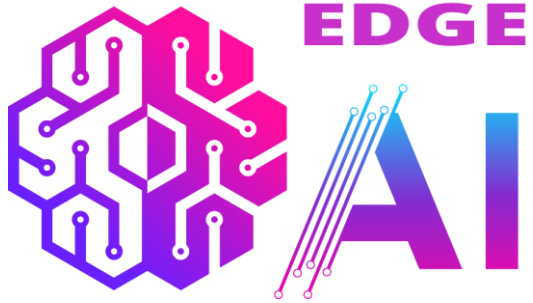
European Conference on EDGE AI Technologies and Applications - EEAI



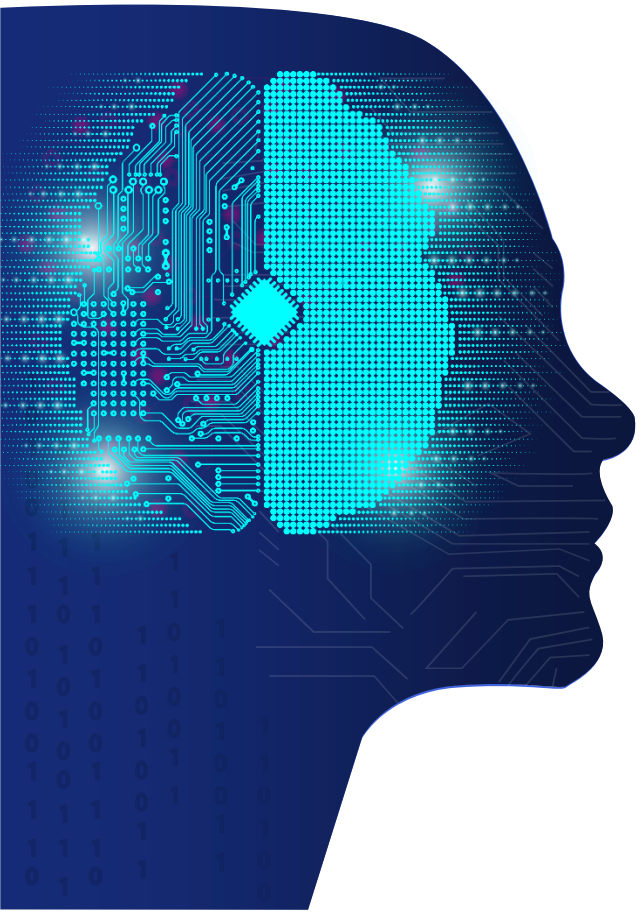
AI-enabled Resource Optimization in Critical Computing Infrastructures

Dr. Prodromos Makris,
Senior Researcher
National Technical University of Athens (NTUA),
Greece

17-19 October 2023 Athens, Greece

Copyright © 2023

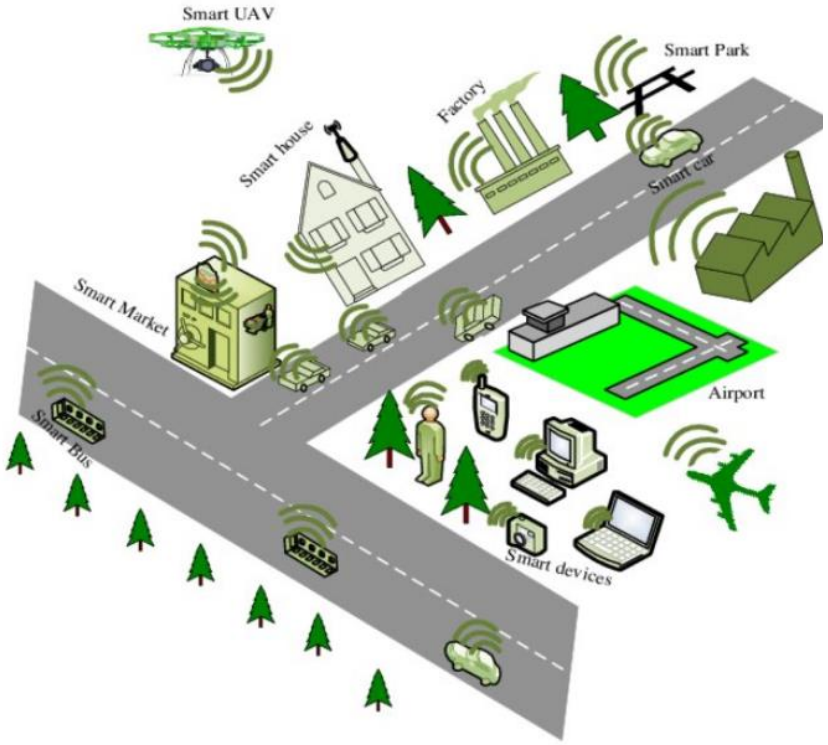
Presentation Outline

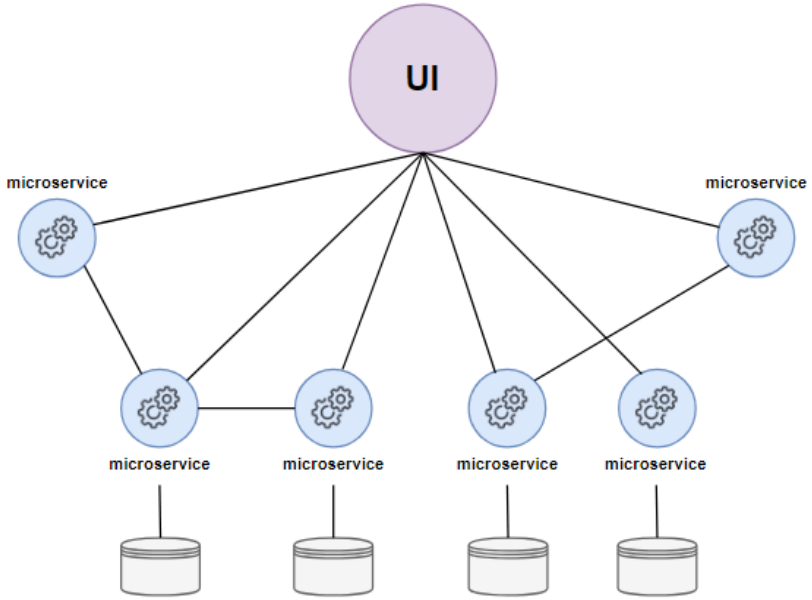


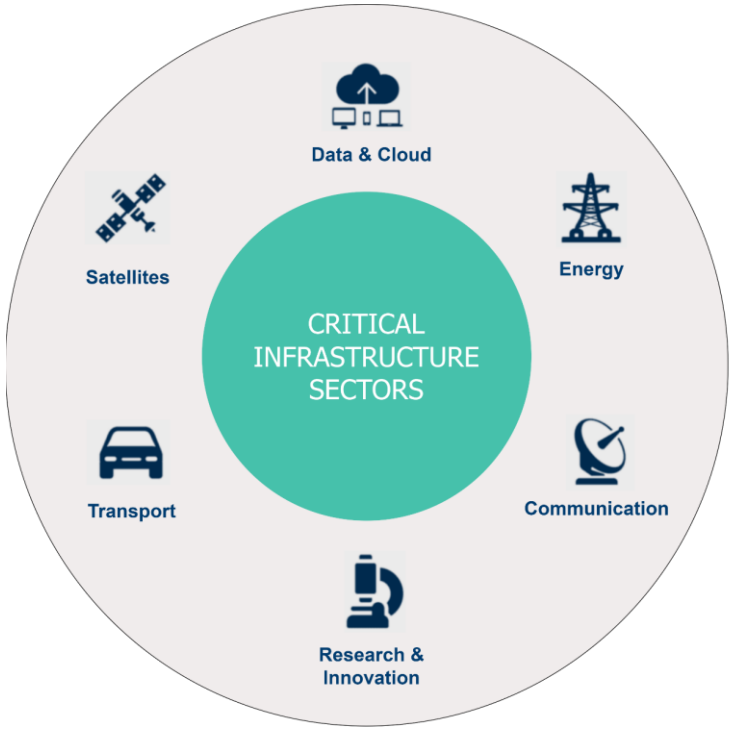
- Introduction
- State-of-the-art
- Research Goal
- Methodology
- Problem Formulation
- Simulation Experiments
- Conclusion and Future Research

Copyright © 2023

Introduction







Next generation smart environments

Optimal resource orchestration across IoT-edge-cloud continuum

Cloud-native applications

Distributed micro-services' design & deployment (requirements set by S/W developers)

Critical infrastructures

Diverse security/communication policies across multiple operators' domains

Copyright © 2023

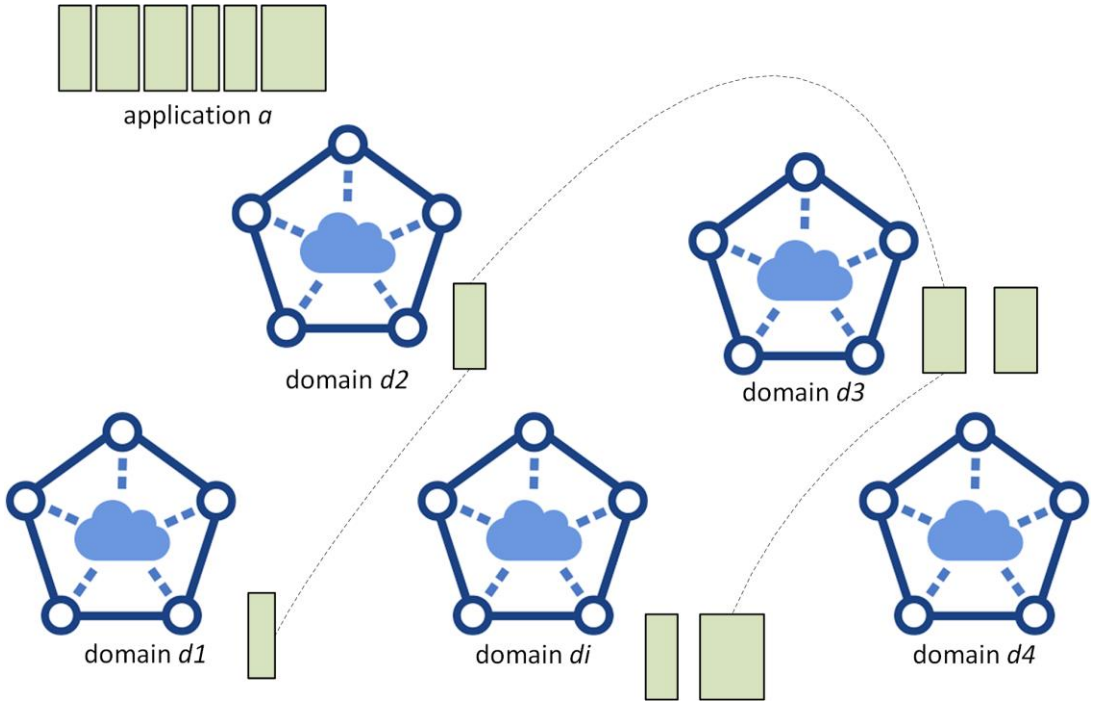
State-of-the-art

- **IoT-edge-cloud continuum**:
  - Multi-layered infrastructure execution & scalability across diverse computational resources.
  - But few works dealing with multiple and conflicting objectives.
- **VM & Networking**:
  - Positioning of VMs to optimally use network resources to minimize congestion/overhead.
  - But few works dealing with inter-dependencies among micro-services/processing tasks.
- **Multi-Agent Systems**:
  - Agent-based dynamic scheduling & prioritization in critical sectors
  - But few works on critical infrastructures, where security constraints are important.
- **Reinforcement Learning**:
  - Few works on Q-learning and DRL for task off-loading, addressing microservice coordination and minimizing waiting times.
- **Neural Networks & DRL**:
  - Combining heuristics and DRL for enhanced resource allocation under uncertainties.
  - But few works on integrating graphs for AI-enabled resource orchestration.

Copyright © 2023

Research Goal

- ✓ To enable **different levels of communication** across distinct infrastructure domains, aiming to safeguard the infrastructure's security while ensuring its smooth operation.
- ✓ To design a **decentralized optimization** model that ensures the careful and selective dissemination of information among different providers and operators.
- ✓ To model the **inter-dependencies among processing tasks** to guarantee various requirements (minimize delay, processing costs, security vulnerabilities, maximize resource utilization, etc.).
- ✓ To develop a **resource allocation mechanism that operates efficiently** (in real-time with negligible optimality gap) under the constraints imposed by both the critical infrastructure and the applications.



Copyright © 2023

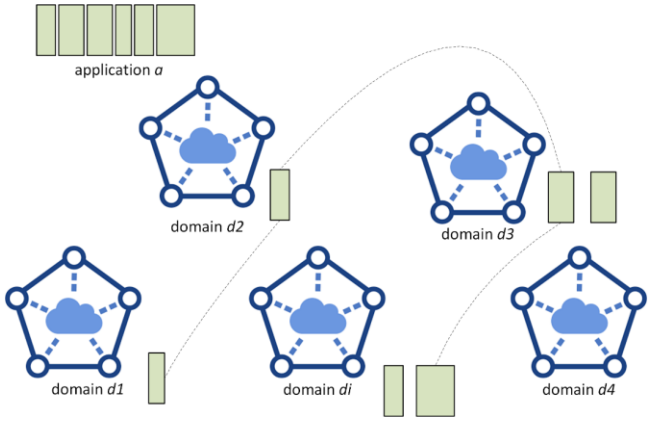
Methodology

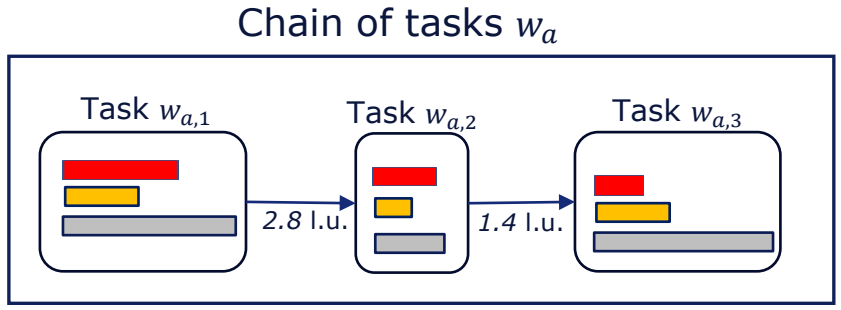
- **Graph-based Representation**:
  - Convert the infrastructure into a graphical model to simplify complex, multi-layered systems.
- **Markov Decision Process (MDP)**:
  - Employ deterministic MDP for modelling and solving the multi-objective resource allocation problem.
- **Deep Q-Learning Integration**:
  - Utilize the power of deep learning for better decision-making in resource allocation under uncertainties.
- **Iterative Training**:
  - Implement iterative training protocols for refining resource allocation decisions and ensuring optimization in real-time contexts.

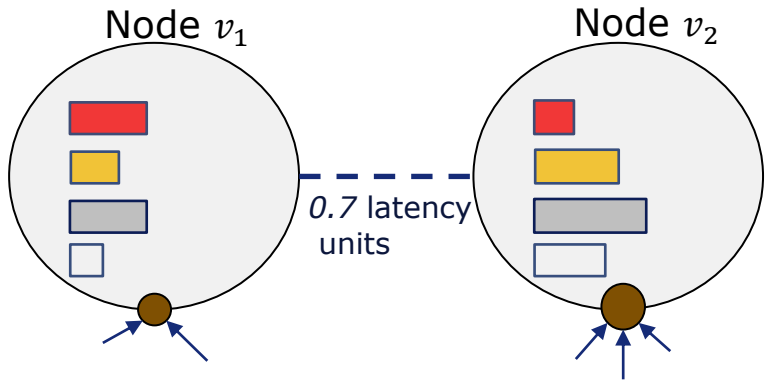
Copyright © 2023

Problem Formulation

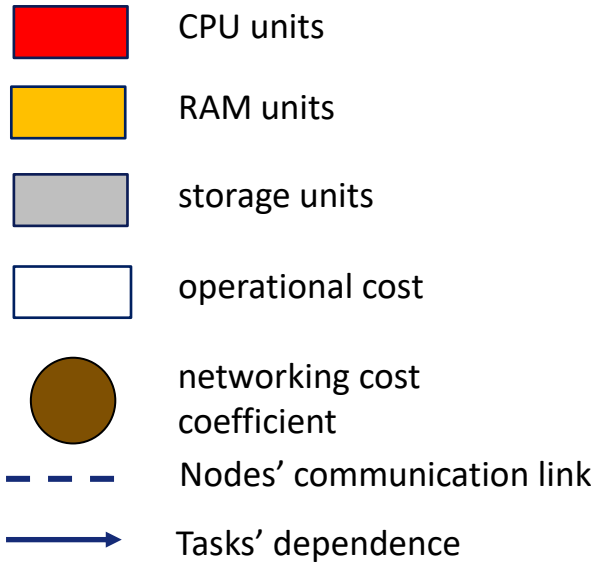
- Computing and networking critical infrastructure as a fully connected directed graph G=(V,E).
- Nodes V equipped with computing (CPU, RAM) and storage resources.
- Links E among nodes embody parameters such as transmission latency, available bandwidth and security constraints.
- Each application is described by the tuple $a = (\pi_a, w_a)$.
- Each app demand $a \in A$ is generated at an infrastructure node π_a and is described by a chain of tasks w_a,each with distinct resource requirements.







Resources and variables:



Copyright © 2023

Problem Formulation - MDP

- A discrete-time dynamic system defined by the tuple (X, U, R).
- $State\ X = (T, W)$ is composed of:
  - the resource status of nodes T
  - the task status W.
- Action space U contains all possible actions from a given state -> each action at state k, $u_k \in U$ corresponds assignment of a task to an infrastructure node.
- For state x_k, a feasible action u_k must satisfy:
  - Capacity Constraint: $CPU_r(x_k) \leq CPU(v, x_k)$, $RAM_r(x_k) \leq RAM(v, x_k)$, $Str_{r(x_k)} \leq Str(v, x_k)$
  - Latency Constraint: The latency between nodes that serve communicating tasks should respect their latency limit. $l_{v,v'} \leq \theta_{a,t,t'}$.
- R is the reward function, given by function $g_k(x_k, u_k)$. It provides a scalar feedback for the action and is defined as a weighted sum of two components:
  - the negative CPU cost
  - the negative task latency
- The aim is to optimize through the reward an objective function that consists of:
  - nodes' operational cost
  - the utilization of the activated infrastructure resources
  - the applications' latency

Copyright © 2023

Multi agent Deep Q Learning

- Multiple Rollout agents collaborate through time to address the workload demands.
  - They utilize multiple DQN agents operating across different domains. Their cooperation is governed by the established communication constraints.

- Each Rollout agent $l \in L$ is assigned with an application demand. Starting from state $x_{l,k}$, it generates a sequence of h actions , acquiring a cumulative reward $\sum_{n=k}^{k+h} g_n\left(x_{l,n}, u_{l,n}\right)$.

- At state $x_{l,k+h}$ it consults a fast sub-optimal policy to evaluate the future cost $\widetilde{J}_k(f_{l,k+h}\left(x_{l,k+h}, u_{l,k+h}\right)$.

- The agent places the microservice where the minimum accumulated cost $\sum_{n=k}^{k+h} g_n\left(x_{l,n}, u_{l,n}\right) + \widetilde{J}_k(f_{l,k+h}\left(x_{l,k+h}, u_{l,k+h}\right)$ is achieved.

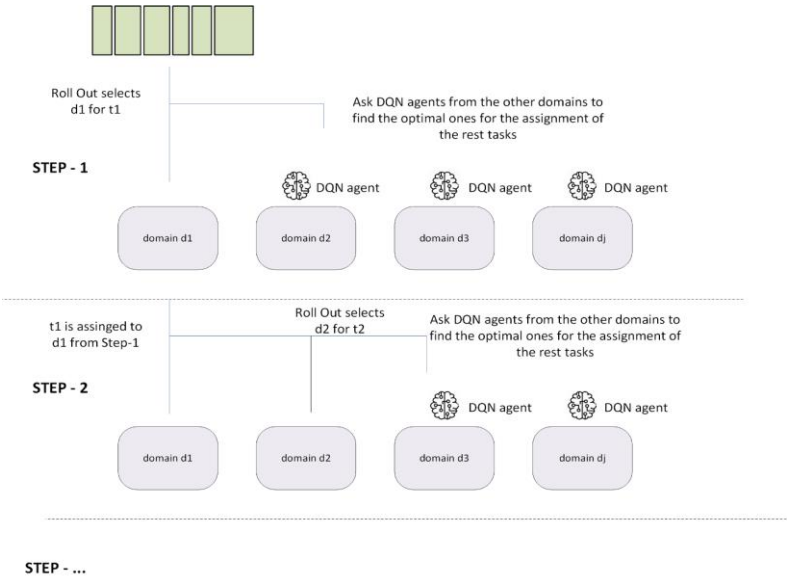
- To curtail the exploration space, only feasible placements are evaluated (i.e., nodes that respect latency, security and capacity constraints).

Copyright © 2023

Multi agent Deep Reinforcement Learning

- A multi-Agent Deep Q-Learning **(DQL)** approach is used as the **sub-optimal policy to approximate future costs**.
- DQN agents estimate Q-values per domain by tuning neural network weights using accumulated experience.
- A mechanism for inter-agent communication is introduced to ensure a global optimization. Agents exchange their network parameters (weights, biases) with communicating agents.
- Upon receiving these parameters, each agent *j* updates using a combination of its current and the received parameters.

The Multi-Agent Rollout Process Facilitated by DQN Agents.



The pseudocode of the training of the multi-agent DQN network.

1. Initialize: θ_{j_d}, ε, c
2. **For** each episode **do**:
3. Initialize state $x_{j,0}$
4. **For** each time step within the episode **do**:
5. **For** each agent j_d with d in D **do**:
6. Initialize window size h
7. **For** each h **do**:
8. Select action $u_{j,k}$ based on current state $x_{j,k}$and ε
9. Take action $u_{j,k}$, observe new state $x_{j,k+1}$ and reward $g(x_{j,k}, u_{j,k})$.
10. Update target Q-value using Bellman equation
11. Update θ_{j_d} based on observed reward and target Q-value.
12. **End For**
13. **If** mod(time step, c) ==0:
14. Exchange $\theta_{j_d}$with communicating agents and update accordingly
15. **End If**
16. **End For**
17. **If** all workloads are assigned:
18. Continue to next episode
19. **End If**
20. **End For**

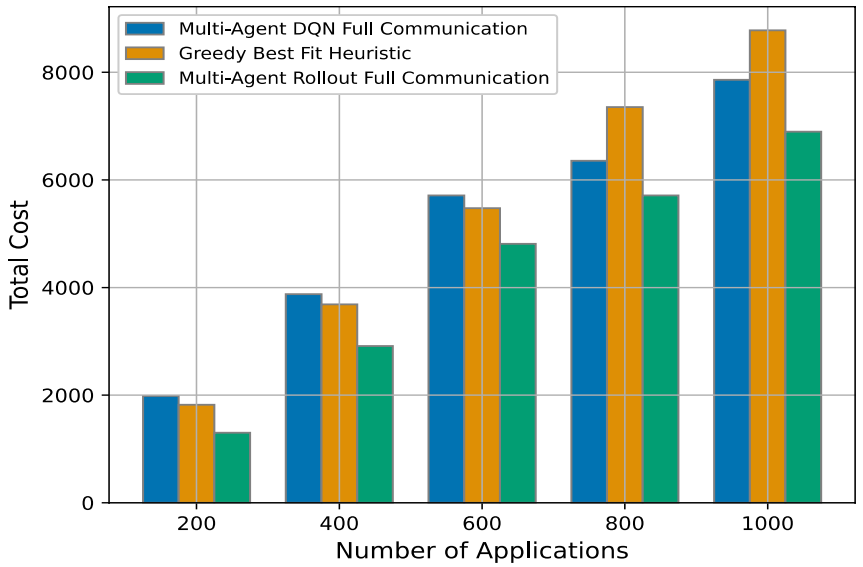
Copyright © 2023

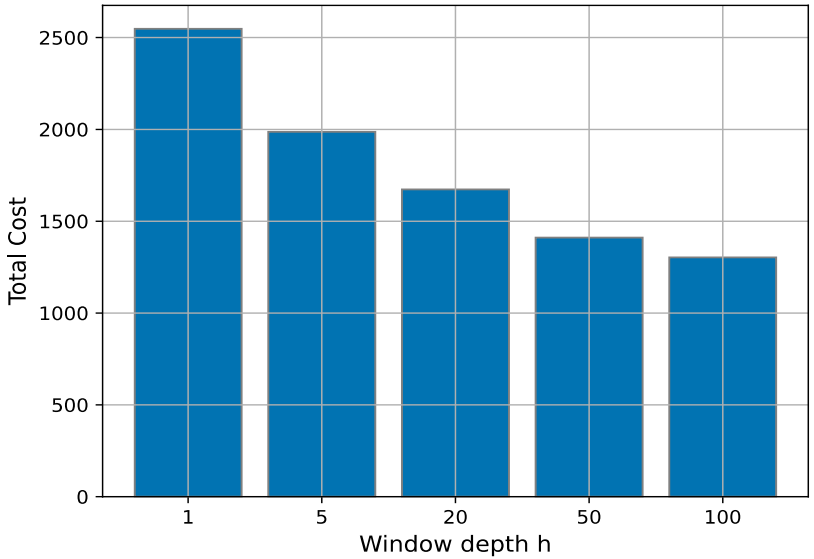
Simulation Experiments

- Development Framework:
  - Python with PyTorch library for DQN agents.
- Neural Network Design:
  - Input layer: Matches state vector elements.
  - Hidden layer: 64 units with ReLU activation.
  - Output layer: Matches available action count.
- DQN Parameters:
  - Exploration vs. Exploitation: parameter ε (=0.9)
  - Learning Rate: parameter γ,η (=0.5)
- Infrastructure Model:
  - 100 nodes, varied resources (CPU, RAM, storage).
  - Monetary costs: Majority at [6,7] C.U., with variations.
  - Delay categorizations based on node types: edge vs. cloud.
- Application Demands:
  - Variable demands: [1,4] tasks per application.
  - Task requirements: Varied CPU, RAM, storage needs.
  - Task latency within applications: [1,3] l.u.
- Training Phase:
  - 500 episodes of iterative learning and adaptation.

Copyright © 2023

Simulation Experiments

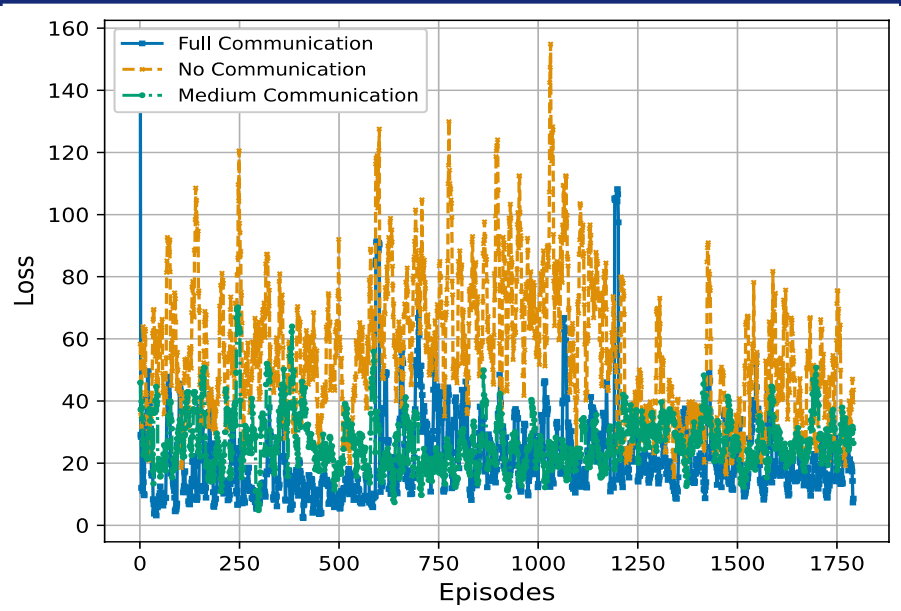


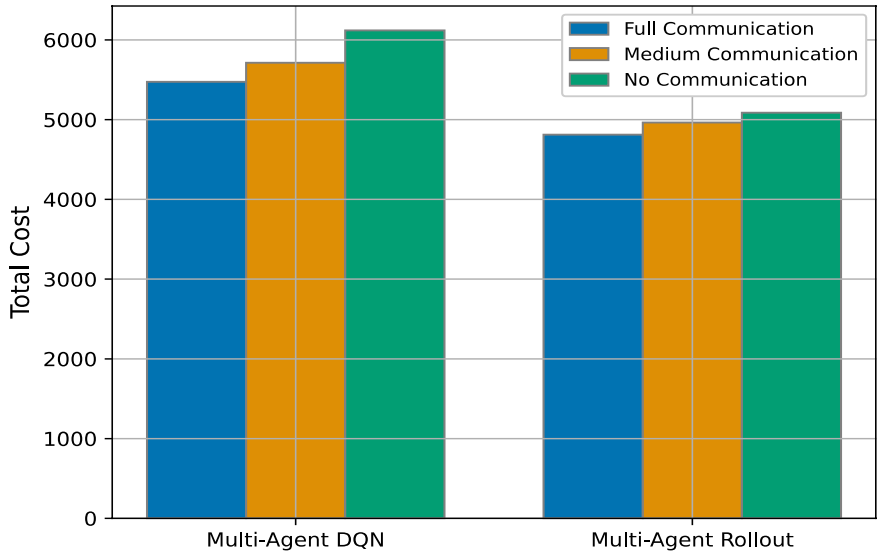


- Rollout yields the lowest cost, exploiting both the look-ahead window and the future approximation (i.e. both immediate and future rewards made by other agents).
- Multi-agent DQN outperforms the greedy heuristic by leveraging past experiences and new knowledge to make more informed decisions.
- As the window depth decreases, the evaluated actions at a given state decrease accordingly → $[(|U|-h)/|U|]$ decrease of execution time.
- Performance increases proportionally to the lookahead window size.
- Even with a window half the size of the action space, the proposed rollout achieves close-to-optimal results.
- **Optimal value of *h* depends on the desired trade-off between execution time and optimality.**

Copyright © 2023

Simulation Experiments



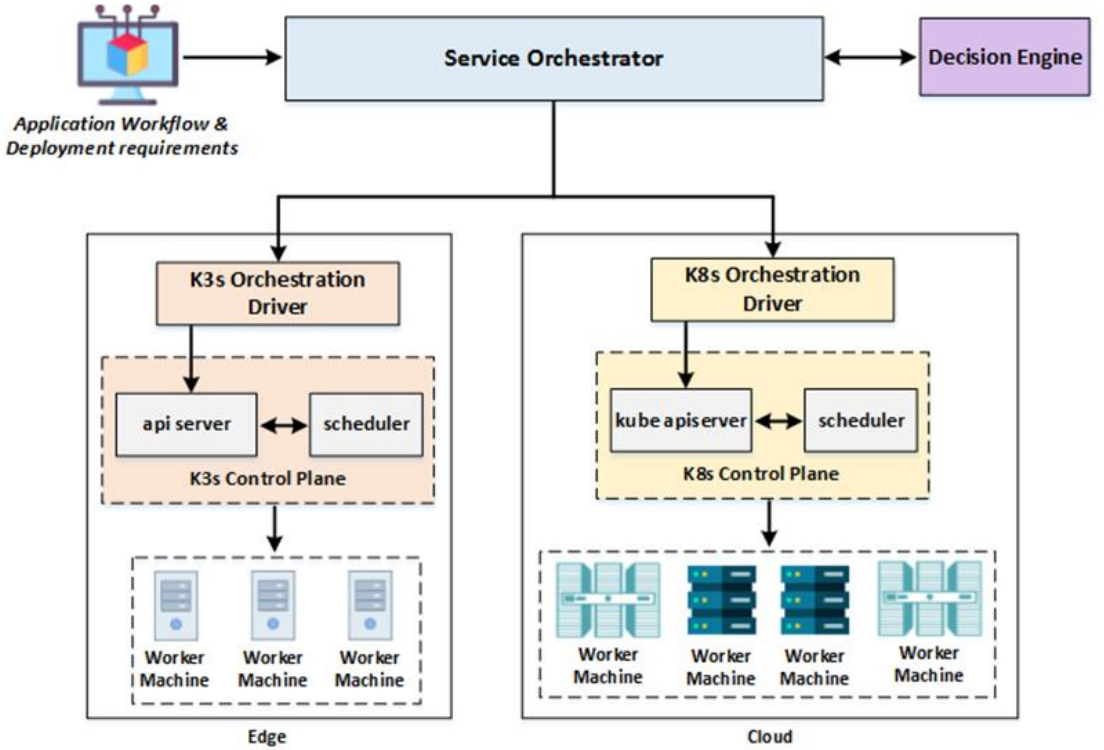


- Loss function is larger when there is limited communication among agents and improves as the level of communication increases.
- In all cases, loss decreases through time as the agents train.
- Even in the absence of communication among agents, the deviation in performance of Rollout mechanism remains tolerant.
- This showcases the resiliency of the multi-Agent Rollout mechanism, despite varying levels of inter-agent communication.

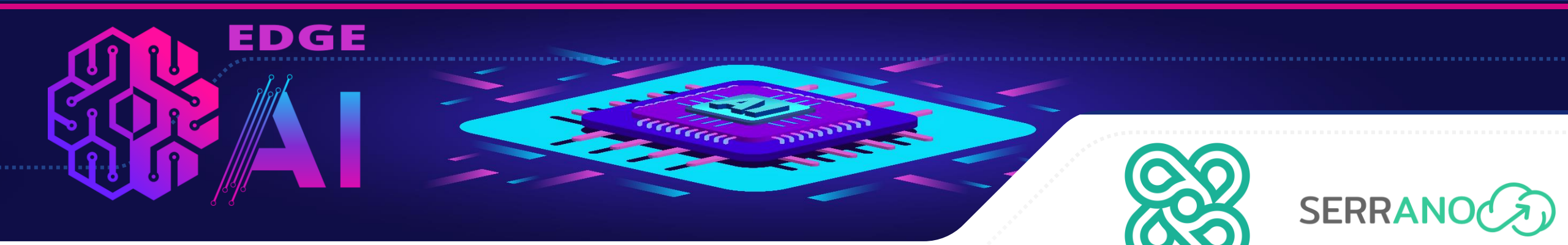
Copyright © 2023

Conclusion & future directions

- Exploit **NTUA's in-house cloud-native application "Service Orchestrator"** implemented in Python
- "Service Orchestrator" is placed on top of existing commercial open-source orchestration platforms (cf. K8s/K3s orchestration drivers)
- Follows a **cloud native design and declarative approach** for interacting with the platform-level orchestration mechanisms
- Currently integrates several resource allocation algorithms (optimization, AI/ML-enabled) in its "Decision Engine"
- "Decision Engine" is populated with new auction-based resource allocation algorithms → "**Broker" that brings closer Service Requesters (demand) and Service Providers (supply) automating the market clearing process.**
- Validated with real-life data/industrial use cases within EU projects:
  - **SERRANO** - https://ict-serrano.eu/
  - **CLEVER** - https://www.cleverproject.eu/
  - **MARSAL** - https://www.marsalproject.eu/
  - **EMPYREAN** – to start in Jan 2024



Copyright © 2023



SERRANO

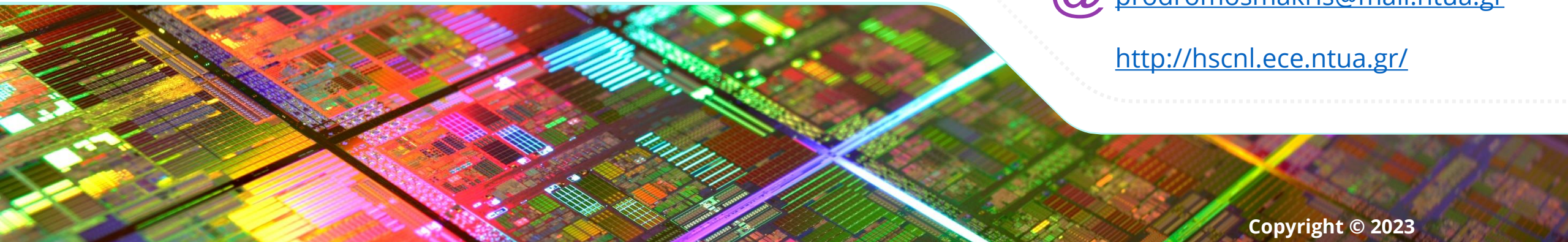
Thank You

For your attention

National Technical University of Athens

@ prodromosmakris@mail.ntua.gr

http://hscnl.ece.ntua.gr/



Copyright © 2023