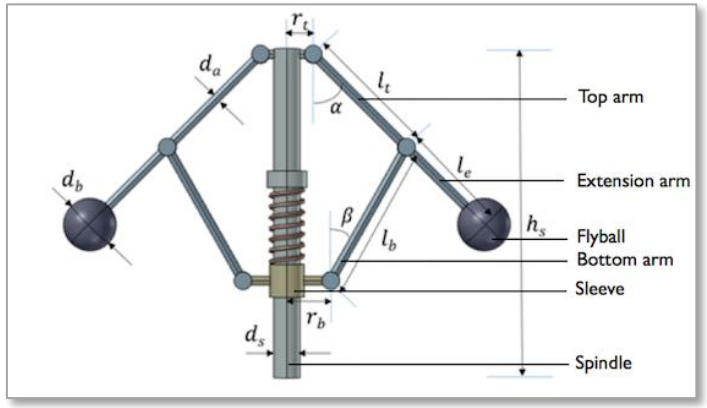
Covering the Pareto Frontier with LLM-Coordinated Interpretable Policy Library

Ni Mu, Yao Luan, Qing-Shan Jia
Tsinghua University
mn23@mails.tsinghua.edu.cn

1.1 Rethink “Industrial Automation”: A 200-Year Journey

- **1788 | *Watt's Centrifugal Governor*** [1]



Workers manually adjust the speed

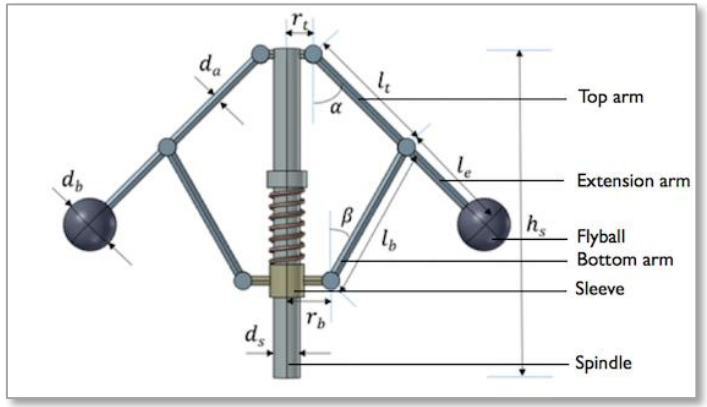
⇩

Machines automatically keep stable

Externalize **Physical Labor**

1.1 Rethink “Industrial Automation”: A 200-Year Journey

➢ **1788 | *Watt's Centrifugal Governor*** [1]



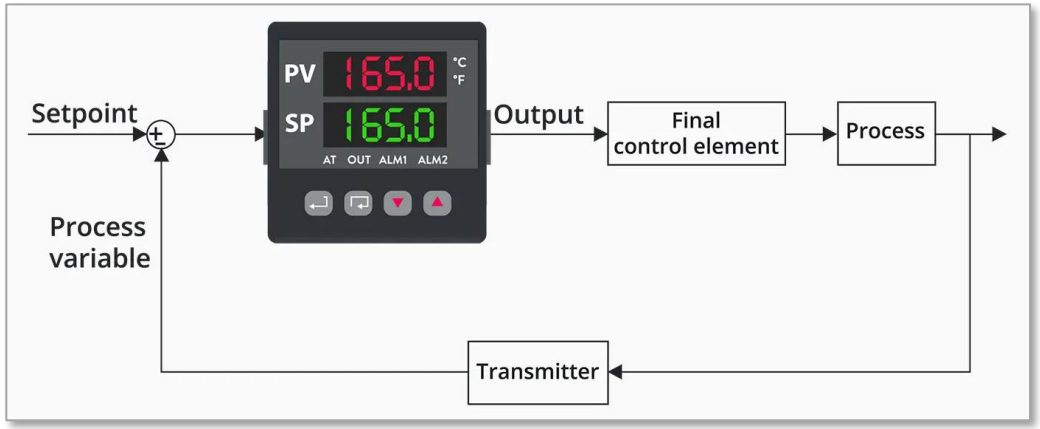
Workers manually adjust the speed

⇩

Machines automatically keep stable

Externalize **Physical Labor**

➢ **1920s+ | *PID Controllers & PLCs*** [2-3]

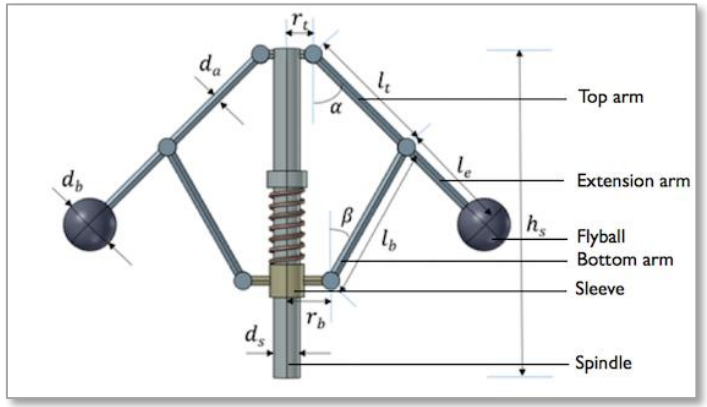


Encoding expert knowledge into fixed rules

Externalize **Human Experience**

1.1 Rethink “Industrial Automation”: A 200-Year Journey

➢ **1788 | *Watt's Centrifugal Governor*** [1]



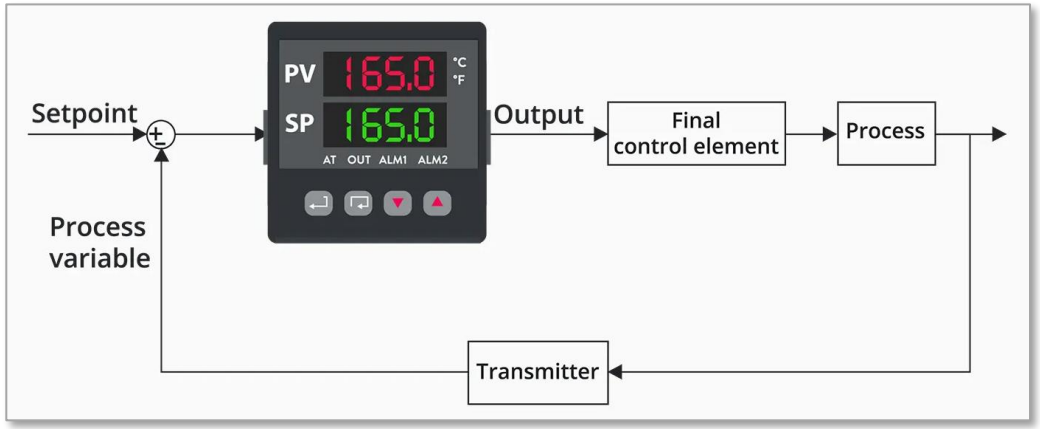
Workers manually adjust the speed

⇩

Machines automatically keep stable

Externalize **Physical Labor**

➢ **1920s+ | *PID Controllers & PLCs*** [2-3]



Encoding expert knowledge into fixed rules

Externalize **Human Experience**

But both share one hidden assumption...

1.2 The *Trilemma* in Control Policy Discovery

The hidden assumption:

- An expert must stay in the loop --- writing rules, deriving formulas, or labeling data.

"Automation" is pre-programmed by experts.

1.2 The *Trilemma* in Control Policy Discovery

The hidden assumption:

- An expert must stay in the loop --- writing rules, deriving formulas, or labeling data.

⇩

"Automation" is pre-programmed by experts.

It causes:

- When, e.g.,
  - Objectives change
  - Constraints change
  - New product lines launch
- We must bring the expert back and redesign from scratch. [4]

1.2 The *Trilemma* in Control Policy Discovery

The hidden assumption:

- An expert must stay in the loop --- writing rules, deriving formulas, or labeling data.

⇩

"Automation" is pre-programmed by experts.

It causes:

- When, e.g.,
  - Objectives change
  - Constraints change
  - New product lines launch
- We must bring the expert back and redesign from scratch. [4]

Specifically, current methods for discovering policy:

Methods	Interpretable	Automated	Sample-Efficient
Rule-based, Mathematical Programming [5-6]	✅ Transparent	❌ Expert-driven	✅ No env-interaction needed
Heuristic Search (GA, NSGA-II) [7-8]	⚠️ Limited by parameterized form	✅ Automated searching	❌ Extensive evaluations
Deep ML, RL [9-10] **(Neural Networks)**	❌ Opaque	✅ Self-learning	❌ Costly trial-and-error

No single method achieves all three (Interpretability, Automation, Sample Efficiency)

1.3 What Can LLMs Do in Industrial Control?

- Large language models (LLMs) were originally designed for text generation [11].
- Now capable of:
  - General Q&A [11-13];
  - Mathematical problem solving [13];
  - Program synthesis [13];
  - …

- ***What role can LLMs play in industrial decision-making scenarios?***
  - ➢ *Can we integrate these capabilities into the control loop?*

1.4 Our Vision: From Automation to Autonomy

- The Shift:
  - Tuning neural network weights ⇨ Searching over ***code logic***

1.4 Our Vision: From Automation to Autonomy

- The Shift:
  - Tuning neural network weights ⇨ Searching over ***code logic***
- The Promise: Breaking the ***Trilemma***
  - ✅ **Interpretable**: Clear as explicit code.
  - ✅ **Automated**: LLMs autonomously discover and refine policies.
  - ✅ **Sample-Efficient**: Leveraging world knowledge for direct code generation.

1.4 Our Vision: From Automation to Autonomy

- The Shift:
  - Tuning neural network weights ⇨ Searching over ***code logic***
- The Promise: Breaking the ***Trilemma***
  - ✅ **Interpretable**: Clear as explicit code.
  - ✅ **Automated**: LLMs autonomously discover and refine policies.
  - ✅ **Sample-Efficient**: Leveraging world knowledge for direct code generation.
- The Ultimate Goal:

Pre-Programmed (Automation) ⇨ **Self-Designing (Autonomy)**

1.4 Our Vision: From Automation to Autonomy

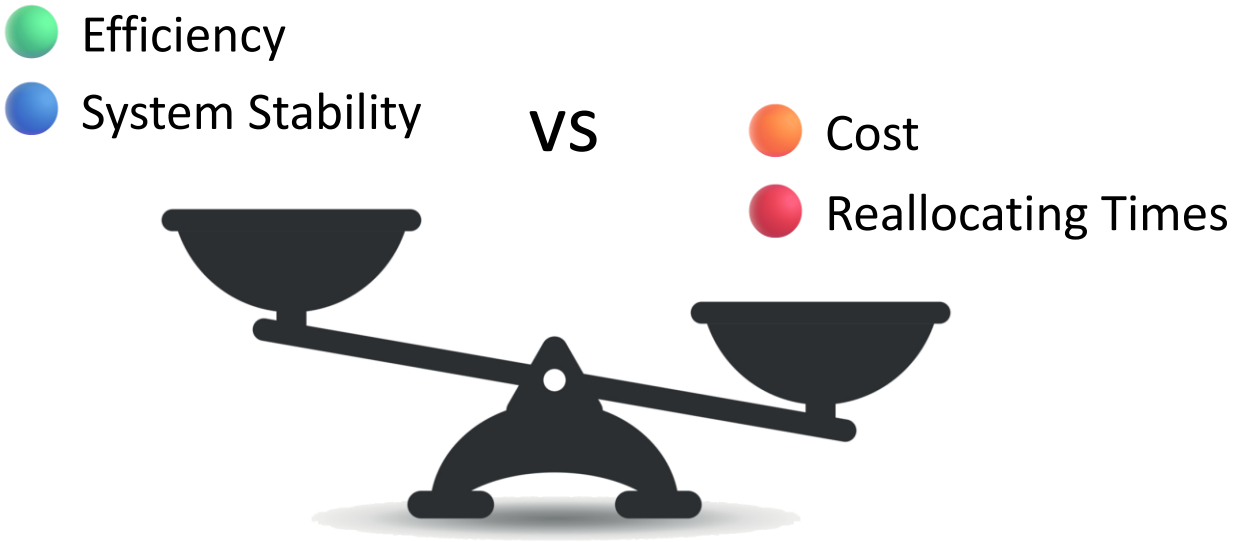
- The Shift:
  - Tuning neural network weights ⇨ Searching over ***code logic***
- The Promise: Breaking the ***Trilemma***
  - ✅ **Interpretable**: Clear as explicit code.
  - ✅ **Automated**: LLMs autonomously discover and refine policies.
  - ✅ **Sample-Efficient**: Leveraging world knowledge for direct code generation.
- The Ultimate Goal:

Pre-Programmed (Automation) ⇨ **Self-Designing (Autonomy)**

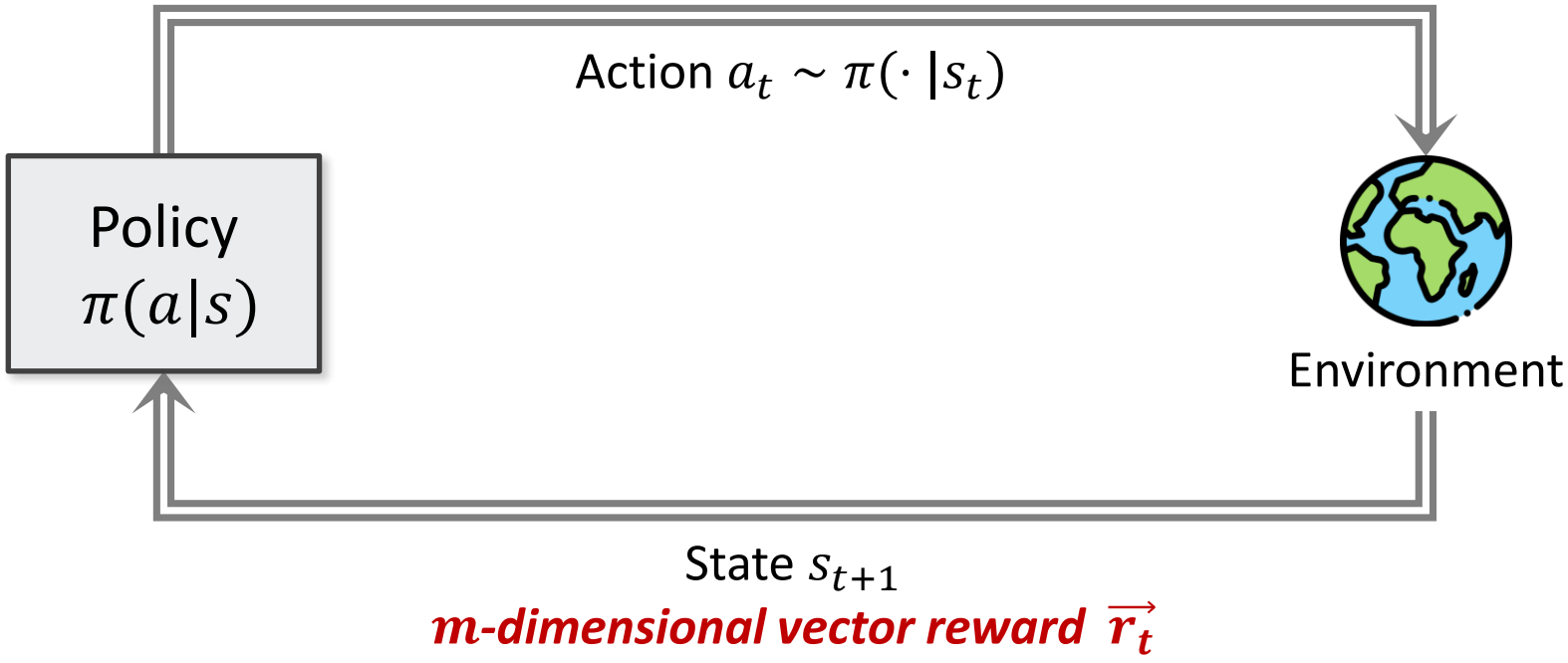
- ***In this work, we provide a proof-of-concept.***

2.1 The Scenario: Multi-Objective Decision-Making

- In this work, we focus on ***multi-objective*** decision-making scenarios.
- Real-world control is inherently multi-objective [7].
- E.g.,

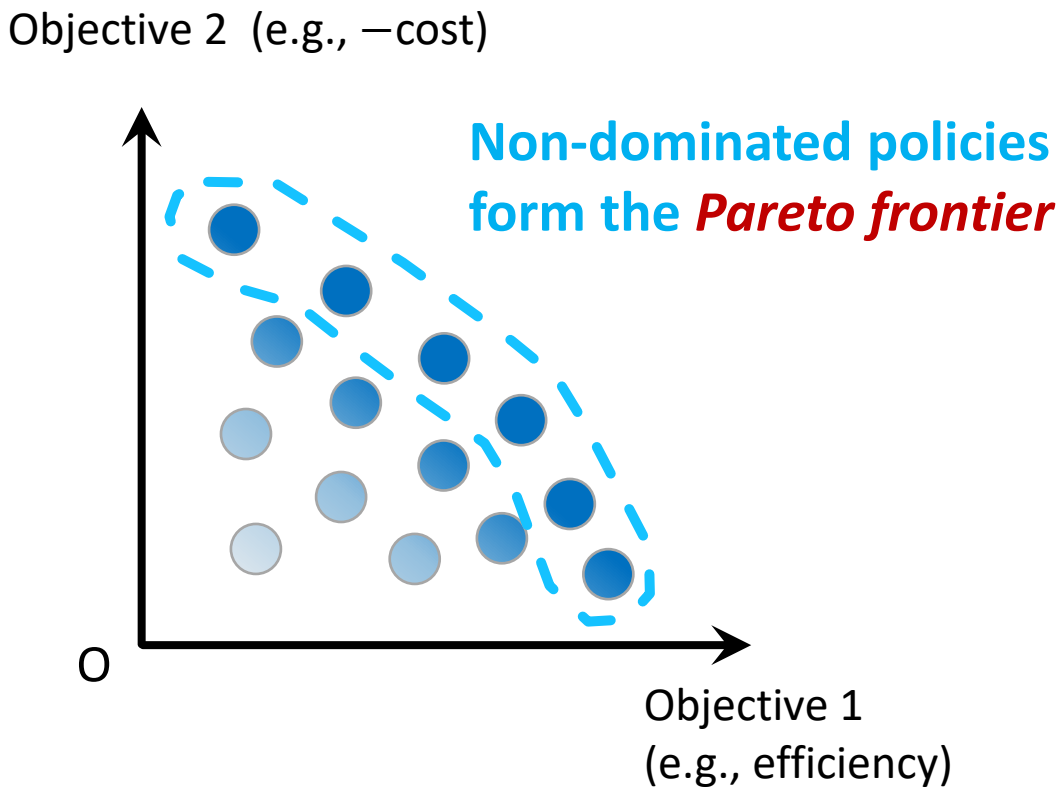


2.2 The Formulation: Multi-Objective MDP



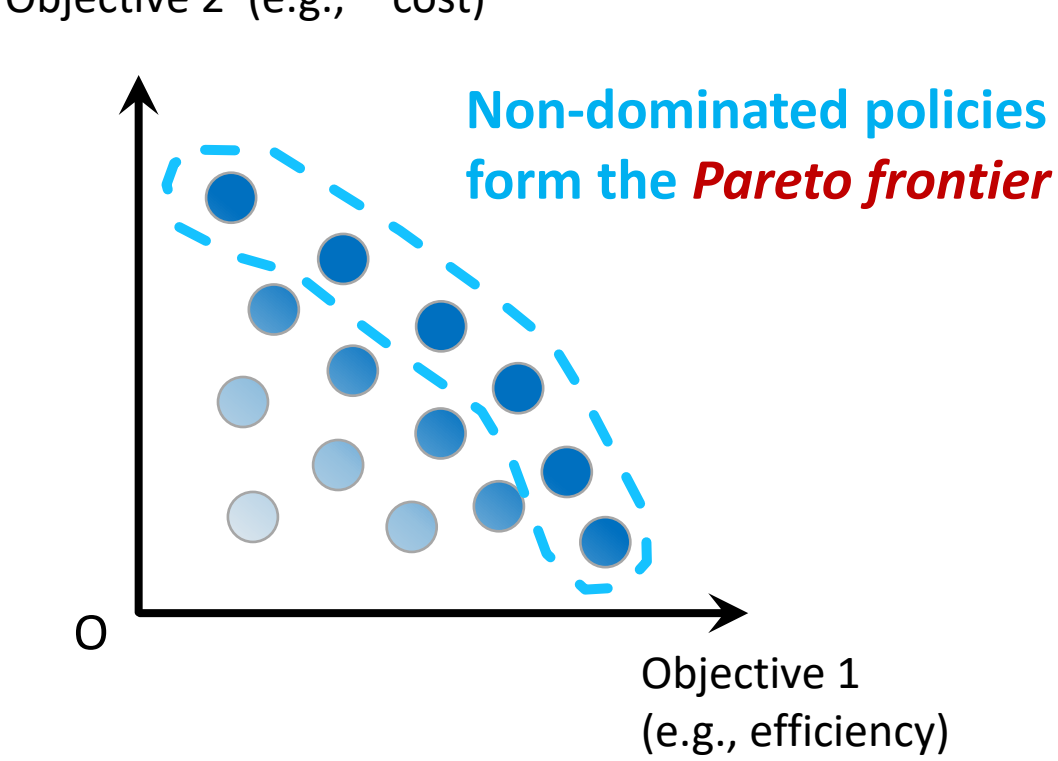
- From single-objective MDP to **multi-objective MDP (MOMDP)** [9]
- Formulation: $M = (S, A, P, \boldsymbol{R}, \gamma)$, where $\boldsymbol{R}$: $S \times A \to \mathbb{R}^m$

2.3 The Goal: Pareto Frontier & Metrics



- Our goal is to obtain a ***policy library*** covering the ***Pareto frontier*** [7, 9].

2.3 The Goal: Pareto Frontier & Metrics



➢ Our goal is to obtain a ***policy library*** covering the ***Pareto frontier*** [7, 9].

➢ Evaluate the policy library:

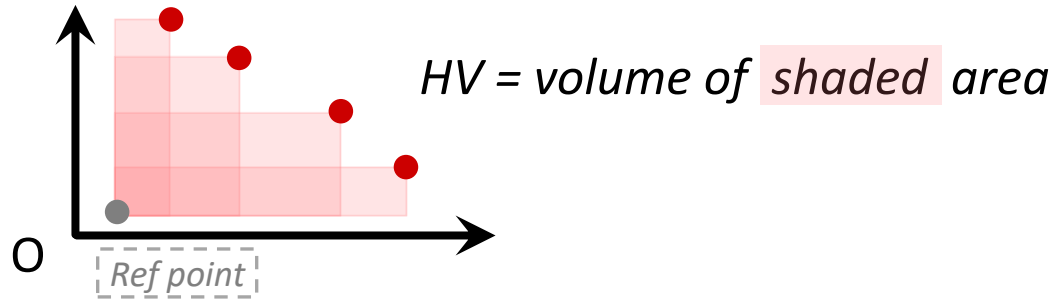
1. Expected Utility (EU) [14]:

$$EU(\pi) = \mathbb{E}_{\vec{w}}\left[\vec{w} \cdot \vec{R}(\pi)\right]$$

- Average performance across random preferences $\vec{w}$.
- EU measures overall quality.

2. Hypervolume (HV) [15-16]:

- Volume of the dominated objective space.
- HV measures policy diversity.



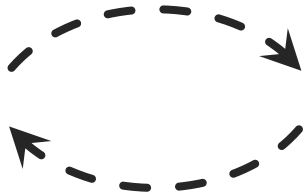
3 Methodology: A Dual-Agent Closed-Loop Architecture

➢ We leverage two LLM agents:

Policy Generator

- *Receives the target objective vector*
- *Generates (Python, C++, etc.) policy code*

Code for the new policy



Target objective vector

Coordinator

- *Analyzes gaps of Pareto frontier*
- *Identifies unexplored trade-offs*

An analogy

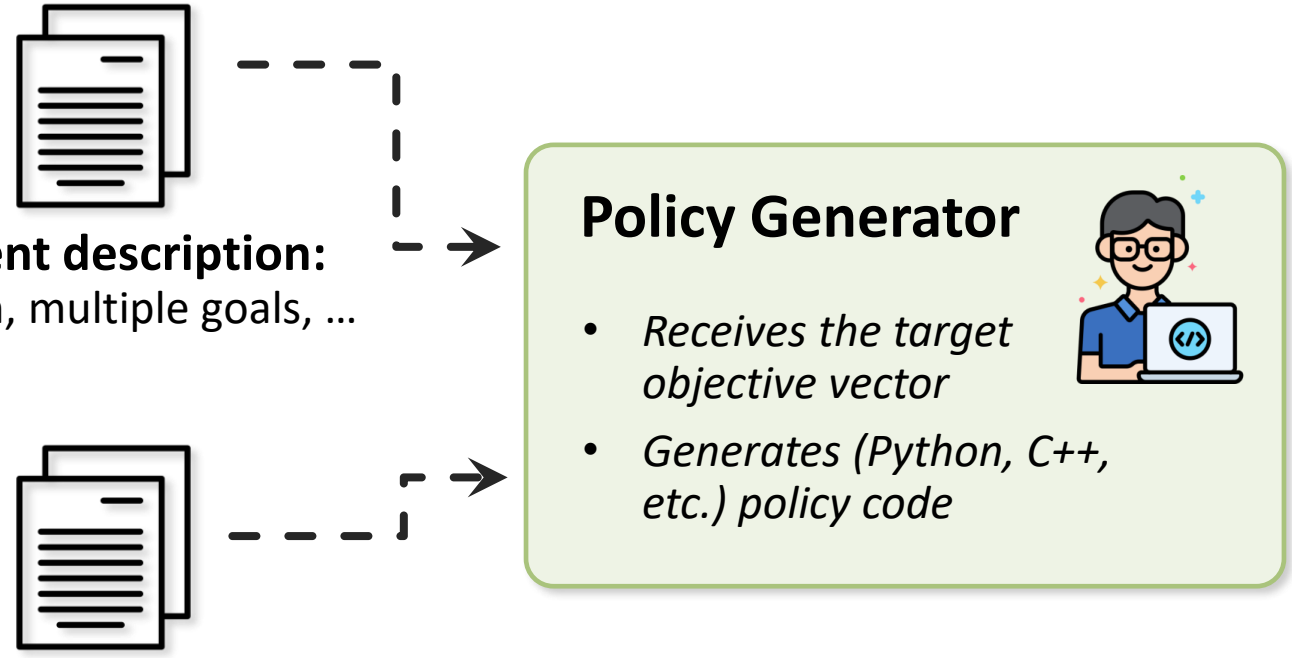
Programmer

We need a policy that prioritizes energy saving while maintaining stability.

OK, I'll fix it.

Product Manager

3.1 The Policy Generator



Prompt:

- Generate a Python policy that achieves the target reward vector `{target}`
- Wrap the code in ```` ```python``` ````

3.1 The Policy Generator

Environment description:
state, action, multiple goals, …

Prompt:

- Generate a Python policy that achieves the target reward vector `{target}`
- Wrap the code in ```` ```python``` ````

Policy Generator

- *Receives the target objective vector*
- *Generates (Python, C++, etc.) policy code*

Evaluation

Executable code

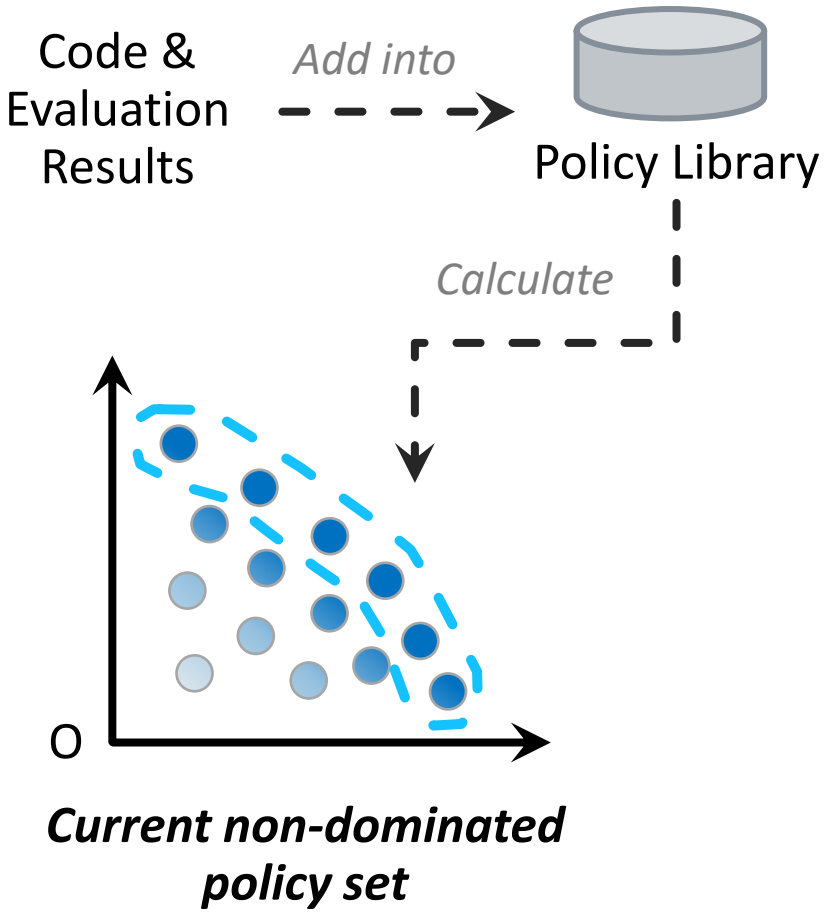
Interact

Environ-ment

Code & Evaluation Results

Coordinator

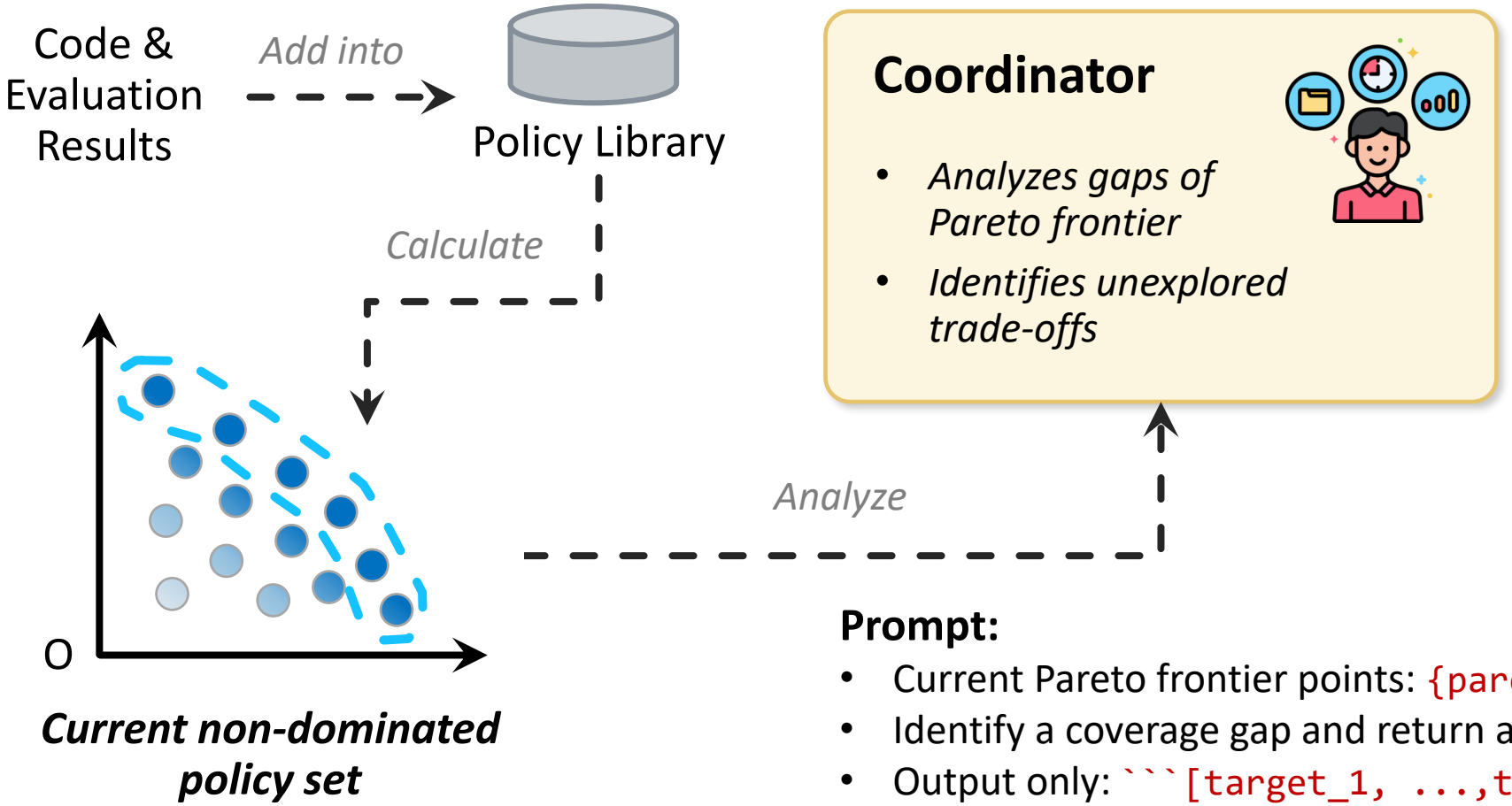
3.2 The Coordinator



Coordinator

- *Analyzes gaps of Pareto frontier*
- *Identifies unexplored trade-offs*

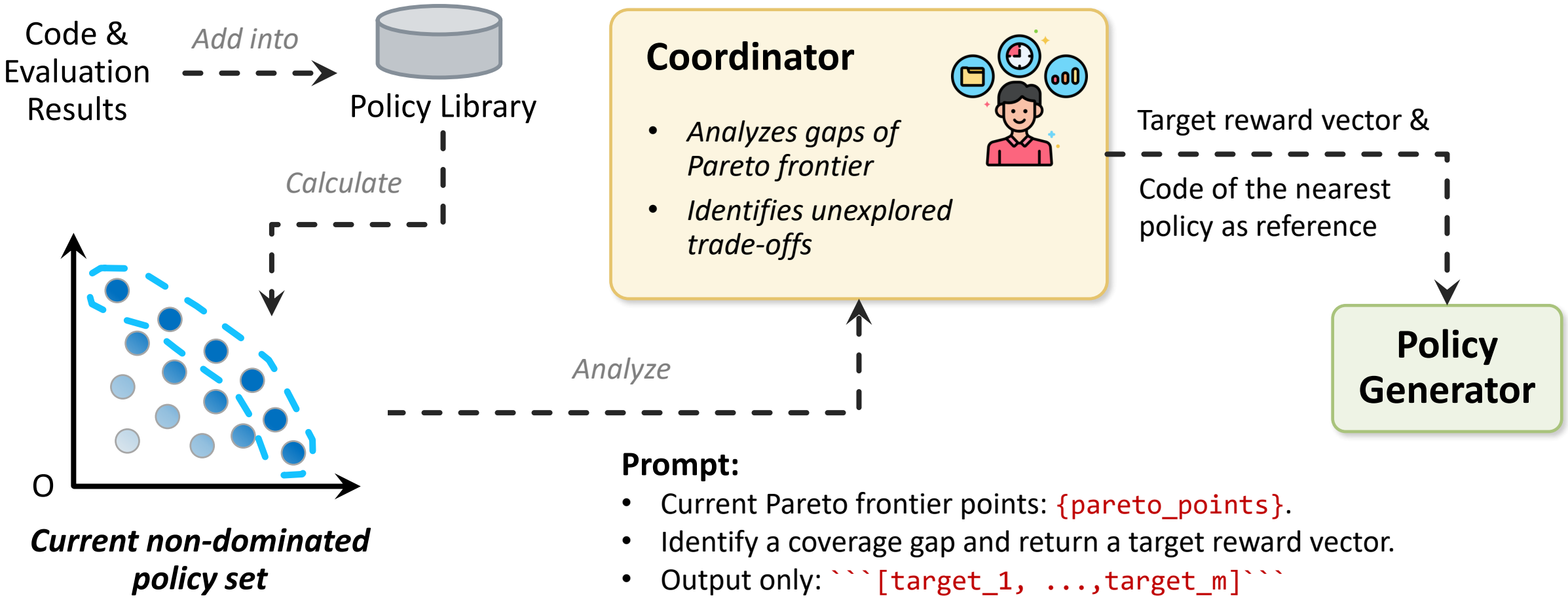
3.2 The Coordinator



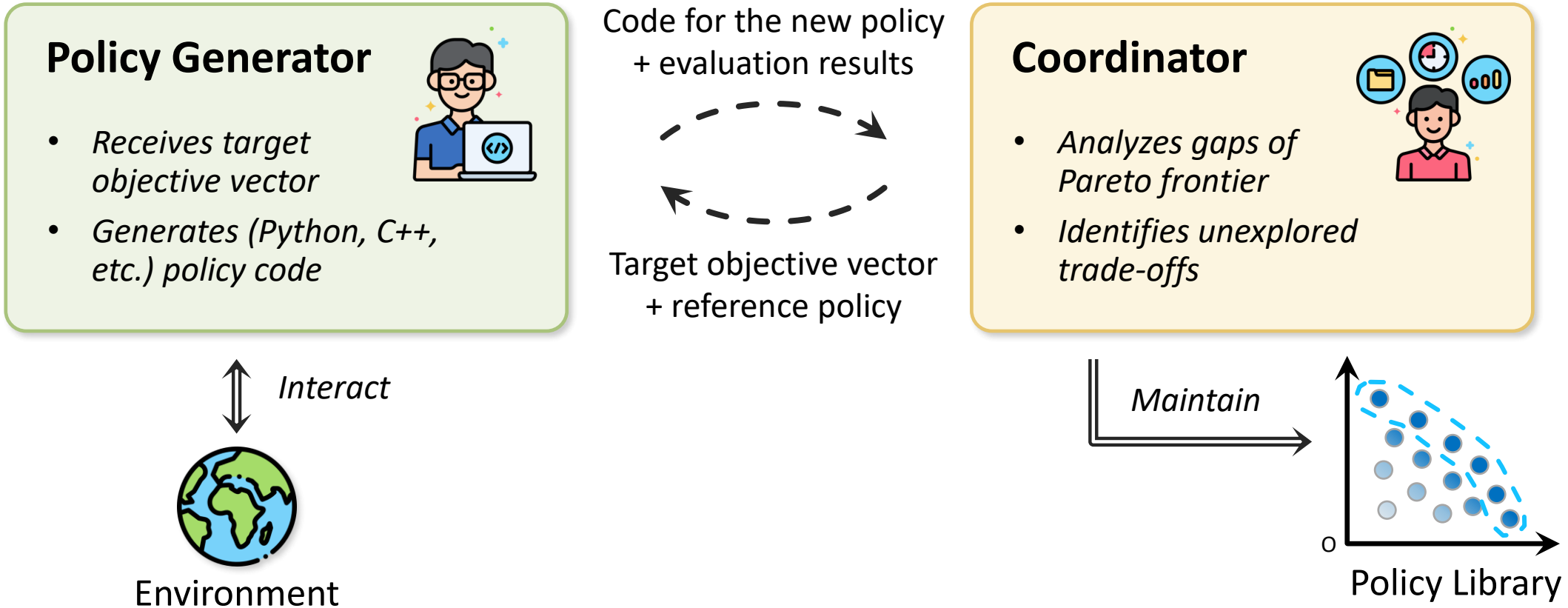
Prompt:

- Current Pareto frontier points: `{pareto_points}`.
- Identify a coverage gap and return a target reward vector.
- Output only: ```[target_1, ...,target_m]```

3.2 The Coordinator



3.3 Overall Process of Policy Library Construction



3.3 Overall Process of Policy Library Construction

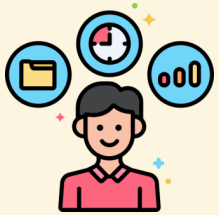
Policy Generator

- *Receives target objective vector*
- *Generates (Python, C++, etc.) policy code*

Code for the new policy + evaluation results

Target objective vector + reference policy

Coordinator

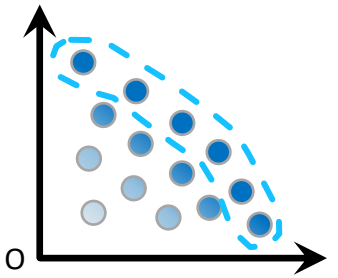


- *Analyzes gaps of Pareto frontier*
- *Identifies unexplored trade-offs*

Interact

Environment

Maintain



Policy Library

Breaks the trilemma:

- ✅ **No training (Sample-efficient)**
- ✅ **Code-based (Interpretable)**
- ✅ **Closed-loop iteration (Automated)**

4.1 Tasks: Two Multi-Objective Scenarios

PCB buffer management in SMT production lines [17]

- Action: Assign incoming board to storage racks.
- Objectives:
  - Minimize rehandles (avoid moving boards)
  - Maximize stability (penalize frequent changes in rack occupancy)
  - Maximize balance (ensure even distribution across all racks)
- Trade-off: Maximizing balance requires frequent reshuffling, which increases rehandles and degrades stability.

Battery storage management in smart grid [18]

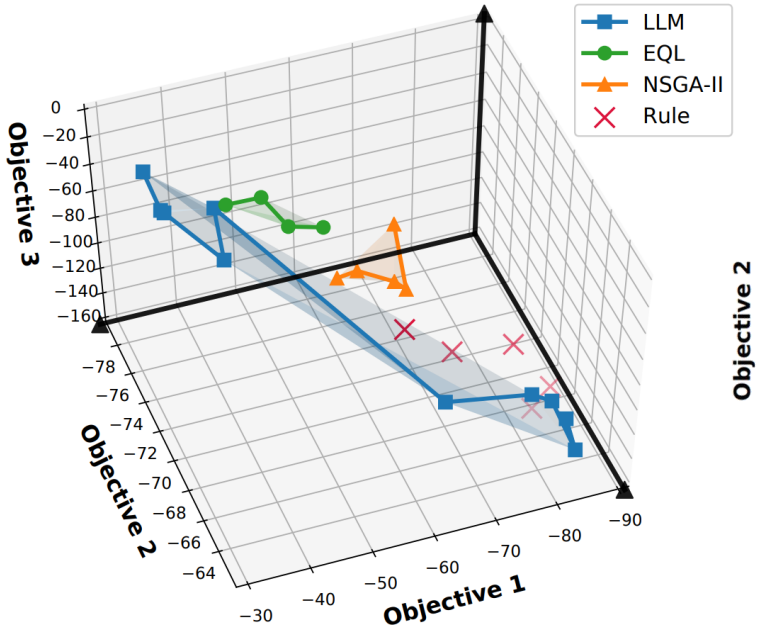
- Action: Charge / discharge power level of the battery.
- Objectives:
  - Minimize grid purchase cost (save money)
  - Minimize discharge cycles (extend battery life)
- Trade-off: Aggressive cost-saving requires frequent cycling.

4.2 Baselines & Experimental Setup

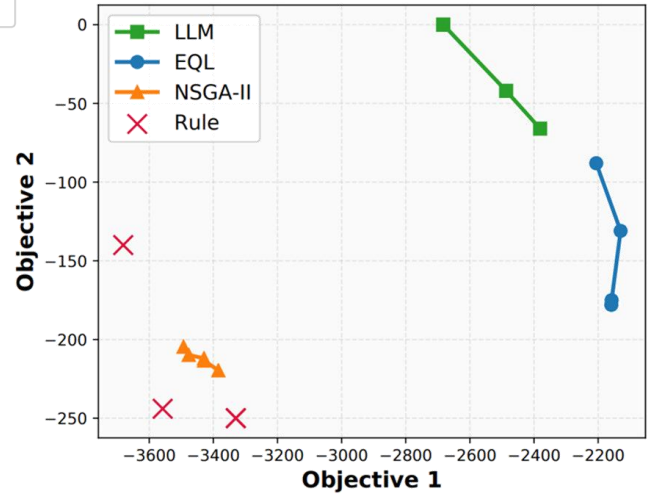
- **Baselines:**
  ① Rule-based: *First-fit, Best-fit, Balanced, Threshold, Cost-saving…*
  ② Evolutionary: *NSGA-II*
  ③ Deep RL: *Envelope Q-Learning (EQL)*
- **Setup:**
  - LLM configuration: `DeepSeek-R1-Distill-Qwen-7B`, temperature=0.7, 30 iterations.
  - NSGA-II uses linear policy representation.
  - EQL: Double Q-network, 1M training timesteps
- **Metrics:**
  - Expected Utility (EU), Hypervolume (HV)

4.3 Numerical Results

Method	PCB Buffer Management		Power Management	
	EU ↑	HV ↑	EU ↑	HV ↑
Ours	-44.62	**137,179**	-1,187.89	**794,272**
EQL	**-45.35**	145,654	**-1,126.10**	767,172
NSGA-II	-57.97	124,147	-1,793.23	180,741
Rule-Based	-61.66	123,248	-1,814.02	203,328



Visualization of PCB Buffer Management



Visualization of Power Management

- **Autonomous high performance**: Our method ***is comparable to EQL (the strongest RL baseline)*** and outperforms others, without manual design or supervision.
- **Sample efficiency**: Our method ***converges in <90 min (30 iterations)***, while EQL requires 1M steps and 10+ GPU hours.
- **Interpretability**: Generates ***transparent, human-readable code*** (demo on the next slide).

4.4 Policy Code Demo

```
def mo_pcb_policy(obs):
    stacks_departure = obs['stacks_departure']
    current_departure = obs['current_departure']
    best_action = 0
    max_score = -float('inf')

    for i in range(3):
        stack_i = stacks_departure[i]
        current_height = len([x for x in stack_i if x != 0])
        if current_height >= 5: continue

        flip_count = sum(1 for x in stack_i if x != 0 and x < current_departure)

        new_height_i = current_height + 1

        heights = [new_height_i if j == i else len([x for x in \
                   stacks_departure[j] if x != 0]) for j in range(3)]
        mean = sum(heights) / 3
        variance = sum((h - mean)**2 for h in heights) / 3

        total = (-flip_count) + (-1) + (-variance)
        if total > max_score:
            max_score = total
            best_action = i

    return best_action
```

4.4 Policy Code Demo

```
def mo_pcb_policy(obs):
    stacks_departure = obs['stacks_departure']
    current_departure = obs['current_departure']
    best_action = 0
    max_score = -float('inf')

    for i in range(3):
        stack_i = stacks_departure[i]
        current_height = len([x for x in stack_i if x != 0])
        if current_height >= 5: continue

        flip_count = sum(1 for x in stack_i if x != 0 and x < current_departure)

        new_height_i = current_height + 1

        heights = [new_height_i if j == i else len([x for x in \
                   stacks_departure[j] if x != 0]) for j in range(3)]
        mean = sum(heights) / 3
        variance = sum((h - mean)**2 for h in heights) / 3

        total = (-flip_count) + (-1) + (-variance)
        if total > max_score:
            max_score = total
            best_action = i

    return best_action
```

Iterate over each rack

Skip if rack is full (safety constraint)

Calculate rehandling count

Predict new height after placement

Calculate load balance (variance)

Composite score: balance all three objectives

Human-readable code, fully interpretable

Conclusion: From Automation to Autonomy

--- Self-Designing Industrial Control Systems

Past & Present

- We externalized human experience.
- Currently, we face the trilemma of ***automation, interpretability, efficiency***.

Conclusion: From Automation to Autonomy

--- Self-Designing Industrial Control Systems

Past & Present ⇨ ***This work***

- We externalized human experience.
- Currently, we face the trilemma of ***automation, interpretability, efficiency***.

- ***LLM-driven*** dual-agent loop: policy generator + coordinator.
- ***Code as policy***, searching the space of code logic.
- Has the potential to ***resolve the trilemma***.

Conclusion: From Automation to Autonomy

--- Self-Designing Industrial Control Systems

Past & Present

- We externalized human experience.
- Currently, we face the trilemma of ***automation, interpretability, efficiency***.

This work

- ***LLM-driven*** dual-agent loop: policy generator + coordinator.
- ***Code as policy***, searching the space of code logic.
- Has the potential to ***resolve the trilemma***.

Future Vision

- From pre-programmed execution to ***self-designing systems***.
- This work provides a ***proof-of-concept*** and reveals the emerging role of LLMs.

Pre-Programmed (Automation) → **Self-Designing (Autonomy)**

References

1. Maxwell, J. C. (1868). On governors. Proceedings of the Royal Society of London, 16(100-107), 270-283.
2. Minorsky, N. (1922). Directional stability of automatically steered bodies. Journal of the American Society of Naval Engineers, 34(2), 280-309.
3. Walker, M., Bissell, C., & Monk, J. (2010). The PLC: a logical development. Measurement + Control, 43(4), 112-117
4. Åström, K. J., & Kumar, P. R. (2014). Control: A perspective. Automatica, 50(1), 3-43.
5. Bernard, J.A. (2002). Use of a rule-based system for process control. IEEE Control Systems Magazine.
6. Craven, B.D. (2012). Mathematical programming and control theory. Springer Science & Business Media.
7. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation, 6(2), 182–197.
8. Zitzler, E. (1999). Evolutionary algorithms for multi-objective optimization: Methods and applications.
9. Yang, R., Sun, X., and Narasimhan, K. (2019). A generalized algorithm for multi-objective reinforcement learning and policy adaptation. Proceedings of the 33rd International Conference on Neural Information Processing Systems (NeurIPS)
10. Mu, N., Hu, X., Jia, Q.S., Zhu, X., and He, X. (2024). Large-scale Data Center Cooling Control via Sample-efficient Reinforcement Learning. 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)
11. OpenAI (2023). Gpt-4 technical report. arXiv preprint arXiv:2303.08774.
12. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., and Cao, Y. (2022). React: Synergizing reasoning and acting in language models. In The eleventh international conference on learning representations (ICLR).
13. Kimi (2025). Kimi k2: Open agentic intelligence. arXiv preprint arXiv:2507.20534.
14. Felten, F., Alegre, L.N., Nowé, A., et al. (2023). A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. Proceedings of the 37th International Conference on Neural Information Processing Systems.
15. Zitzler, E. (1999). Evolutionary algorithms for multi-objective optimization: Methods and applications, volume 63. Shaker Ithaca.
16. Zitzler, E., and Thiele, L. (1999). Multi-objective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation.
17. Gebus, S., Martin, O., Soulas, A., and Juuso, E. (2004). Production optimization on PCB assembly lines using discrete-event simulation.
18. Branco, H., Castro, R., and Lopes, A.S. (2018). Battery energy storage systems as a way to integrate renewable energy in small isolated power systems. Energy for Sustainable Development, 43, 90–99.

Thanks for listening!

Ni Mu Tsinghua University

E-mail: mn23@mails.tsinghua.edu.cn

Weixin / WeChat: MoonOutCloudBack

放映结束，单击鼠标退出。