Scheduling Jobs on Multi-Instance GPUs (MIG) using Reinforcement Learning

An Improved Implementation and Analysis

Tanvi Hisaria, Devyani Vij, Aayush Srivastava

Technical Report for COMS E6998: Reinforcement Learning
Columbia University in the City of New York

December 7, 2025

1 The Challenge: Optimizing for Energy and Performance on MIG

1.1 Context

NVIDIA's Multi-Instance GPU (MIG) technology partitions a single GPU into up to seven isolated "slices." This allows multiple jobs to run concurrently, improving utilization for smaller AI/ML workloads.

- Data centers consumed 4.4% of total US electricity in 2023
- Projections suggest this could rise to 12% by 2028
- Efficient GPU scheduling is critical for sustainable AI infrastructure

1.2 The Core Trade-Off

Energy Efficiency: Achieved by packing the GPU with jobs to maximize utilization. Power consumption is *concave*—using 4/7 of a GPU consumes nearly the same power as using 7/7.

Tardiness: A measure of how late past the deadline a job completes. Packing more jobs onto smaller slices can increase their runtime, negatively impacting tardiness.

Key Insight

Objective: Design a scheduling agent that minimizes a weighted measure of Energy and Tardiness—the **ET metric**.

2 Hypothesis 1: A Standard PPO Formulation

Based on the initial proposal, we model the scheduling problem as a Markov Decision Process (MDP) and apply Proximal Policy Optimization (PPO).

2.1 State Space ($\mathcal{S}$)

A vector representing the current environment state:

- Characteristics of the next job to be scheduled (e.g., deadline, expected runtimes)
- Statistics of the job queue (e.g., histograms of job durations)
- Binary status of GPU slices (busy or free)

2.2 Action Space ($\mathcal{A}$)

A discrete space representing all available GPU slices:

- `Action` = Assign job j to slice i
- Action masking is used to prevent assignment to occupied slices

2.3 Reward Function (R)

A sparse, terminal reward calculated only at the end of an episode (after all jobs are scheduled):

$$R = \frac{-\text{total\_tardiness} - \alpha \cdot \text{total\_energy}}{\text{num\_jobs} \cdot \alpha + 1} \tag{1}$$

where α is a scaling parameter. **No reward is given for intermediate actions.**

2.4 Goal

Train a PPO agent with this formulation to outperform deterministic greedy algorithms.

3 Initial Results: RL Agent Fails

When evaluated on a simulated 24-hour workload, the initial PPO agent produced unexpectedly poor results:

Metric	Result
Late Jobs %	∼87%
PPO Agent vs. Best Heuristic	Loses by ∼5%

Table 1: Initial PPO agent performance

Key Insight

The Central Question: Why did a sophisticated learning algorithm fail against a basic, hand-coded strategy? The formulation, which seemed logical, was fundamentally flawed.

4 Diagnosis: Four Critical Flaws

A systematic analysis revealed that the problem was not the algorithm (PPO) but the environment and problem definition provided to it.

4.1 Flaw 1: The Problem Was Greedy-Optimal

Cause: Extremely tight deadlines, set at uniform$(1.0, 1.5) \times t_{\text{fastest}}$

Effect: The only viable strategy was to always pick the largest available slice to minimize runtime. There was no room for intelligent trade-offs, making the problem trivial for a greedy heuristic but difficult for RL to discover.

4.2 Flaw 2: The Agent Was Flying Blind

Cause: The observation space $\mathcal{S}$ contained the binary status of slices (busy/free) but critically **lacked their sizes** (e.g., 1g, 4g, 7g).

Effect: The agent could not learn the fundamental concept that larger slices are faster. It could not distinguish between assigning a job to a fast 7g slice or a slow 1g slice.

4.3 Flaw 3: The Feedback Loop Was Broken

Cause: A sparse, end-of-episode reward signal.

Effect: The agent received feedback only after scheduling hundreds of jobs, making it nearly impossible to assign credit or blame to any of the individual scheduling decisions that led to the final outcome.

4.4 Flaw 4: Experimentation Was Impractical

Cause: The simulation environment was implemented in Pandas.

Effect: Slow performance bottlenecked training and iteration, making systematic improvement difficult.

5 A Revised Hypothesis: Systematic Enhancements

We addressed each identified flaw with a specific enhancement to the environment, observation, and reward structure.

Table 2: Original vs. Enhanced Approach

Flaw	Original (Hypothesis 1)	Enhanced (Hypothesis 2)
Greedy-Optimal Problem	Deadlines: uniform$(1.0, 1.5) \times t_{\text{fastest}}$	Relaxed Deadlines: uniform$(2.0, 4.0) \times t_{\text{fastest}}$
Agent is Blind	Observation: Missing slice sizes	Enriched Observation: Added slice sizes, job "urgency," and max available slice size
Broken Feedback	Reward: Sparse, terminal only	Shaped Rewards: Added immediate, per-step rewards
Slow Experimentation	Environment: Pandas-based	Optimized Environment: NumPy for 10-50× speedup
Training Protocol	200k steps, fixed learning rate	Advanced Training: 500k steps, deeper network, LR annealing, entropy decay

6 Formalizing the Enhancements

6.1 Enhanced Observation Space ($\mathcal{S}'$)

We augment the state with critical information:

- **Normalized Slice Sizes:** A vector indicating the relative size of each slice.
- **Urgency Ratio:** A continuous value measuring how close a job is to its deadline:

$$\text{urgency} = \min\left(10, \frac{t_{\text{fastest}}}{\max(t_{\text{deadline}} - t_{\text{now}}, \epsilon)}\right) \tag{2}$$

where ϵ is a small constant to prevent division by zero.

6.2 Immediate Reward Shaping (R')

We introduce a dense, immediate reward $r_{\text{immediate}}$ awarded at each step t *in addition* to the terminal reward:

$$r_{\text{immediate}} = 0.01 \cdot \frac{\text{slice\_size}}{7} + r_{\text{on-time}} \tag{3}$$

where:

$$r_{\text{on-time}} = \begin{cases} 0.05 & \text{if expected\_finish} < \text{deadline} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

- The first term encourages using larger, more efficient slices
- The second term rewards actions projected to meet the deadline

7 Validation: Enhanced RL Agent Outperforms Baselines

After training for 500,000 timesteps with the enhanced formulation, the new agent demonstrates significant performance improvement.

Table 3: Final Performance Comparison (Averaged over multiple runs)

Method	Late Jobs (%)↓	Avg. Tardiness (s)↓	Energy (MJ)
RL-PPO (Enhanced)	**37.7 ± 5.7**	**0.91 ± 0.48**	2.50 ± 0.04
Largest-First (Best Heuristic)	42.7 ± 5.7	1.02 ± 0.73	2.51 ± 0.05
EFT (Earliest Finish Time)	43.1 ± 5.6	1.04 ± 0.63	2.48 ± 0.05
Smallest-First	54.3 ± 4.7	1.21 ± 0.58	2.42 ± 0.05
Random	50.1 ± 4.7	1.13 ± 0.57	2.49 ± 0.04

Result

Key Observations:

- The RL agent achieves the **lowest percentage of late jobs** and the **lowest average tardiness**
- It reduces late jobs by **11.7%** relative to the strongest heuristic (Largest-First)
- Energy consumption remains comparable to other methods, demonstrating an effective balance in the energy-tardiness trade-off

8 The Decisive Result

The primary goal was to create a learning agent that could discover a policy superior to fixed, greedy approaches. The enhanced formulation achieved this.

Best Heuristic: Largest-First

42.7%

Late Jobs

Our Enhanced RL Agent

37.7%

Late Jobs

+11.7% On-Time Jobs

Key Insight

By providing the agent with the right information (slice sizes), the right incentives (immediate rewards), and a meaningful problem space (relaxed deadlines), it successfully learned a non-trivial scheduling policy that outperforms the greedy strategy that dominated the original, flawed problem.

9 Theoretical Context: Lower Bounds

To rigorously benchmark our system's performance, we derive theoretical lower bounds for energy and tardiness.

9.1 Methodology

9.1.1 1. Linearization

We first simplify the problem by converting all jobs with sublinear throughput curves to their more efficient "linearized" counterparts.

$L(j)$ is a job with processing time $p_{j,i} = p_{j,1}/i$ on a slice of size i.

9.1.2 2. Configuration Simplification

We prove that any schedule on m MIGs with various configurations can be converted to a schedule on m GPUs, each with a single 7g slice, without increasing energy or tardiness.

9.2 Mathematical Formulation of Bounds

9.2.1 Energy Lower Bound

Reduced to a minimum makespan problem on uniform machines, which is polynomially solvable:

$$E_{\text{opt}}(L(J)) = \sum_j (p_{j,7} \cdot P_7) + (m \cdot C^* - \sum_j a_j) \cdot P_0 \tag{5}$$

9.2.2 Tardiness Lower Bound

Formulated as a time-indexed Mixed Integer Program (MIP):

$$\text{Objective: } \min \sum_j T_j \tag{6}$$

subject to processing, machine capacity, and release time constraints.

10 The Key Insight: RL Thrives on Information and Meaningful Decisions

This investigation reveals that the success of a reinforcement learning agent is critically dependent on the formulation of the problem itself.

Key Insight

The Bottom Line: Reinforcement learning is not a black box. Its ability to outperform heuristics is unlocked only when the environment is designed to make learning both *possible* and *necessary*.

11 Conclusion

By diagnosing and systematically correcting the core issues in the initial RL formulation, we successfully developed a scheduling agent that learns a policy superior to strong heuristic baselines for the MIG scheduling problem.

Table 4: When RL Fails vs. When RL Succeeds

When RL Fails...	**When RL Succeeds...**
The problem is "greedy-optimal." Tight constraints leave no room for optimization.	The problem has "slack." Relaxed constraints create a complex trade-off space.
The observation is incomplete. The agent lacks necessary information.	The observation is informative. All relevant state variables are provided.
Rewards are sparse and delayed. Feedback is disconnected from actions.	Rewards are immediate and shaped. Direct, per-step feedback guides learning.

Table 5: Supporting Data: Impact of Problem Formulation

Configuration	**Deadline Slack**	**RL Late %**	**RL vs. Best Heuristic**
Original	1.0–1.5×	∼87%	Loses by ∼5%
Enhanced	2.0–4.0×	37.7%	**Wins by 11.7%**

11.1 Summary of Contributions

1. **A 10-50× Faster Simulation Environment:** Re-implementation in NumPy enabled rapid experimentation and extensive training.
2. **Demonstrated RL Superiority:** Achieved an **11.7% reduction** in late jobs compared to the best-performing heuristic baseline.
3. **Formal Analysis of Problem Structure:** Identified and proved how overly tight deadlines can render a complex scheduling problem "greedy-optimal," making RL ineffective.
4. **Principled Agent Design:** Developed an enhanced observation space and immediate reward shaping function, providing a template for applying RL to similar resource management tasks.

"This work underscores a fundamental principle: in applied reinforcement learning, the careful formulation of the problem is as important as the learning algorithm itself."

References

1. E. Lipe, N. Karia, C. Espenshade, C. Stein, A. Tantawi and O. Tardieu, "Energy Efficient Scheduling of AI/ML Workloads on Multi Instance GPUs with Dynamic Repartitioning," *2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, Tromsø, Norway, 2025, pp. 53-62. DOI: 10.1109/CCGRID64434.2025.00066.