COLUMBIA | ENGINEERING

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

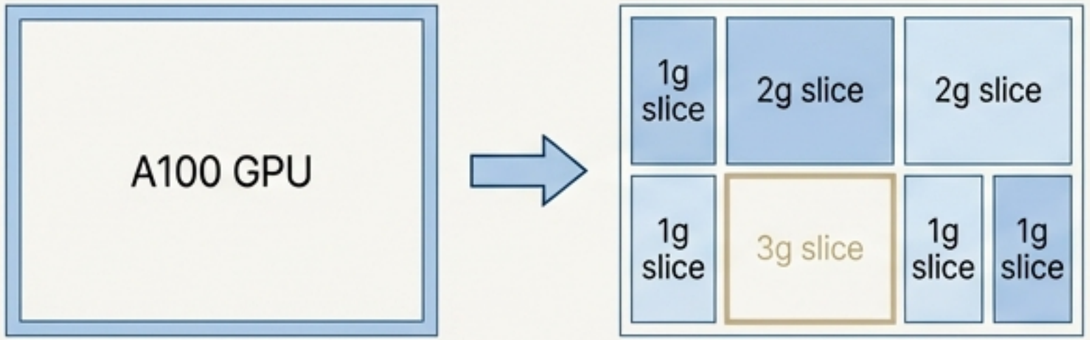
The Challenge: Optimizing for Energy and Performance on Multi-Instance GPUs

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, with a projected increase to 12% by 2028. Efficient scheduling is critical.



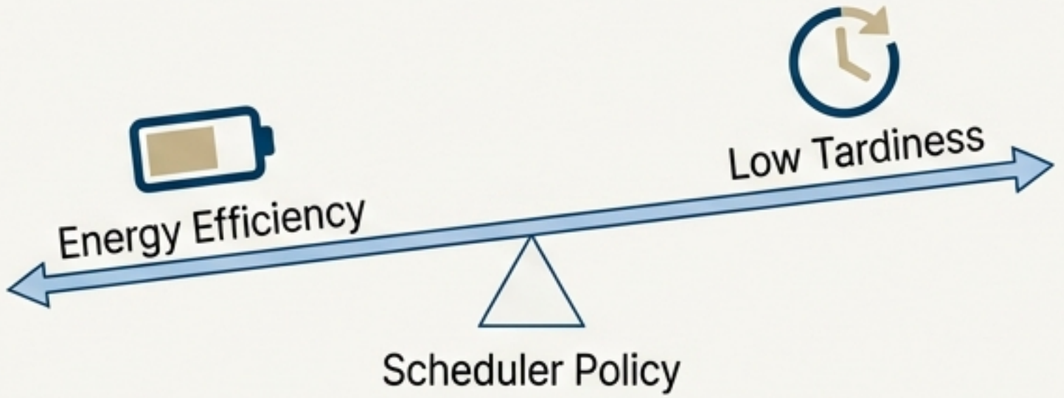
The Core Trade-Off

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 far past its deadline a job completes. Packing more jobs onto smaller slices can increase their runtime, negatively impacting tardiness.



Objective

To design a scheduling agent that minimizes a weighted measure of Energy and Tardiness, the **ET metric**.

Hypothesis 1: A Standard PPO Formulation for MIG Scheduling

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

S

State Space (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).

A

Action Space (A): A discrete space representing all available GPU slices. The agent selects one slice to assign the current job.

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

R

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

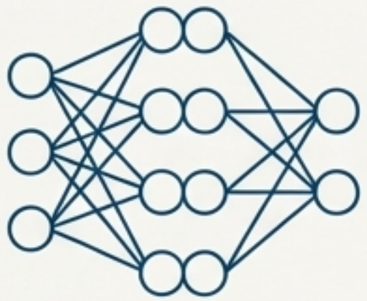
$$R = \frac{-total\_tardiness - \alpha * total\_energy}{num\_jobs * \alpha + 1}$$

- Where `$\alpha$` is a scaling parameter. No reward is given for intermediate actions.

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

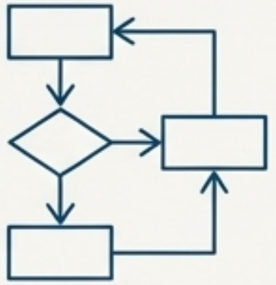
Initial Results: The RL Agent Fails to Outperform Simple Heuristics

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



PPO Agent

Metric	Result
Late Jobs %	~87%
vs. Best Heuristic	Loses by ~5%



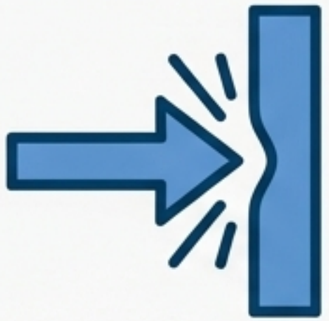
Simple Heuristic

The Central Question:

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

Diagnosis: Identifying the Four Critical Flaws in the Formulation

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



The Problem Was Greedy-Optimal

Cause: Extremely tight deadlines, set at *uniform*(1.0, 1.5) × t_*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.

The Agent Was Flying Blind

Cause: The observation space *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.

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.

Experimentation Was Impractical

Cause: The simulation environment was implemented in Pandas.

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

A Revised Hypothesis: Systematically Enhancing the RL Formulation

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

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

Formalizing the Enhancements I: A Richer State and Denser Reward

To empower the agent, we mathematically redefined the observation and reward functions.

1. Enhanced Observation Space (S')

We augment the state with critical information. The new features include:

- **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, forcing the agent to prioritize. Defined as:

$$\text{urgency} = \min\left(1.0, \frac{t_{\text{fastest}}}{\max(t_{\text{deadline}} - t_{\text{now}}, \epsilon)}\right)$$

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

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. This provides a direct feedback signal.

$$r_{\text{immediate}} = \left(0.01 \cdot \frac{\text{slice\_size}}{7}\right) + r_{\text{ontime}}$$

where:

$$r_{\text{ontime}} = \begin{cases} 0.05 & \text{if expected\_finish} \leq \text{deadline} \\ 0 & \text{otherwise} \end{cases}$$

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

Validation: The Enhanced RL Agent Outperforms All Heuristic Baselines

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

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

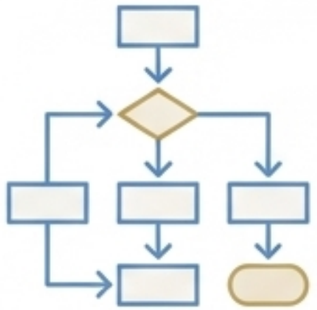
Key Observations

- 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.

The Decisive Result: An 11.7% Improvement Over the Strongest Heuristic

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

+11.7%

Relative Improvement in On-Time Jobs

Our Enhanced RL Agent

37.7%

Late Jobs

Conclusion: 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.

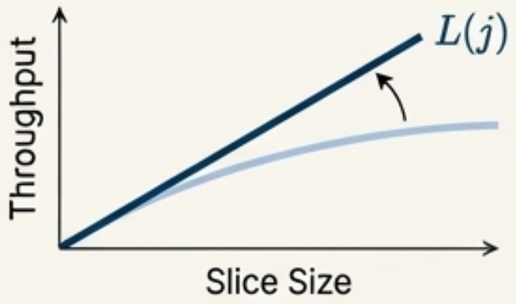
Theoretical Context: Establishing Lower Bounds for Energy and Tardiness

To rigorously benchmark our system's performance, we derive theoretical lower bounds for energy and tardiness. These bounds represent an idealized optimum that may not be achievable in practice but serve as a crucial reference point.

Methodology

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.

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.

Mathematical Formulation of Bounds

Energy Lower Bound

Reduced to a minimum makespan problem on uniform machines, which is polynomially solvable. The total energy is a function of the minimum makespan C^*:

$$E_{\text{LB}}(L(J)) = \sum(p_{j,7} \cdot P_7) + \left(mC^* - \sum p_{j,7}\right) * P_0$$

Tardiness Lower Bound

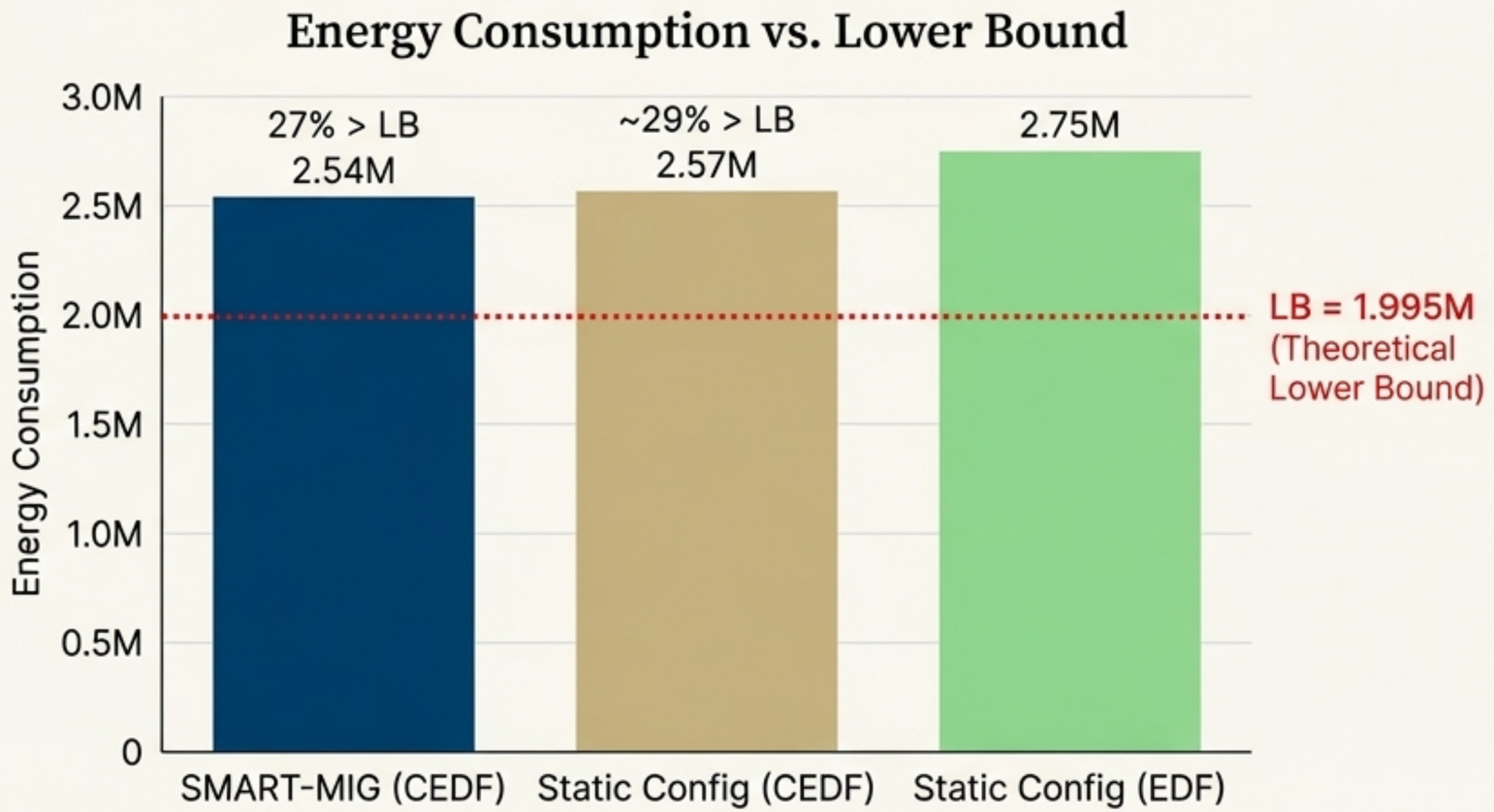
Formulated as a time-indexed Mixed Integer Program (MIP), which is NP-hard and solved numerically to find the minimum possible total tardiness.

$$\text{Objective: } \min \sum T_j$$

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

Performance in Perspective: Closing the Gap to the Theoretical Optimum

When compared against the derived lower bounds, our dynamic repartitioning framework demonstrates strong performance, operating within a reasonable margin of the idealized best-case scenario.



Key Insight:

Considering that the lower bound is not tight (due to job linearization) and our primary goal is multi-objective optimization (balancing energy *and* tardiness), this result indicates highly effective energy management. management. The dynamic repartitioning learns to adapt to workloads, achieving a better trade-off than any fixed configuration.

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.

Main Comparison Table

When RL Fails...	When RL Succeeds...
The problem is 'greedy-optimal." Tight constraints (e.g., deadlines `1.0-1.5×`) leave no room for optimization. A simple heuristic is sufficient.	**The problem has "slack."** Relaxed constraints (e.g., deadlines `2.0-4.0×`) create a complex trade-off space where learning is valuable.
The observation is incomplete. The agent lacks the necessary information to make correct decisions (e.g., cannot see slice sizes).	**The observation is informative.** The agent is provided all relevant state variables to learn the underlying dynamics (e.g., slice sizes, urgency).
Rewards are sparse and delayed. Feedback is disconnected from the actions that caused it, making credit assignment impossible.	**Rewards are immediate and shaped.** The agent receives direct, per-step feedback that guides it toward desirable behaviors.

Supporting Data Table

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%**

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.

Conclusion: From Flawed Formulation to Superior Performance

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.

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."