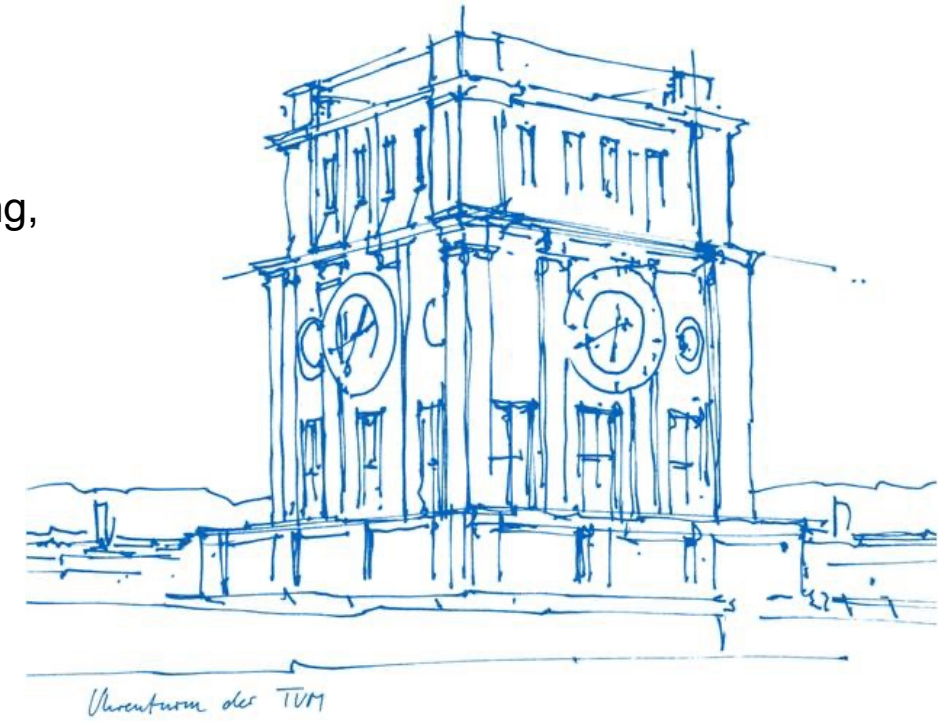
Deep Reinforcement Learning for Real-Time Resource Management

Marco Caccamo

Chair of Cyber Physical Systems in Production Engineering,

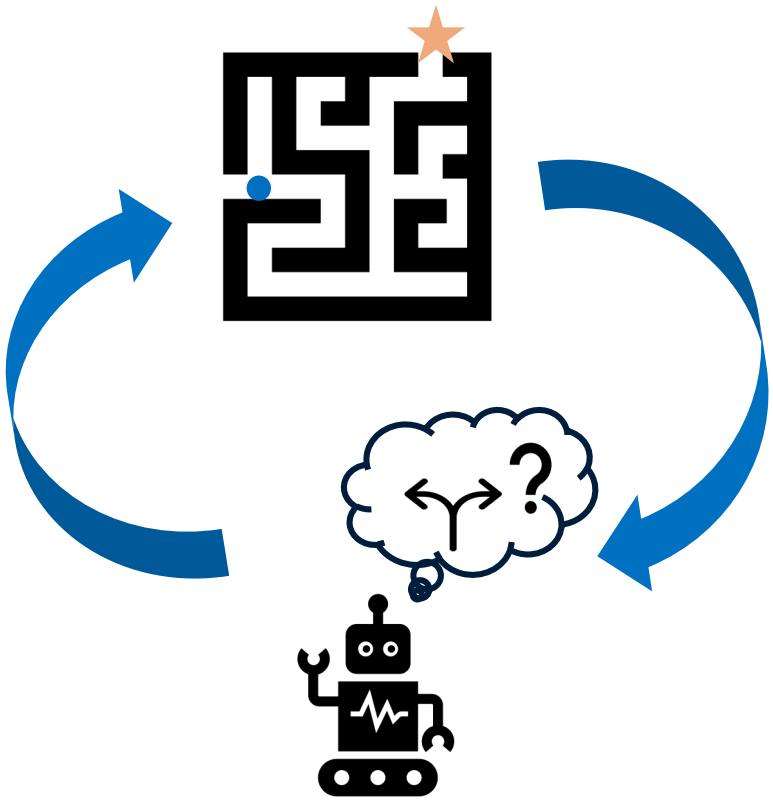
Technical University of Munich (TUM)

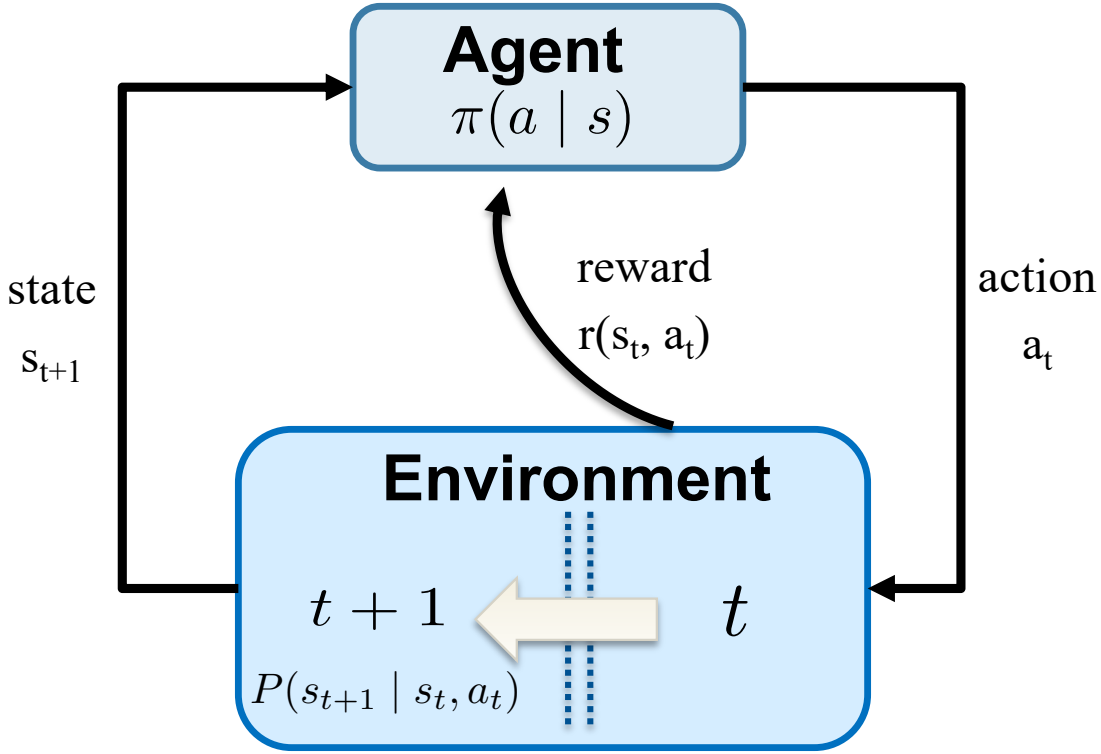


Outline

- Background: Deep Reinforcement Learning (DRL)
- DRL applied to Real-Time Resource Management
  - DAG scheduling problem
  - Edge Generation Scheduling
  - DRL architecture with PPO
  - Evaluation Results
- Future Research
- Conclusions

Reinforcement Learning (RL)





Agent – Environment Interaction

Reinforcement Learning (RL)

Markov Decision Process (MDP):

$$\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$$

- State Space $\mathcal{S}$
- Action Space $\mathcal{A}$
- State Transition Probability $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$
- Reward Function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Discount Factor $\gamma \in [0, 1]$

Policy: $a = \pi(s)$ Deterministic **or** $a \sim \pi(\cdot \mid s)$ Stochastic

Objective of Reinforcement Learning

$$J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

$$\pi^{\star} = \arg\max_{\pi} J(\pi)$$

$\mathbb{E}_{\pi}$: expected total reward over trajectories generated by the policy

Q(s,a) Action-Value function and **V(s)** Value function

Action-Value Function:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right]$$

Value Function: $V^{\pi}(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[\, Q^{\pi}(s,a)\,]$

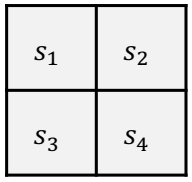
➔ **how good on average a state S is under policy π?**

Q-iteration with tabular Q-function

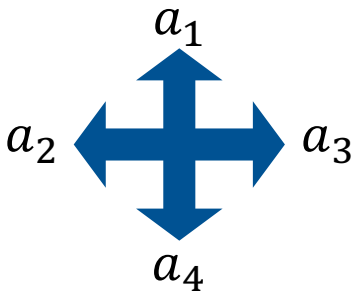
Bellman Optimality Equation for Action-Value Function:

➔ Q*(s, a) can be iteratively computed

$$\forall s, \forall a \quad Q^{\star}(s,a) = r(s,a) + \gamma \sum_{s'} P(s' \mid s,a) \max_{a'} Q^{\star}(s',a')$$



Grid World



Actions

	a_1	a_2	a_3	a_4
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_1, a_3)$	$Q(s_1, a_4)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$	$Q(s_2, a_4)$
s_3	$Q(s_3, a_1)$	$Q(s_3, a_2)$	$Q(s_3, a_3)$	$Q(s_3, a_4)$
s_4	$Q(s_4, a_1)$	$Q(s_4, a_2)$	$Q(s_4, a_3)$	$Q(s_4, a_4)$

Q Table

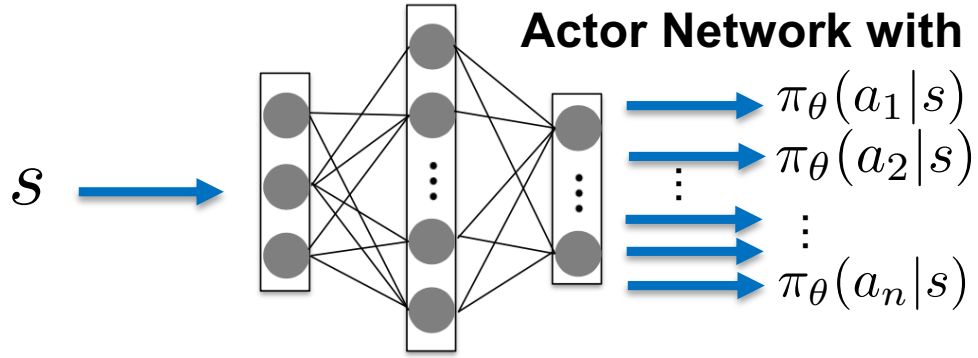
Problems:

- **curse of dimensionality**
- **state transition probability is often unknown (model free RL)**

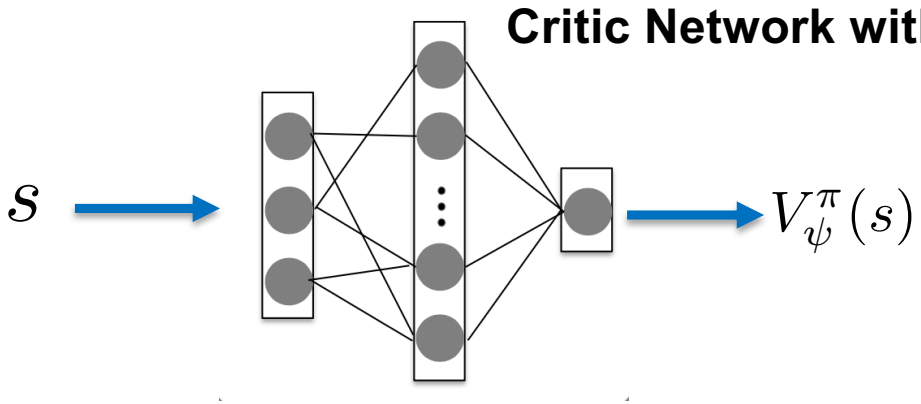
Solutions:

- **Approximate Q-table with deep neural net.**
- **Use Q-learning or policy gradient algorithms**

Proximal Policy Optimization (PPO)

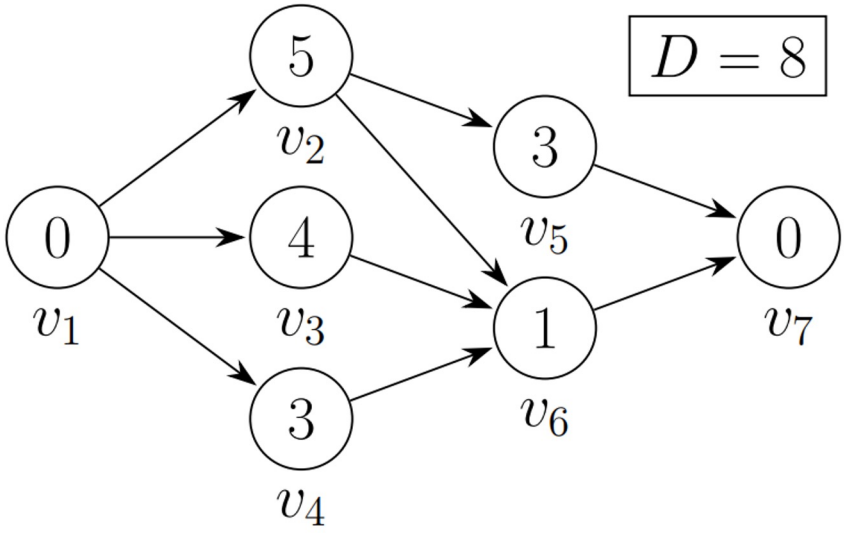


$$\rightarrow \quad a = \arg\max_a \pi_\theta(a \mid s)$$



- **PPO uses an approximation of the Value function to "guide" its learning process**
- **PPO's actor relies on "specialized clipping" in its objective for more stable learning**
- **Critic is trained by using supervised regression with L_2 norm**

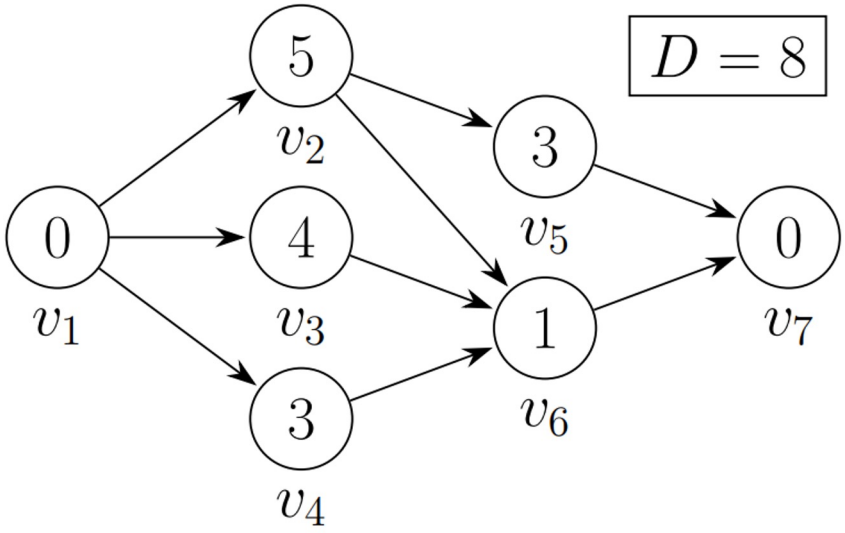
DRL applied to RT Resource Management:
DAG scheduling problem



DAG scheduling problem consists of scheduling nodes on M processors such that all the nodes can finish before a deadline D

Typical solution: list scheduling (i.e., work-conserving algorithm based on priorities)

Trivial Schedulability

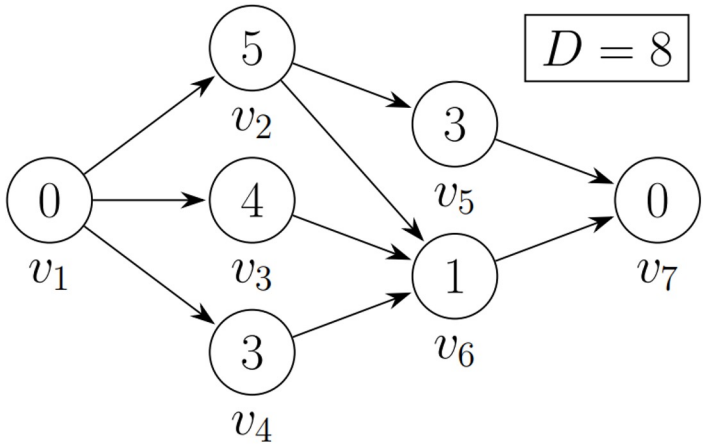


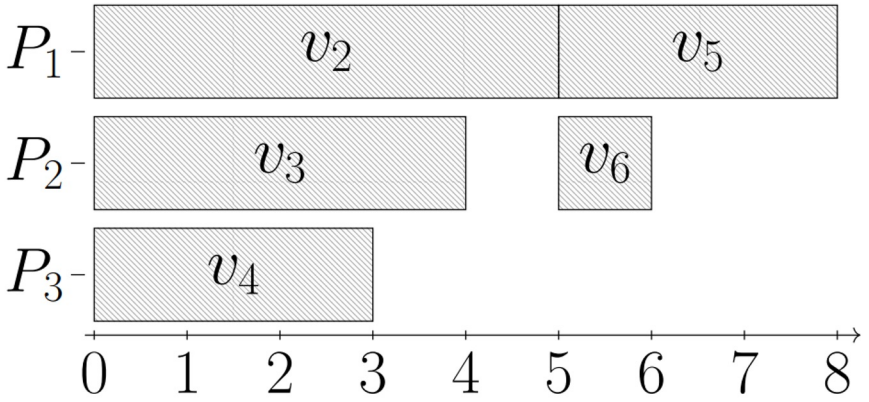
Definition

A DAG task $(\mathcal{G}, D)$ is *trivially schedulable* on M processors if it satisfies the following two conditions:

1) $\mathrm{L}(\mathcal{G}) \leq D$
2) $\mathrm{W}(\mathcal{G}) \leq M$

Trivial Schedulability



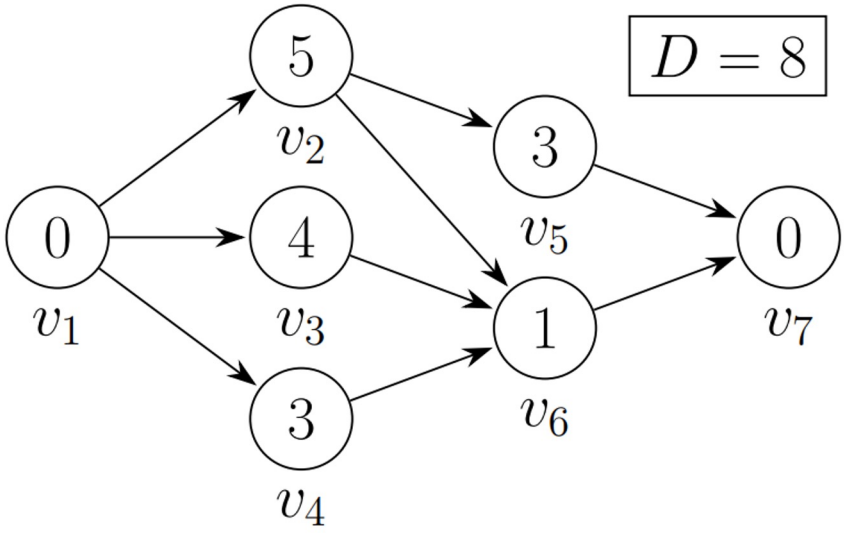


Definition

A DAG task $(\mathcal{G}, D)$ is *trivially schedulable* on M processors if it satisfies the following two conditions:

1) $\mathrm{L}(\mathcal{G}) \leq D$
2) $\mathrm{W}(\mathcal{G}) \leq M$

Trivial Schedulability



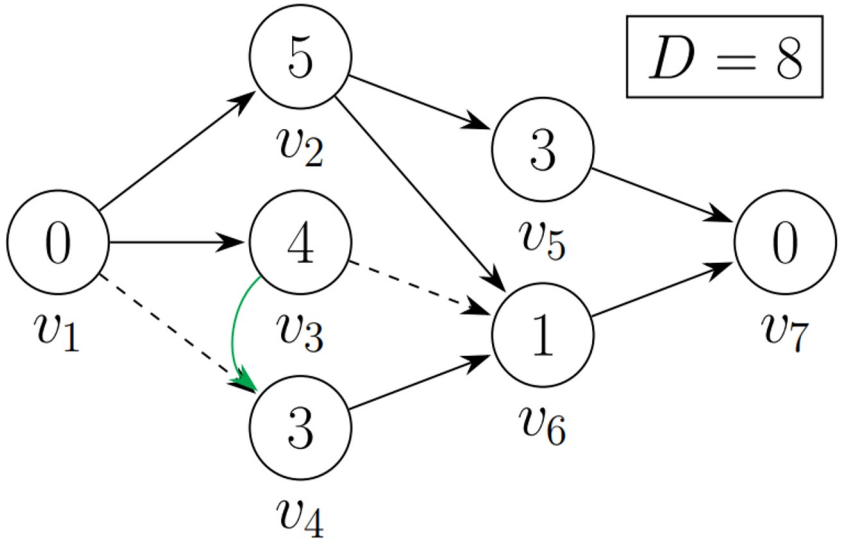
Is this DAG task also schedulable on 2 processors?

Definition

A DAG task $(\mathcal{G}, D)$ is *trivially schedulable* on M processors if it satisfies the following two conditions:

1) $\mathrm{L}(\mathcal{G}) \leq D$
2) $\mathrm{W}(\mathcal{G}) \leq M$

Trivial Schedulability



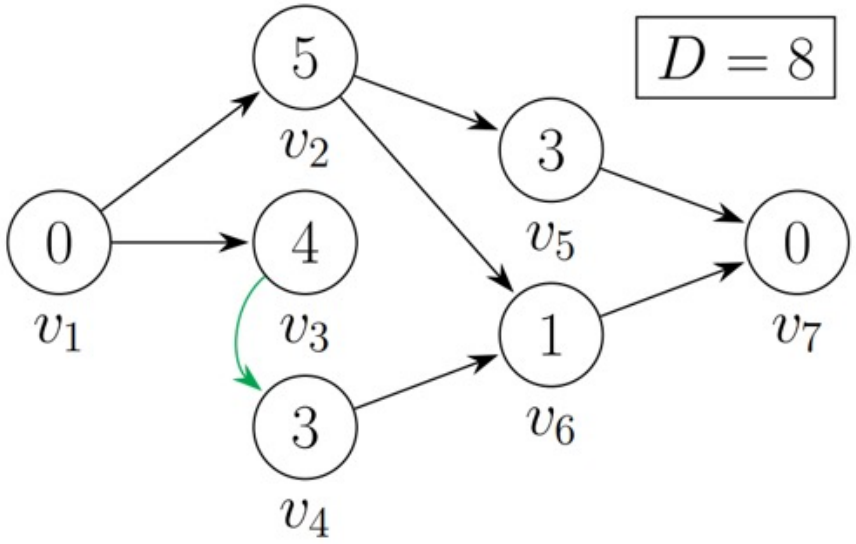
Is this DAG task also schedulable on 2 processors?

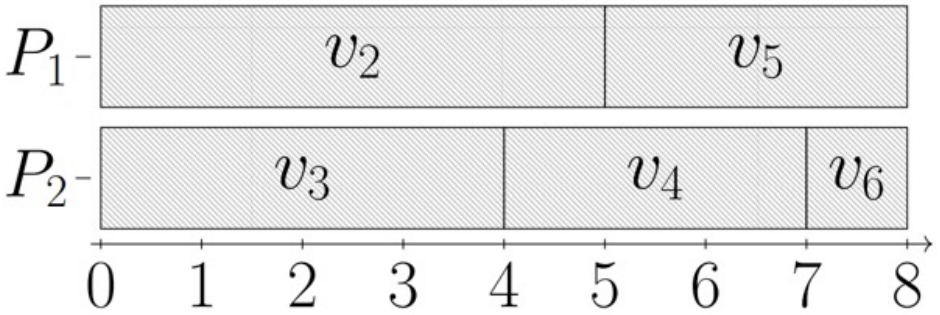
Definition

A DAG task $(\mathcal{G}, D)$ is *trivially schedulable* on M processors if it satisfies the following two conditions:

1) $\mathrm{L}(\mathcal{G}) \leq D$
2) $\mathrm{W}(\mathcal{G}) \leq M$

Trivial Schedulability





Definition

A DAG task $(\mathcal{G}, D)$ is *trivially schedulable* on M processors if it satisfies the following two conditions:

1) $\mathrm{L}(\mathcal{G}) \leq D$
2) $\mathrm{W}(\mathcal{G}) \leq M$

Edge Generation Scheduling (EGS)

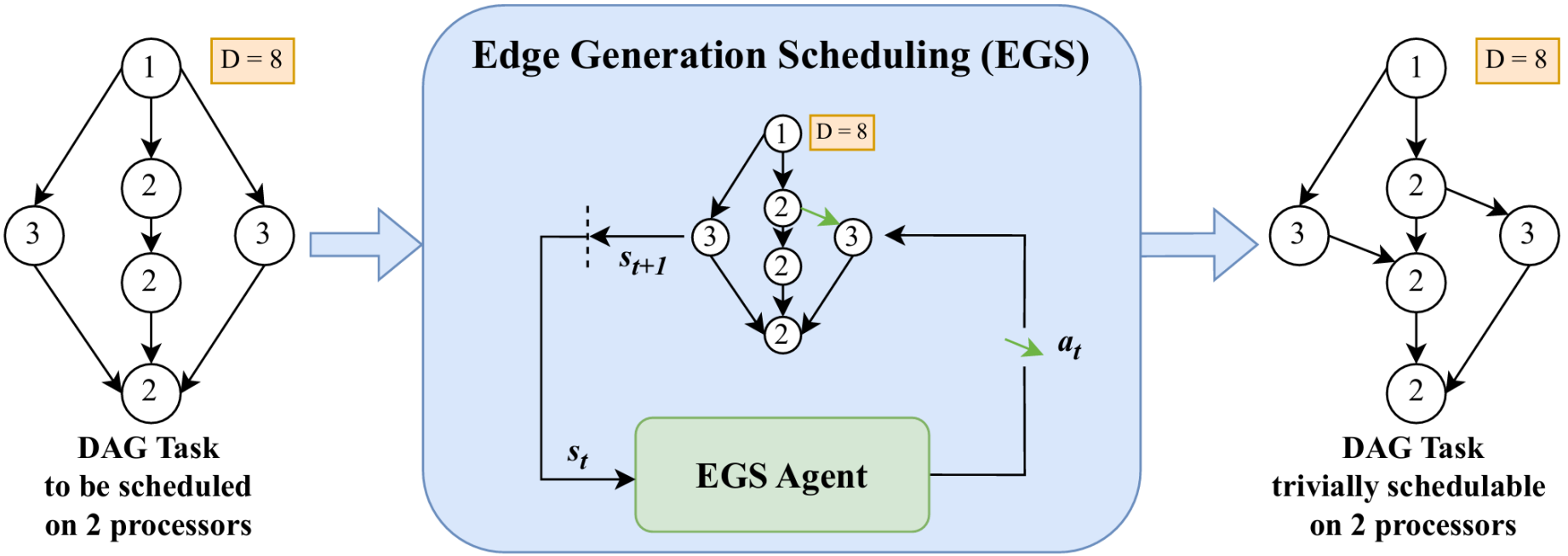
- **The goal of this work is to learn an EGS policy that can be effective at solving the following optimization problem:**

Optimization Problem

$$\begin{aligned} \underset{\mathcal{G}' \supseteq_E \mathcal{G}}{\text{minimize}} \quad & \mathrm{W}(\mathcal{G}') \\ \text{subject to} \quad & \mathrm{L}(\mathcal{G}') \leq D \end{aligned}$$

The policy will add edges to the original DAG until no edges can be added without violating deadline D or current DAG width reaches its lower bound

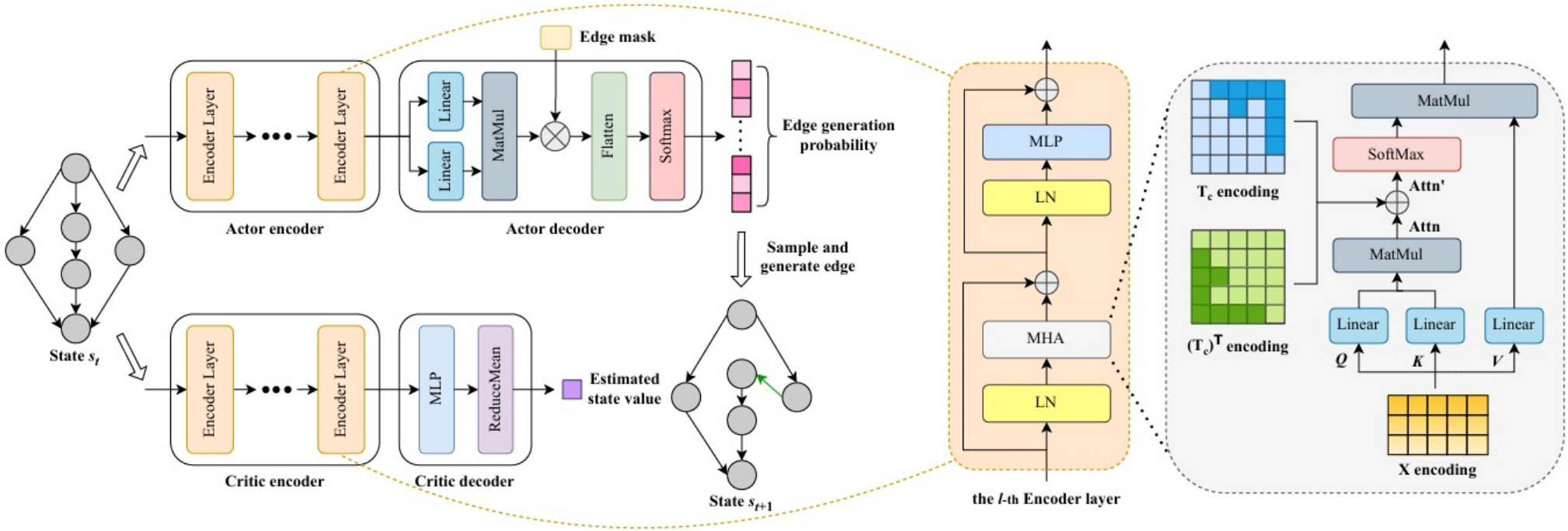
Edge Generation Scheduling



To minimize number of cores needed by the DAG, we use following Reward function:

$$\mathrm{R}(s_t, a_t) = \mathrm{W}(\mathcal{G}_t) - \mathrm{W}(\mathcal{G}_{t+1})$$

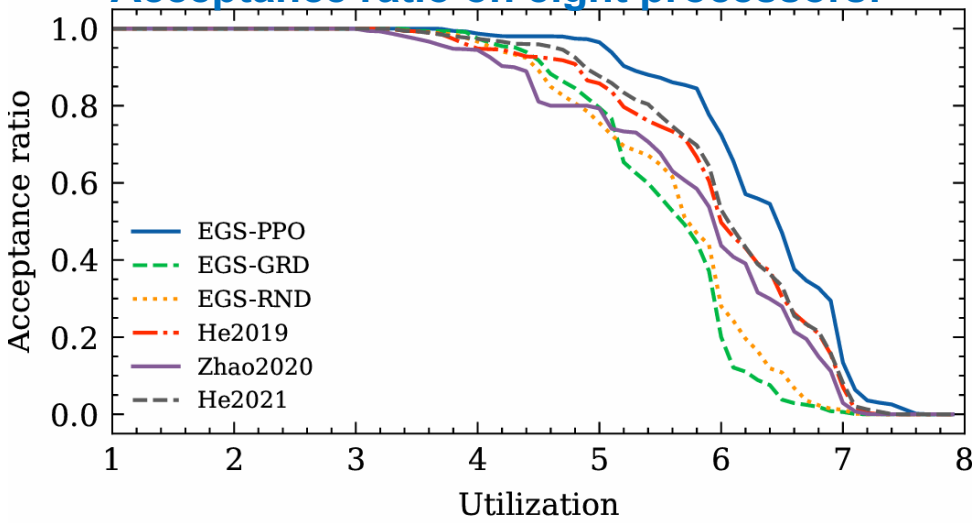
DRL with Proximal Policy Optimization (PPO)

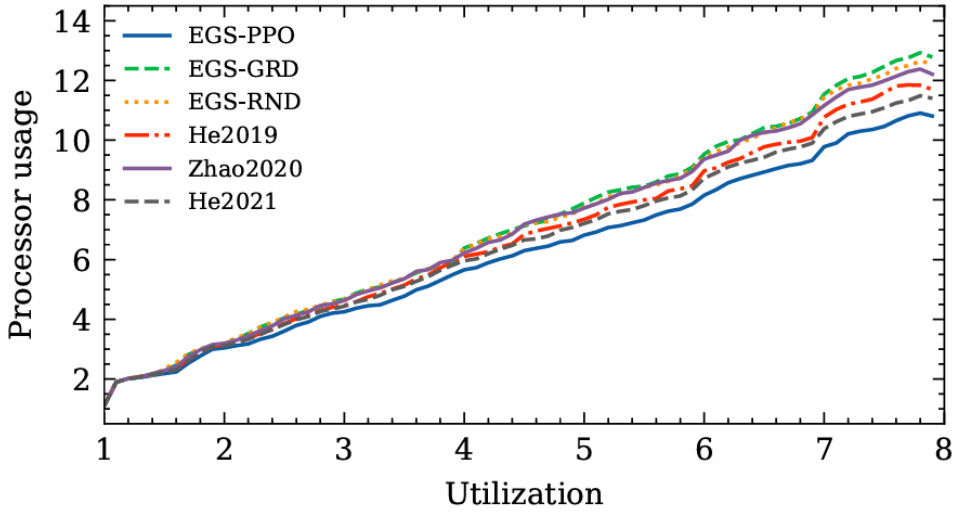


Evaluation Results

Sun B, Theile M, Qin Z, Bernardini D, Roy D, Bastoni A, Caccamo M. Edge Generation Scheduling for DAG Tasks using Deep Reinforcement Learning. IEEE Transactions on Computers. 2024

Acceptance ratio on eight processors:





Q.He et al., “Intra-task priority assignment in real-time scheduling of DAG tasks on multi-cores,” TPDS, 2019

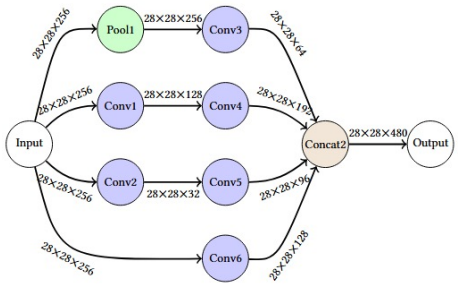
S.Zhao,X.Dai, I.Bate,A.Burns, and W.Chang, “DAG scheduling and analysis on multiprocessor systems: Exploitation of parallelism and dependency,” in RTSS, 2020

Q.He,M. Lv, andN.Guan, “Response time bounds for DAG tasks with arbitrary intra-task priority assignment,” in ECRTS, 2021

Example of Future Research

Example: DNN Pipelining on Hardware Accelerators*

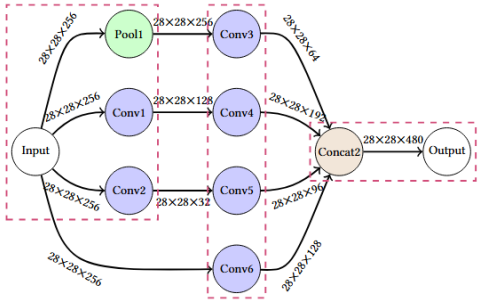
Input: An original neural network



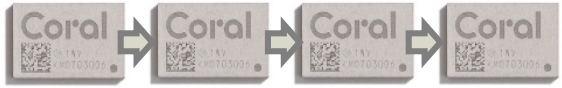
DRL Agent

DRL Training

Output: A partitioned neural network



Deploy



Edge TPU Accelerator Pipeline

Performance Profiler
(latency, throughput, etc.)
➔ Reward generation

*B Sun, B Zou, Y Hu, T Kloda, L Wang, T Abdelzaher, M Caccamo, "SAPar: A Surrogate-Assisted DNN Partitioner for Efficient Inferences on Edge TPU Pipelines", in EMSOFT, 2025.

Conclusions

- RL is a framework for sequential decision-making in which an agent interacts with the environment to *maximize its expected cumulative reward*.
- Unlike imitation learning, *RL does not depend on expert demonstrations*. In fact, *RL enables autonomous policy discovery through trial and error*.
- We are exploring the potential of DRL applied to real-time resource management
  - ➔ DRL does not guarantee globally optimal solutions but instead it produces learned heuristics

- *DRL has some limitations too*:
  - ➢ *It lacks formal guarantees (however, DAG RT constraints are enforced through "**action masking**")*
  - ➢ *It demands problem-specific MDP design*
  - ➢ *It can incur high training costs*, especially when real hardware interaction is needed.