Course Project

Code of Honor. All external resources used in the project, including research papers, open-source repositories, datasets, and any content or code generated using AI tools, e.g., ChatGPT, GitHub Copilot, Claude, Gemini, must be *clearly cited* in the final submission. The final report must also include *a clear breakdown of individual group member contributions.* Any lack of transparency in the use of external resources or in reporting group contributions will be considered academic dishonesty and will significantly impact the final evaluation.

Topic	RL for Job Scheduling in a Compute Cluster
Category	Resource Allocation

Objective Design and implement an RL agent for job scheduling in a simulated compute cluster. The agent should decide which jobs to execute in order to optimize performance metrics such as throughput, average waiting time, or fairness. The project aims to compare RL-based scheduling policies with classical heuristics such as First-Come-First-Served (FCFS) or Shortest-Job-First (SJF).

Motivation Job scheduling is a fundamental resource allocation problem in computer systems and datacenters. Traditional scheduling policies such as FCFS and SJF are simple but cannot adapt to dynamic workloads with varying arrival rates and resource requirements. Reinforcement learning provides a framework for adaptive scheduling, where the agent can learn to make better decisions under uncertainty [2, 3, 4]. This project introduces students to RL applications in systems and networking, connecting algorithmic design with real-world ECE problems.

Requirements The final submission should address the following requirements while the details can be freely decided by the group members.

1. Implementation: in this respect, you should
   - design a simple cluster simulation where jobs arrive randomly with different lengths and resource requirements; see, for instance, the reference environment in [1],
   - implement an RL agent (e.g., DQN, PPO) that chooses which job to schedule next, and
   - implement baseline policies (e.g., FCFS, SJF) for comparison.
2. Environment modification: to ensure originality, you **must** extend a pre-implemented environment with **at least one** of the following modifications:
   - introduce *bursty* job arrivals or variable workloads,
   - add queue size limits (job dropping),
   - penalize long jobs, or
   - add multiple servers with heterogeneous speeds.
3. Evaluation: the final project should report key evaluation metrics in the modified environment. In this respect, the results should

- compare RL scheduling policies with baseline heuristics,
- analyze average job completion time, queue length, and fairness, and
- report robustness under different workload patterns.

4. The results should be elaborated through
   - ablation experiments, e.g., different reward designs (throughput vs fairness), and
   - providing discussions on trade-offs between simplicity and adaptation.

MILESTONES The following milestones are to be accomplished through semester.

1. Literature Review and Setup
   - Review RL applications in scheduling. In this respect, you are suggested to focus on some line of work that can be feasibly implemented (see references to get some idea).
   - Design a simple job scheduling simulation.
   - Finalize workload models and environment modifications.
2. Implementation
   - Implement baseline scheduling policies (FCFS, SJF).
   - Implement RL agent (DQN or PPO) and validate on a simple workload.
   - Train on modified environments.
3. Evaluation and Analysis
   - Collect and plot scheduling performance metrics (throughput, waiting time).
   - Compare RL agent vs baselines under different workloads.
   - Perform ablation experiments (reward variations, workload variations).
4. Final Report and Presentation

SUBMISSION GUIDELINES The main body of work is submitted through Git. In addition, each group submits a final paper and gives a presentation. In this respect, please follow these steps.

- Each group must maintain a Git repository, e.g., GitHub or GitLab, for the project. By the time of final submission, the repository should have
  - Well-documented codebase
  - Clear `README.md` with setup and usage instructions
  - A `requirements.txt` file listing all required packages or an `environment.yaml` file with a reproducible environment setup
  - Demo script or notebook showing sample input-output
  - *If applicable,* a `/doc` folder with extended documentation
- A final report (maximum *5 pages*) must be submitted in a PDF format. The report should be written in the provided formal style, including an abstract, introduction, method, experiments, results, and conclusion.
  Important: Submissions that do not use template are considered *incomplete.*
- A 5-minute presentation (maximum *5 slides including the title slide*) is given on the internal seminar on Week 14, i.e., *Dec 1 to Dec 5*, by the group. For presentation, any template can be used.

Final Notes While planning for the milestones please consider the following points.

1. Creativity in modifying the workload model and designing reward functions is encouraged.
2. Training should remain feasible by keeping the simulated cluster small (e.g., 1–2 servers, a few queues).
3. Students are expected to balance research-oriented exploration with achievable implementation.
4. Teams are expected to manage their computing needs and are advised to perform early tests to estimate runtime and training feasibility. As graduate students, team members can use facilities provided by the university, e.g., ECE Facility. Teams are expected to inform themselves about the limitations of the available computing resources and design accordingly.

References

[1] Hongzi Mao. Deep RM. `https://github.com/hongzimao/deeprm?tab=readme-ov-file`, 2017.

[2] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM workshop on hot topics in networks*, pages 50–56, 2016.

[3] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM special interest group on data communication*, pages 270–288. 2019.

[4] Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in neural information processing systems*, 33:1621–1632, 2020.