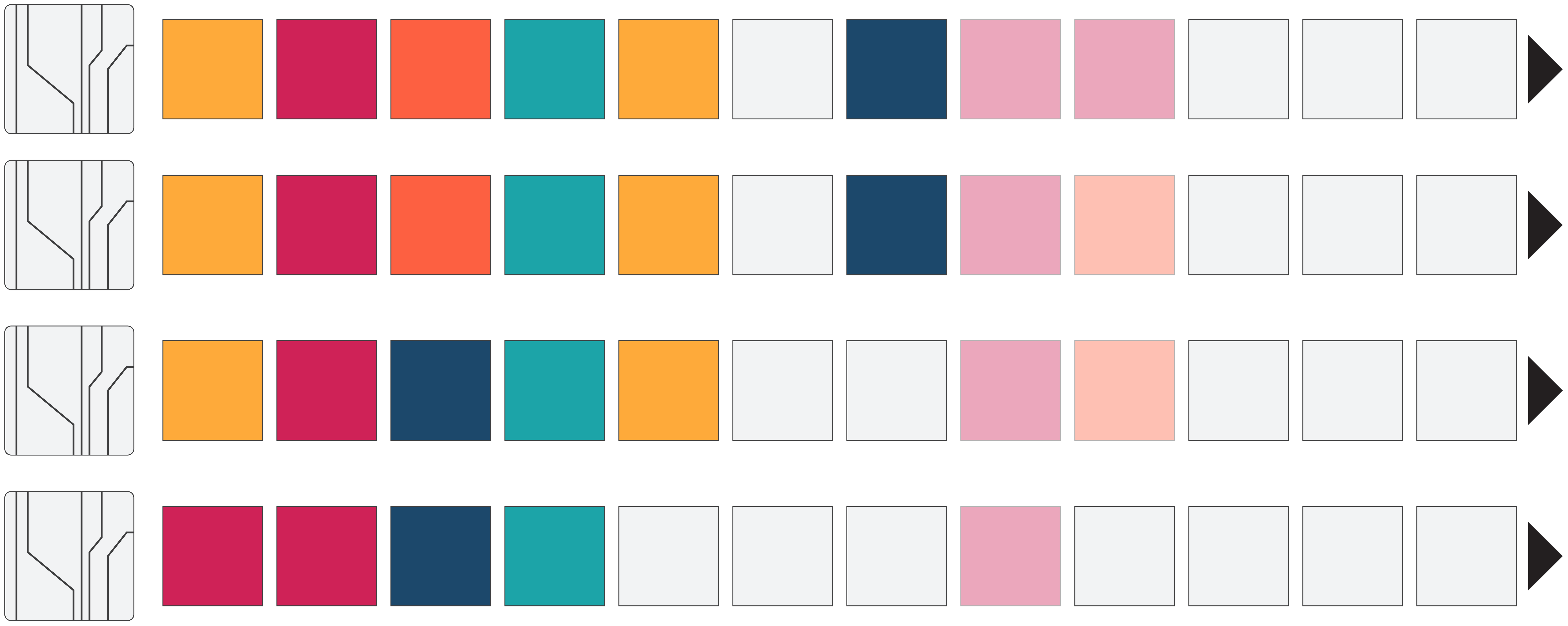
Parallel schedule



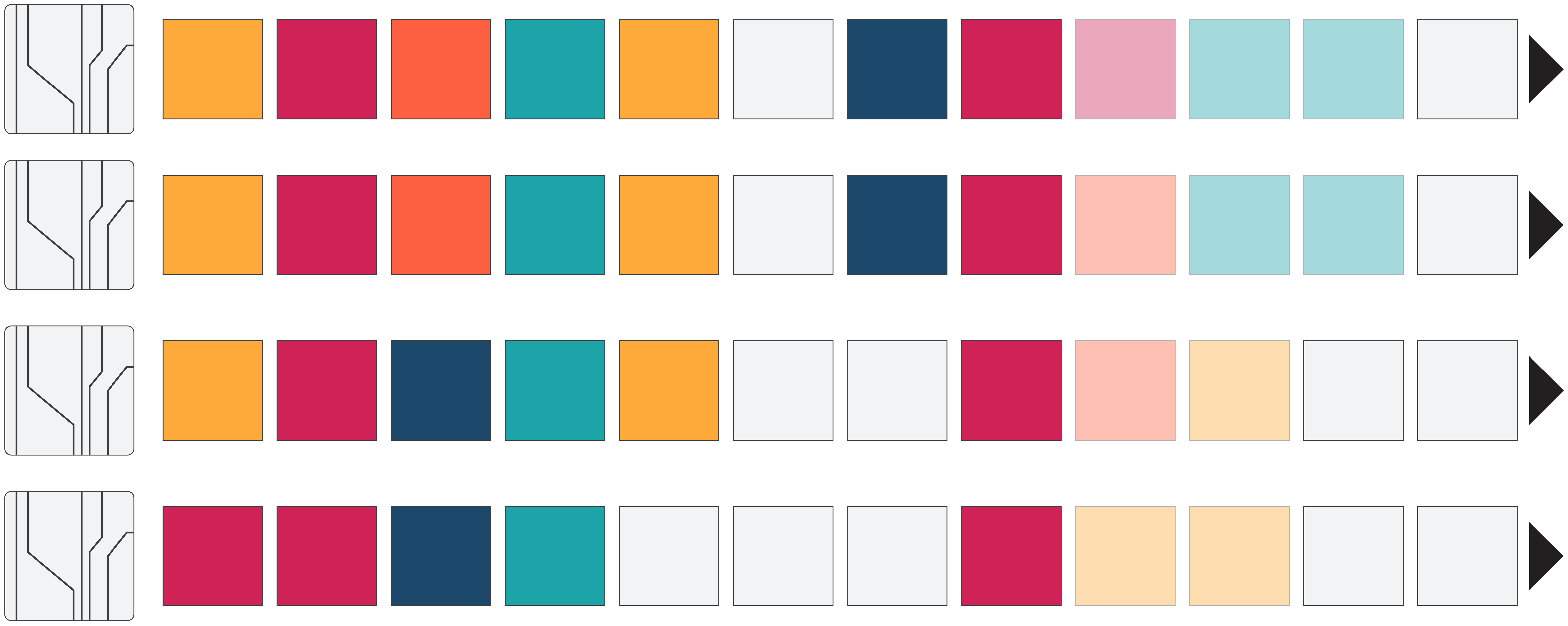
Parallel schedule



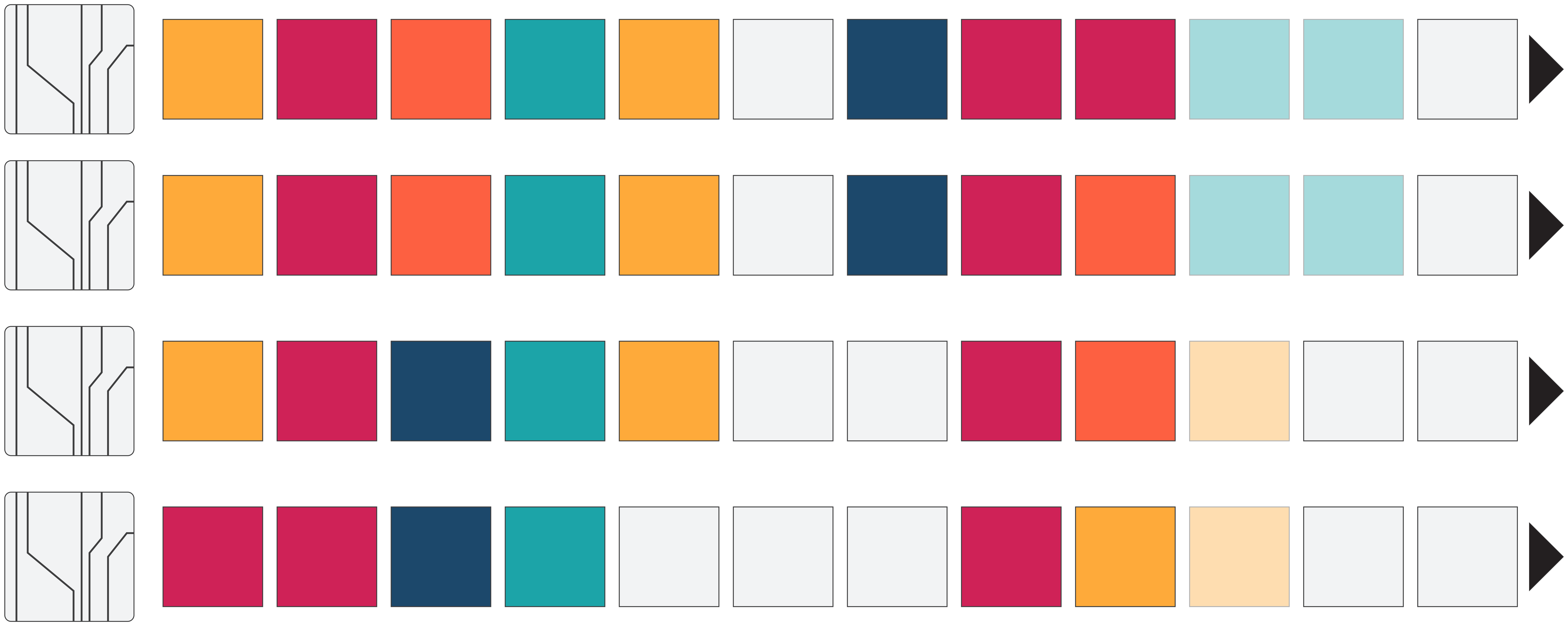
Parallel schedule



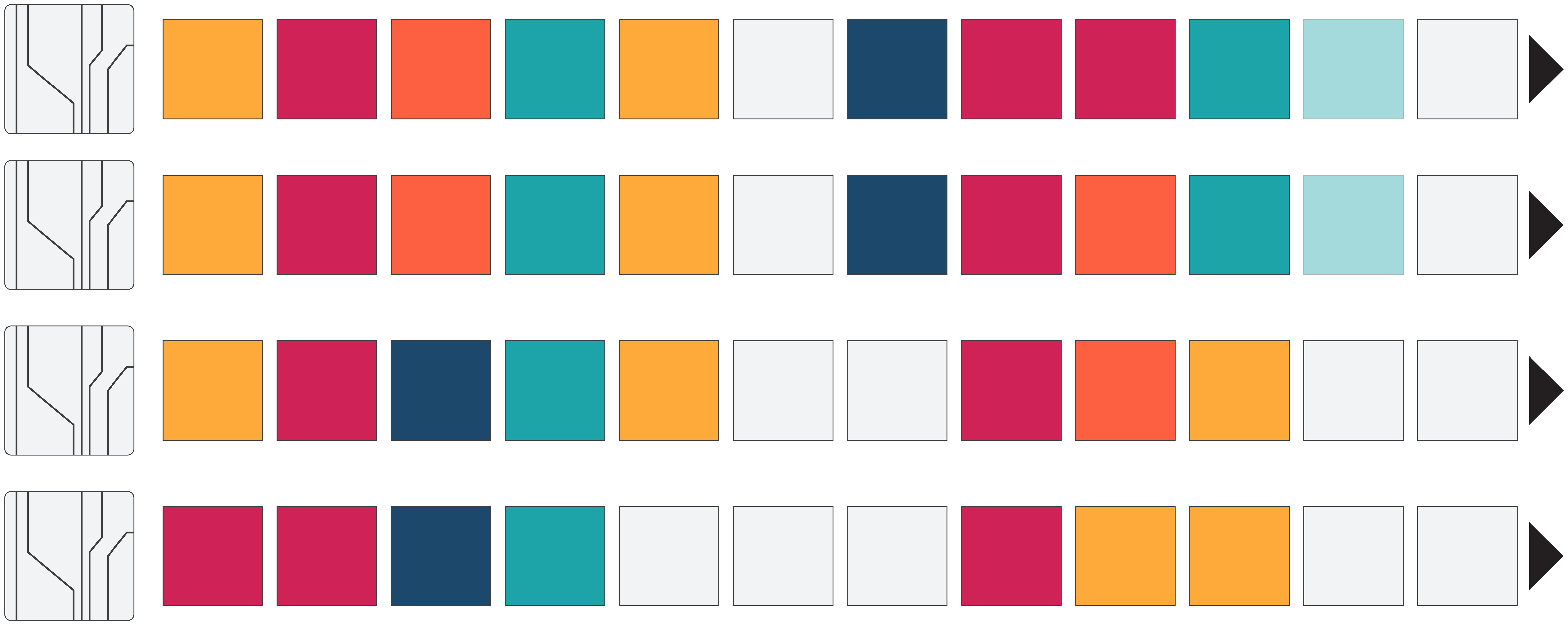
Parallel schedule



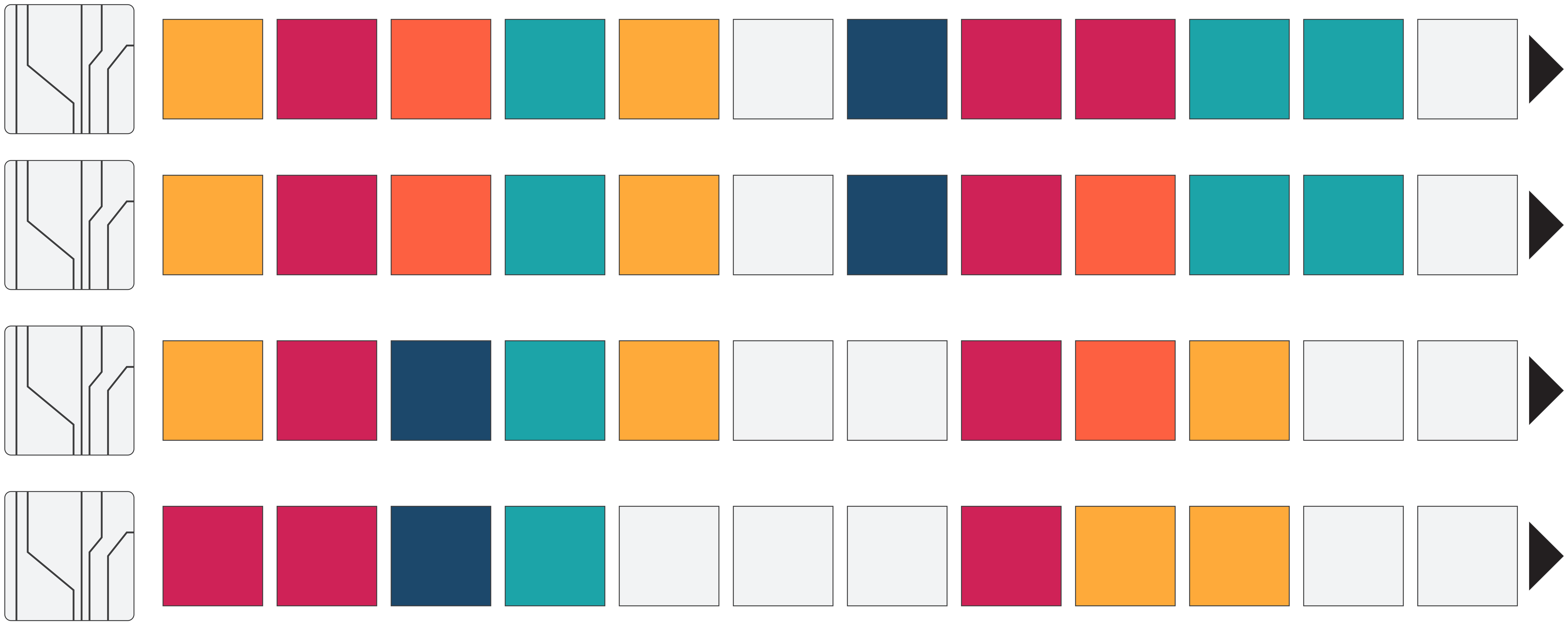
Parallel schedule



Parallel schedule

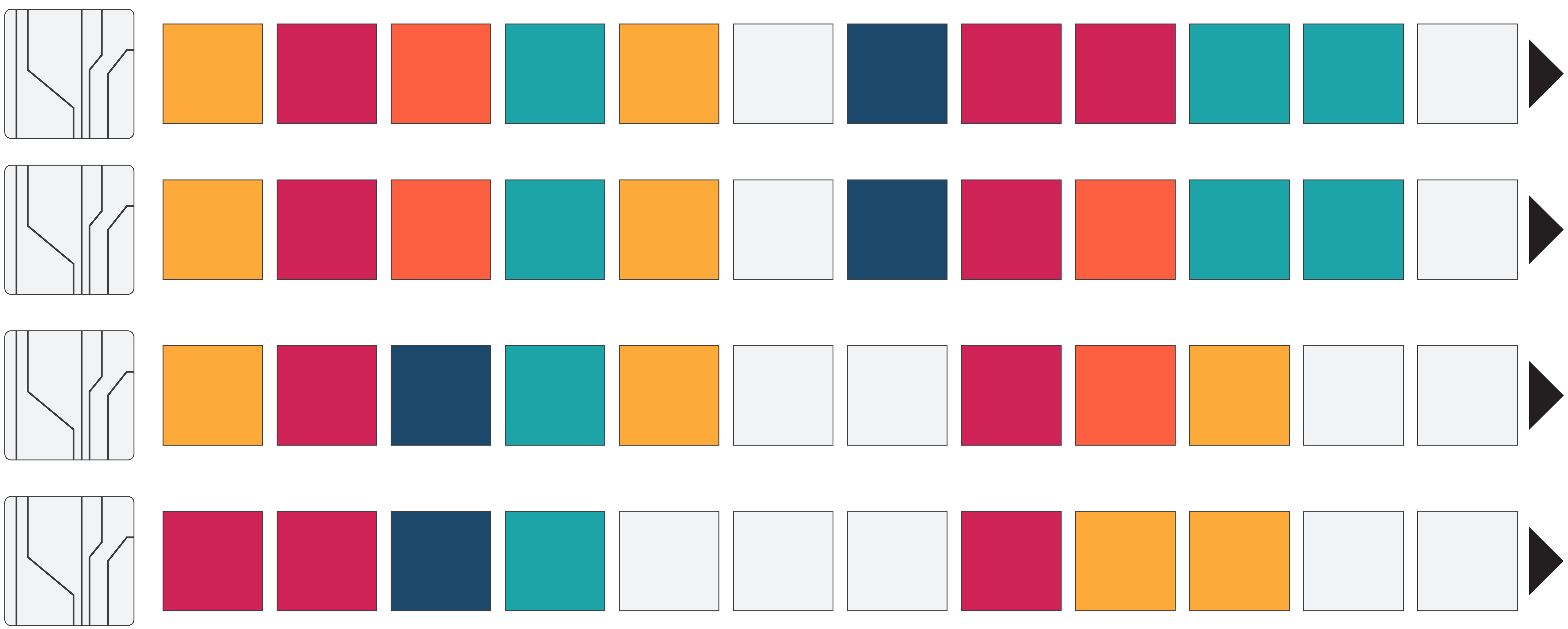


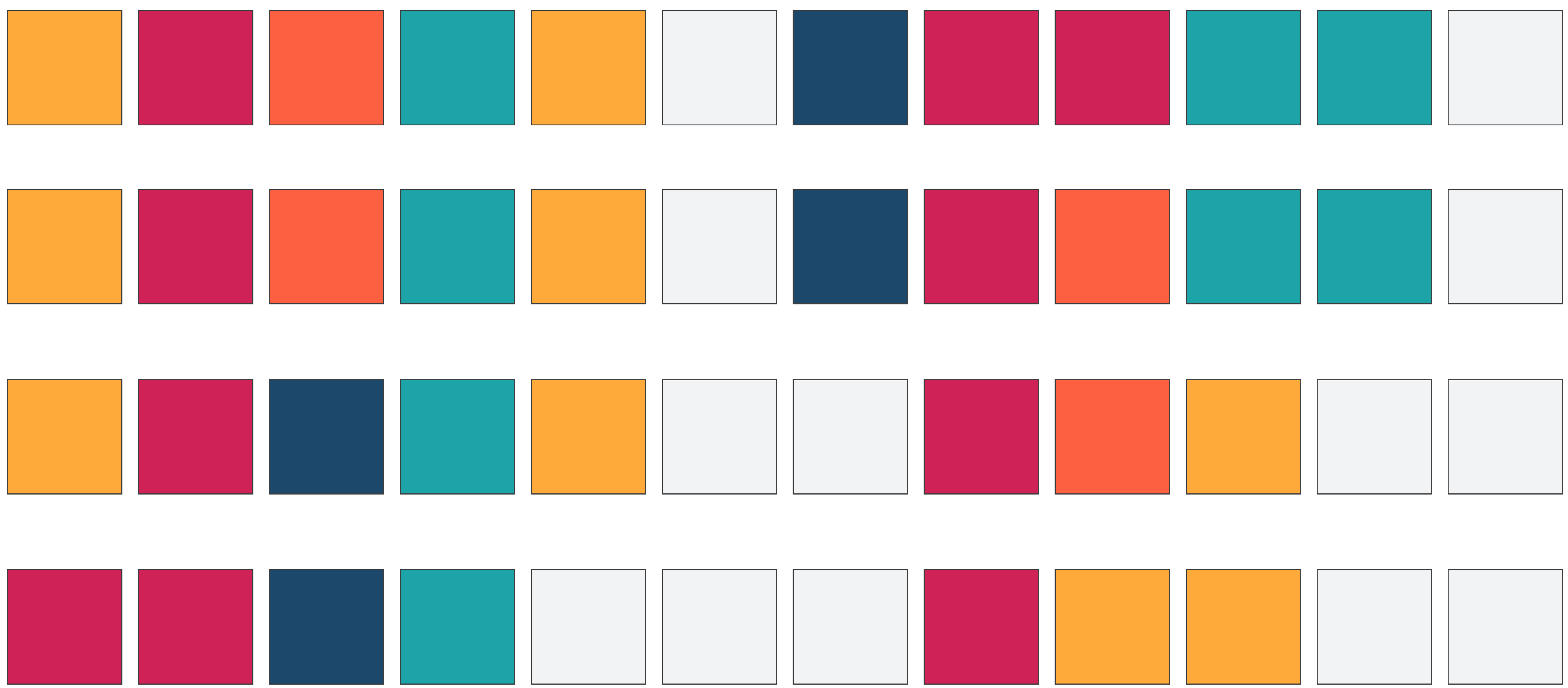
Parallel schedule

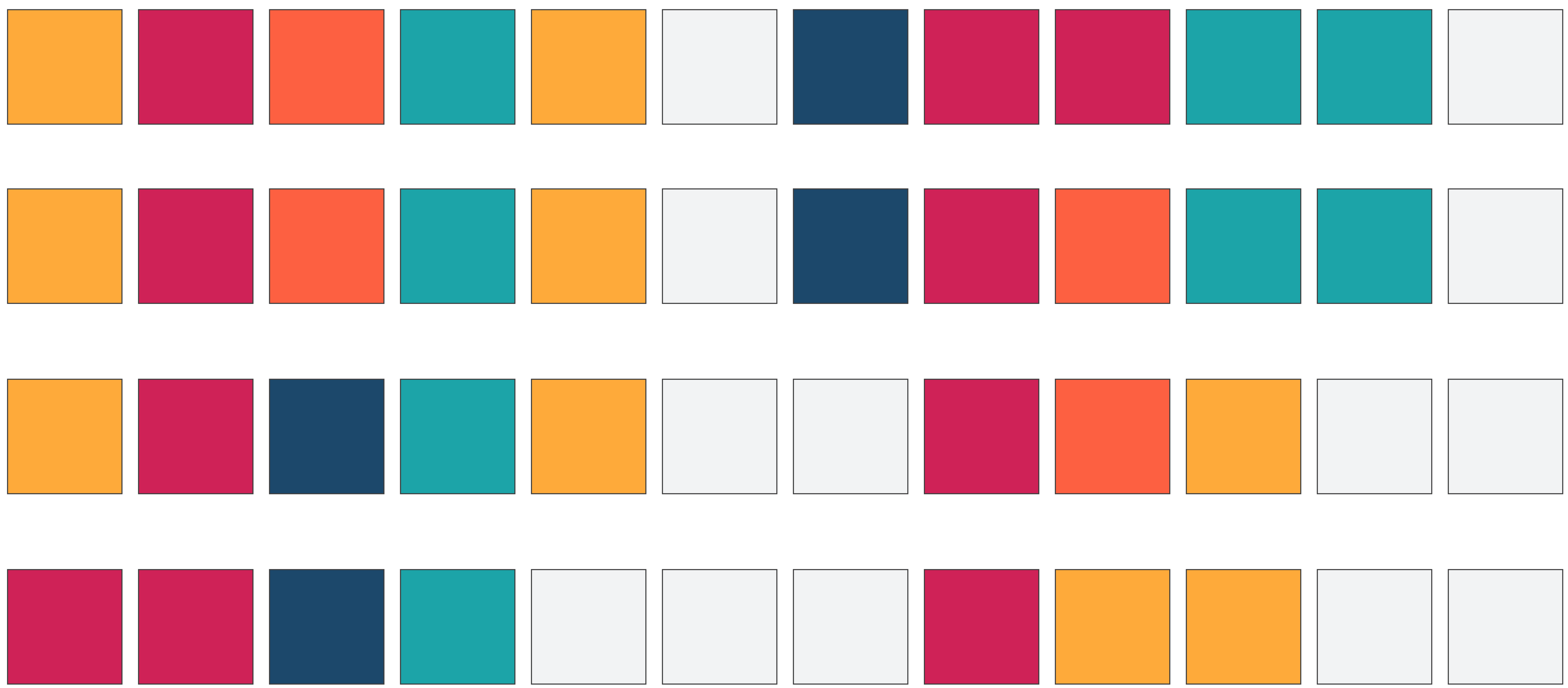


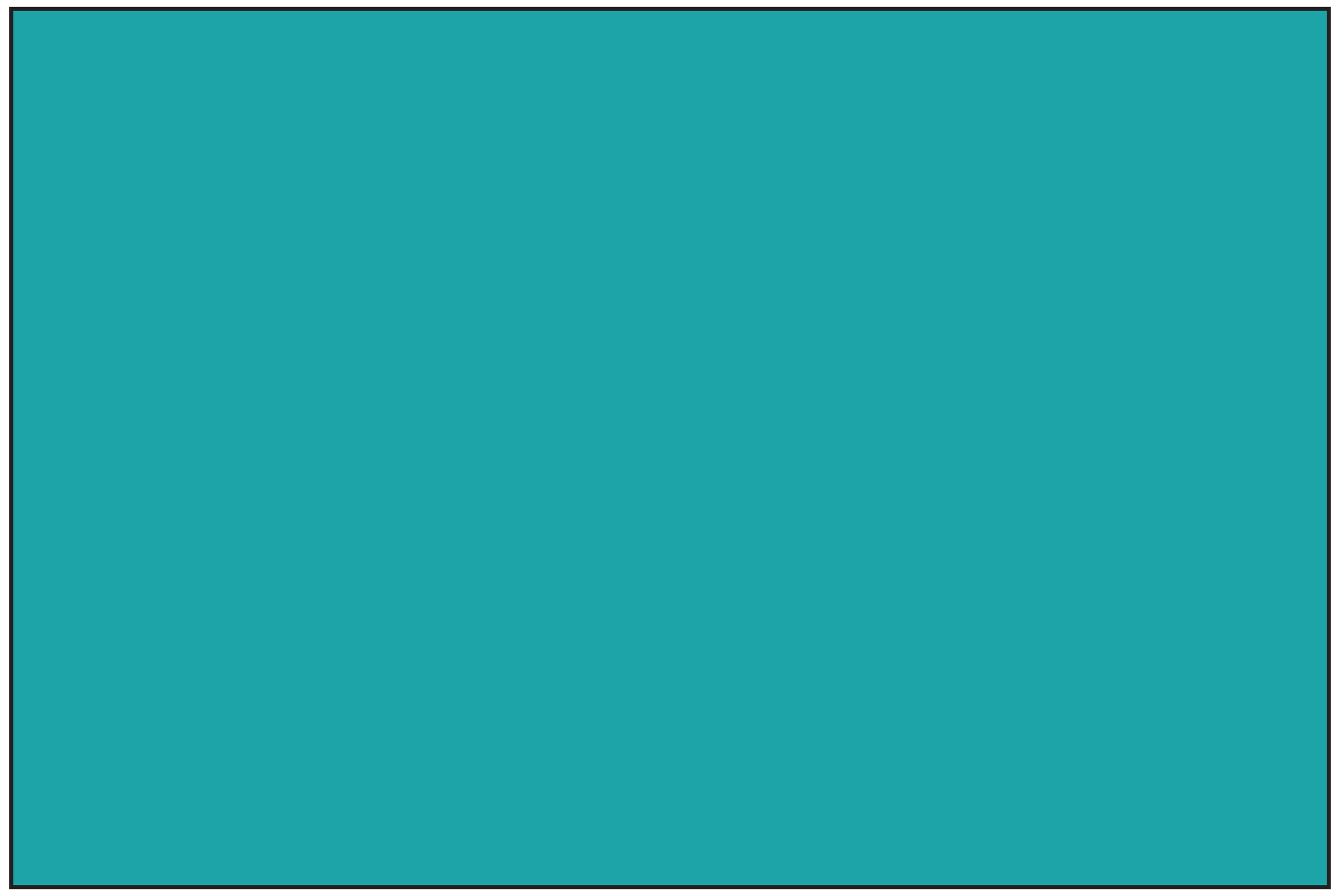
Can we save **more** with **parallelism**?

Parallel schedule

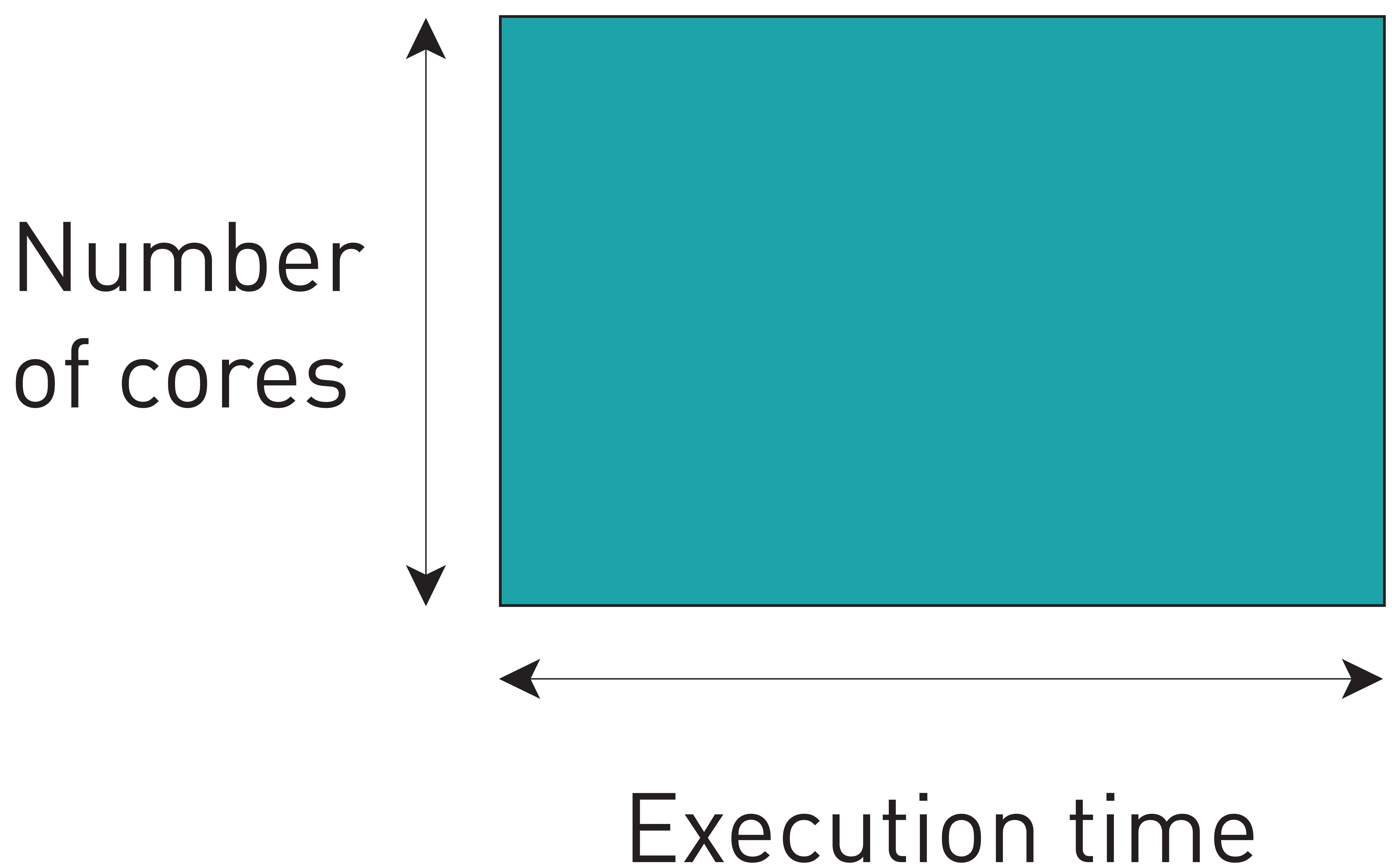




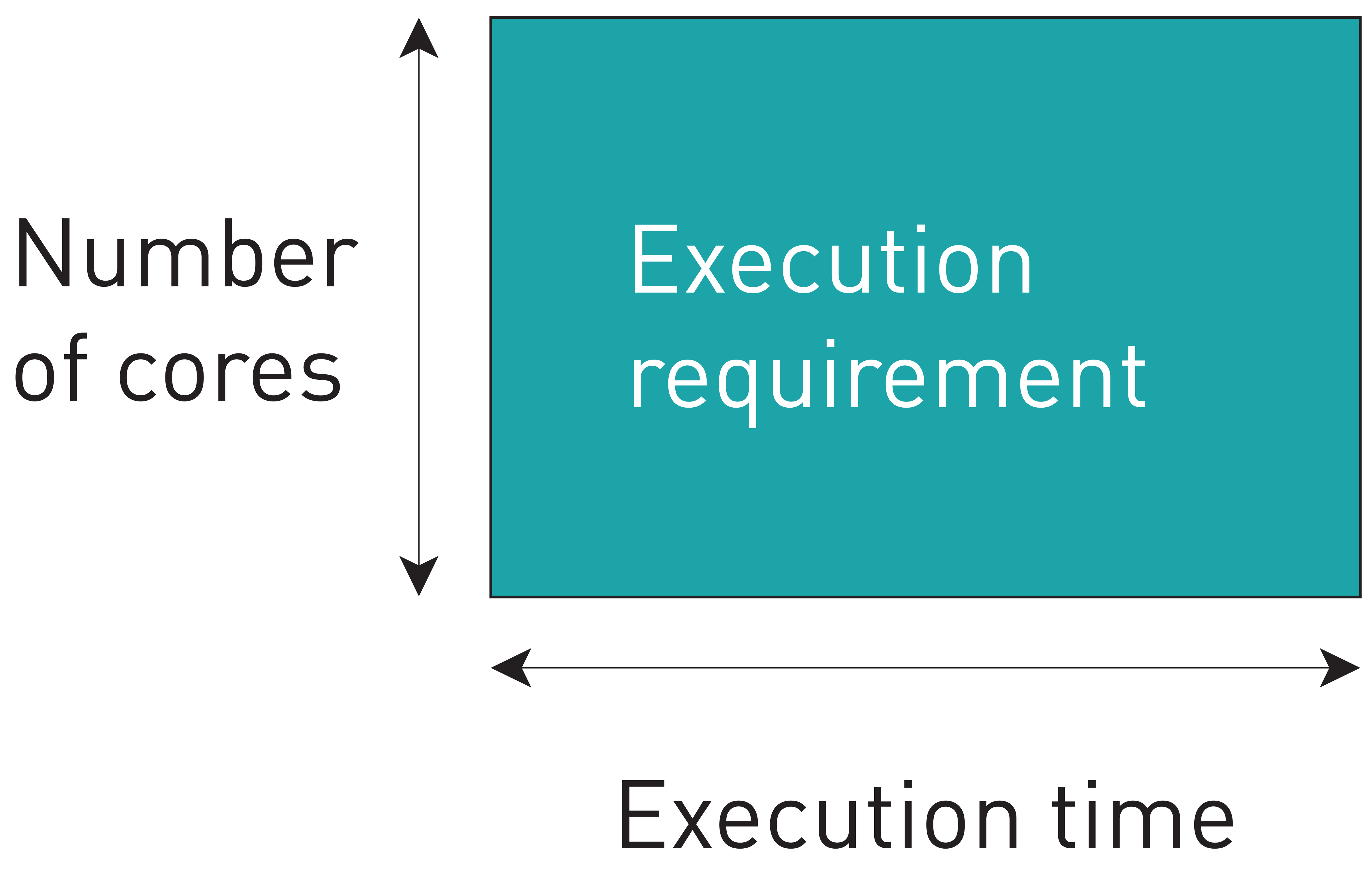




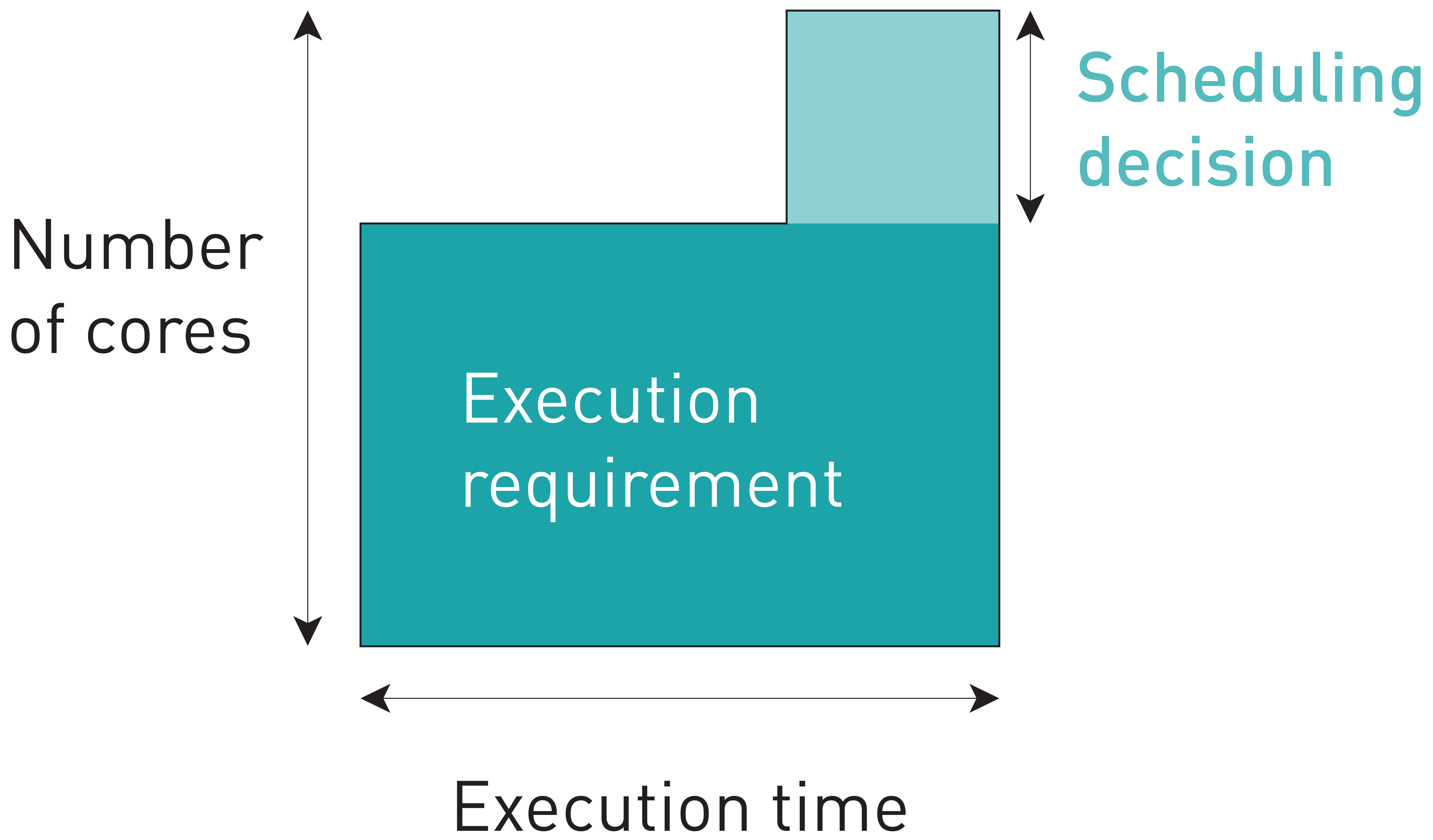
Parallel job model

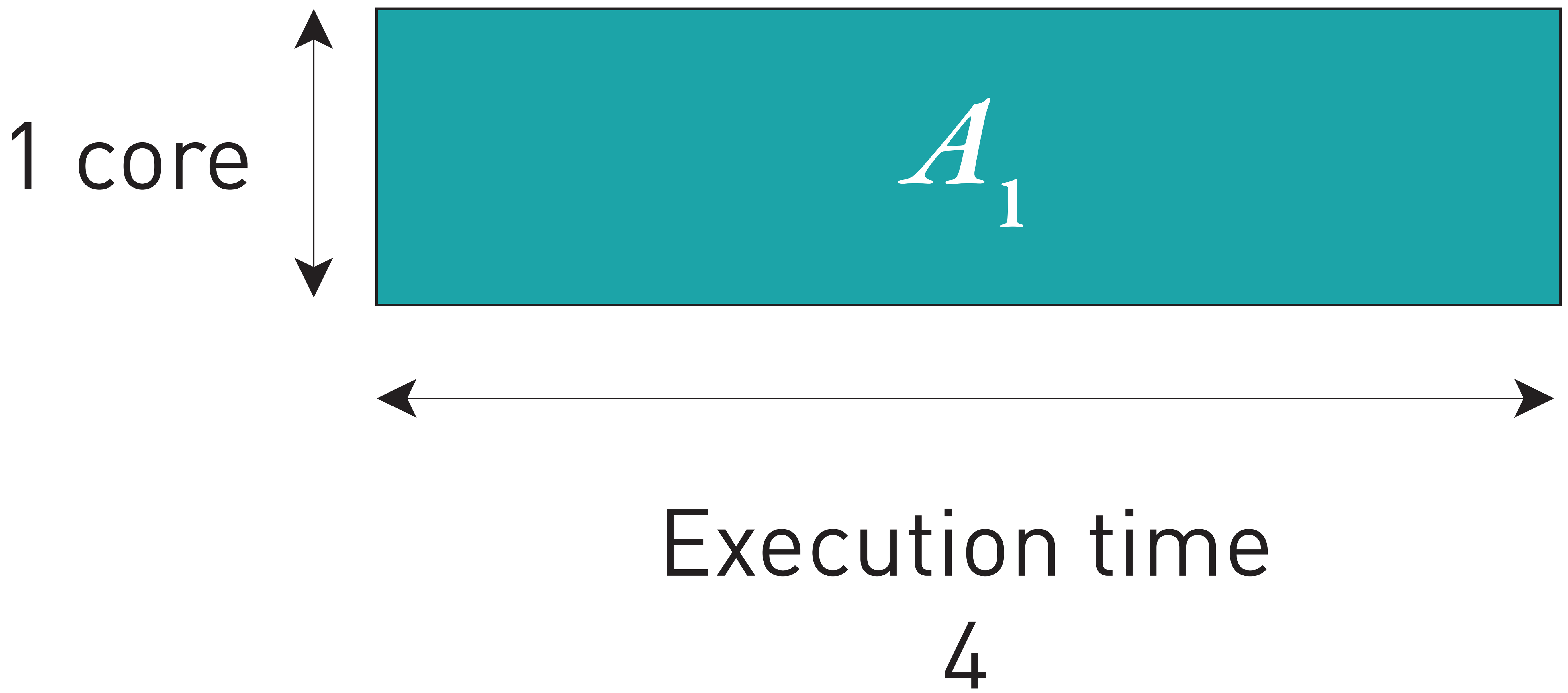


Parallel job model

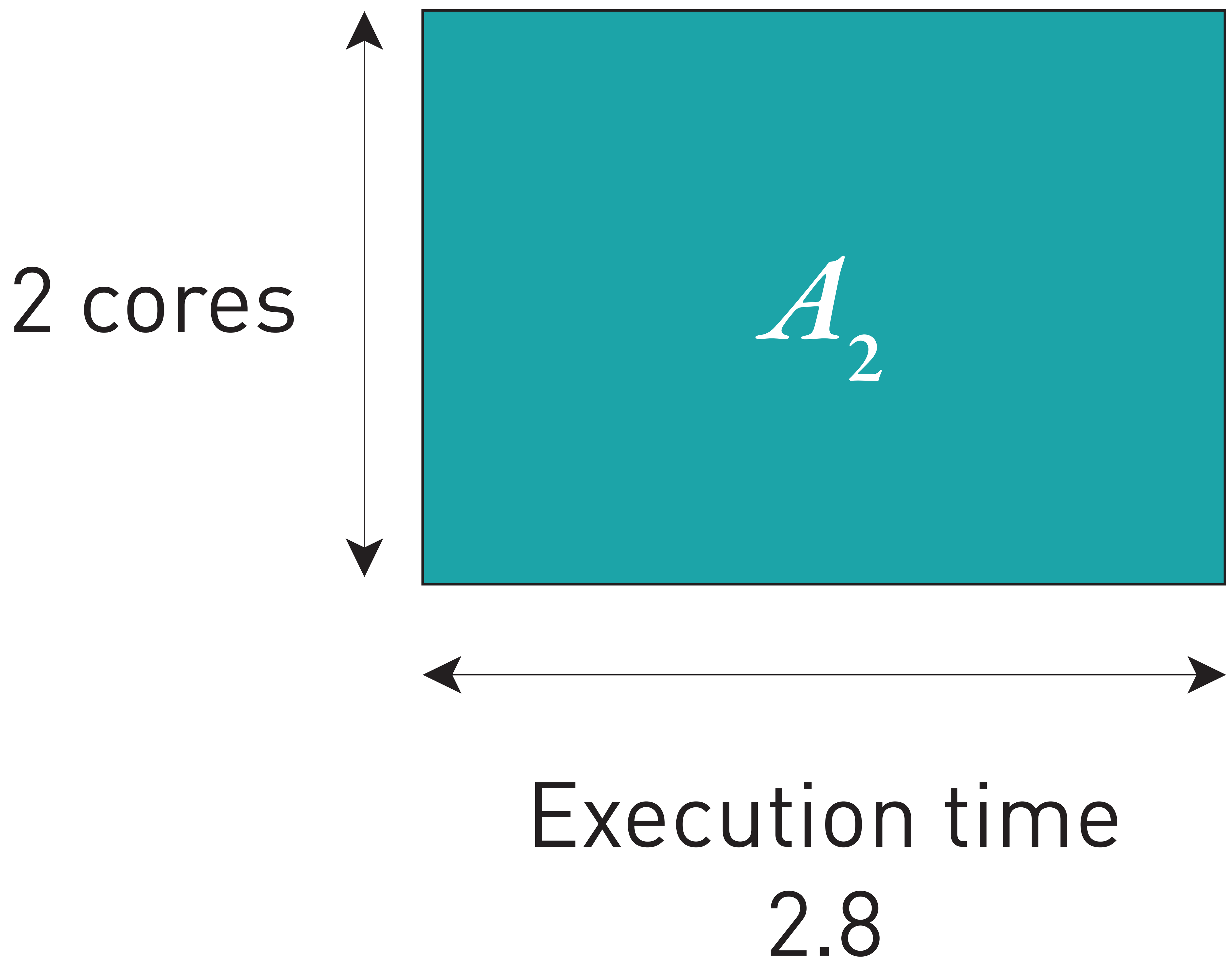


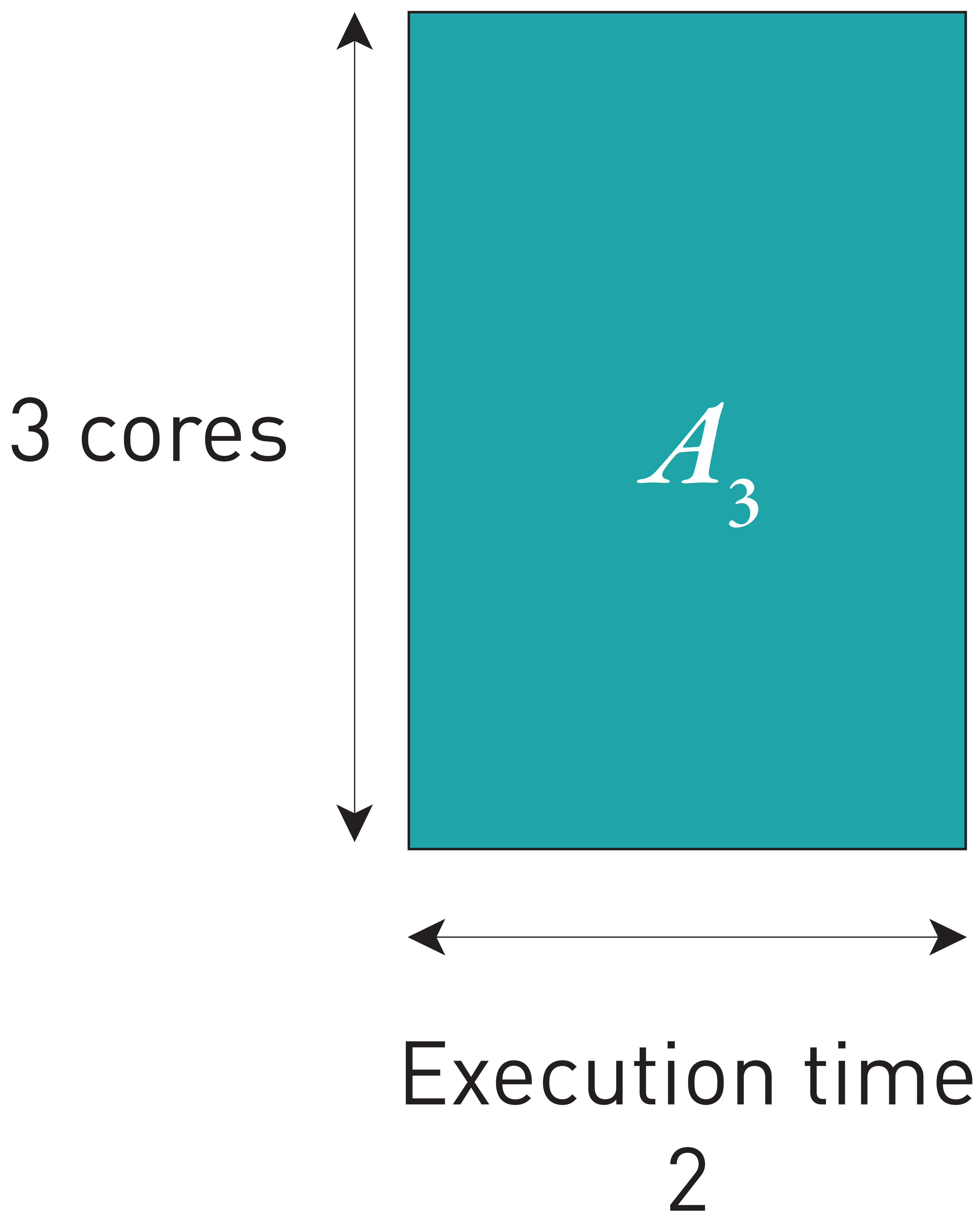
Malleable job





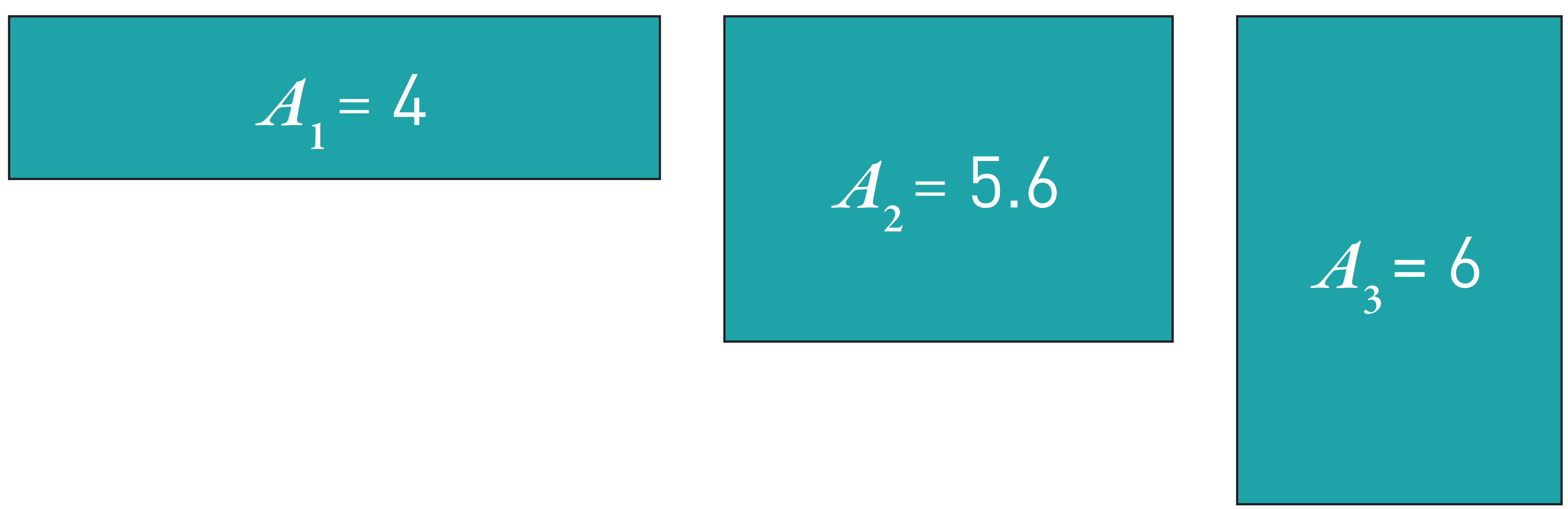
1 core
A_1
Execution time
4


2 cores
A_2
Execution time
2.8

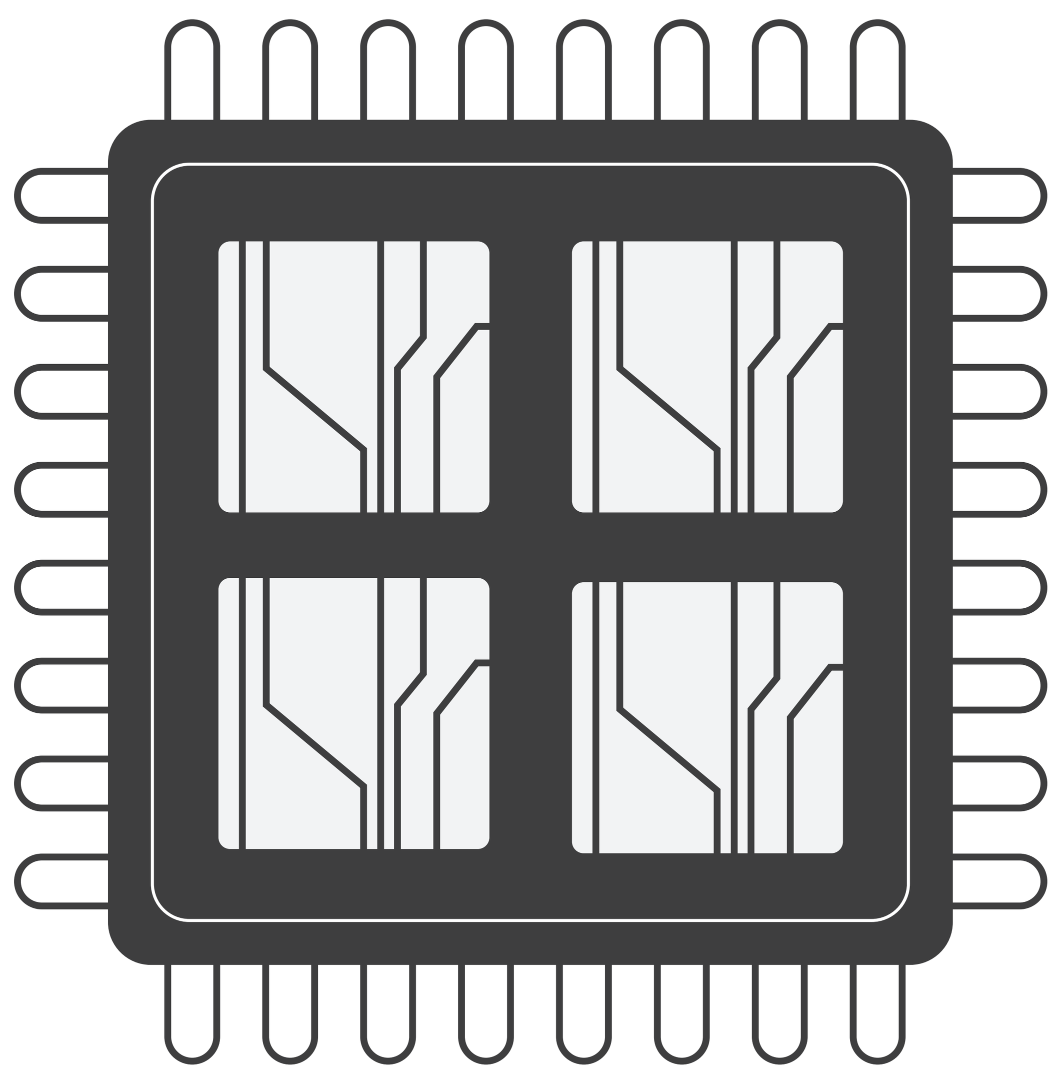

3 cores
A_3
Execution time
2

Sub-linear speedup ratio

$$A_k < A_{k+1}$$



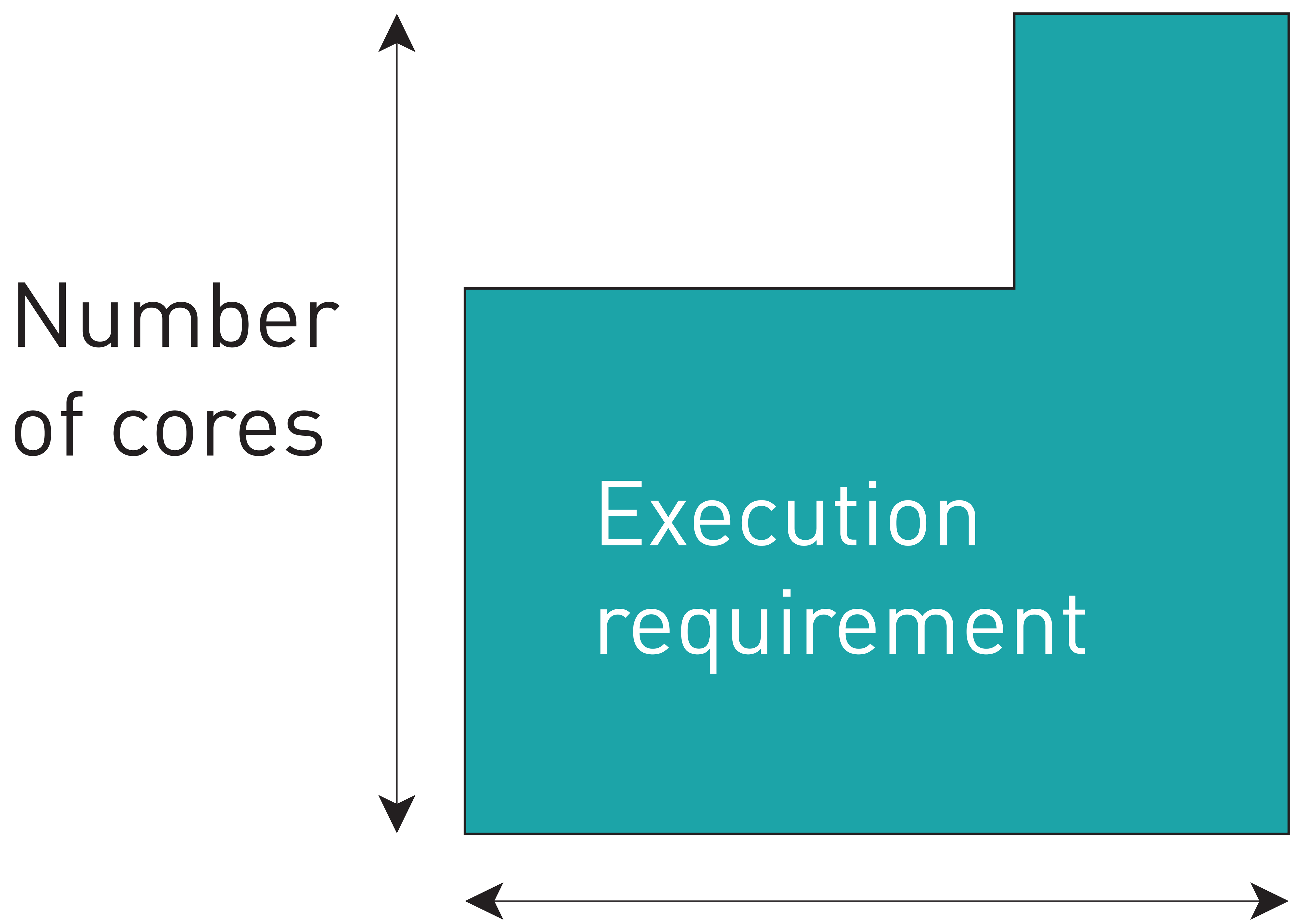
Processor Model



m cores, k are actives

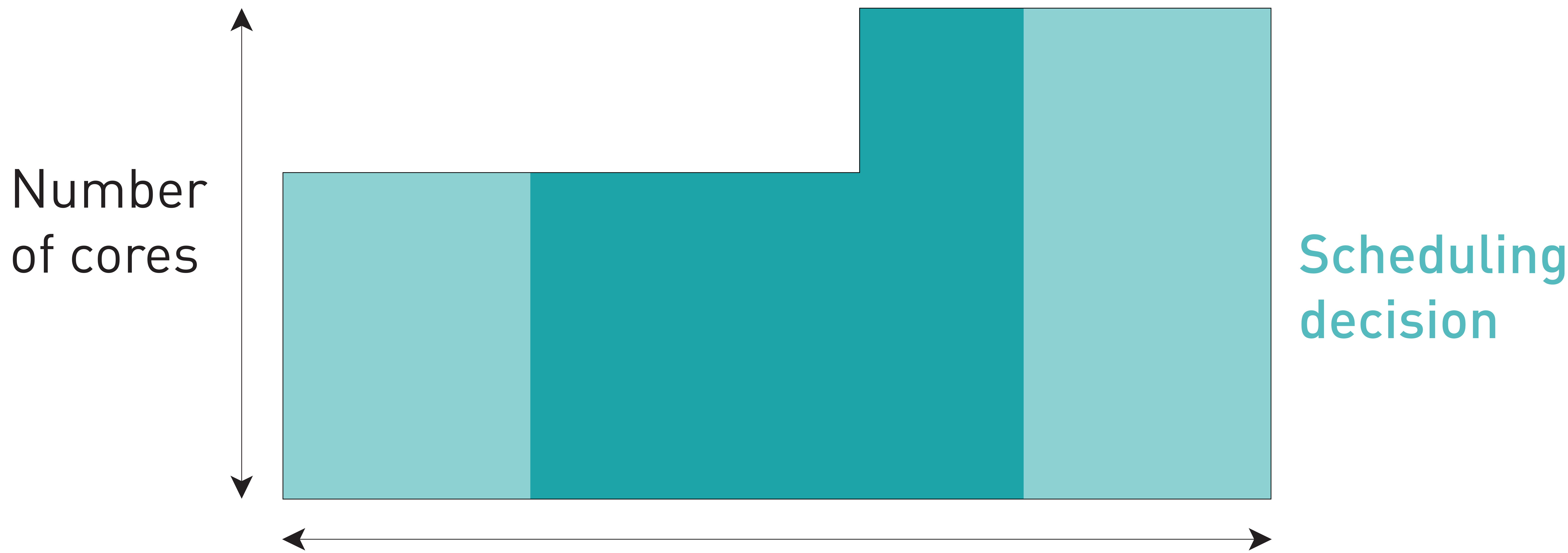
homogeneous frequency f

Frequency Scaling



$$\frac{\text{Execution time}}{f} \qquad f = 1$$

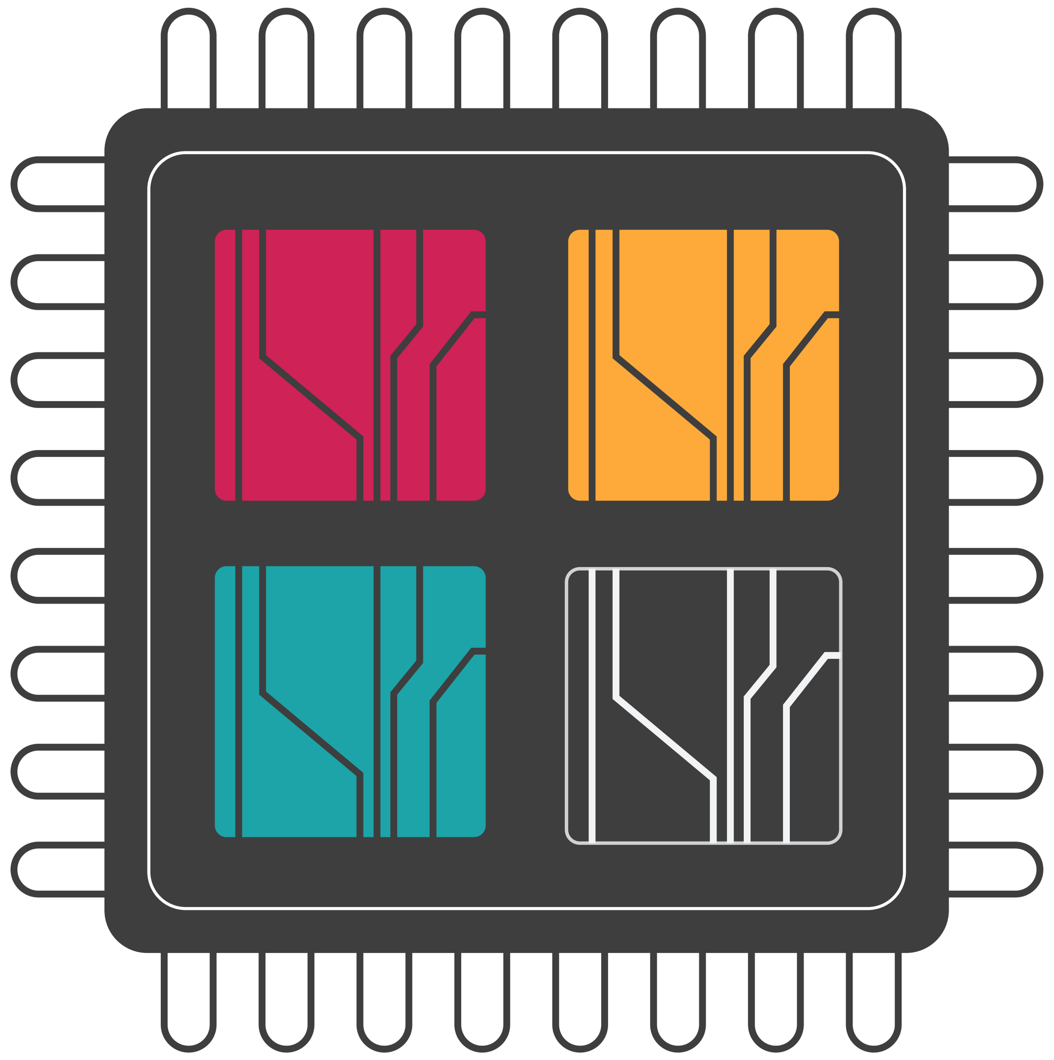
Frequency Scaling



$$\frac{\text{Execution time}}{f}$$

$f = 0.5$

Power Model

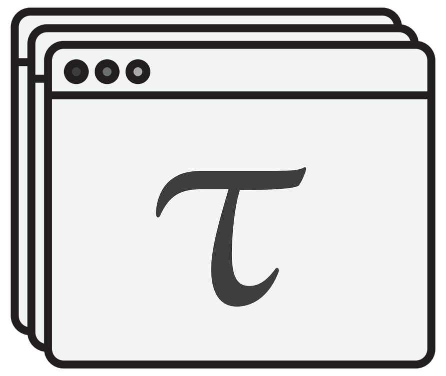


k active cores

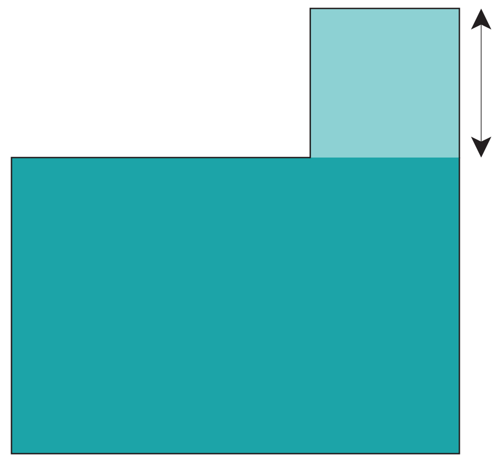
homogeneous frequency f

power function $P(f, k)$

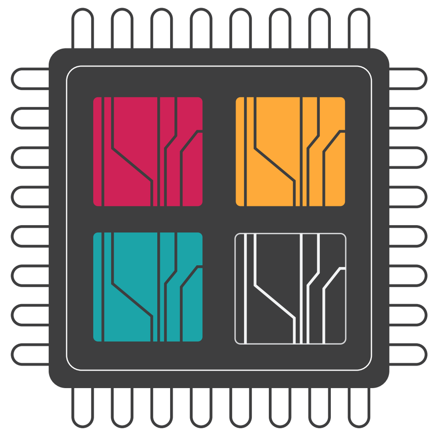
e.g. $P(f, k) = f^3 k + 0.15 \times k$



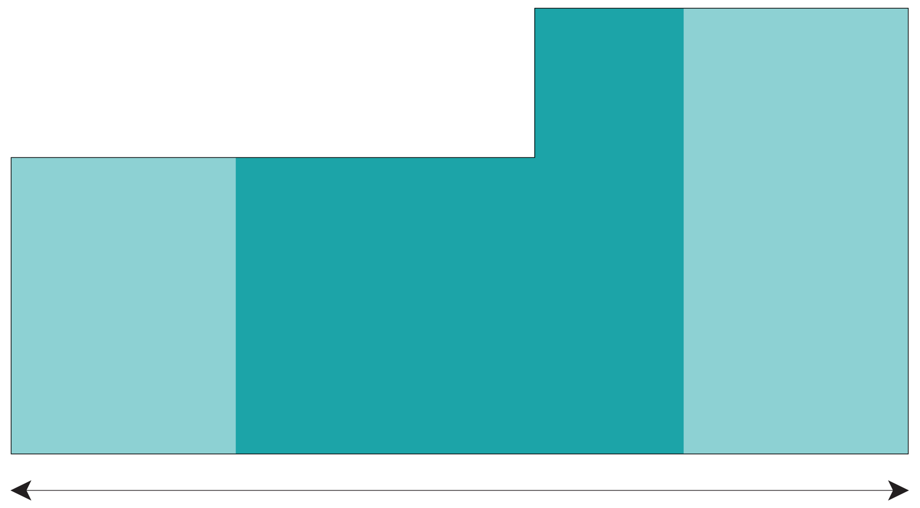
Implicit-deadline sporadic tasks



Malleable jobs



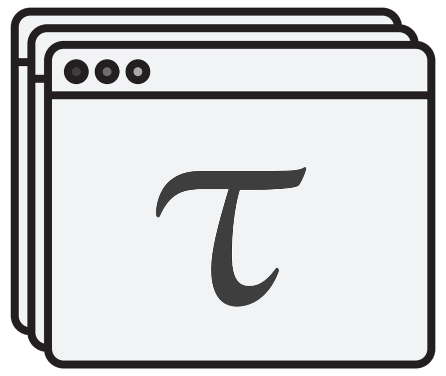
DVFS/DPM-enabled processor with m cores



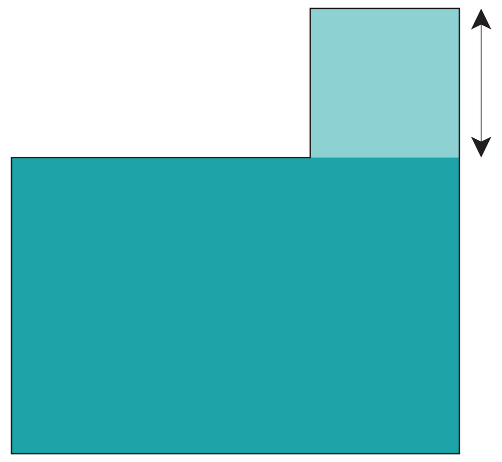
Homogeneous frequency

Canonical optimal scheduling

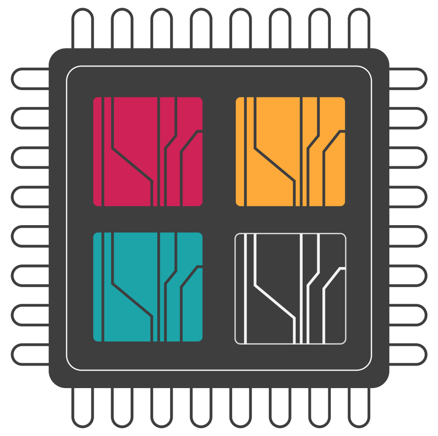
Find f and k such that $P(f, k)$ is minimized



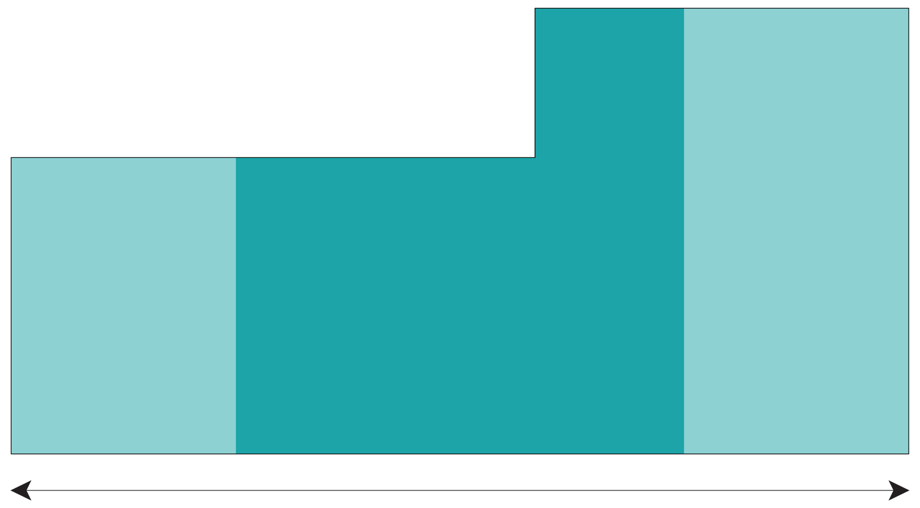
Implicit-deadline sporadic tasks



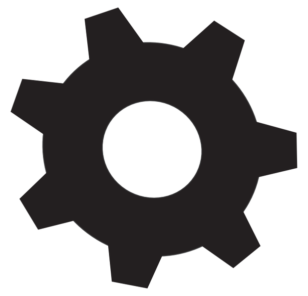
Malleable jobs



DVFS/DPM-enabled processor with m cores



Homogeneous frequency



Canonical optimal scheduling

Simulations

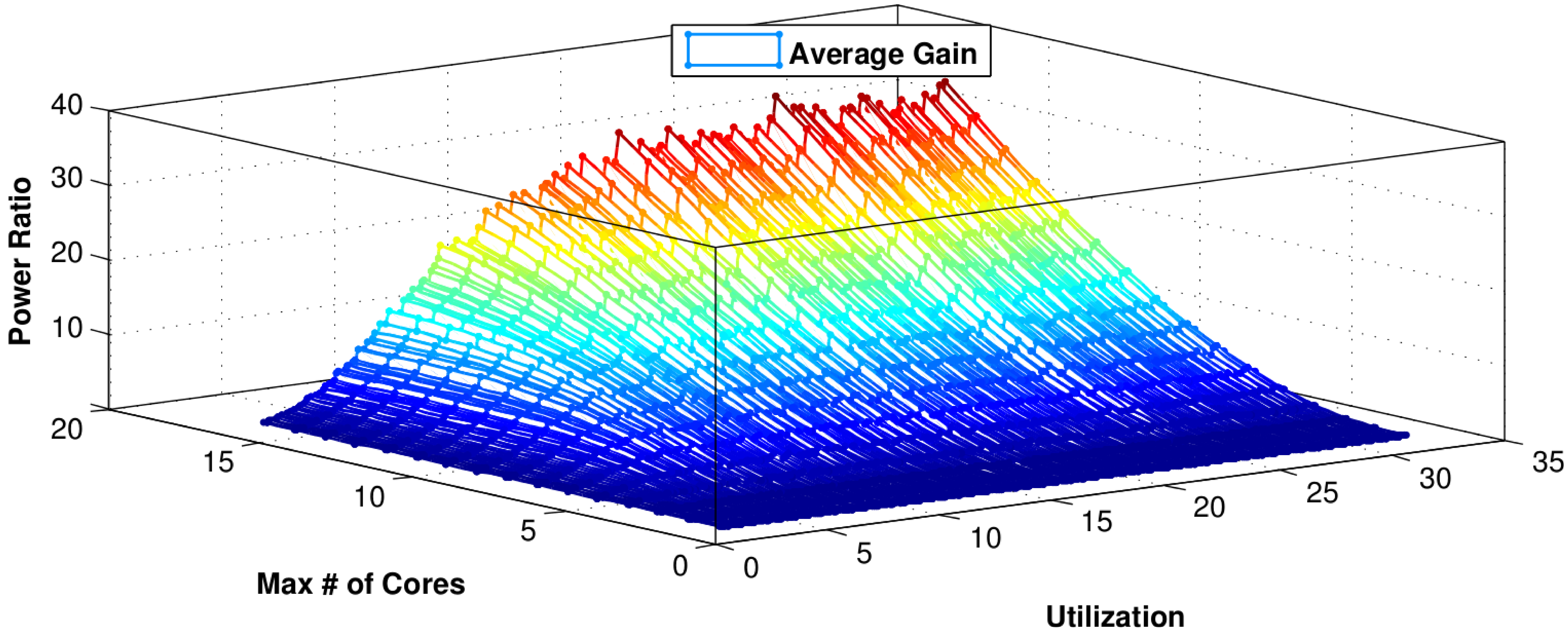
Randomly generated task systems

Compute minimum frequency
for sequential and malleable

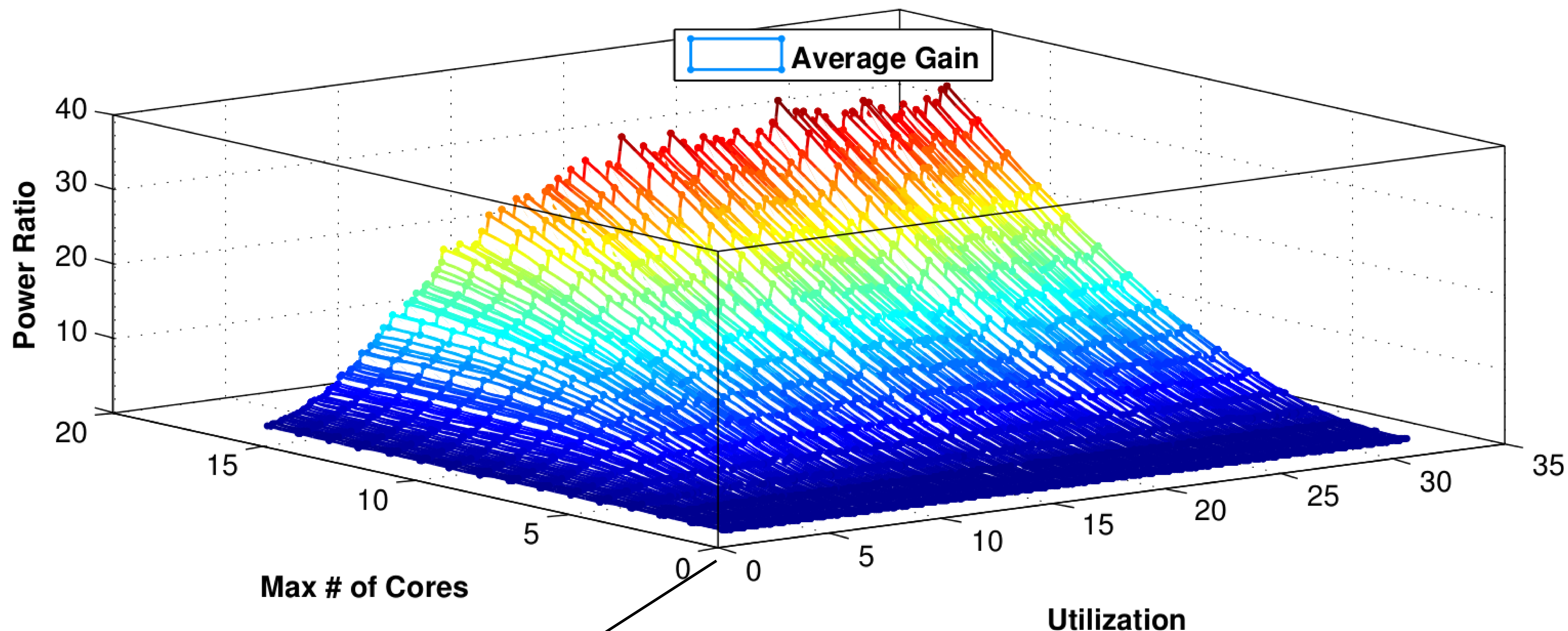
Evaluate savings with:

$$\frac{P(f_{seq}, k_{seq})}{P(f_{mal}, k_{mal})}$$

Simulation results



Simulation results



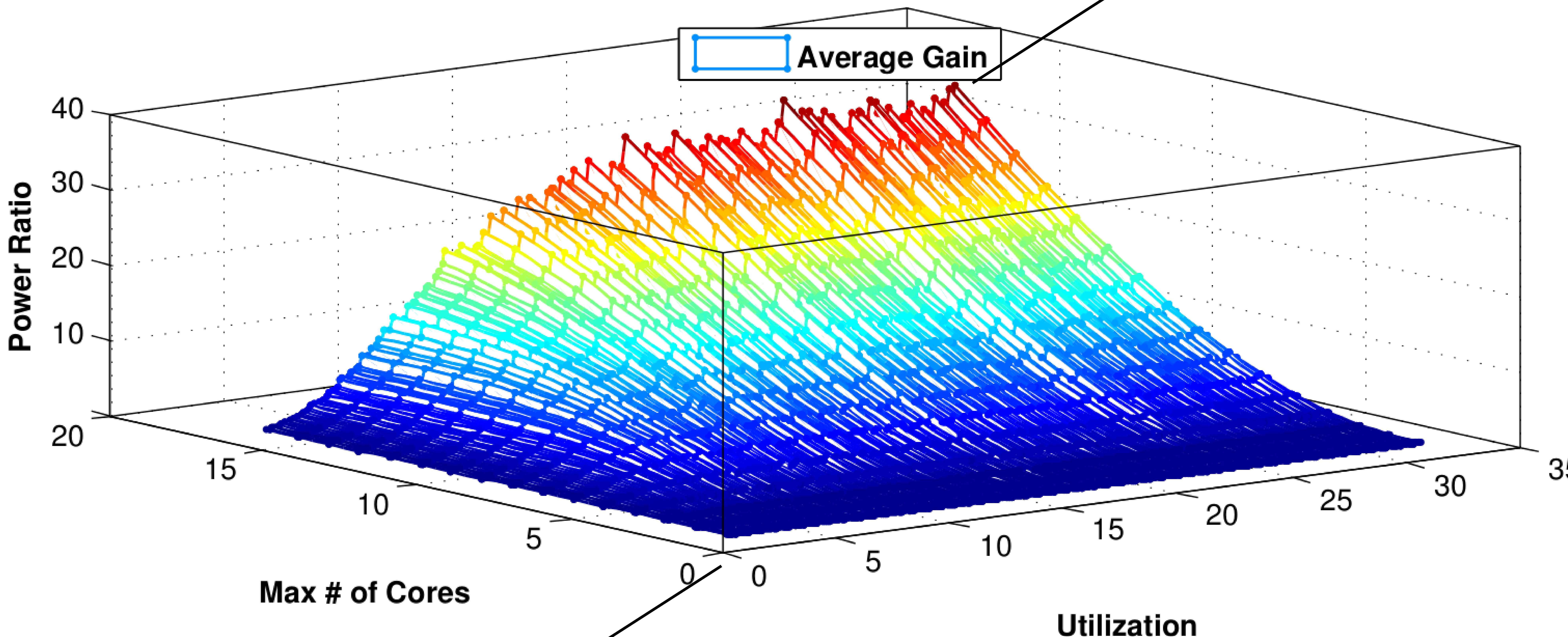
$$P_{mal} = P_{seq}$$

Ratio = 1

Simulation results

$P_{mal} << P_{seq}$

Ratio ≃ 36



$P_{mal} = P_{seq}$

Ratio = 1

Conclusion

Parallel schedule helps
to save power **in theory**.

Conclusion

Parallel schedule helps
to save power **in theory**.

More **practical** model
must be tested.

RTOS **implementation**
must be evaluated.

More in paper

- *Sub-linear speedup ratio [13]*:

$$1 < \frac{\gamma_{i,j'}}{\gamma_{i,j}} < \frac{j'}{j}$$
$$\text{where} \quad 0 < j < j' \leqslant m\,.$$

- *Work-limited parallelism [4]*:

$$\gamma_{i,(j'+1)} - \gamma_{i,j'} \leqslant \gamma_{i,(j+1)} - \gamma_{i,j}$$
$$\text{where} \quad 0 \leqslant j < j' < m\,.$$

Algorithm 2: minimumOptimalFrequency(τ, m)

for $i \in \{1, 2, \ldots, n\}$ **do**
  if schedulable$(\tau, m, \frac{u_i}{\gamma_{i,m}})$ **then**
    $\bar{\kappa}_i \leftarrow m - 1$
  else
    $\bar{\kappa}_i \leftarrow \min_{\kappa=0}^{m-1}\{\kappa \mid$ **not** schedulable$(\tau, m, \frac{u_i}{\gamma_{i,\kappa+1}})\}$
$\bar{\kappa} \stackrel{\text{def}}{=} \langle \bar{\kappa}_1, \bar{\kappa}_2, \ldots, \bar{\kappa}_n \rangle$
return $\Psi_\tau(m, \bar{\kappa})$

Algorithm 3: frequencyCoreSelection(τ, m)

$\ell_{\min} \leftarrow 1$
$f_{\ell_{\min}} \leftarrow$ minimumOptimalFrequency$(\tau, 1)$
for $\ell \in \{2, 3, \ldots, m\}$ **do**
  $f_\ell \leftarrow$ minimumOptimalFrequency(τ, ℓ)
  if $P(f_\ell, \ell) < P(f_{\ell_{\min}}, \ell_{\min})$ **then**
    $\ell_{\min} \leftarrow \ell$
    $f_{\ell_{\min}} \leftarrow f_\ell$
return $\langle f_{\ell_{\min}}, \ell_{\min} \rangle$

Theorem 1 (extended from Collette et al. [4])**.** *A necessary and sufficient condition for an implicit-deadlines sporadic malleable task system τ respecting* sub-linear speedup ratio *and* work-limited parallelism, *to be schedulable by the canonical schedule on m processors at frequency f is given by:*

$$\begin{cases} \max_{i=1}^{n}\{k_i(f)\} < m \quad \textit{and} \\ \sum_{i=1}^{n}\left(k_i(f) + \frac{u_i - \gamma_{i,k_i(f)} f}{(\gamma_{i,k_i(f)+1} - \gamma_{i,k_i(f)})f}\right) \leqslant m\,. \end{cases} \tag{4}$$

Other useful specific values are $u_{\max} \stackrel{\text{def}}{=} \max_{i=1}^{n}\{u_i\}$ and $u_{\text{sum}} \stackrel{\text{def}}{=} \sum_{i=1}^{n} u_i$.

A collection of sporadic tasks $\tau \stackrel{\text{def}}{=} \{\tau_1, \tau_2, \ldots, \tau_n\}$ is called a *sporadic task system*. In this paper, we assume a common subclass of sporadic task systems called *implicit-deadline sporadic task systems* where each $\tau_i \in \tau$ must have its relative deadline equal to its period (i.e., $d_i = p_i$).

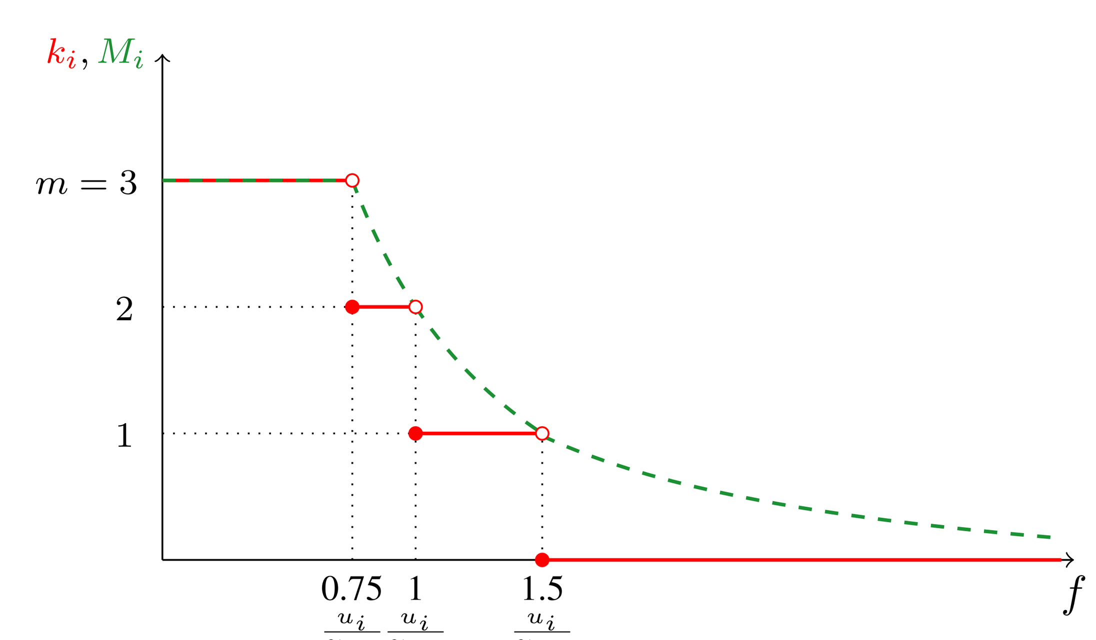
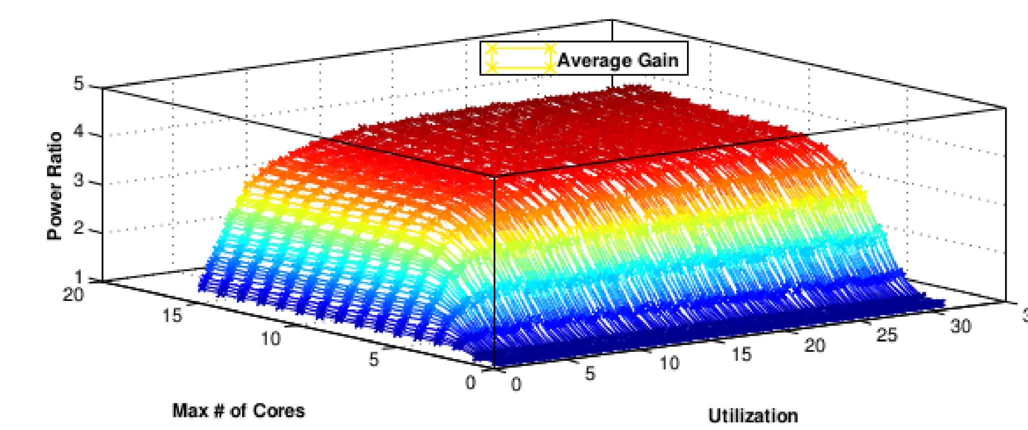


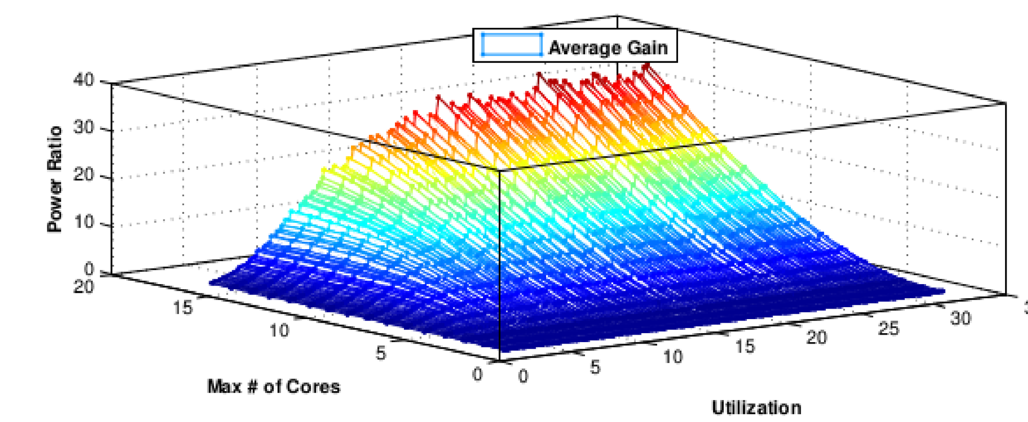
Fig. 1: Plot of k_i and M_i for $m = 3, \tau_i = (6, 4, \langle 1.0, 1.5, 2.0 \rangle)$

$$k_i(f) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if } u_i \leq \gamma_{i,1} f \\ \max_{k=1}^{m}\{k \mid \gamma_{i,k} f < u_i\}, & \text{otherwise.} \end{cases} \tag{3}$$

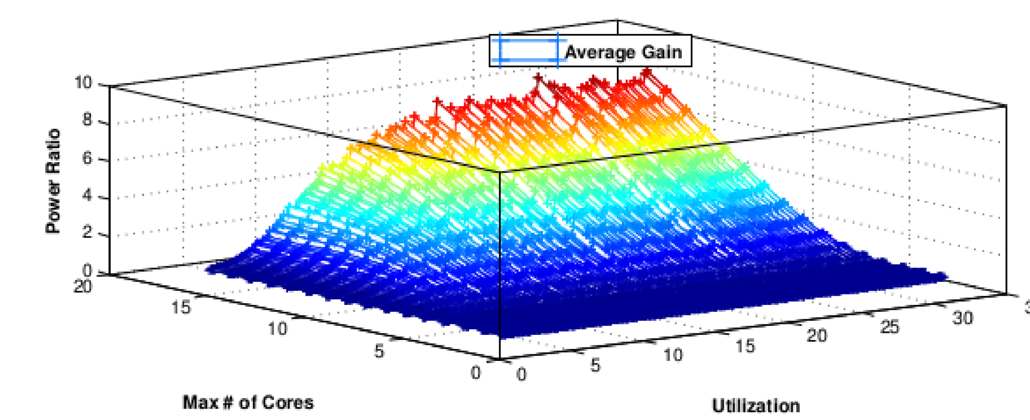
The *canonical schedule* fully assigns $k_i(f)$ processor(s) to τ_i and at most one additional processor is partially assigned (see [4] for details). This definition extends the original definition of k_i from non-power-aware parallel systems [4].



(a) Power savings for WPS over SEQ.



(b) Power savings for SPS over SEQ.



(c) Power savings for SPS over WPS.

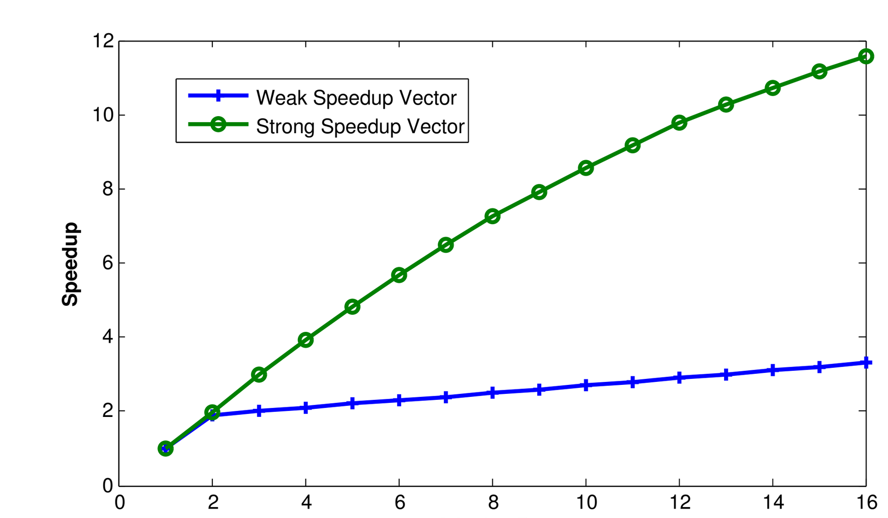


Fig. 2: Speedup Vectors for *strong* and *weak* parallelized systems.

# cores	Γ_{WPS}	Γ_{SPS}	# cores	Γ_{WPS}	Γ_{SPS}
1	1.0	1.000	9	2.6	7.926
2	1.9	1.990	10	2.7	8.559
3	2.0	2.970	11	2.8	9.185
4	2.1	3.913	12	2.9	9.771
5	2.2	4.801	13	3.0	10.271
6	2.3	5.670	14	3.1	10.726
7	2.4	6.505	15	3.2	11.148
8	2.5	7.255	16	3.3	11.558

Fig. 3: Values of speedup vectors for WPS and SPS.