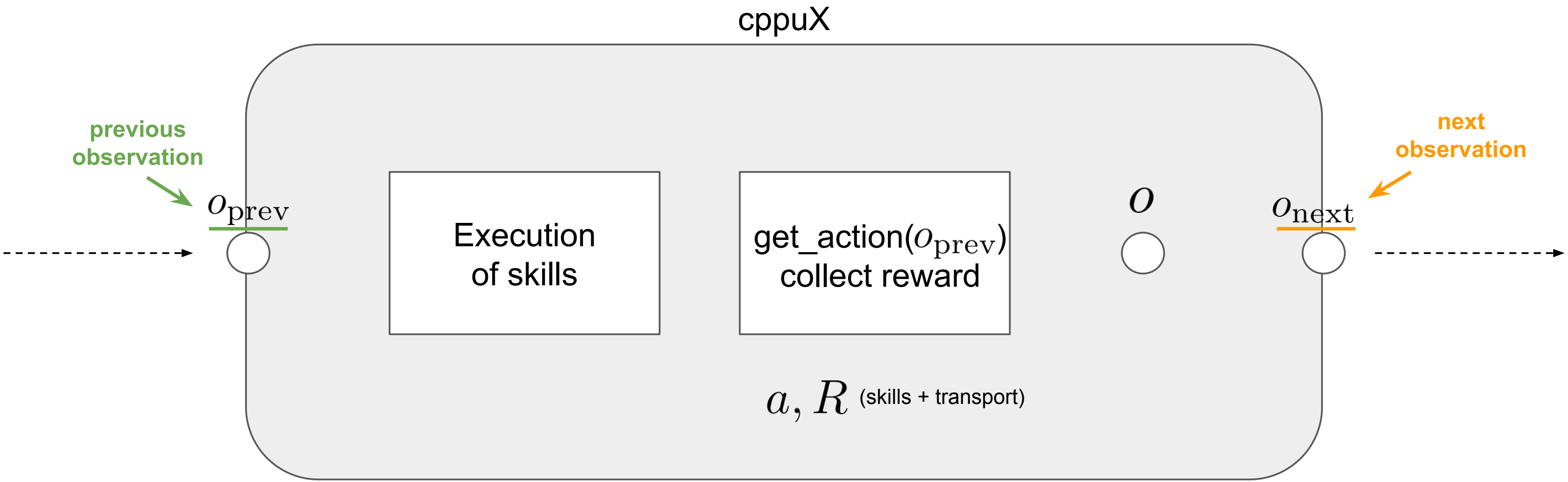
New observations

Agent-environment interface



Q-learning update

$$Q(\underline{o_{\text{prev}}}, a) = \cdots + \alpha \Big[R(\underline{o_{\text{prev}}}, \pi(\underline{o_{\text{prev}}})) + \gamma \max_{a'} Q^{\text{next}}(\underline{o_{\text{next}}}, a') \Big]$$

All the fields are computed from Skill History except counter

previous observation

next skill | product name | remaining skills to be executed | counter | previous cppu

o_{prev}

```
('supply', 'Product0', [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0], 0, 'cppu0')
```

get/update on whiteboard

SAME

o

```
('fasten', 'Product0', [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 0, 'cppu0')
```

next observation

o_{next}

```
('fasten', 'Product0', [0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 0, 'cppu1')
```

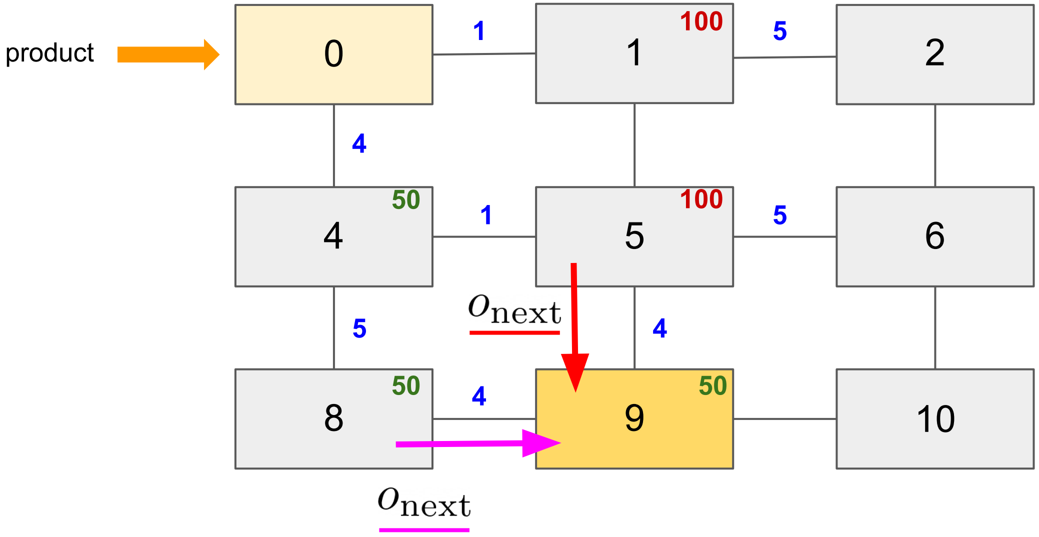
self.cppu_name

from whiteboard of next cppu

New end episode

in Q-learning

Q-learning: New End episode



FINAL AGENT

function end_episode

cppu9: $Q(o_{\text{prev}}, :) = [\dots 0 \dots]$

o_{next} o_{next}

SECOND-TO-LAST AGENTS

Once end_episode has been called at least once on the last agent

cppu8: $Q(\dots) = \dots + \cancel{Q^{\text{next}}(o_{\text{next}}, a')}$

cppu5: $Q(\dots) = \dots + \cancel{Q^{\text{next}}(o_{\text{next}}, a')}$

Initialization

in Q-learning

Q learning init.

By definition $$\frac{R_{\min}}{1-\gamma} \leq Q \leq \frac{R_{\max}}{1-\gamma}$$

Q-learning update $$Q = (1-\alpha)Q^{-} + \alpha Q^{+}$$
$$\alpha \in (0,1)$$

Optimistic init. (spontaneously encourages exploration)

The maximum Q computed according to the Q-learning update must always be smaller than Q_opt

$$\underbrace{(1-\alpha)Q_{\text{opt}} + \alpha(R_{\max} + \gamma Q_{\text{opt}})}_{\min Q} \leq Q_{\text{opt}} \implies Q_{\text{opt}} \geq \frac{R_{\max}}{1-\gamma}$$

Pessimistic init. (spontaneously discourages exploration)

The minimum Q computed according to the Q-learning update must always be larger than Q_pess

$$\underbrace{(1-\alpha)Q_{\text{pess}} + \alpha(R_{\min} + \gamma Q_{\text{pess}})}_{\max Q} \geq Q_{\text{pess}} \implies Q_{\text{pess}} \leq \frac{R_{\min}}{1-\gamma}$$

Decreasing learning rate

Learning rate

Learning rate = $\frac{\alpha}{T^d}$

History-based LR

T = number of interactions the agent has had with the environment (<u>reset</u> at the end of each episode)

Transition-based LR

T = counter of how many times the agent has seen the tuple (observation, action) for which is going to update the Q value (<u>no reset</u> at the end of the episode)

With $\alpha = 0.7$

$d = 0.3 \implies [0.7, 0.57, 0.50, 0.46, 0.43, 0.41, 0.39, 0.37, 0.36, 0.35]$ <1e-3 with T ~ 2000

$d = 1/0.8 \implies [0.7, 0.29, 0.18, 0.12, 0.09, 0.07, 0.06, 0.05, 0.04]$ <1e-3 with T ~ 200

$d = 2 \implies [0.7, 0.17, 0.08, 0.04, 0.03, 0.02, 0.01, \ldots]$ <1e-3 with T ~ 30

Reward

In all the experiments:

- After the cppu selects an action, the reward is computed as:

  -1 * duration of skills executed by itself (if any) + transport (related to the action chosen)

In some experiments:

- **positive reward for skills** means that if a skill is executed the reward is computed as before but converted to positive (by adding a scale factor of +200)

  -1 * duration of skills executed by itself (if any) + transport (related to the action chosen) **+ 200**

Code

Main changes

- created script for hyperparameter tuning
  - Launch multiple experiments (grid search)
  - Automatically set the desired scenario
  - Can be used with local as well as RLlib algorithms
- created functions to compute previous/next observations
- exported skill history to save the full path of each episode (training/evaluation)
- created python script for estimating skill durations

Q-learning

- introduced softmax action selection
- updated reward
  - Negative duration of the transport plus the duration of the executed skills (if any)
- introduced end episode

RLlib

- updated compatibility layer
  - Automatically launch and connect to the servers
  - Synchronize start/end episode
- updated reward
  - based on the full trajectory

Experiments on Scenario 1-2-3

Q-learning with epsilon-greedy policy

- ai_optimizer → riccardo/test-reprodubility-rllib → New counter for the decreasing learning rate (28 Jul 2023 15:16 cf695f9f)
- skill_runtime → master → Logging level cleanup (22 Jun 2023 14:19, d61f733f) ----> **skill runtime v. 1.68**
- skill-runtime-common → master → Logging Level Cleanup: Moving log about the number of loaded libraries to TRACE level (20 Jun 2023 16:52, bb082780)

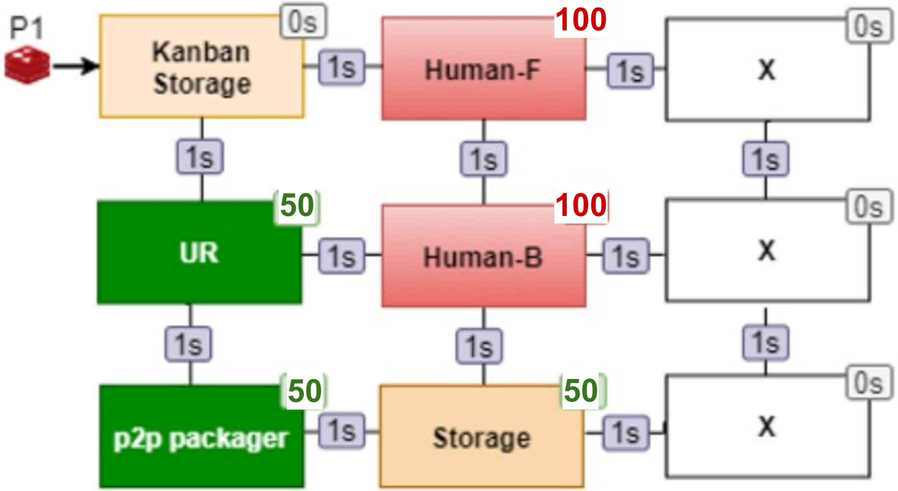
A few technical notes on the following plots

- Training/Evaluation curves
    - y-axis = duration of the entire production (mean over multiple runs with 90% confidence interval)

- 4/5 run in the legend means 4 runs out of 5 produced the final output.
    - Unless explicitly stated, this means that the execution of 1 run failed for some reasons not related to the training process (e.g., the simulation didn’t start properly or was randomly stopped)

- The baseline is the minimum-hop policy

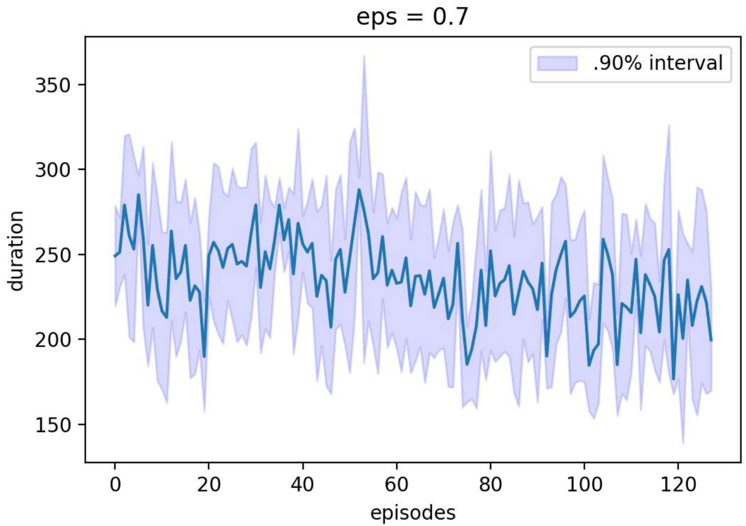
Scenario 1

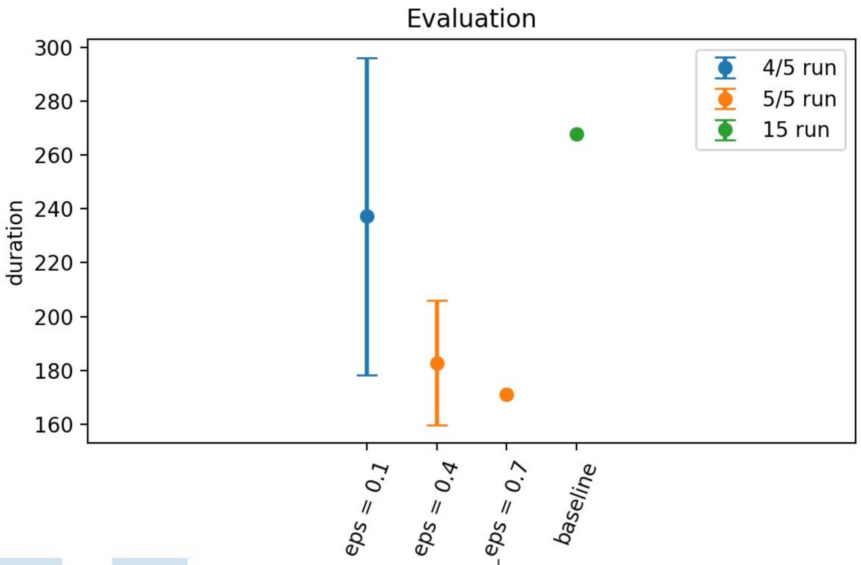


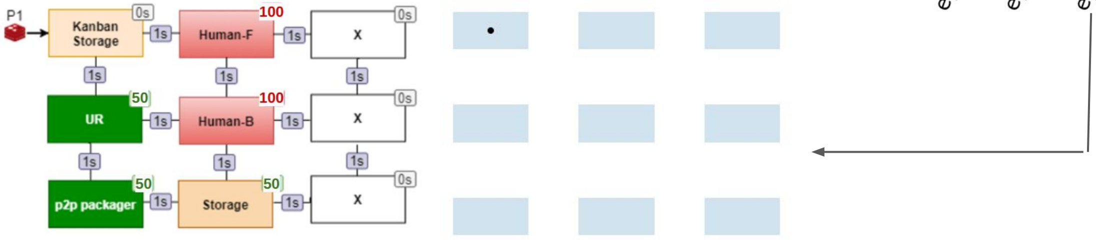
Modified duration of skills: **50**s or **100**s
All the transport are in the range 1-5s

Scenario 1

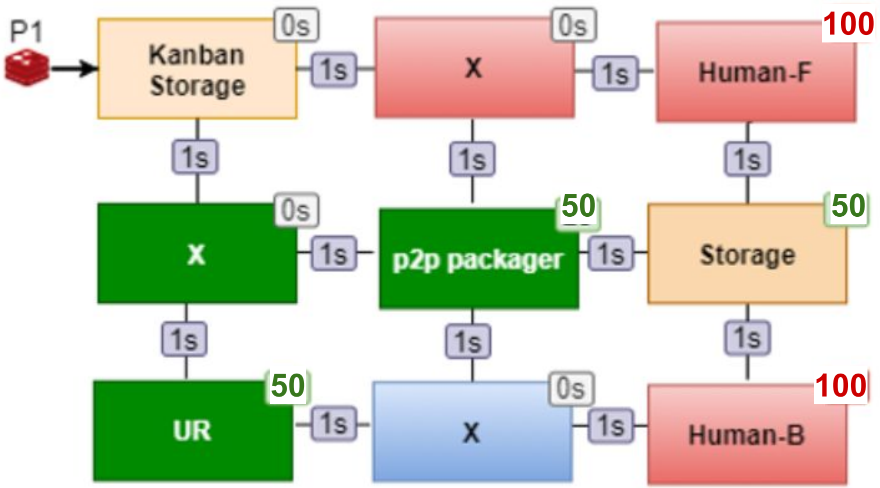
Constant learning rate= 0.7 | gamma = 0.98 | q_init = -10000 | **variable epsilon** | 128 episodes | no counter in obs.







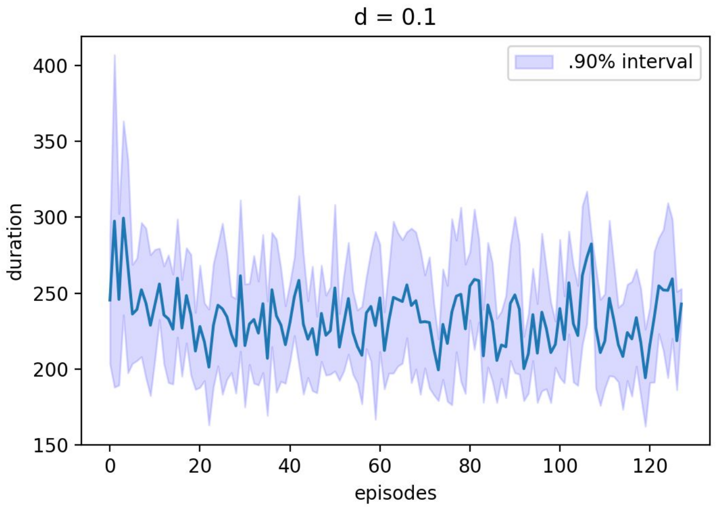
Scenario 2

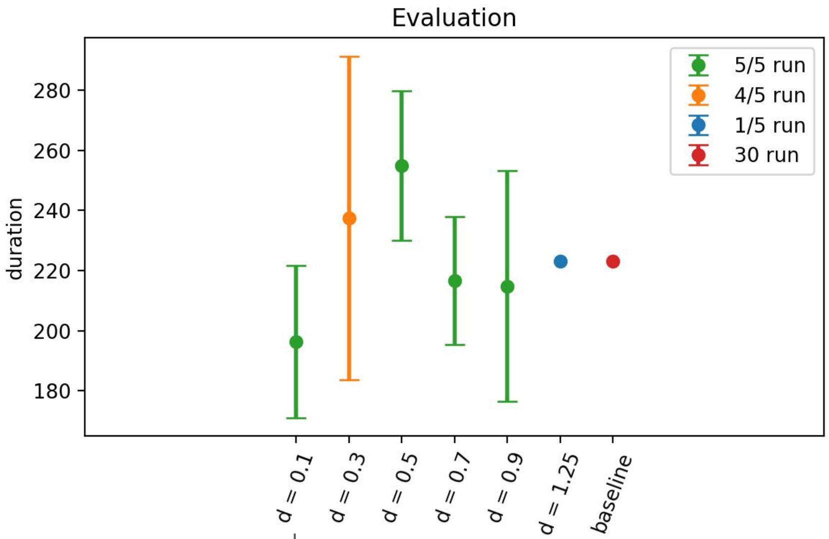


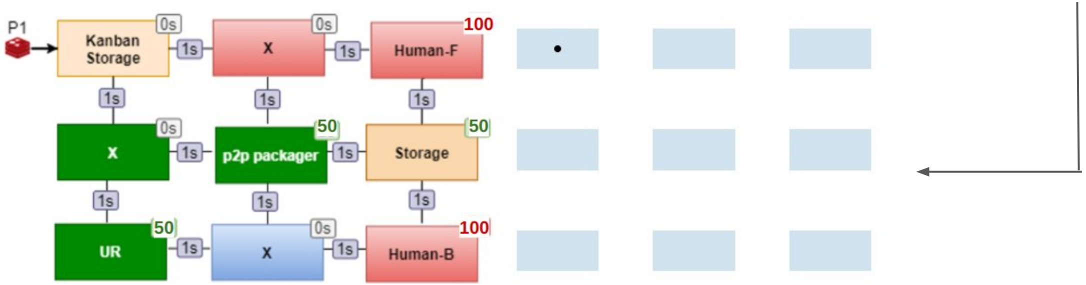
Modified duration of skills: **50**s or **100**s
All the transport are in the range 1-5s

Scenario 2

Transition-based learning rate: alpha = 0.7, **variable d** | gamma = 0.98 | q_init = -10000 | **epsilon = 0.7** | 128 episodes | **positive reward for skills**

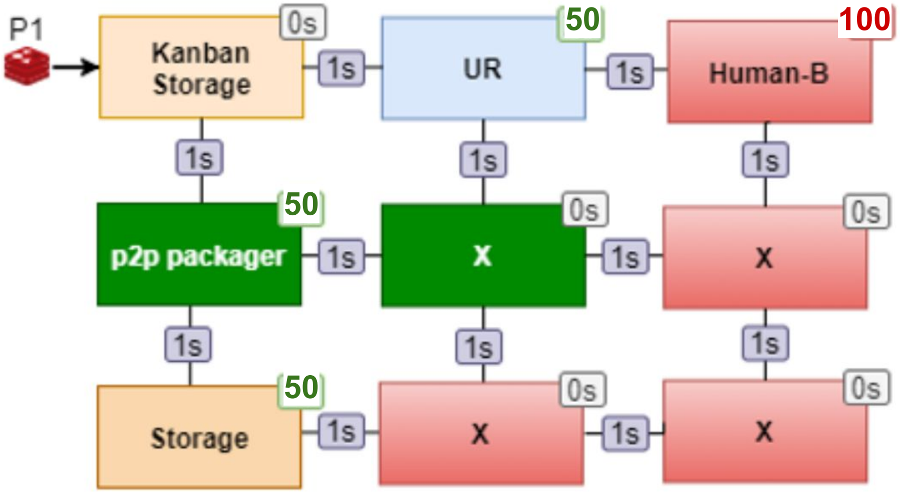






no infinite loop
COMPARED TO SLIDE 26
(just simulation errors)

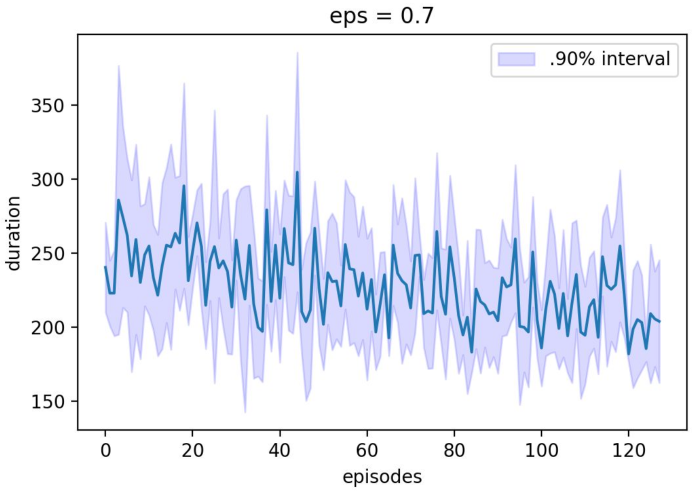
Scenario 3

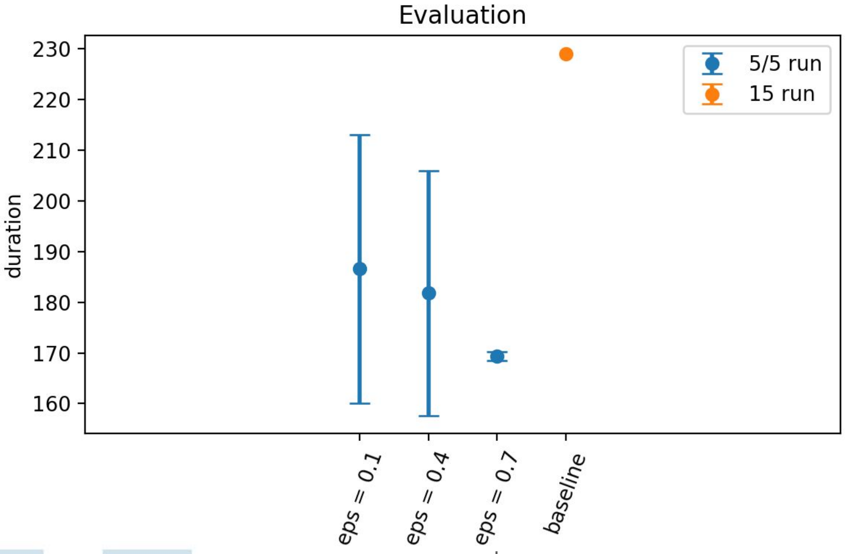


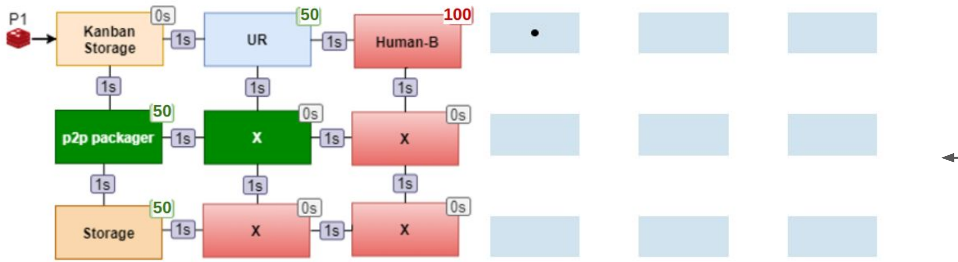
Modified duration of skills: **50**s or **100**s
All the transport are in the range 1-5s

Scenario 3

Constant learning rate= 0.7 | gamma = 0.98 | q_init = -10000 | **variable epsilon** | 128 episodes | **positive reward for skills**







no infinite loop

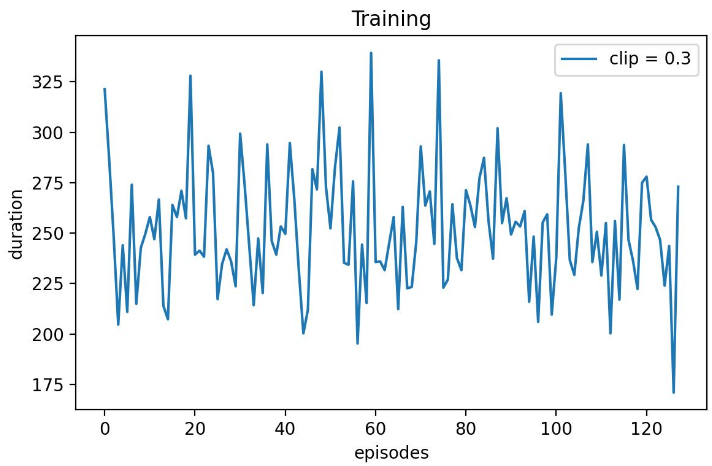
COMPARED TO SLIDE 31

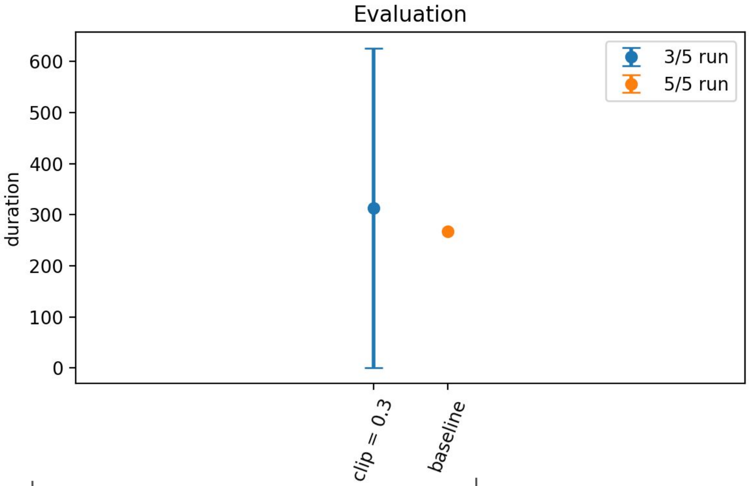
Avg time of experiments

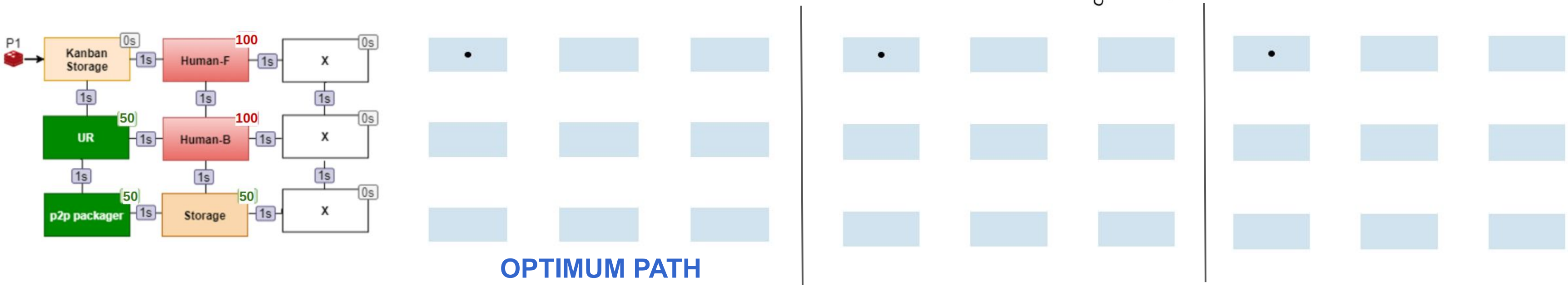
Algorithm	N. of episodes	Avg. time	Avg. time per episode
Q-learning	128	~10 m	~5 s

Experiment with PPO

- Reward is computed once the full trajectory has been collected
- This is a preliminary test (just a few episodes)







Final notes

- Refined Q-learning shows promising results in all the 3 scenarios
  - Scenario 1: baseline is most of the time outperformed, in some cases reaching the optimum path
  - Scenario 2: baseline is sometimes outperformed, but in some cases infinite loops are generated
  - Scenario 3: baseline is most of the time outperformed, but in some cases infinite loops are generated

Next steps

- Experiments with RLlib algorithms (e.g., PPO, DQN)
- Code refactoring
- Simpler output
- Reproducibility of experiments
- Save/Checkpoint models
- Migration to the production of multiple products