adesso business. people. technology.

Reinforcement Learning in a Smart Factory

SAKI Exercise 3

Jürgen Kasperek

Jan Marvin Wickert

Content

- Who we are?
- Introduction to Reinforcement Learning
- Markov Decision Process (MDP)
- Exercise: Smart Factory
- MDP Algorithms
- MDP Examples

Who we are?

People

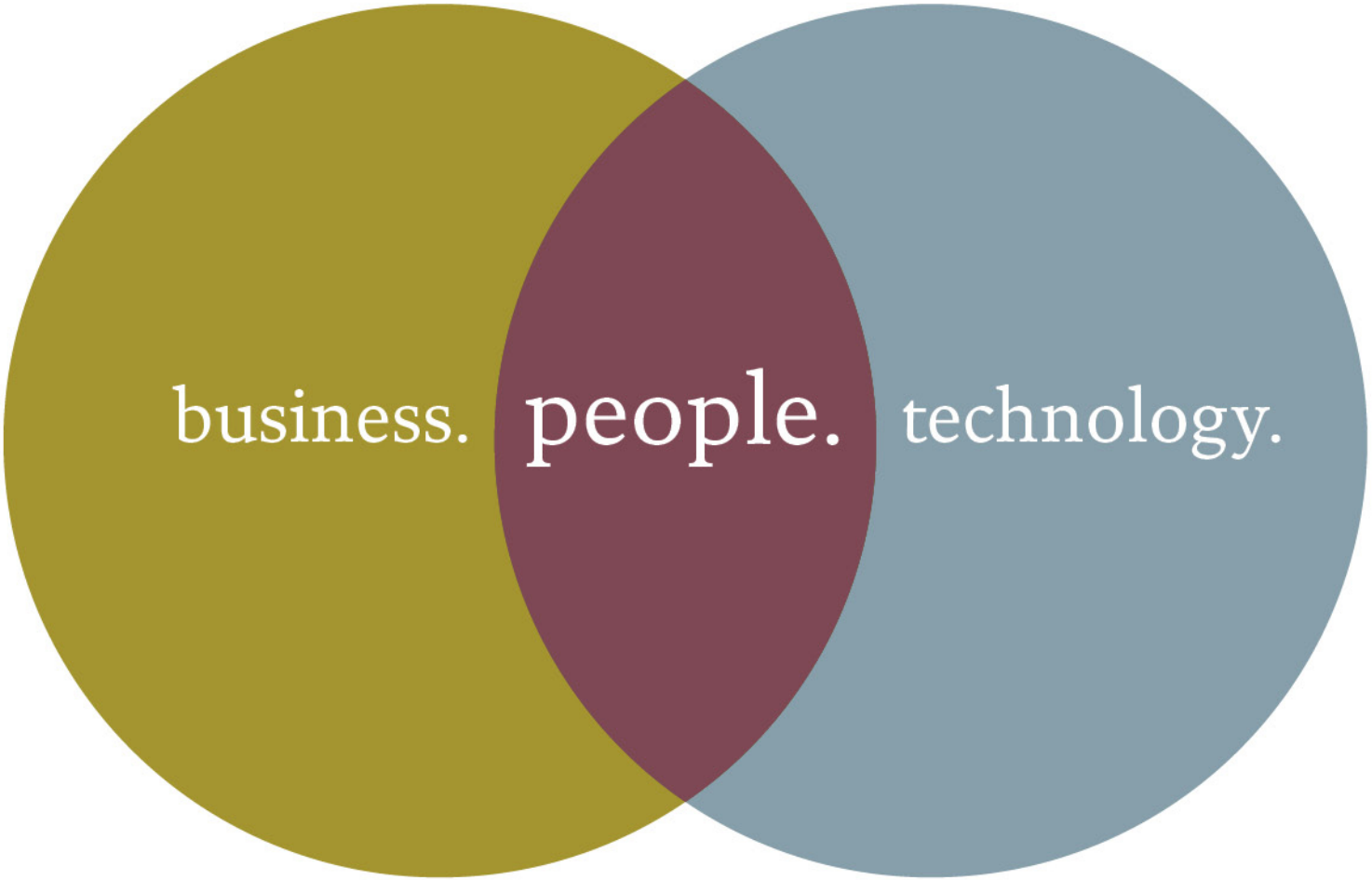
Dipl. Ing. and Dipl. Kfm. Jürgen Kasperek
Head of Competence Center IT-Consulting Manufacturing Industry

- Economics at Fernuniversität Hagen (Dipl. Kfm.)
- Electronical Engineering and Information Technology at University of Bundeswehr München (Dipl. Ing.)

M.Sc Jan Marvin Wickert
Trainee in IT-Consulting Manufacturing Industry

- Electronical Engineering and Information Technology at University Stuttgart (M.Sc.)
- Mechatronics at University of Applied Science Bochum (B.Eng.)

Philosophy at Adesso



Partner in digital transformation

adesso optimises the core business processes of companies by providing advice and developing individual software and industry solutions.

Success factors and growth

over 3.200
Employees
adesso Group

Group sales **2018**
375,5 Million €

Group sales 2016
260,4 Million €

Group sales 2017
321,6 Million €

Awards
as employer

GREAT PLACE TO WORK 2018 Beste Arbeitgeber ITK

1. Platz IN DER GRÖSSEN-KATEGORIE: > 1.000 MITARBEITENDE

GREAT PLACE TO WORK 2018 Beste Arbeitgeber Deutschland

TOP NATIONALER ARBEITGEBER 2018 FOCUS

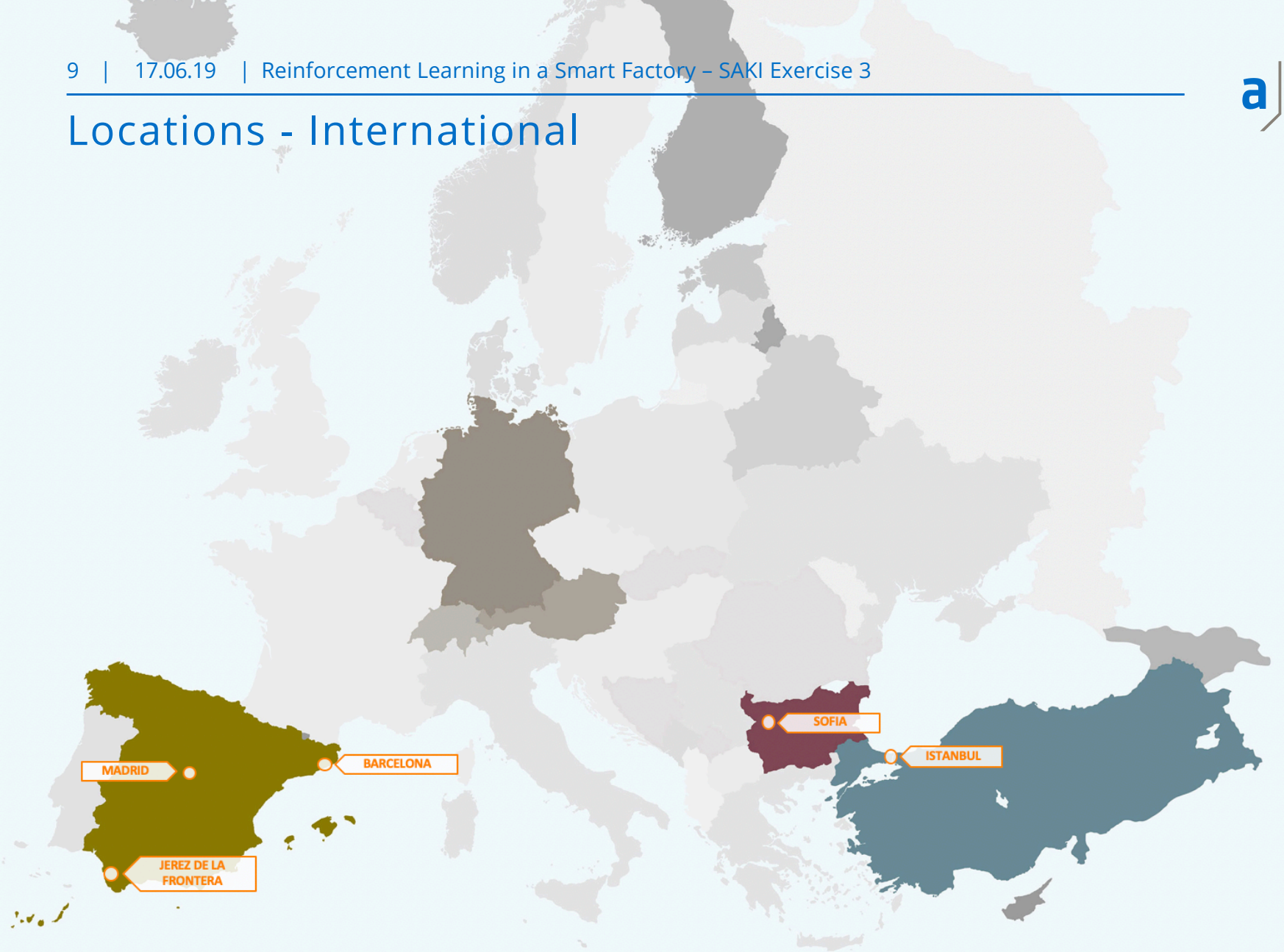
GREAT PLACE TO WORK BESTE ARBEITGEBER ITK 2016

GREAT PLACE TO WORK DEUTSCHLANDS BESTE ARBEITGEBER 2016

kununu TOP COMPANY VON MITARBEITERN EMPFOHLEN!

Locations - German-speaking area

Locations - International

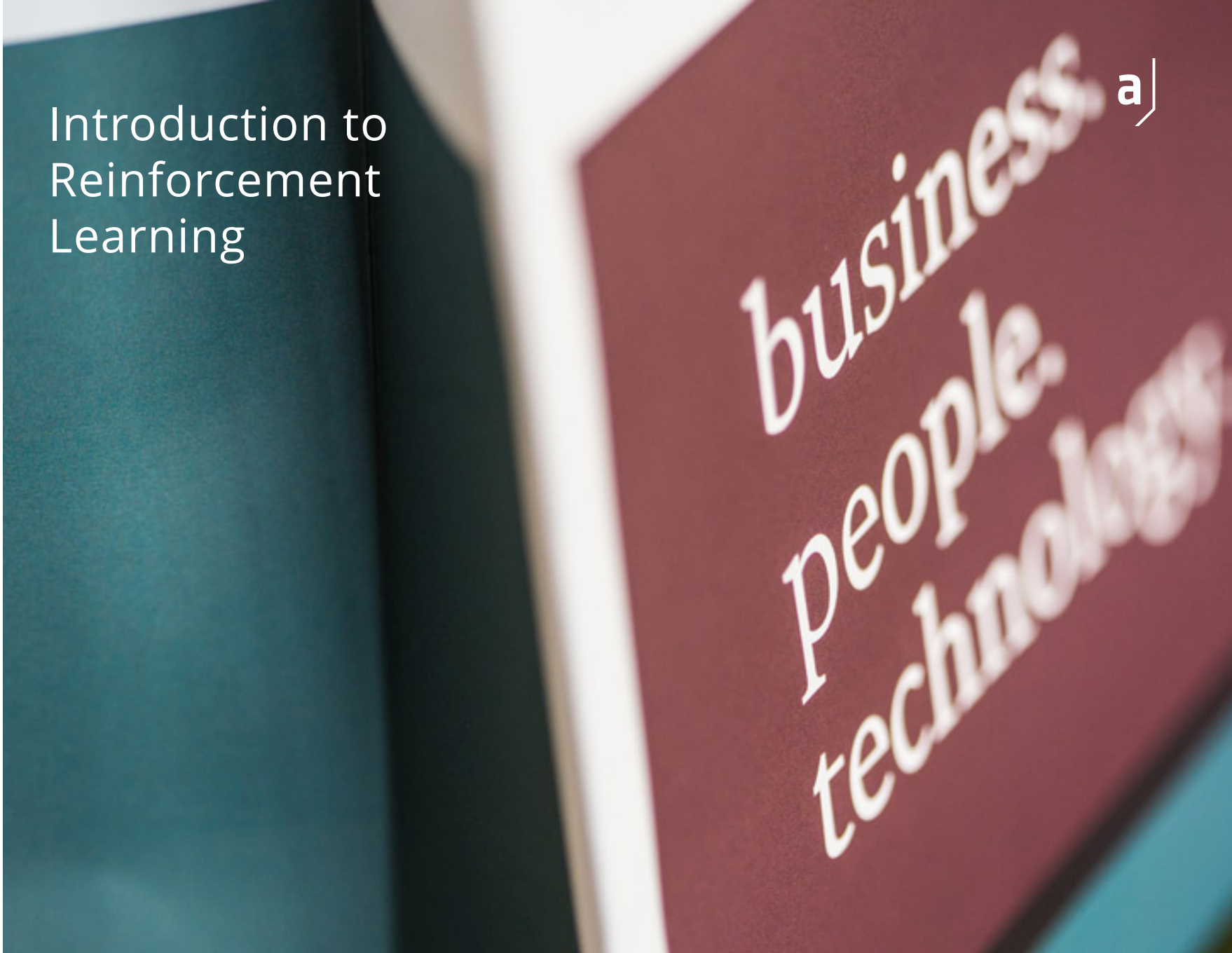


Services

>>> business

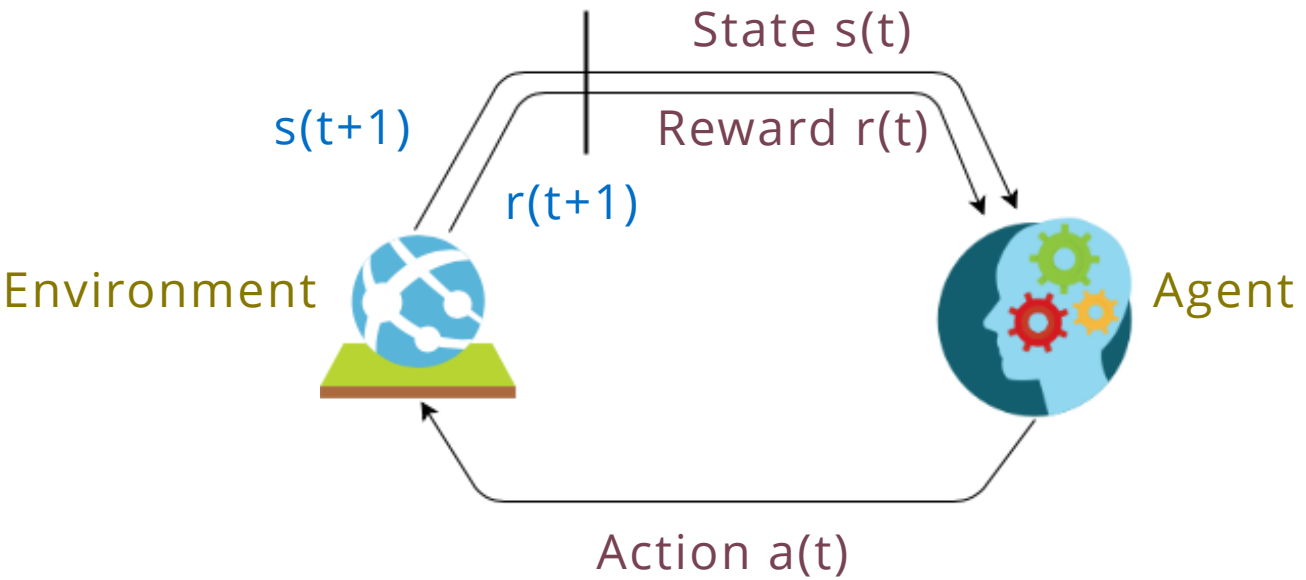
technology <<<

Introduction to Reinforcement Learning



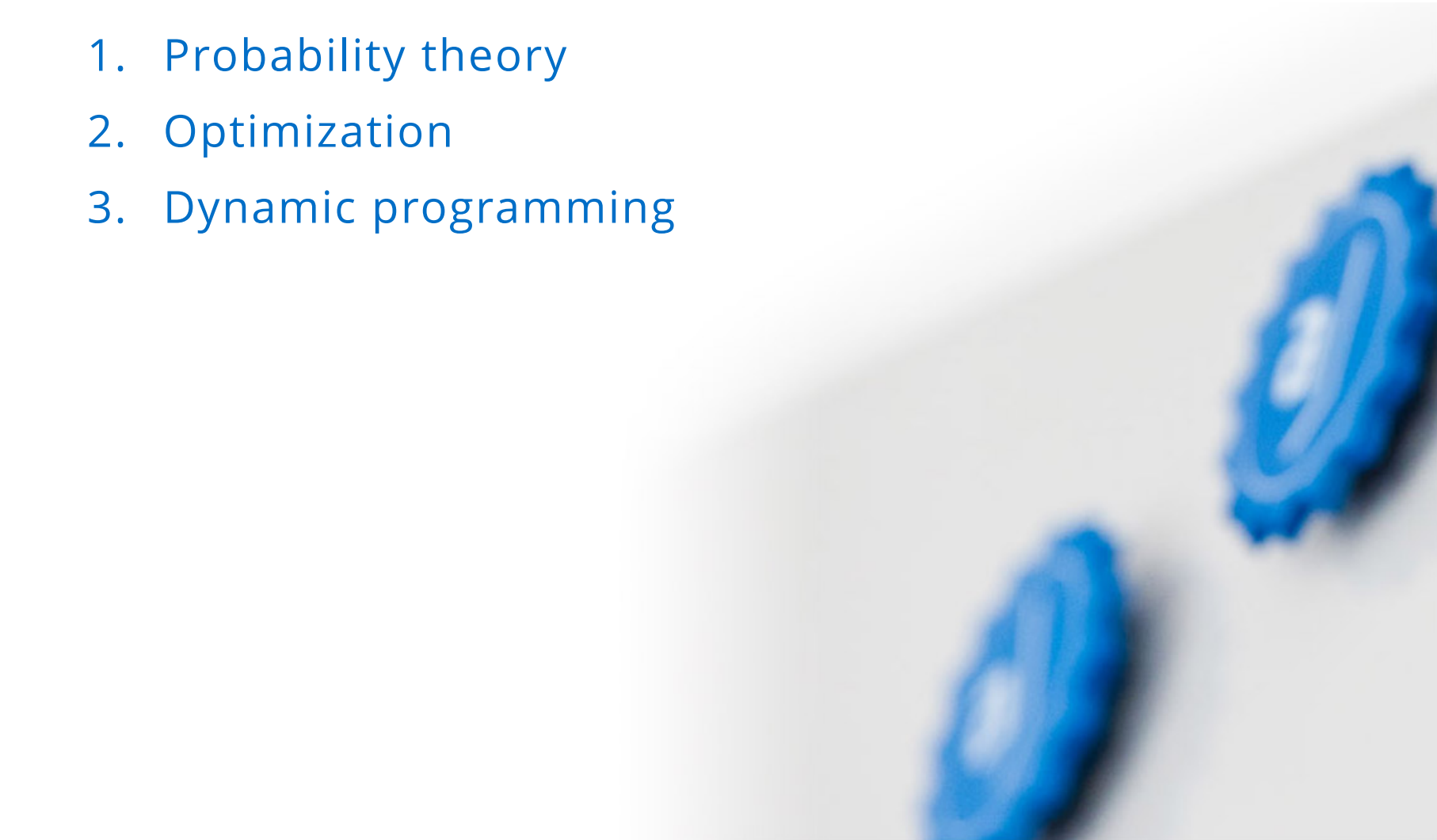
Introduction to Reinforcement Learning

"Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal" (Sutton, Reinforcement Learning: An introduction)



Background Material of Reinforcement Learning

1. Probability theory
2. Optimization
3. Dynamic programming



Background Material - Probability theory

➡ Example with dice consist out of 3 different colors

Source: www.dice-collection.com

Probability to roll red two times:

P = 1/3 * 1/3 = 1/9

Background Material - Optimization

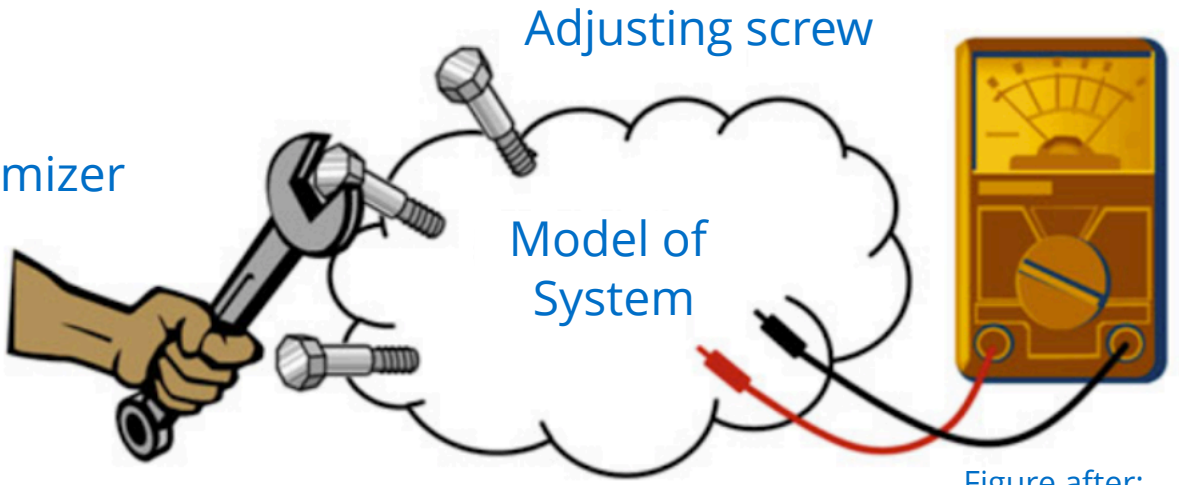


Figure after:
Christian Grimme and Jakob Bossek, Einführung in die Optimierung (2018)

Definition of optimization formula:

$f: S \rightarrow Z$

$f \rightarrow min$ or $f \rightarrow max$

$max\ f = -\ min(-f)$

$min\ f = -\ max(-f)$ <- duality principle

Background Material - Optimization

Optimization example: knapsack problem

We have a backpack with 10kg capacity.

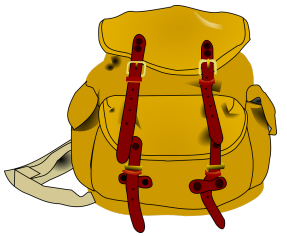
There are k elements we can take with us. Each have a weight of w(i) and a value of v(i). The goal is to maximize the value inside the backpack. For example you want to rob a jeweler and you have to decide which jewelry with a defined price you put into the bag.

1. Define the Problem mathematically

$$f_{value}(x) = v_1 \times x_1 + \cdots + v_k \times x_k$$

$$\sum_{i=1}^{k} x_i \times w_i \leq 10kg \; with \; x = (x_1, \ldots, x_k) \in S$$

Background Material - Optimization



Optimization example: knapsack problem

We have a backpack with 10kg capacity.

There are k numbers of elements we can take with us. Each have a weight of w(i) and a value of v(i). The goal is to maximize the value inside the backpack. For example you want to rob a jeweler and you have decide which jewelry with a defined price you put into the bag.

Values of items: {10, 40, 30, 50}

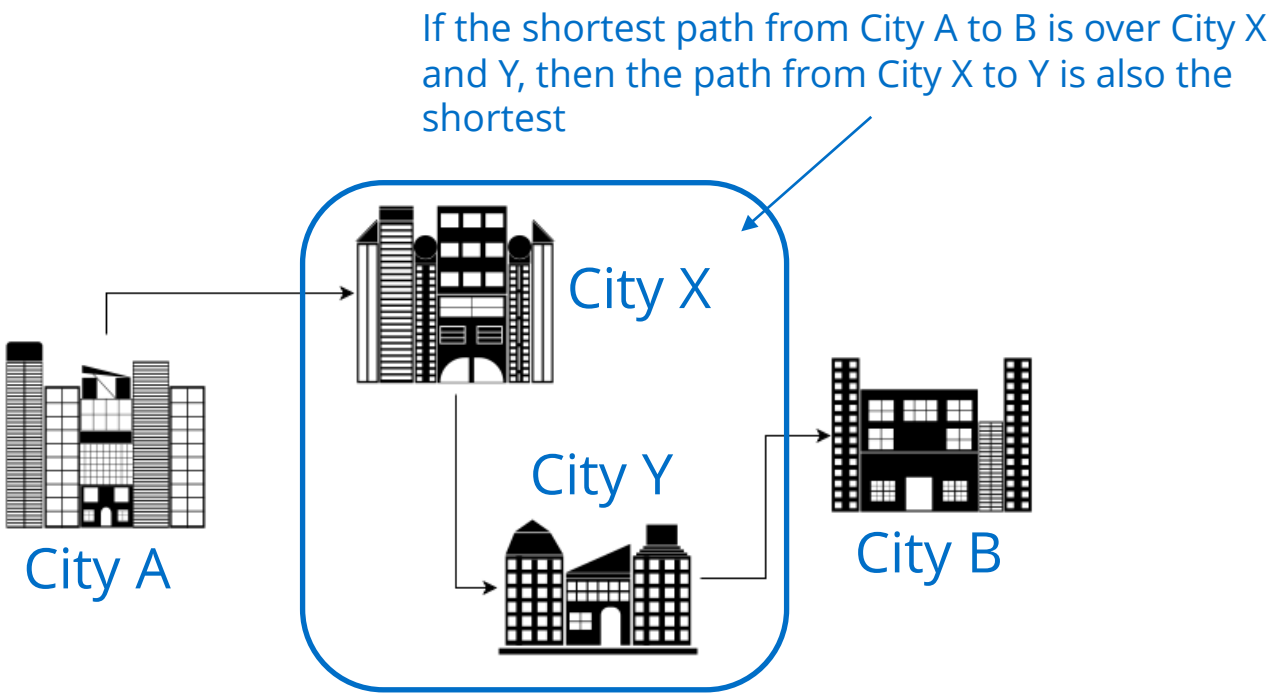
Weight of items: {5, 4, 6, 3}

Which items will you put into the backpack?

[40,4] + [50,3]

Background Material - Dynamic programming

„An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.“ – (Bellman, 1957)



Background Material - Dynamic programming

Knapsack problem with dynamic programming:

- Values of items: {10, 40, 30, 50}
- Weight of items: {5, 4, 6, 3}

How can we solve the task with dynamic programming?

(Hint: create subproblems)

	0	1	2	3	4	5	6	7	8	9	10
0 items	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
item 1	0	0	0	0	0	10	10	10	10	10	10
item 2	0	0	0	0	40	40	40	40	40	50	50
item 3	0	0	0	0	40	40	40	40	40	50	70
item 4	0	0	0	50	50	50	50	90	90	90	90

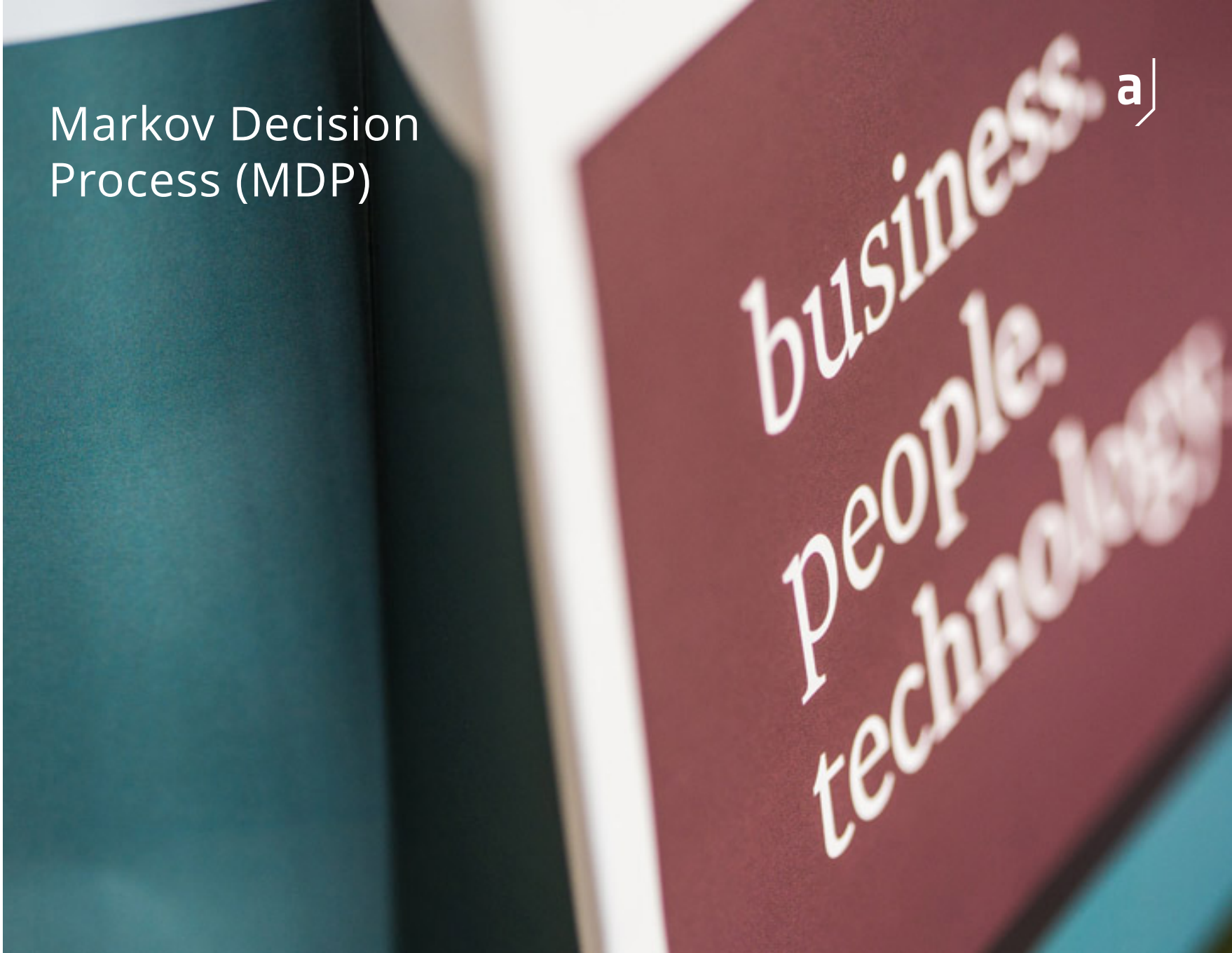
Background Material - Dynamic programming

Knapsack problem with dynamic programming:

Lets create some fancy python code!

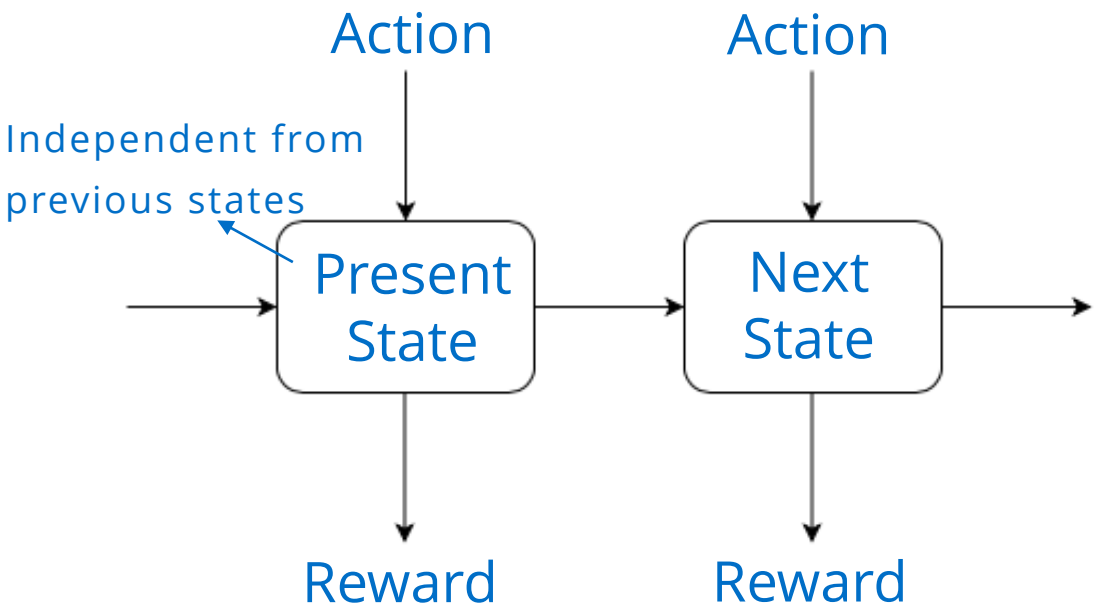
```python
def dynamicknapsack(wmax, val, wt):
    n = len(val)
    valTable = [[0 for x in range(wmax+1)] for x in range(n+1)]
    for i in range(n+1):
        for n in range(wmax+1):
            if i == 0 or n == 0:
                valTable[i][n] = 0
            elif wt[i-1] <= n:
                valTable[i][n] = max(val[i-1]+valTable[i-1][n-wt[i-1]],valTable[i-1][n])
            else:
                valTable[i][n] = valTable[i-1][n]
    print(valTable)
    return

wmax = 10
val = [10,40,30,50]
wt = [5,4,6,3]

dynamicknapsack(wmax, val, wt)
```

Markov Decision Process (MDP)



Markov Decision Process

A Makrov descision process is a discrete time stochastic control process where the markov property is satisfied



Markov Decision Process

Definition of parameters:

A MDP is a 4-tuple (S, A, P_a, R_a)

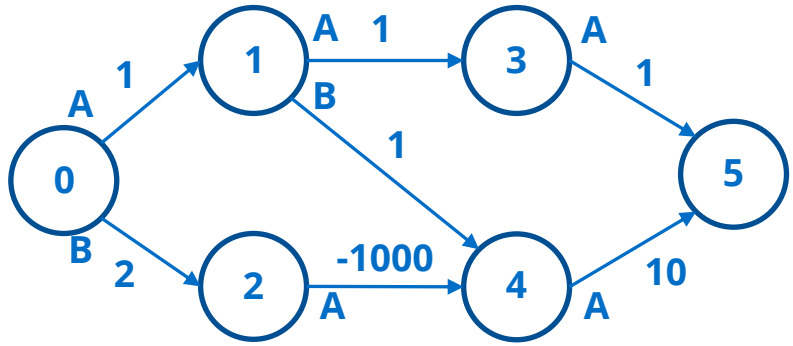
S is a finite set of states
S = $\{s_1, s_2, \ldots, s_n\}$

A is a finite set of actions A
= $\{a_1, a_2, \ldots, a_n\}$

$P_a(s, s')$ is the transition probability matrix with the probabilities to lead from state s another state s' within the action a

$R_a(s, s')$ is the reward matrix receiving a defined reward after action a in state s to reach state s'

Markov Decision Process – The policy



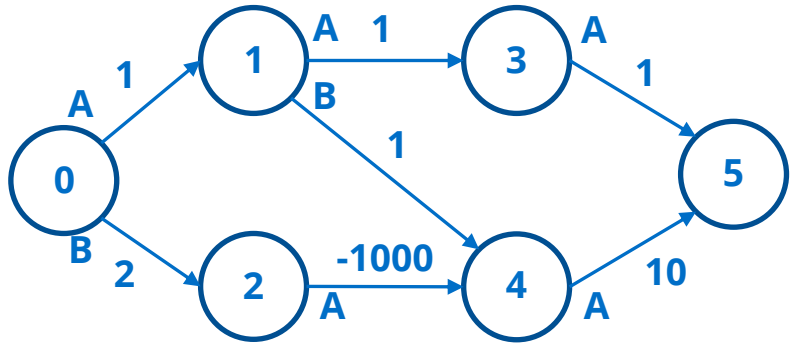
The path example:

- States: S = {0,1,2,3,4,5}
- Actions: A = {A,B}
- Transition probabilitymatrix? Rewardmatrix?

Probabilitymatrix for action A:

	0	1	2	3	4	5
0	**0**	**1**	**0**	**0**	**0**	**0**
1	0	0	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	0	1
4	0	0	0	0	0	1
5	0	0	0	0	0	1

Markov Decision Process – The policy



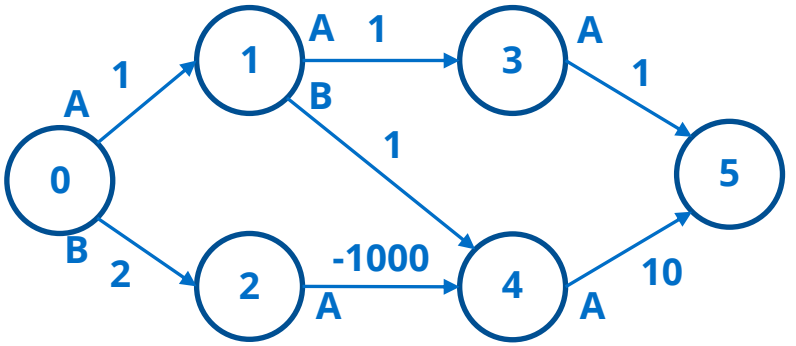
The path example:

- States: S = {0,1,2,3,4,5}
- Actions: A = {A,B}
- Transition probabilitymatrix? Rewardmatrix?

Probabilitymatrix for action B:

	0	1	2	3	4	5
0	**0**	**0**	**1**	**0**	**0**	**0**
1	0	0	0	0	1	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

Markov Decision Process – The policy



The path example:

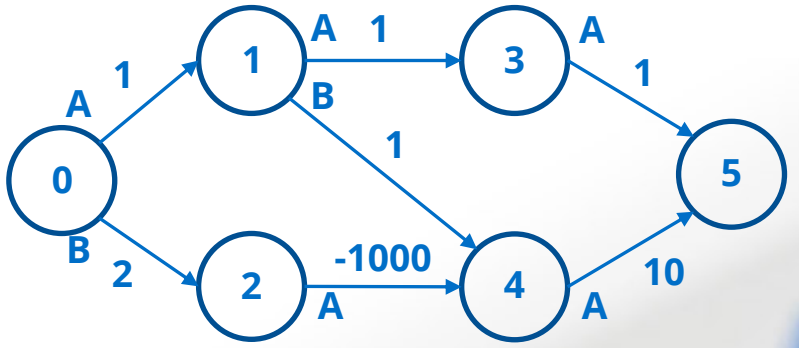
- States: S = {0,1,2,3,4,5}
- Actions: A = {A,B}
- Transition probabilitymatrix? Rewardmatrix?

Rewardmatrices:

	Action A
0	**1**
1	1
2	-1000
3	1
4	10
5	0

	Action B
0	**2**
1	1
2	0
3	0
4	0
5	0

Markov Decision Process – The policy



The path example:

- States: S = {0,1,2,3,4,5}
- Actions: A = {A,B}

There are 3 policies for this MDP:

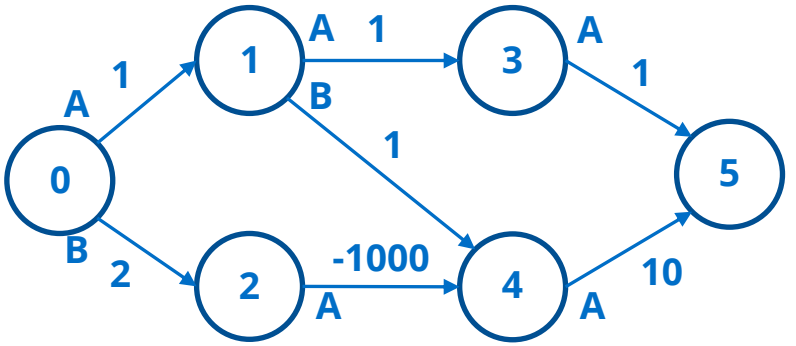
1. 0->1->3->5 = 1+1+1 = 3
2. 0->1->4->5 = 1+1+10 = 12 ✓
3. 0->2->4->5 = 2-1000+10 = -988

Which one is the best?

Markov Decision Process – Value functions

The path example:

- States: S = {0,1,2,3,4,5}
- Actions: A = {A,B}



The state value function V describes a value for each state with a given policy

$V_1\ (s_0) = 3$ $V_1\ (s_1) = 2$ $V_1\ (s_3) = 1$

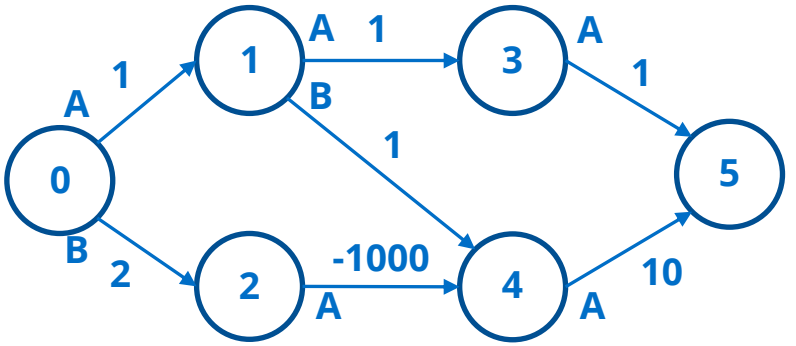
$V_2\ (s_0) = 12$ $V_2\ (s_1) = 11$ $V_2\ (s_4) = 10$

$V_3\ (s_0) = -988$ $V_3\ (s_2) = -990$ $V_3\ (s_4) = 10$

Markov Decision Process – Value functions

The path example:

- States: S = {0,1,2,3,4,5}
- Actions: A = {A,B}



It is possible to define a value without the policy in the following way:

Q(0,A) = 12
Q(0,B) = -988

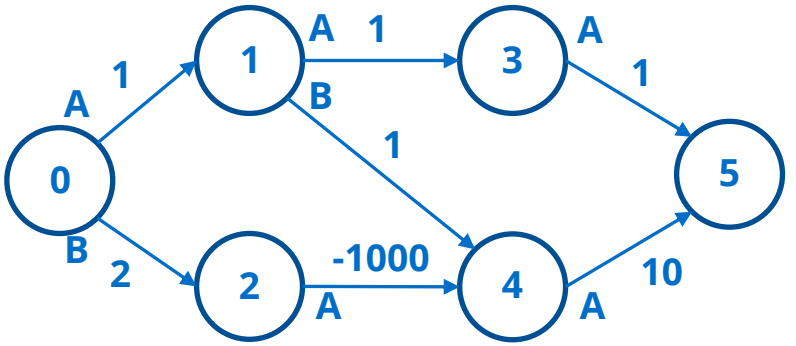
Q(1,A) = 2
Q(1,B) = 11
Q(2,A) = -990

Q(3,A) = 1
Q(4,A) = 10

Markov Decision Process – Discount factor

The path example:

- States: S = {0,1,2,3,4,5}
- Actions: A = {A,B}

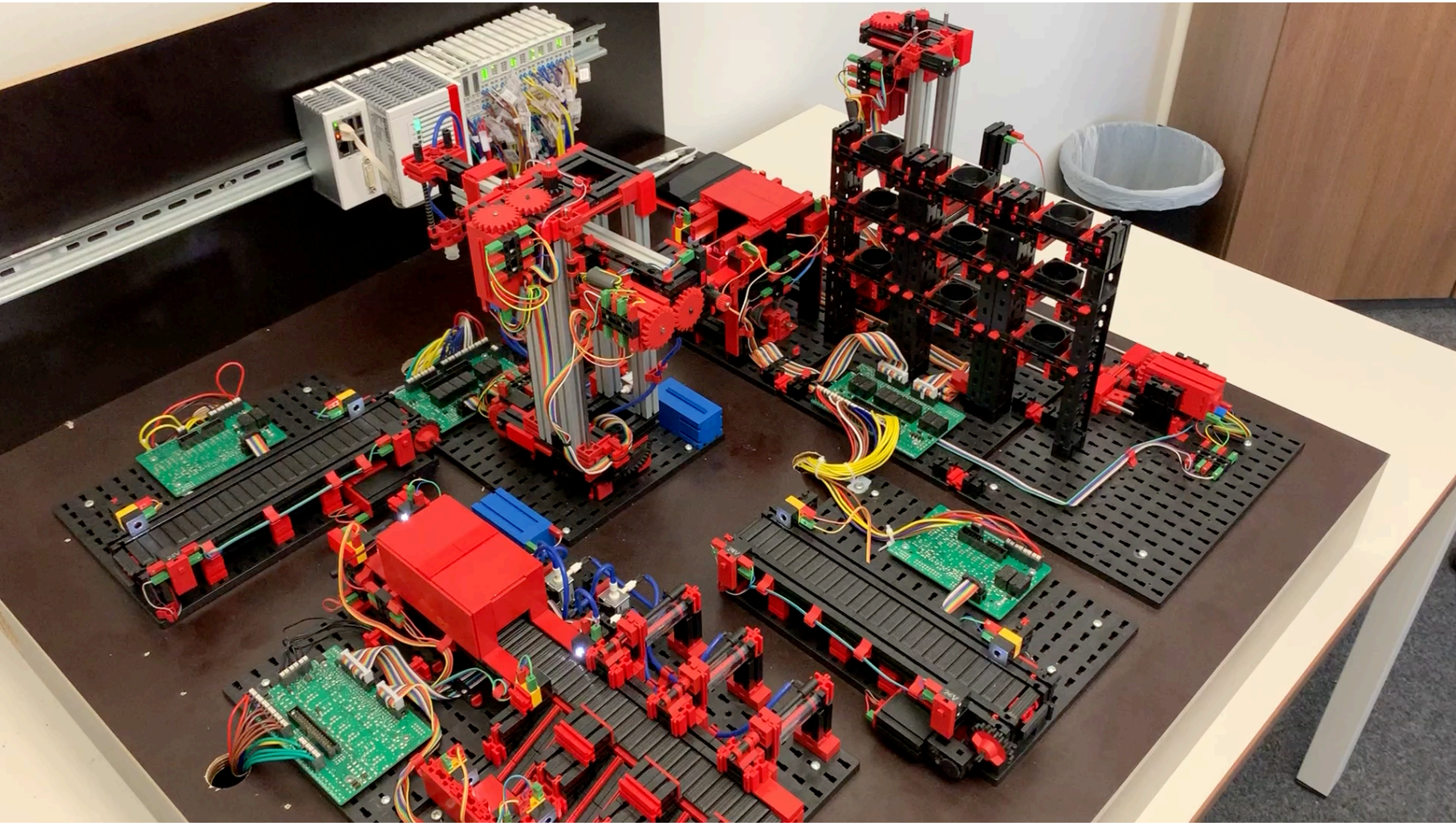


The discount factor γ is used to discount the reward for the future that infinite MDPs are solveable!

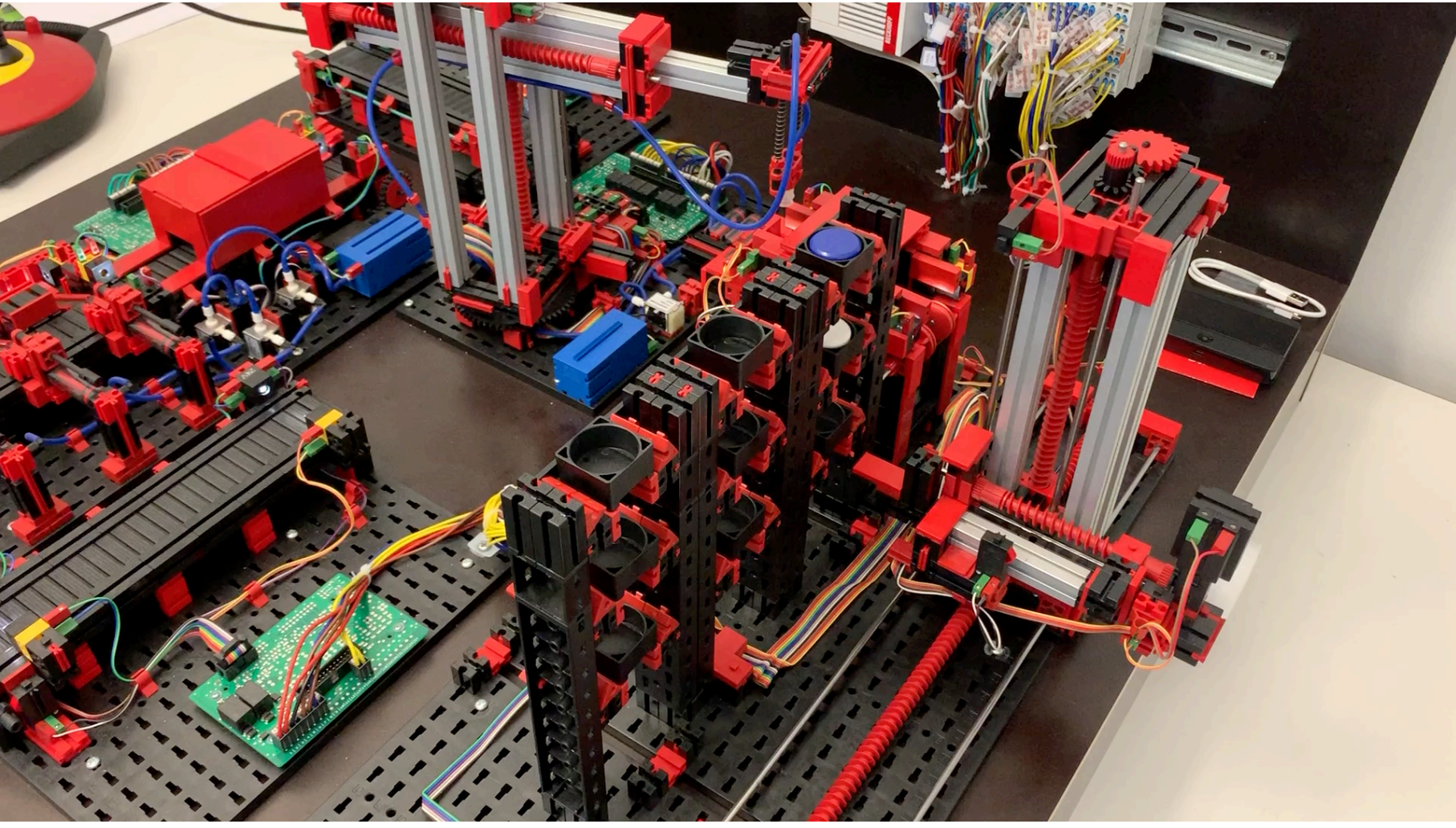
$$V^*(s) = max_a\left[R(s,a) + \gamma \sum_{s'} P(s'|s,a)V^*(s')\right]$$

Exercise:
Smart Factory

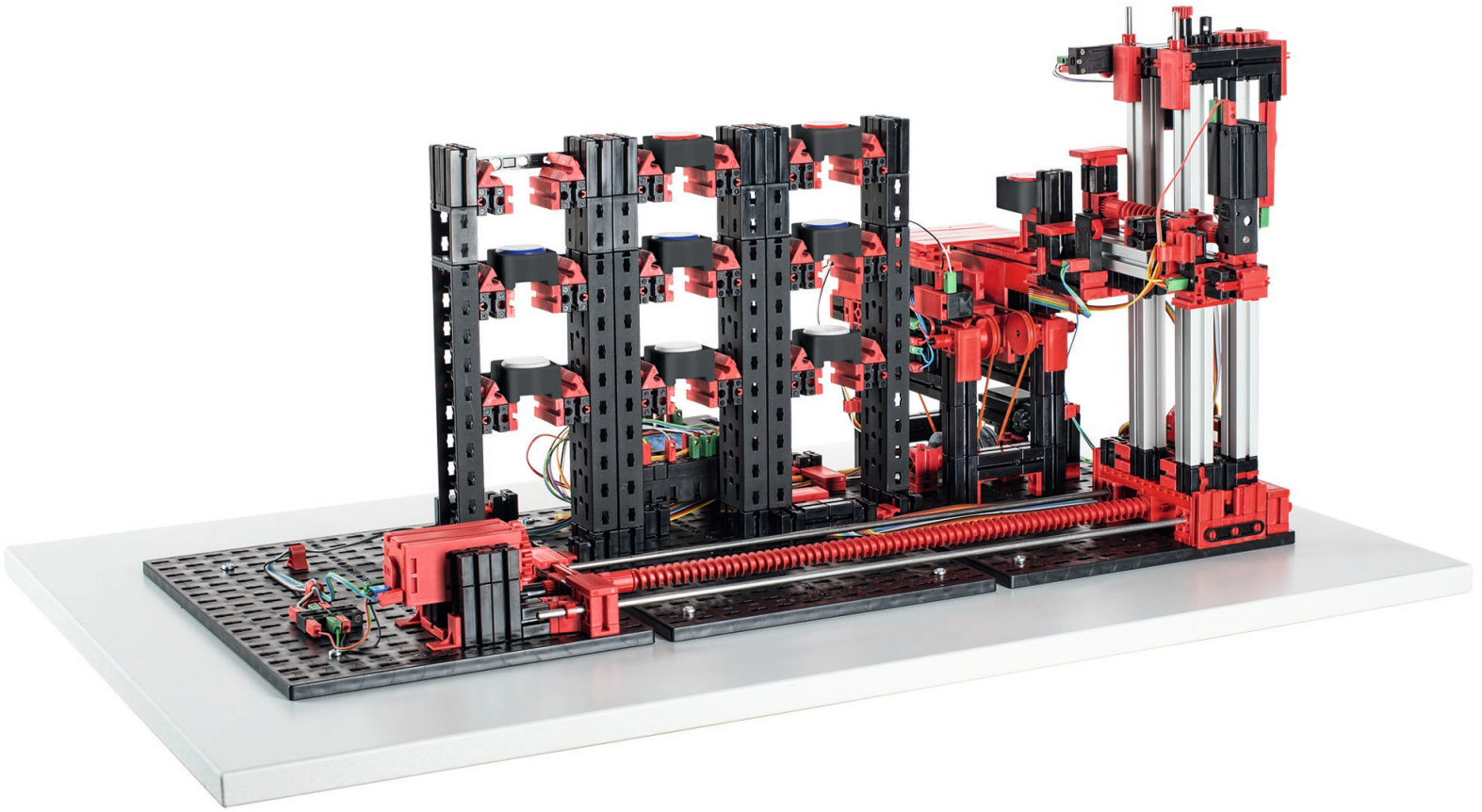
Exercise: Smart Factory



Exercise: Smart Factory



Exercise: Smart Factory

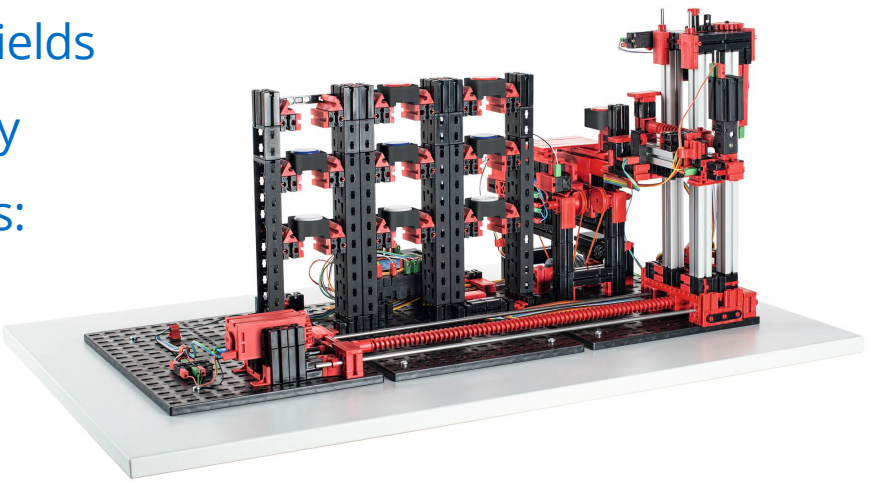


Goal: Optimization of the robots route for pick-up and storage of items in a warehouse

Exercise: Smart Factory

Optimize the route with following constraints:

- Size of warehouse is {1..3} x {1..3}
- Separate start/stop position outside the 3x3 storage space where the robot have to go at the end of storage and pick-up
- The first position the robot can move into is always (1,1)
- Robot can move to adjacent fields
- Robot cannot move diagonally
- There are three types of items: (white, blue, red)

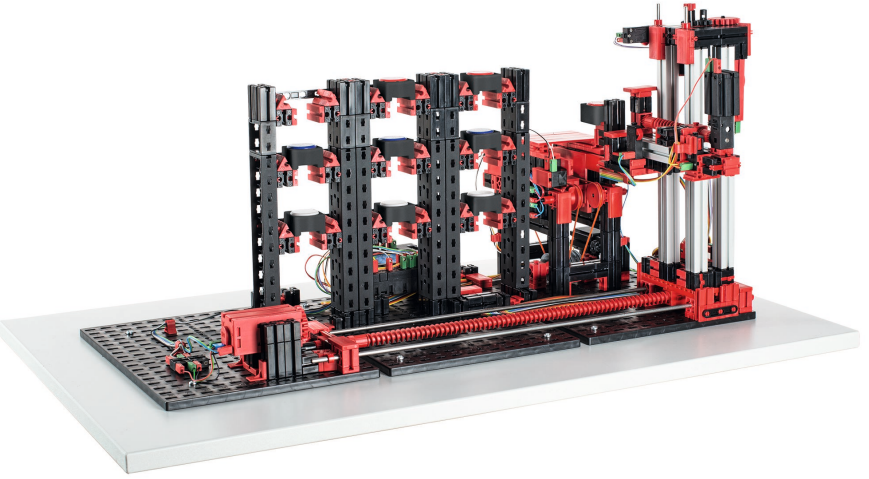


Exercise: Smart Factory

Approch:

- Implement a reinforcement-learning-based algorithm
- The robot is the agent and decides where to place the next part
- Use the markov decision process toolbox for your solution
- Choose the best performing MDP

- Hints:
  - Define the states (1572864 states)
  - The distance influences the reward



Thank you!

adesso business. people. technology.

adesso AG
Adessoplatz 1
D-44269 Dortmund
T +49 231 7000-7000
F +49 231 7000-1000
www.adesso.de