LECTURE 9
Scheduling of robot actions

Andrea Zanchettin

Agenda

- Introduction and motivations

- Petri Nets: a mathematical tool for discrete event systems
  - Generalities
  - Time Petri Nets

- Scheduling with time Petri Nets
  - The cost function
  - Scheduling algorithm

Introduction and motivations

Scheduling, or more formally **job shop scheduling**, is an optimization problem in which jobs are assigned to resources at particular times.

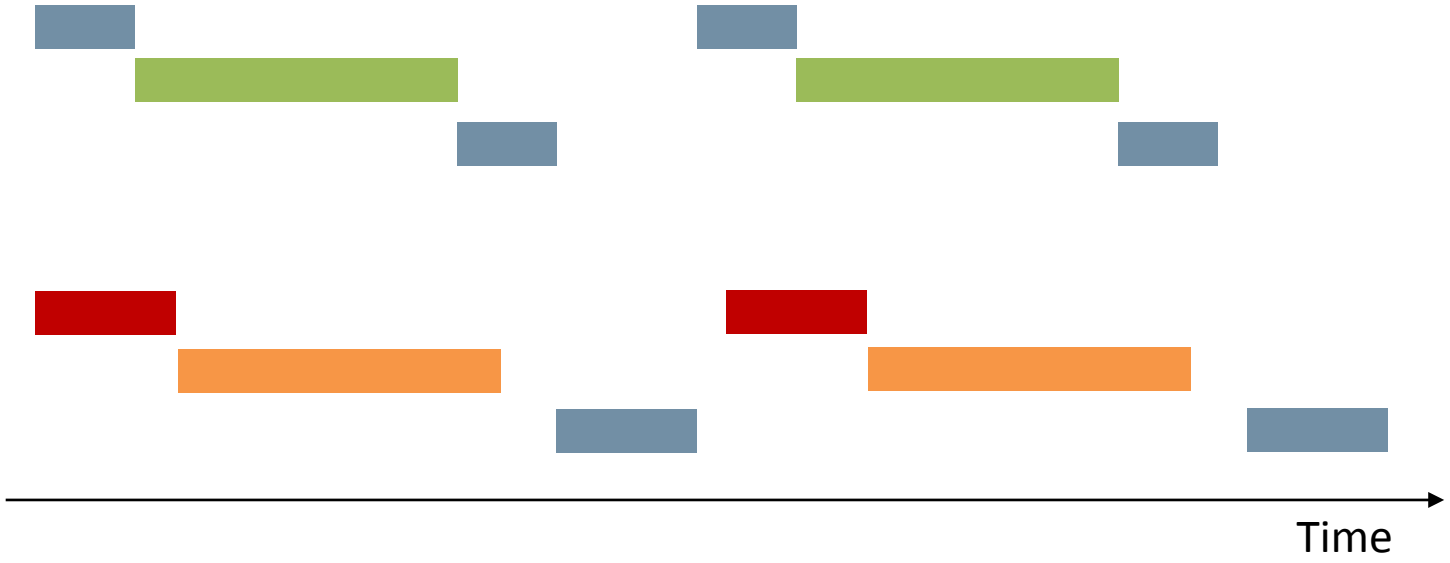
We are given *n* jobs $J_1, J_2, ..., J_n$ which need to be scheduled, and m agents (robots, machines, operators, …).

Within each job there is a set of operations $O_1, O_2, ..., O_n$ together with precedence constraints (preconditions on operations). Additionally, agents (especially machines and robots):

- might have operation-dependent setups
- might fail in performing operations
- might have uncertain durations when assigned to operations

The most common tool practitioners adopt to perform (and visualize) scheduling is the Gantt chart:

Operations	**Resource**
Job A	
Operation A.1	Robot 1
Operation A.2	Machine 2
Operation A.3	Robot 1
Job D	
Operation D.1	Operator 1
Operation D.2	Machine 1
Operation D.3	Robot 1



Introduction and motivations

There are several variations (simplifying assumptions) for the scheduling problem.

Each assumption generates a simplified problem:

- **dispatching problem**: solves the problem of assigning operations to resources
- **sequencing problem**: solves the problem of prioritizing the execution of operations

In the following, we will address how to efficiently describe and optimally solve sequencing problems.

Introduction and motivations

What should the robot do?
- Assembly #1 …?
- Testing …?
- Waiting …?

Agenda

- Introduction and motivations
- Petri Nets: a mathematical tool for discrete event systems
  - Generalities
  - Time Petri Nets
- Scheduling with time Petri Nets
  - The cost function
  - Scheduling algorithm

Petri Nets - generalities

Petri nets are mathematical modeling languages for the description of discrete event dynamic system.

A Petri net is a directed and weighted bipartite graph. Nodes are either places (P) or transitions (T). Directed edges (arrows) connect one place with one transition or vice versa.

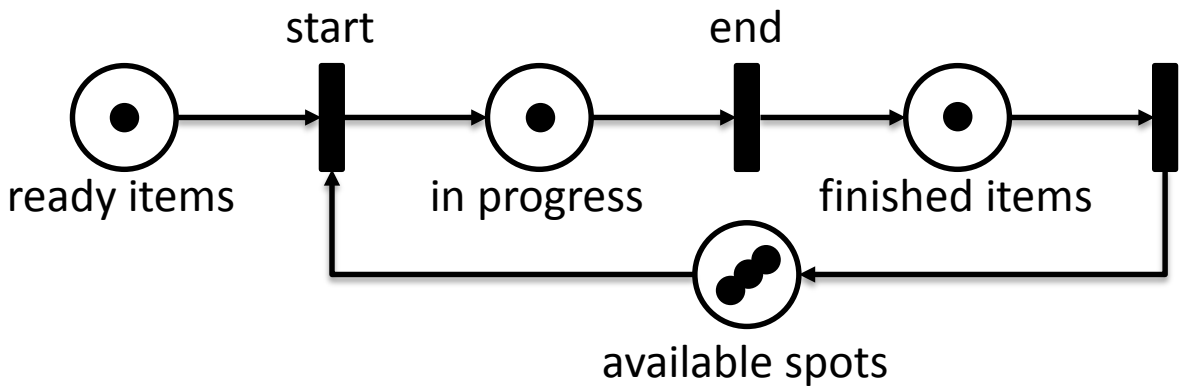
More formally, a Petri net *PN* is $\mathrm{PN} = \{P, T, W^+, W^-, M_0\}$ where

$$P \cap T = \emptyset \qquad M_k \in \left(\mathbb{N}^+ \cup \{0\}\right)^{|P|}$$

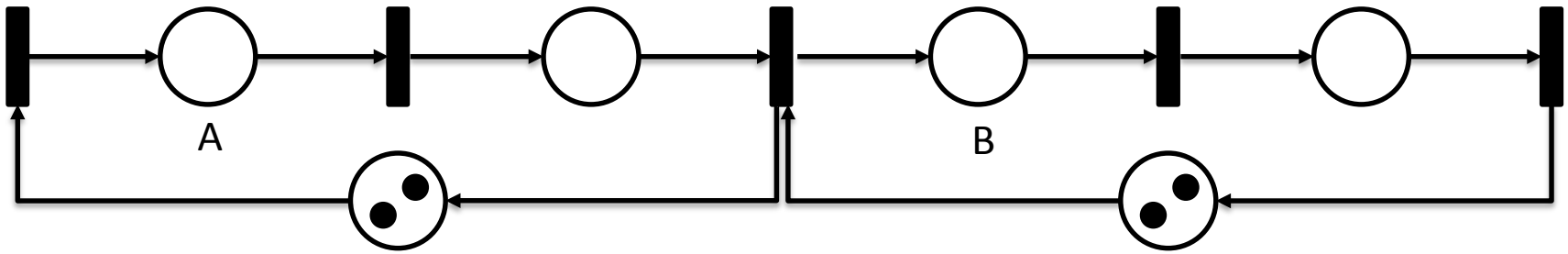
$$W^-, W^+ \in \left(\mathbb{N}^+ \cup \{0\}\right)^{|P|} \times \left(\mathbb{N}^+ \cup \{0\}\right)^{|T|}$$

Petri Nets - generalities

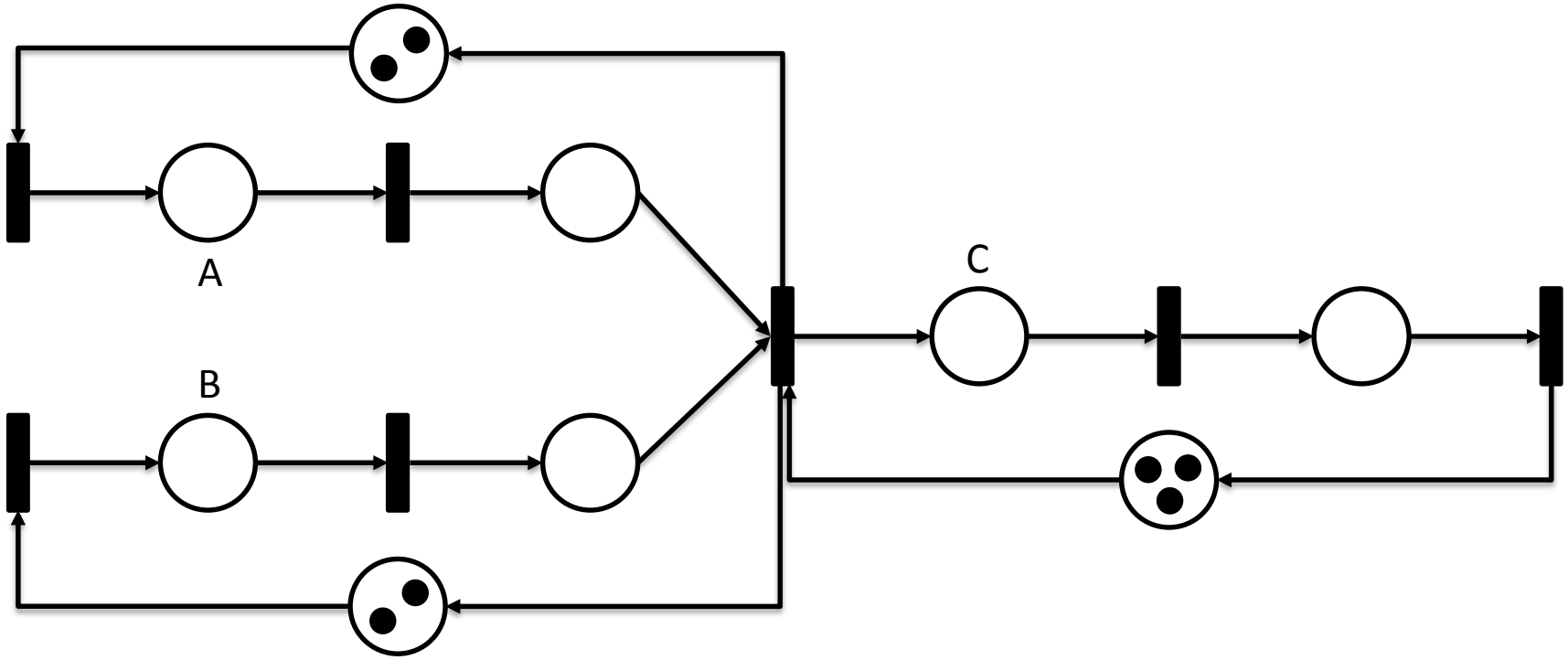
In the following, we will first address the problem of how to model typical structures (priorities between operations, storage, …) without considering how the operations are actually performed.



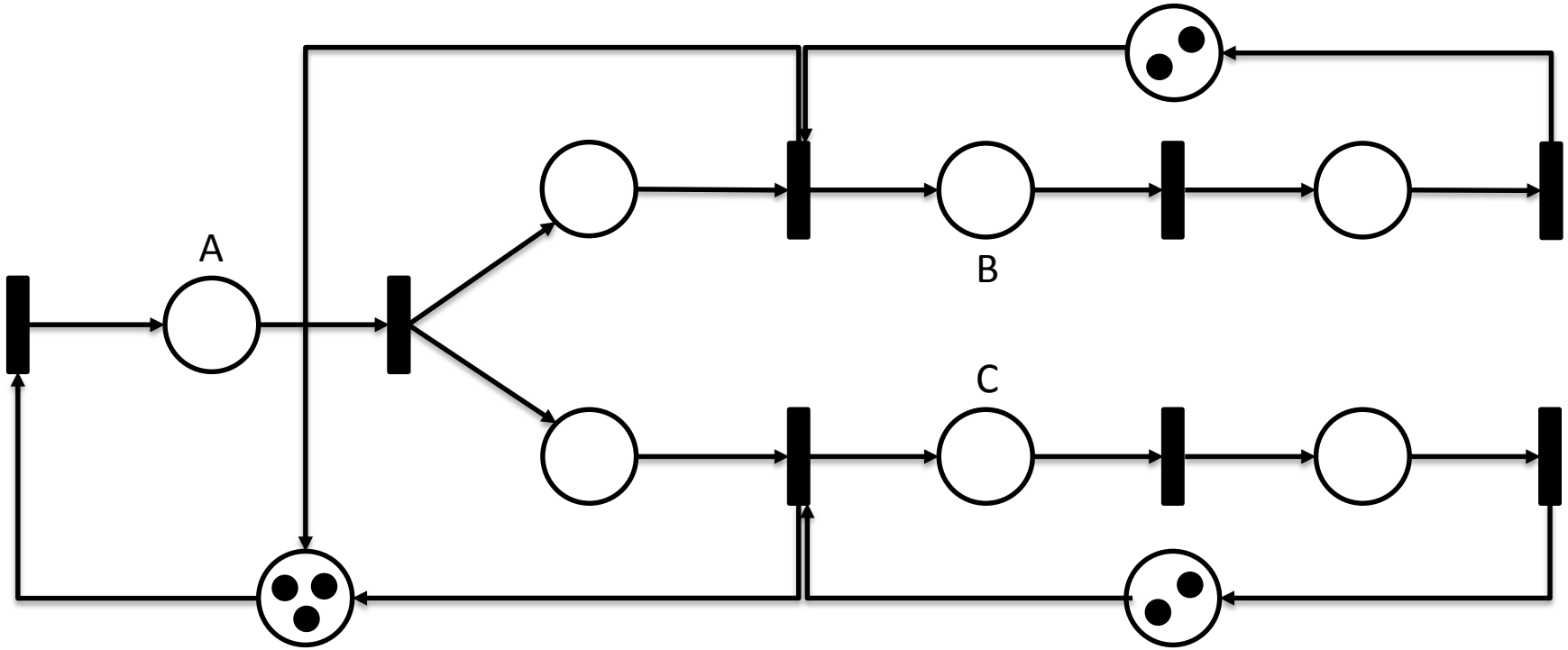
Strict sequence (B must be done after A):



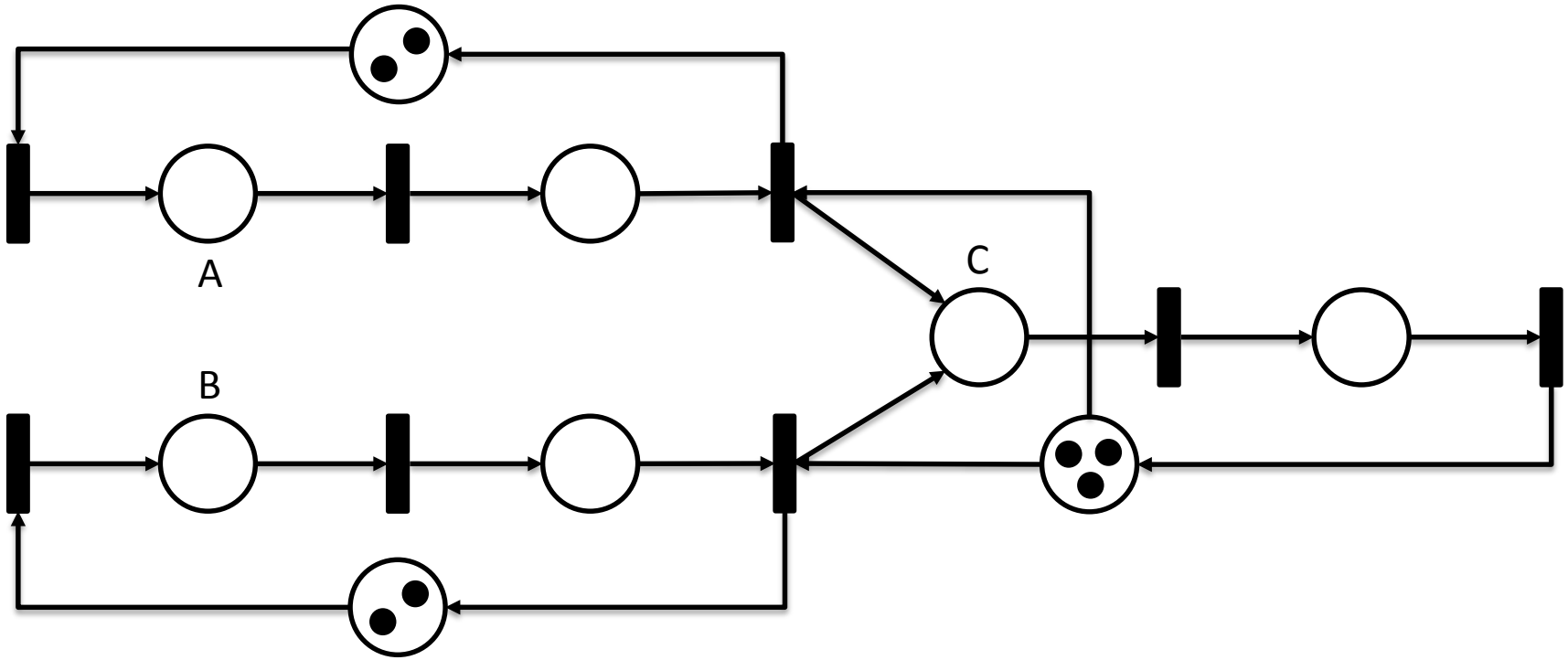
AND convergence (C must be done after A and B, but their relative order is irrelevant):



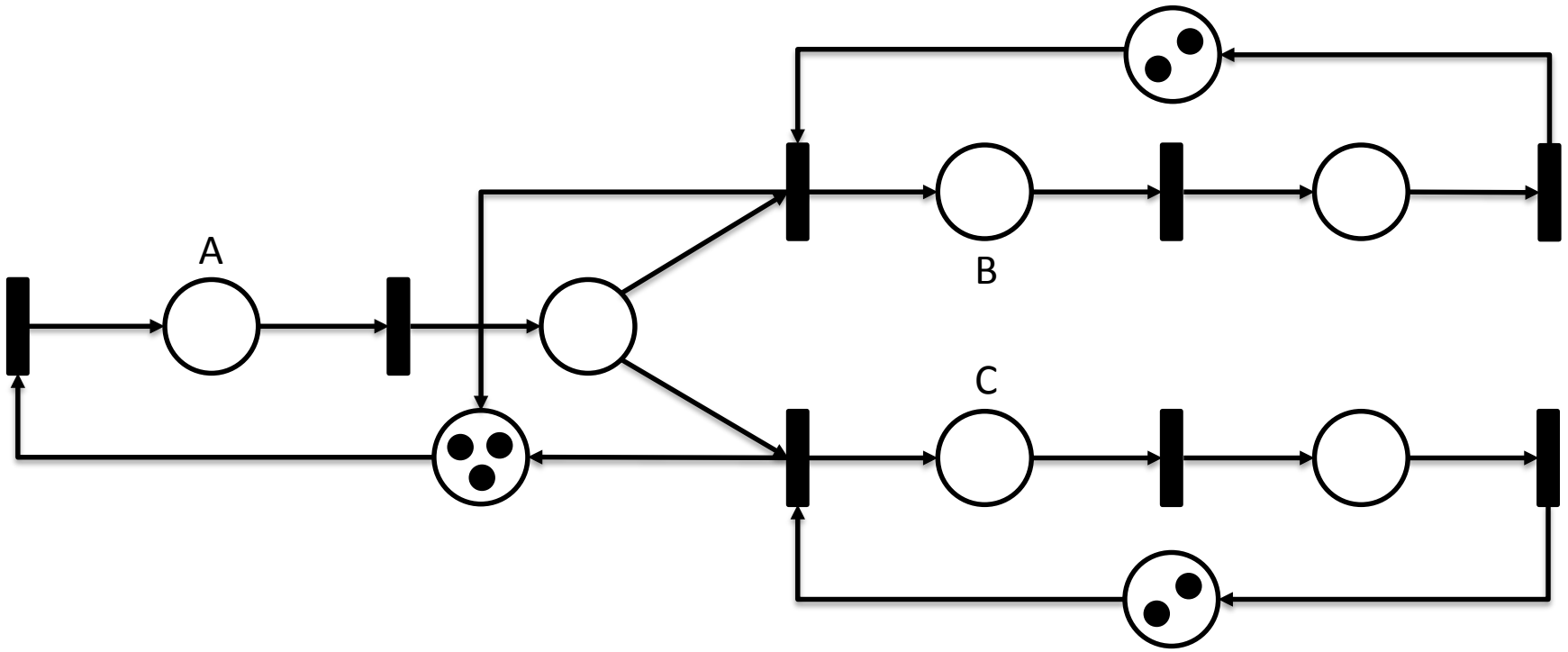
AND divergence (B and C must be done after A, but their relative order is irrelevant):



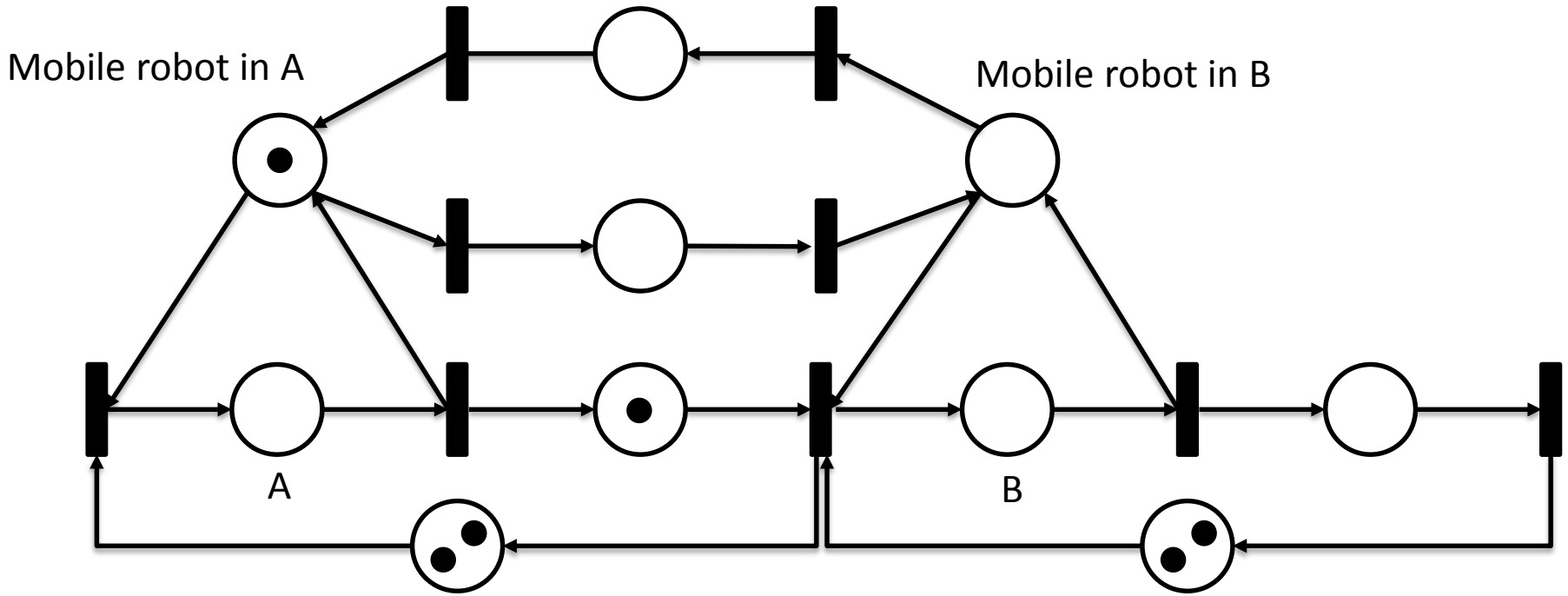
XOR convergence (C must be done after either A or B):



XOR divergence (either B or C must be done after A):

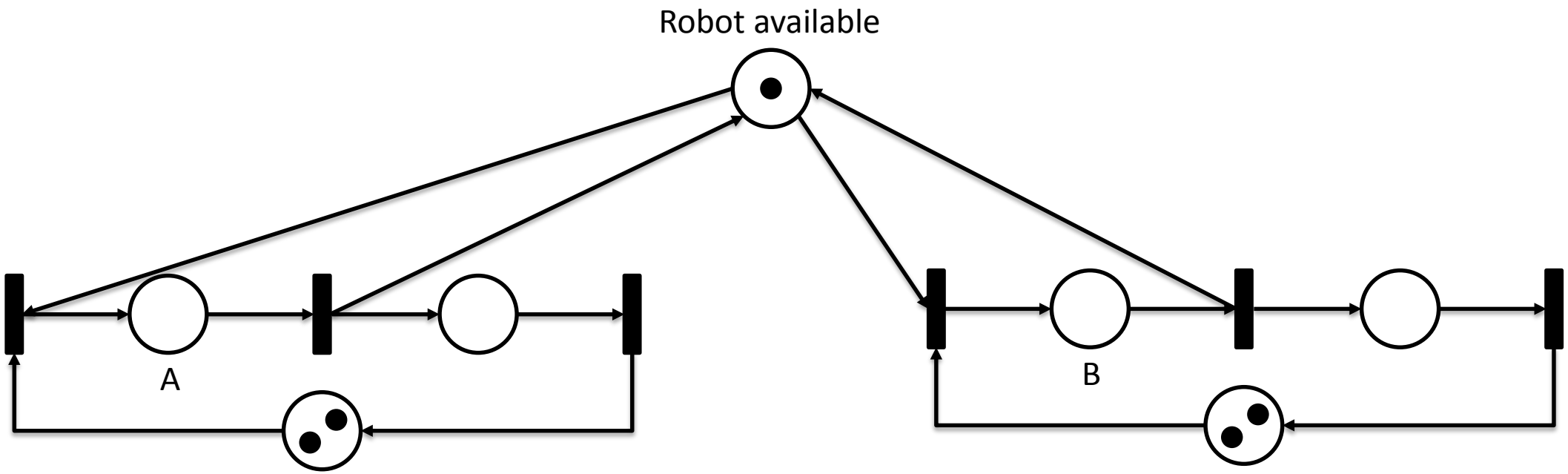


Operations might require some time to move parts (e.g. using mobile robots) to another department before being processed again:



Petri Nets - generalities

Since we will possibly have **more operations than agents** (as in the motivating example) we need to model how performing a certain operation will make an agent unavailable to perform other operations.



Petri Nets - time Petri Nets

So far, only sequential constraints and on the availability of resources and work-in-progress (WiP) have been specified.

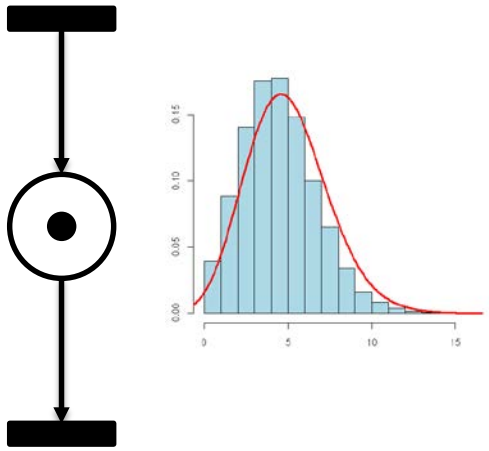
In order to model the **durations of operations**, we need to complicate the Petri net model in order to embed the concept of delay.

In general, within a T-Time Petri Net **delays** are attached to each transition, specifying that, after being enabled, a certain amount of time must pass before the actual firing.

Petri Nets - time Petri Nets

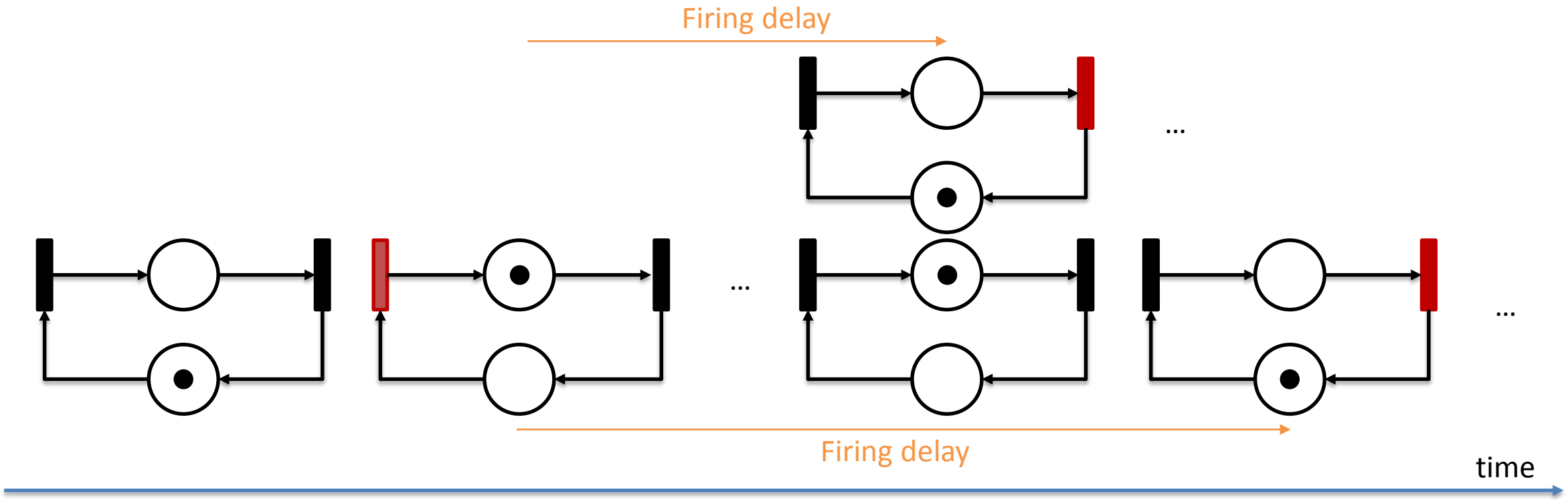
Delays attached to each transition can be:

- deterministic
- stochastic
    - with finite support;
    - exponentially distributed (Generalized Stochastic Petri Nets);
    - arbitrarily distributed.



We will cover the more general case of arbitrarily distributed (stochastic) firing delays (with finite support).

As we introduced stochastic firing delays, the marking of the net at a certain time instant *t* is no longer a deterministic quantity.



Agenda

- Introduction and motivations
- Petri Nets: a mathematical tool for discrete event systems
  - Generalities
  - Time Petri Nets
- Scheduling with time Petri Nets
  - The cost function
  - Scheduling algorithm

Scheduling with Time Petri Nets - the cost function

A scheduling algorithm can be introduced in order to minimize the following cost function:

$$J = \int_{t}^{t+T_p} w^T M(\tau)\, d\tau, \;\; w^T = \left[w_1, \ldots, w_{|P|}\right], w_i \geq 0$$

where $M(\tau), \tau \in [t, t+T_p]$ is the marking, w_i are-non negative weights.

The scheduler will decide which transitions to fire in order to minimize the total cost.

Casalino, A., Zanchettin, A.M., Piroddi, L., Rocco, P. IEEE Tran ASE (under review)

Scheduling with Time Petri Nets - the cost function

The **optimal control problem** we want to solve is therefore the following one:

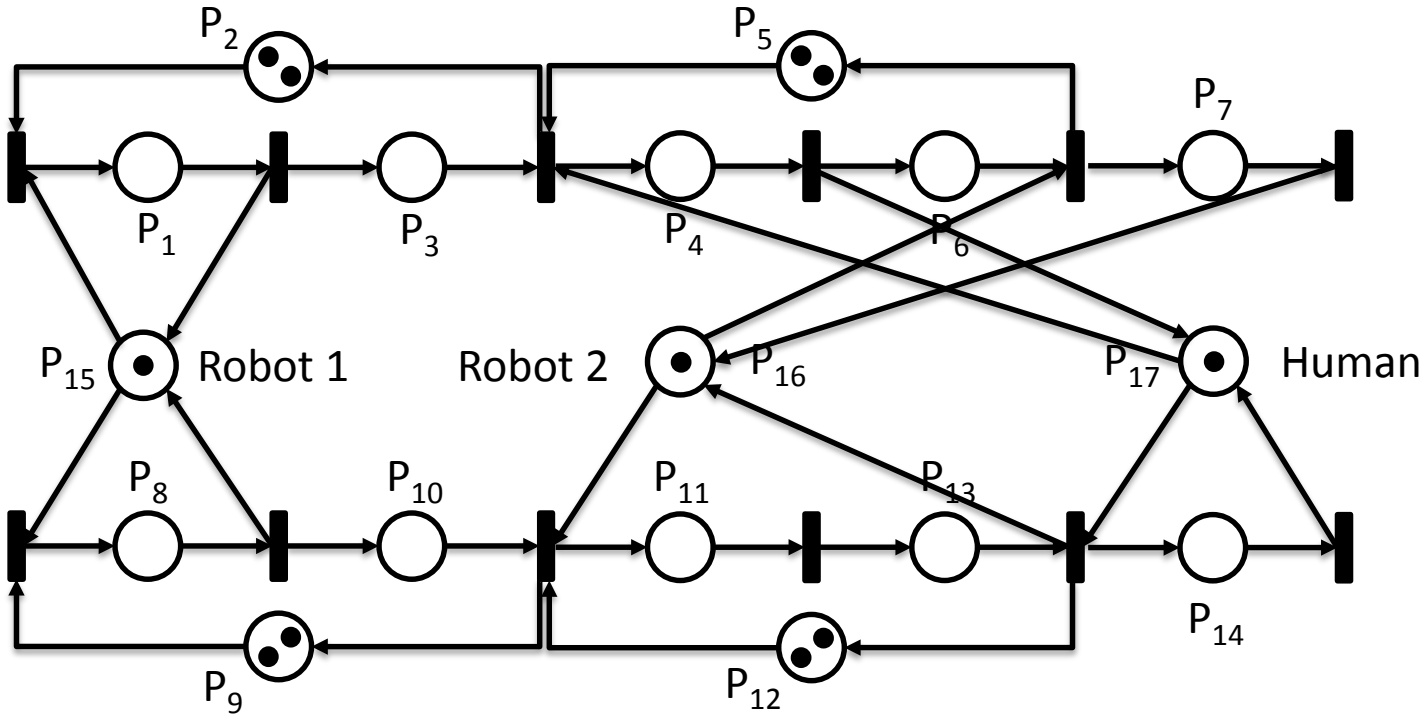
$$\min_{u(\cdot)} \int_t^{t+T_p} w^T M(\tau)\, d\tau$$

Remarks:

- the cost function is stochastic, as one possible decision may have several associated costs (one may account for that in different ways):
  - use the average value (i.e. the first moment) $\min_{u(\cdot)} \mathbb{E}\left[\int_t^{t+T_p} w^T M(\tau)\, d\tau\right]$
  - use the second moment $\min_{u(\cdot)} \mathbb{E}\left[\left(\int_t^{t+T_p} w^T M(\tau)\, d\tau\right)^2\right]$
- idle time is penalised, hence we expect the flow to be maximised.

Scheduling with Time Petri Nets - the cost function

For example, consider two jobs consisting of three sequential operations each. Operations are performed by two robots and a human operator:



Scheduling with Time Petri Nets - the cost function

The objective might be to **minimize the inactivity** of all agents, i.e.

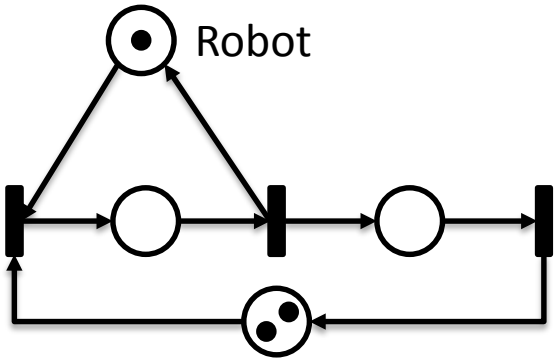
$$\min_{u(\cdot)} \int_{t}^{t+T_p} w^T M(\tau)\, d\tau \qquad w^T = \begin{bmatrix} 0 & \dots & 0 & \underbrace{1 \quad 1 \quad 1}_{\text{weights associated to } P_{15},\, P_{16},\, P_{15}} \end{bmatrix}$$

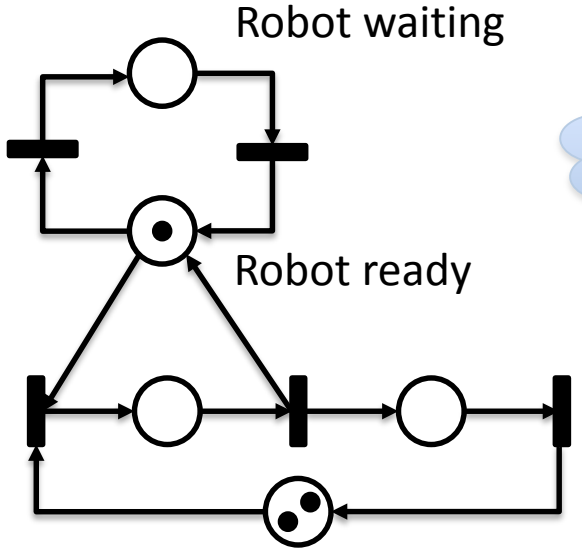
The cost function then tends to minimize situations in which the corresponding places are marked, i.e. agents are not active (idle).

Scheduling with Time Petri Nets - scheduling algorithm

In order to solve efficiently the optimal control problem, it is convenient to explicitly model the idle time as an activity with a given and fixed duration.

Therefore, we introduce the following modification:





Scheduling with Time Petri Nets - scheduling algorithm

With the previous modelling assumption, the scheduler is responsible only for

understanding which transition to fire *and no longer when*.

Weightings within the cost function are associated to places corresponding to idle agents.

We need to understand **how to actually solve** the optimal control/decision problem.

Scheduling with Time Petri Nets - scheduling algorithm

The key idea to find the solution of the optimal control is to explore all possible executions and evaluate the corresponding costs.

Hence, the scheduling algorithm will explore all possible behavior of the net accounting for:

- possible decisions to take in due to course;
- possible evolutions of the stochastic part (durations).

The mostly convenient data structure is then the **net reachability tree**. During each simulation, a benches are created whenever a conflict appears.

POLITECNICO
MILANO 1863