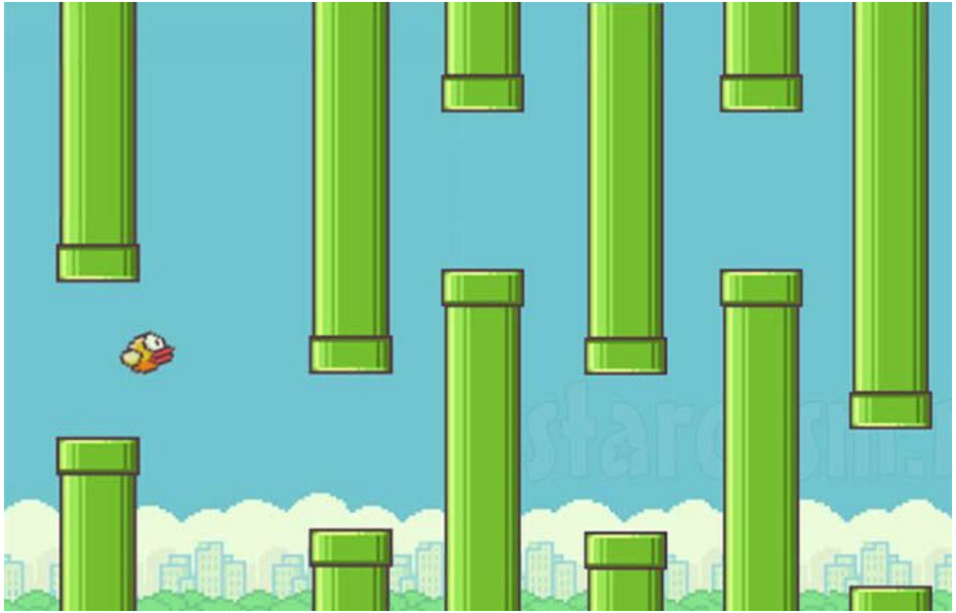
Multi-dimensional Autoscaling: A Comparison of Agent-based Methods

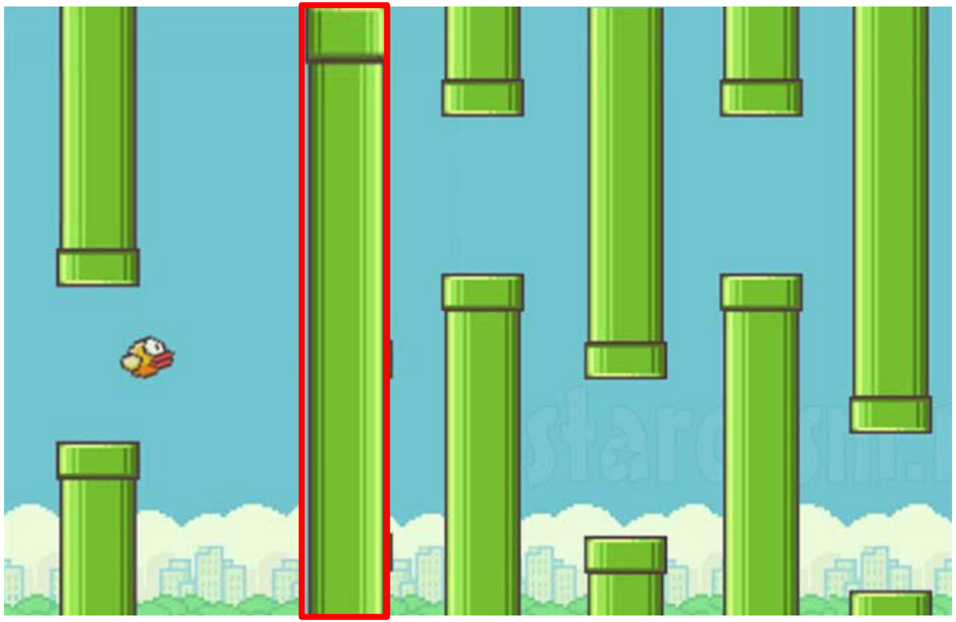
Boris Sedlak, Alireza Furutanpey, Zihang Wang,
Víctor Casamayor Pujol, and Schahram Dustdar

TU WIEN

upf. Universitat Pompeu Fabra Barcelona

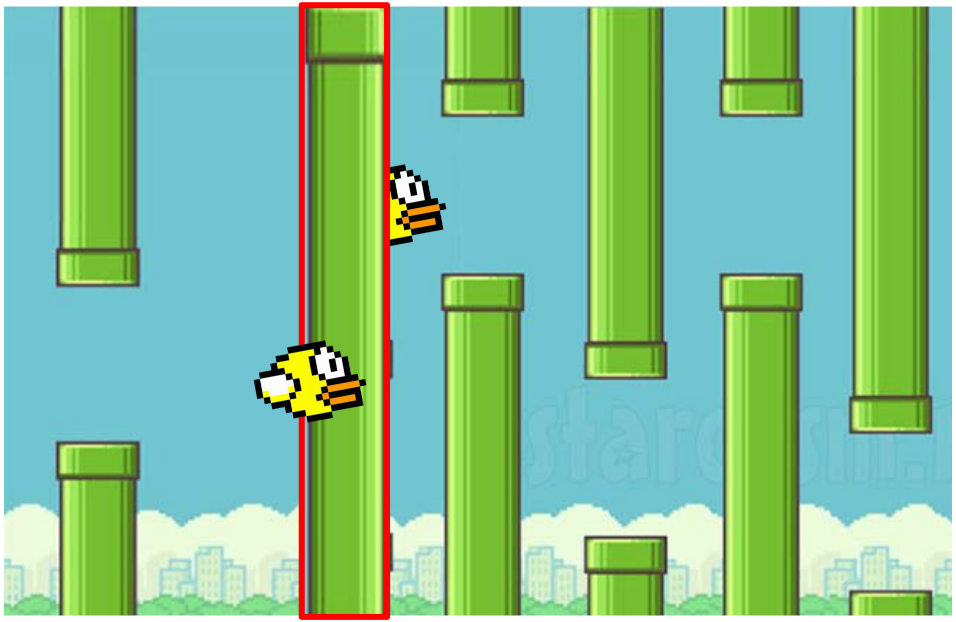


Need to find a policy that keeps flappy bird alive with **one** possible action only (i.e., jump)



Need to find a policy that keeps flappy bird alive with **one** possible action only (i.e., jump)

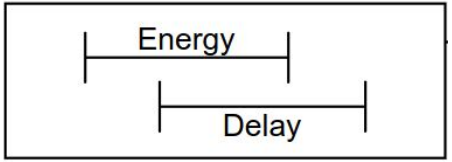
Can't pass obstacle?



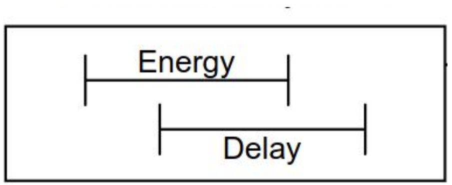
Need to find a policy that keeps flappy bird alive with **one** possible action only (i.e., jump)

Can't pass obstacle?

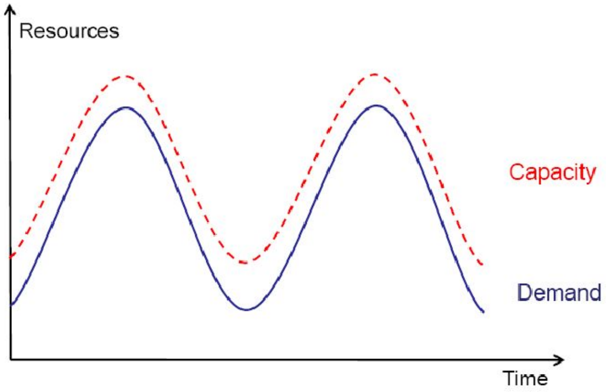
Use another dimension to improve flexibility of agent (i.e., action space)



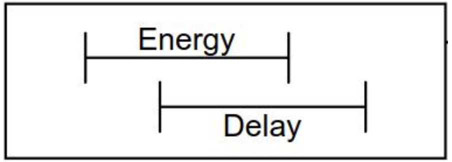
want to ensure **requirements** (= goals)
of systems, e.g., latency of OpenReview



want to ensure **requirements** (= goals) of systems, e.g., latency of OpenReview

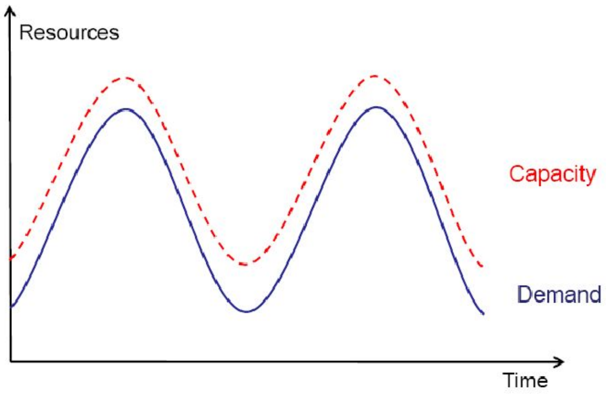


dynamic resource allocation according to current **demand**, e.g., submission time



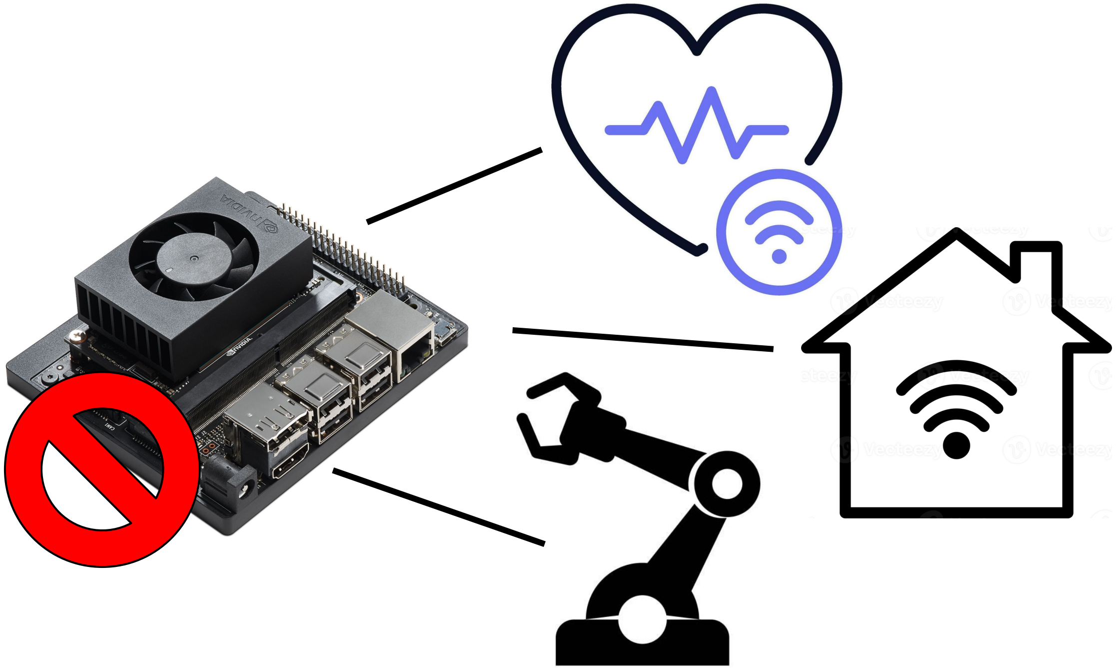
want to ensure **requirements** (= goals) of systems, e.g., latency of OpenReview

works well with infinite resources

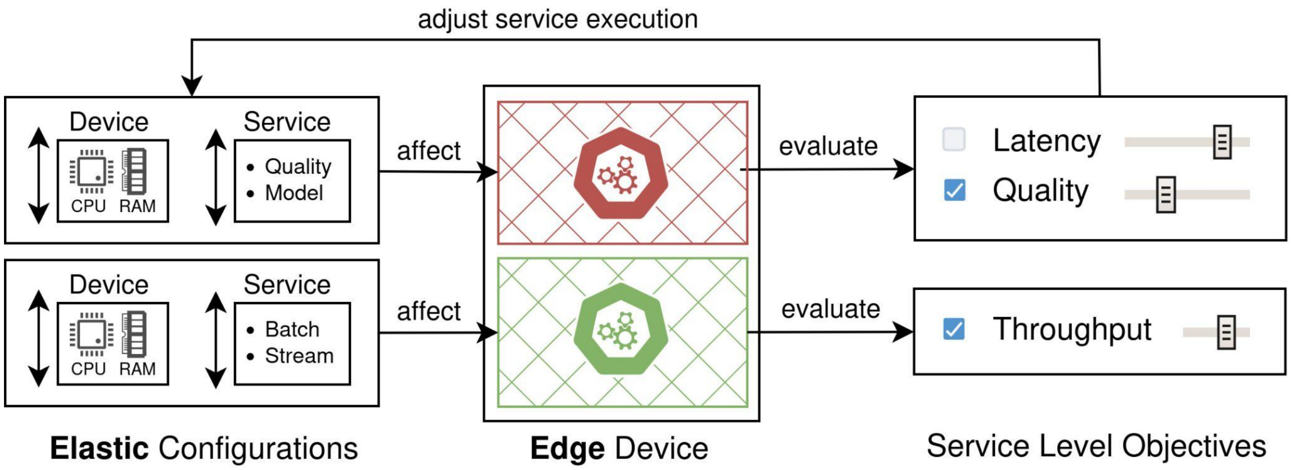


dynamic resource allocation according to current **demand**, e.g., submission time

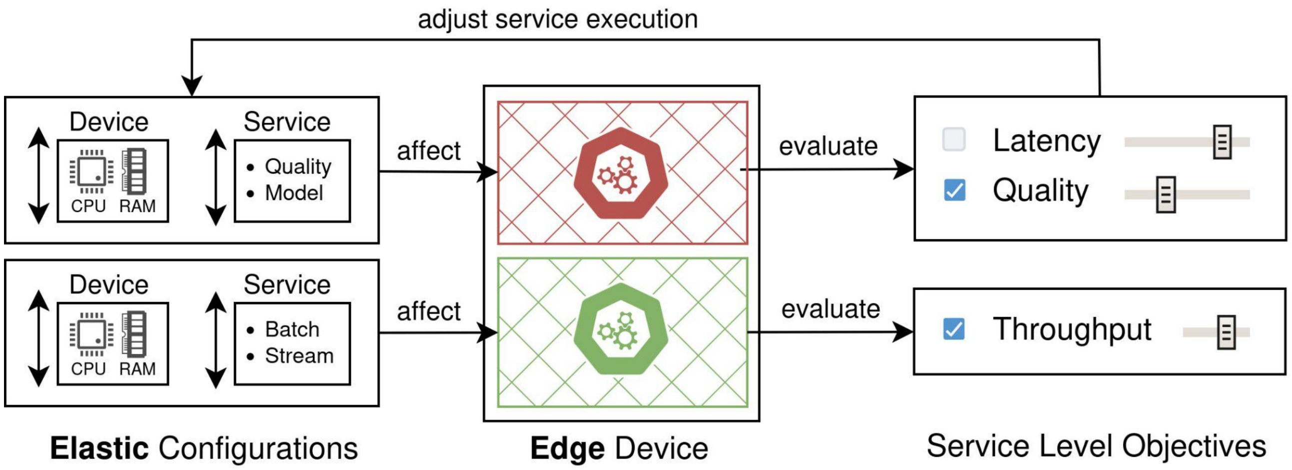
doesn’t work under resource limits



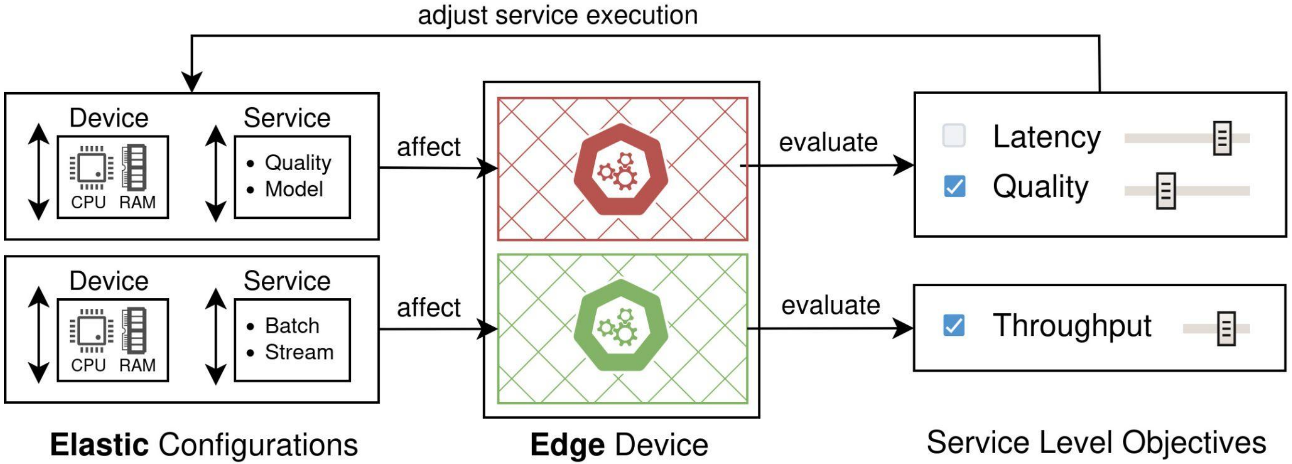
Multi-Dimensional Elasticity: Systems Perspective



Multi-Dimensional Elasticity: Systems Perspective

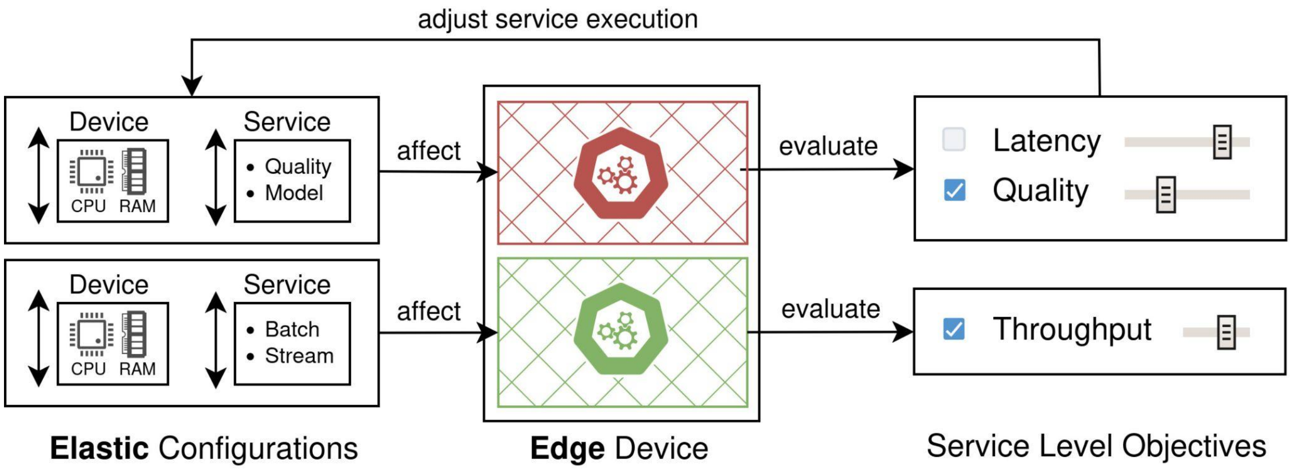


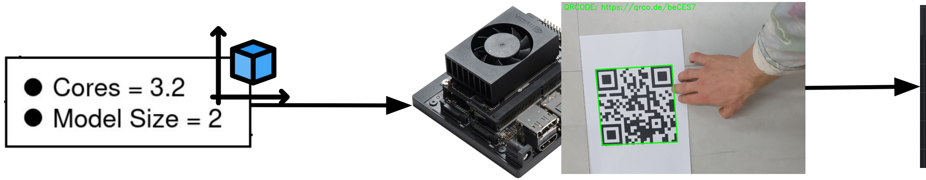
Multi-Dimensional Elasticity: Systems Perspective



fps	pixel	cores	change_flag
31	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False

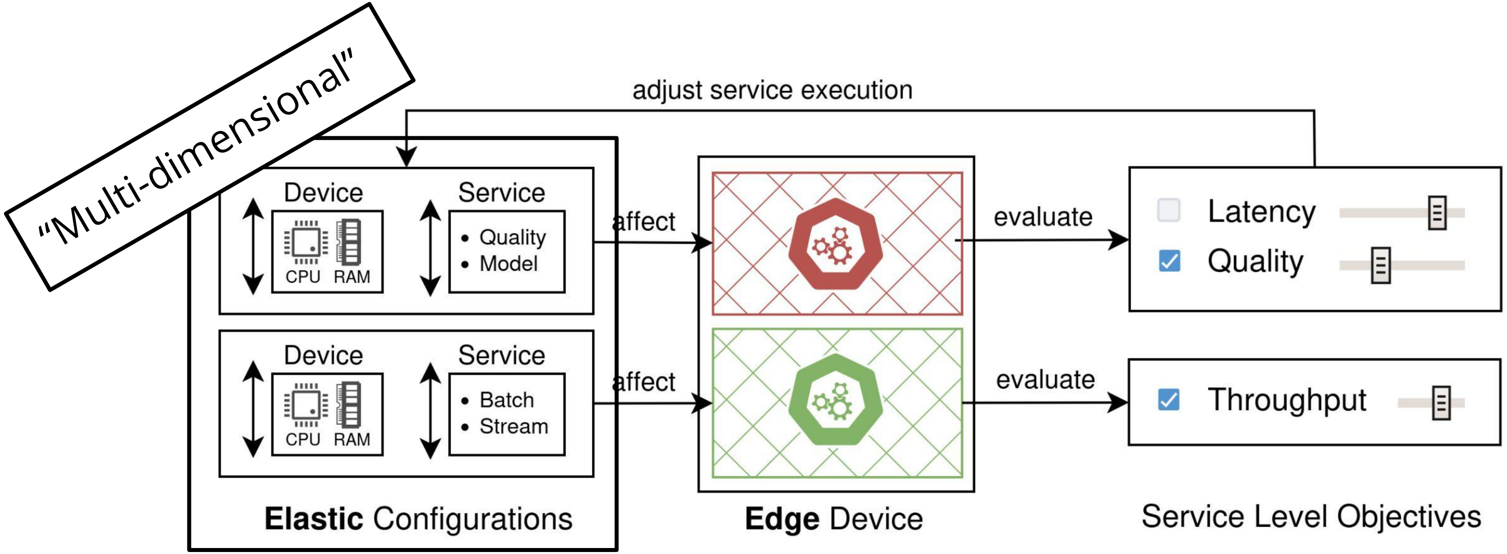
Multi-Dimensional Elasticity: Systems Perspective

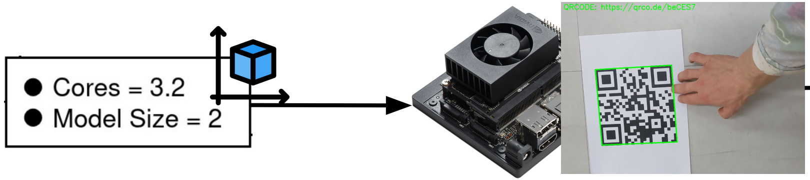




fps	pixel	cores	change_flag
31	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False

Multi-Dimensional Elasticity: Systems Perspective





fps	pixel	cores	change_flag
31	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False

Reformulated in AIF Terms: Mapping

Why not analyze system and solve exact / heuristic?

Reformulated in AIF Terms: Mapping

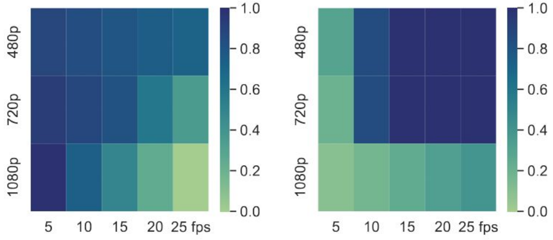
Why not analyze system and solve exact / heuristic?

1. Expected behavior unknown a priori for combinations of service types and devices

Reformulated in AIF Terms: Mapping

Why not analyze system and solve exact / heuristic?

1. Expected behavior unknown a priori for combinations of service types and devices
2. Large amount of parameter combinations make it hard for random/exhaustive search

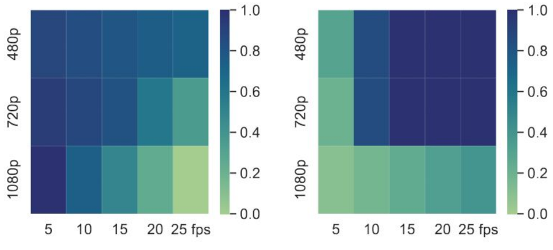


(b) Pragmatic value (c) Information gain

Reformulated in AIF Terms: Mapping

Why not analyze system and solve exact / heuristic?

1. Expected behavior unknown a priori for combinations of service types and devices
2. Large amount of parameter combinations make it hard for random/exhaustive search
3. Variable distributions change over time

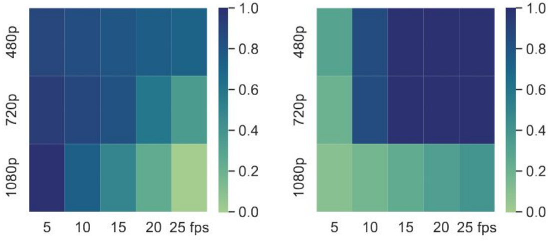


(b) Pragmatic value (c) Information gain

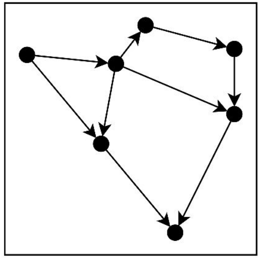
Reformulated in AIF Terms: Mapping

Why not analyze system and solve exact / heuristic?

1. Expected behavior unknown a priori for combinations of service types and devices
2. Large amount of parameter combinations make it hard for random/exhaustive search
3. Variable distributions change over time
4. Changes to complex systems can jeopardize dependent parts; causality or state model



(b) Pragmatic value (c) Information gain

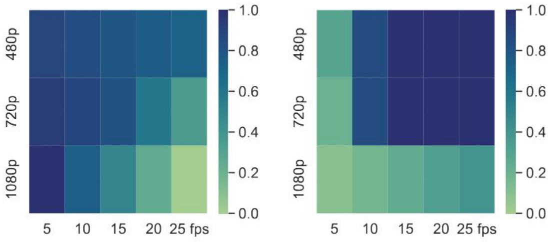


Reformulated in AIF Terms: Mapping

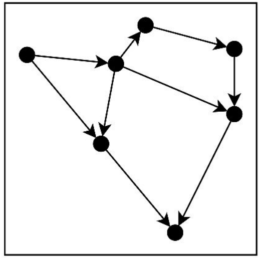
Why not analyze system and solve exact / heuristic?

1. Expected behavior unknown a priori for combinations of service types and devices
2. Large amount of parameter combinations make it hard for random/exhaustive search
3. Variable distributions change over time
4. Changes to complex systems can jeopardize dependent parts; causality or state model

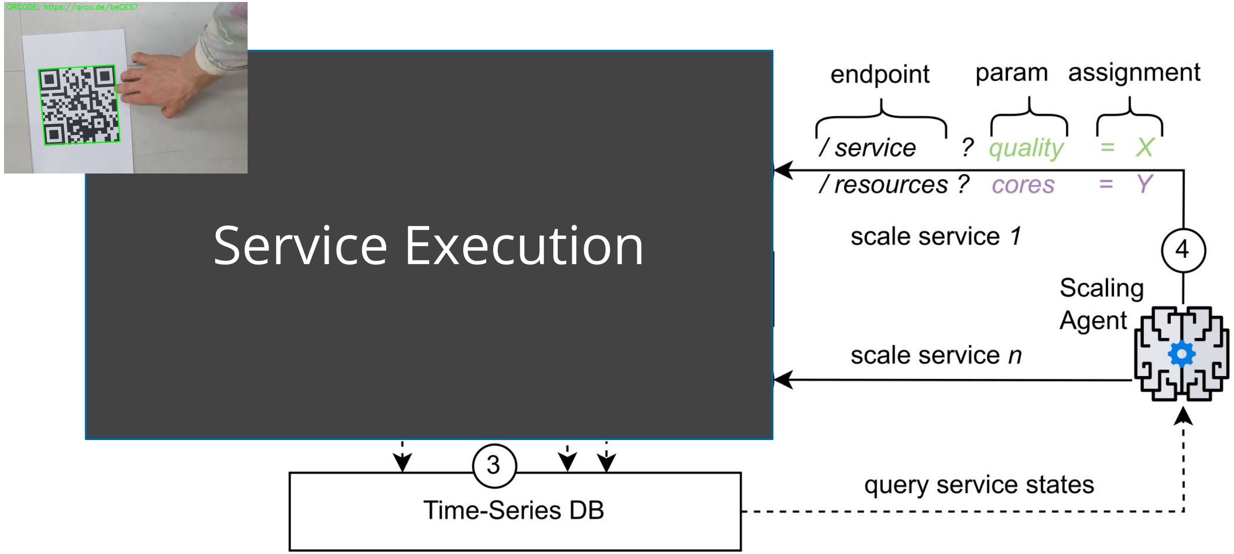
→ Formulate as POMDP, create interfaces for agents to **sense** the environment and **act** on services/devices; playground to compare performance of agent types



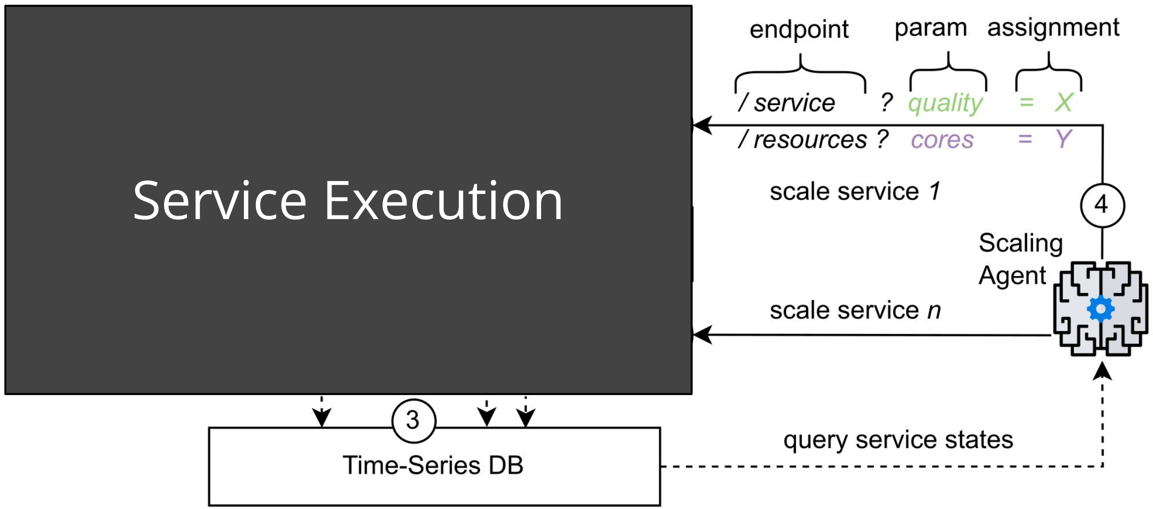
(b) Pragmatic value (c) Information gain



Experimental Setup: Environment



Experimental Setup: Environment

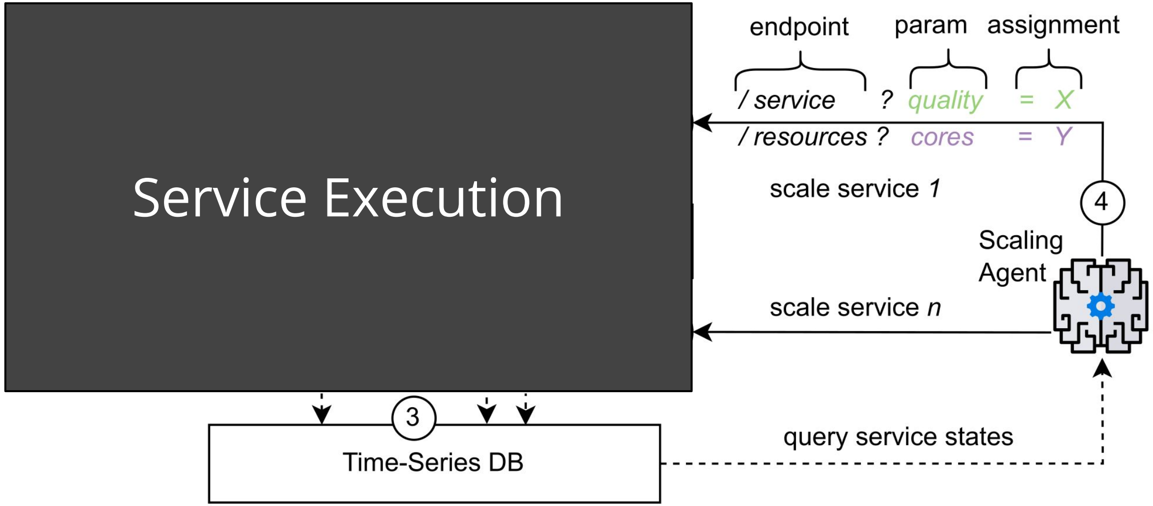


Action frequency **limited** by domain's temporal dynamics

"cool down period"

Choose autoscaling interval of 5s for picking new action

Experimental Setup: Environment



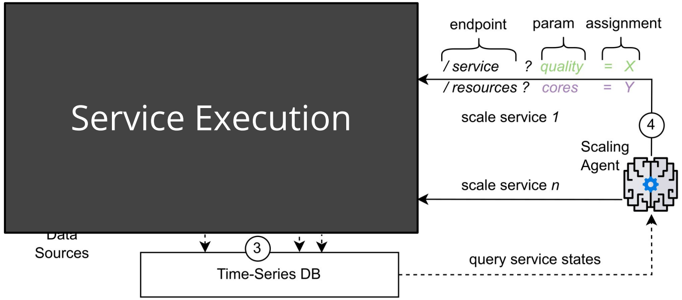
Action frequency **limited** by domain's temporal dynamics

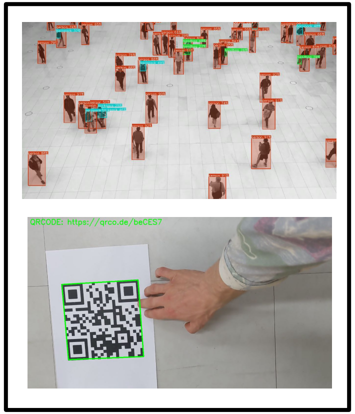
"cool down period"

Choose autoscaling interval of 5s for picking new action

Must be **sample-efficient**; conflicts with **common** RL

Experimental Setup: Environment (2)





Find **scaling policy** that optimizes performance

Problem instance:
2 services (QR, CV)

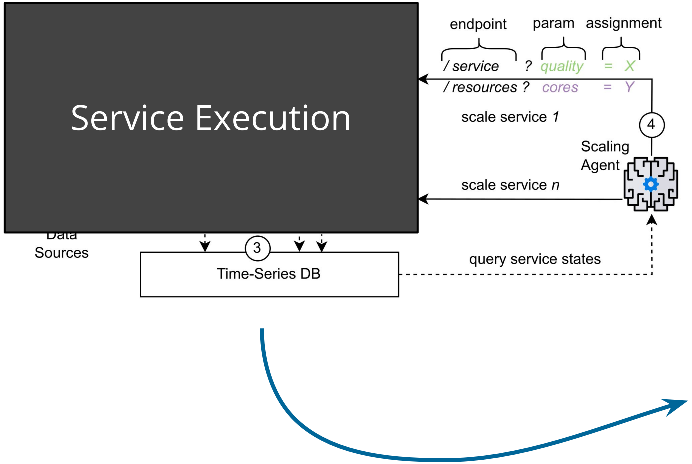
3 **interventional** parameters (quality, model size, resource allocation between services)

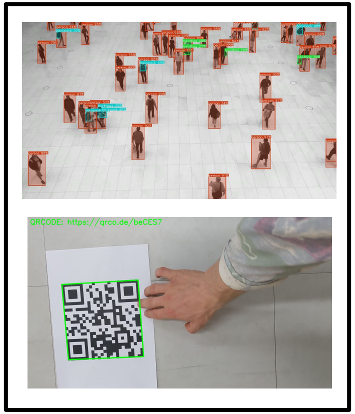
1 **dependent** parameter (service throughput / rps)

1 **constraint** (cores ≤ max)

\+ **preferred** observations (high quality and throughput)

Experimental Setup: Environment (2)





Find **scaling policy** that optimizes performance

Problem instance:
2 services (QR, CV)

3 **interventional** parameters (quality, model size, resource allocation between services)

1 **dependent** parameter (service throughput / rps)

1 **constraint** (cores ≤ max)

+ **preferred** observations (high quality and throughput)

→ **Optimize** the requirements fulfillment through 4 different agents (DQN, 2x AIF, Regression)

Experimental Setup: Agents

DQN agent work on heavily discretized space; requires offline pretraining in custom gymnasium environment

AIF agent uses pymdp [1] with equally discretized space; 35 policy options and 300k different state combination

DACI agent uses MCTS [2] methods for mapping high-dimensional observations into compressed latent space

RASK agent explores dependencies in the processing environment through continuous regression functions; uses numerical solver for finding optimal policy

[1] Heins, Millidge, Demekas, Klein, Friston, Couzin, Tschantz.: **pymdp: A Python library for active inference in discrete state spaces** (2022)
[2] Fountas, Z., Sajid, N., Mediano, P., Friston, K.: **Deep active inference agents using monte-carlo methods** (2020)

Experimental Setup: Agents

DQN agent work on heavily discretized space; requires offline pretraining in custom gymnasium environment

AIF agent uses pymdp [1] with equally discretized space: 35 policy options and 300k different state combination

DACI agent uses MCTS [2] methods for mapping high-dimensional observations into compressed latent space

RASK agent explores dependencies in the processing environment through continuous regression functions; uses numerical solver for finding optimal policy

[1] Heins, Millidge, Demekas, Klein, Friston, Couzin, Tschantz.: **pymdp: A Python library for active inference in discrete state spaces** (2022)
[2] Fountas, Z., Sajid, N., Mediano, P., Friston, K.: **Deep active inference agents using monte-carlo methods** (2020)

Experimental Setup: Agents

DQN agent work on heavily discretized space; requires offline pretraining in custom gymnasium environment

AIF agent uses pymdp [1] with equally discretized space; 35 policy options and 300k different state combination

DACI agent uses MCTS [2] methods for mapping high-dimensional observations into compressed latent space

RASK agent explores dependencies in the processing environment through continuous regression functions; uses numerical solver for finding optimal policy

[1] Heins, Millidge, Demekas, Klein, Friston, Couzin, Tschantz.: **pymdp: A Python library for active inference in discrete state spaces** (2022)
[2] Fountas, Z., Sajid, N., Mediano, P., Friston, K.: **Deep active inference agents using monte-carlo methods** (2020)

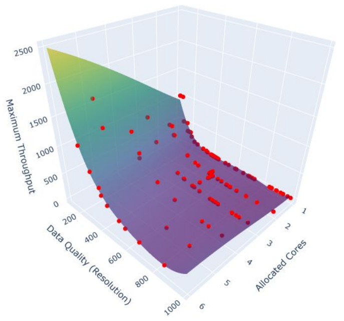
Experimental Setup: Agents

DQN agent work on heavily discretized space; requires offline pretraining in custom gymnasium environment

AIF agent uses pymdp [1] with equally discretized space; 35 policy options and 300k different state combination

DACI agent uses MCTS [2] methods for mapping high-dimensional observations into compressed latent space

RASK agent explores dependencies in the processing environment through continuous regression functions; uses numerical solver for finding optimal policy

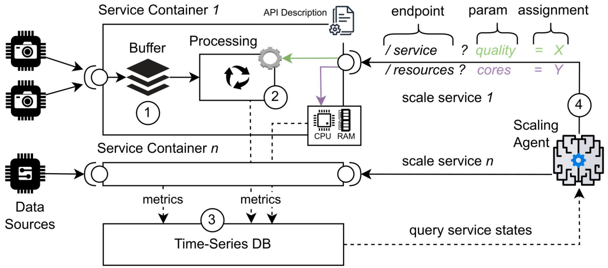


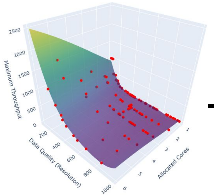
[1] Heins, Millidge, Demekas, Klein, Friston, Couzin, Tschantz.: **pymdp: A Python library for active inference in discrete state spaces** (2022)
[2] Fountas, Z., Sajid, N., Mediano, P., Friston, K.: **Deep active inference agents using monte-carlo methods** (2020)

Experimental Setup: Scenarios

Start processing services and one of the scaling agents; let agent operate for **250s** (i.e., sense and act in env.)

Capture **reward** (i.e., requirements fulfillment) and the **time** that agents require to infer a scaling policy

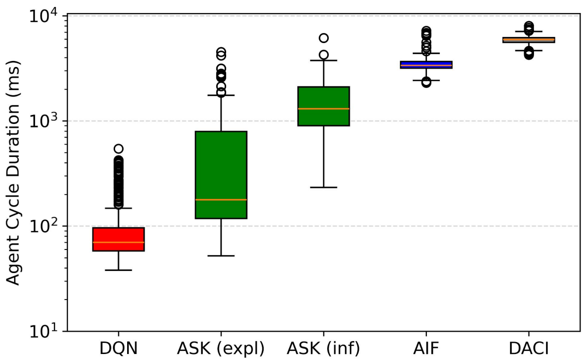




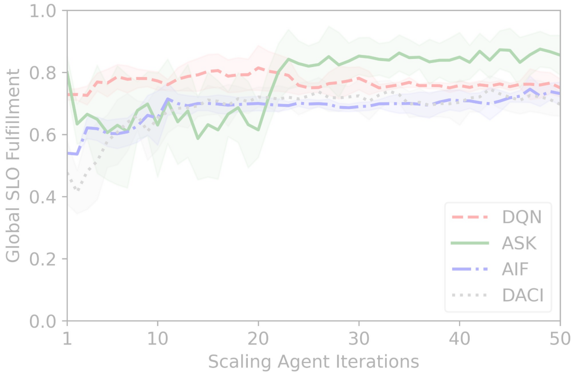
fps	pixel	cores	change_flag
31	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False
32	800	3	False

Experimental Results

Runtime: DQN performs best with runtimes < 100ms; highly optimized on hardware; AIF and DACI most computations

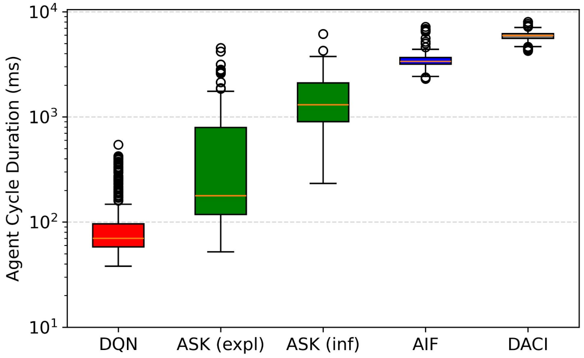


Reward: ASK superior, other agents similar. ASK operates in **continuous** space and makes fine-grained scaling actions

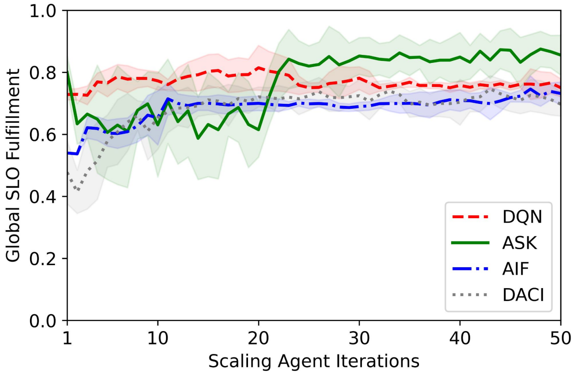


Experimental Results

Runtime: DQN performs best with runtimes < 100ms; highly optimized on hardware; AIF and DACI most computations



Reward: ASK superior, other agents similar. ASK operates in **continuous** space and makes fine-grained scaling actions



Conclusion

Motivation: Large-scale computing systems pose infinity of optimization problems; must explore behavior during runtime due changing variable distributions.

Solution: Model processing environment through POMDP and train state transition models; compare four agents.

Benefit: Create stable autoscaling policies; embed AIF agents into common use cases and allow comparison with contemporary ML approaches (e.g., DQN).

Future work: sophisticated exploration schemes for continuous variables built on Gaussian processes.