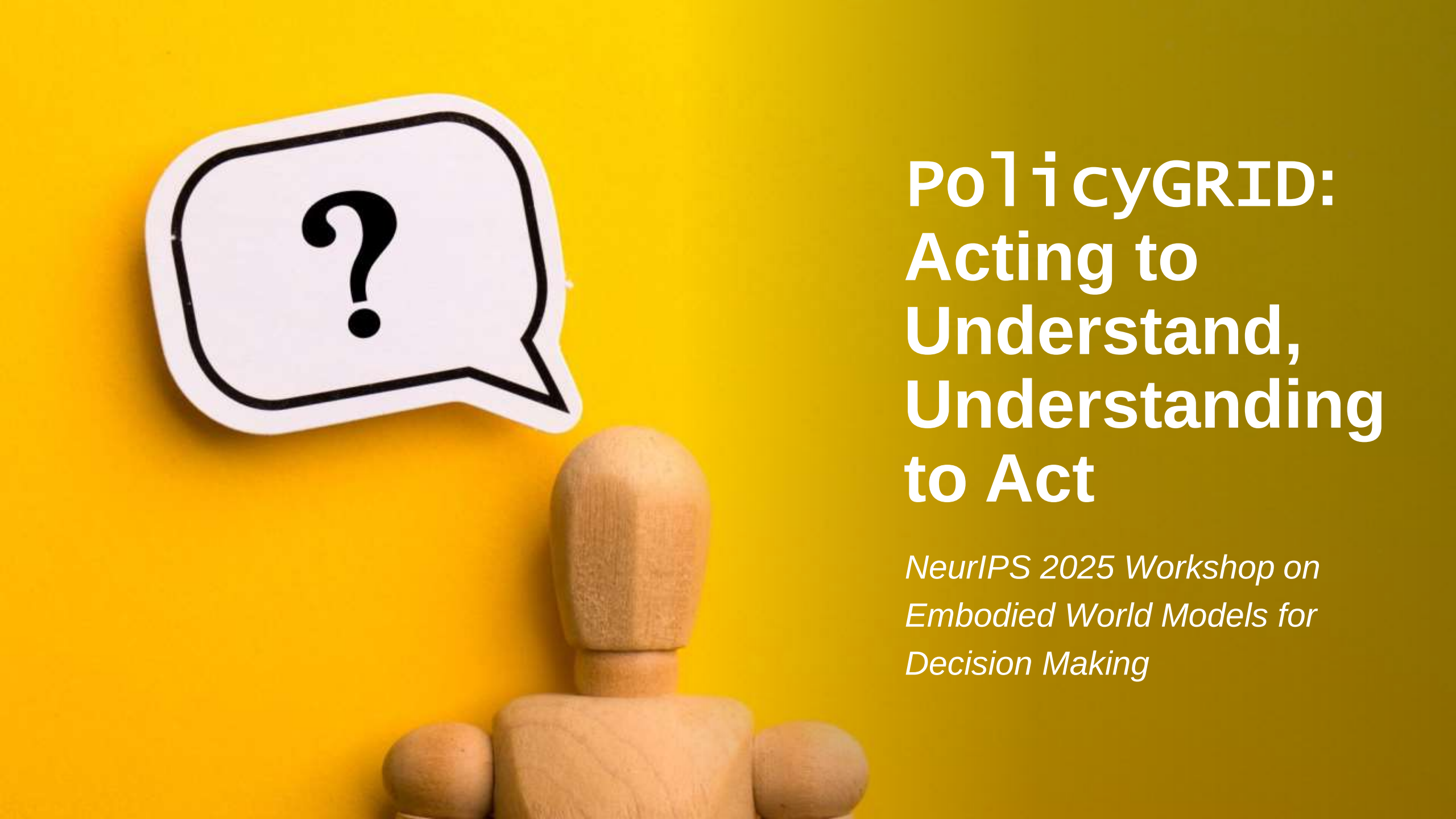
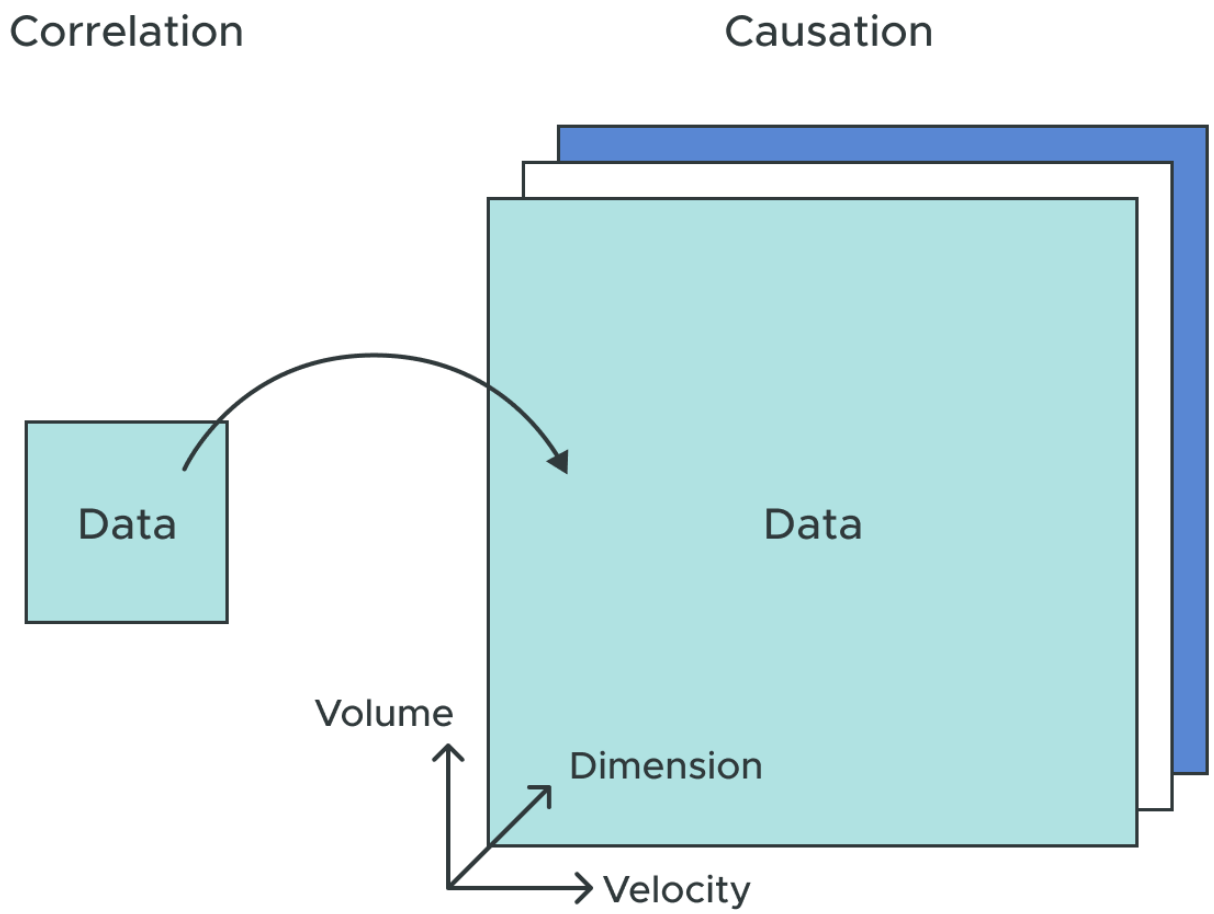


PolicyGRID: Acting to Understand, Understanding to Act

NeurIPS 2025 Workshop on Embodied World Models for Decision Making

Correlation vs Causation

Correlation vs Causation

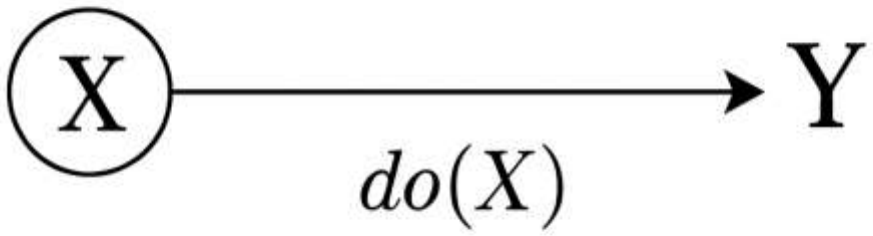


- correlation != causation
- models fail when agents intervene
- current systems either:
  - ignore causal structures
  - treat causality as an offline pre-processing step

Why Causality Matters

- **Prediction ≠ Understanding**
  - Correlations describe *what tends to happen*
  - But actions change the world — correlations break under intervention
  - Real-world systems need *stability*, *transferability*, and *control*
- **Goal:** move from *seeing* patterns → *understanding* mechanisms

From Observation to Intervention



Two Questions:

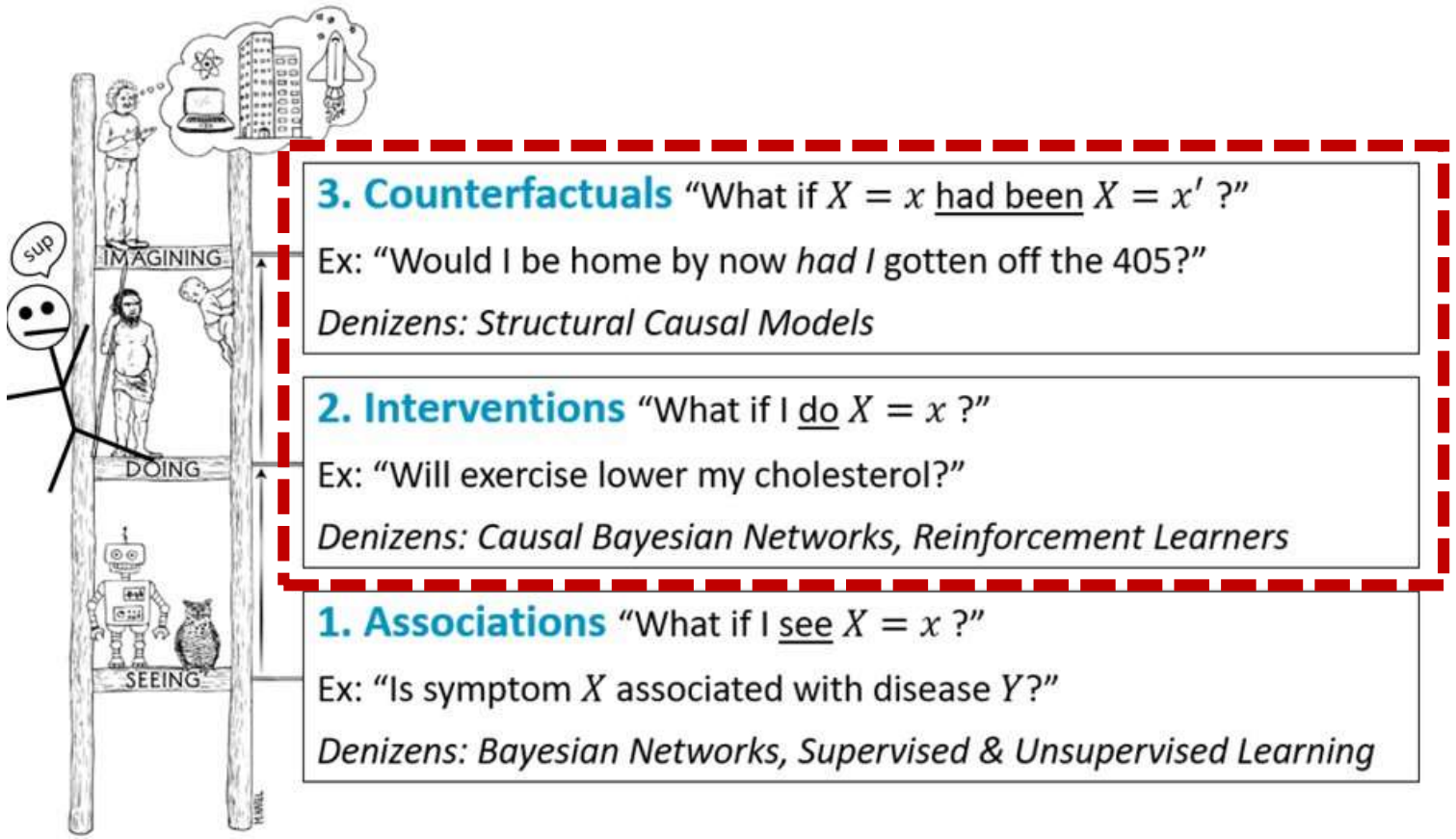
- *Observation:* "What happens when X is seen?" $\rightarrow P(Y \mid X)$
- *Intervention:* "What happens when X is done?" $\rightarrow P\big(Y \mid do(X)\big)$

Why this matters

- $P(Y \mid X)$ may fail when environment shifts
- $P\big(Y \mid do(X)\big)$ lets agents *predict consequences* of their actions

The Causal Ladder

Levels of Reasoning *(Pearl, 2019)*



`PolicyGRID` operates at 2–3

- Learns *how actions change outcomes*

- Uses causal understanding to design better control policies

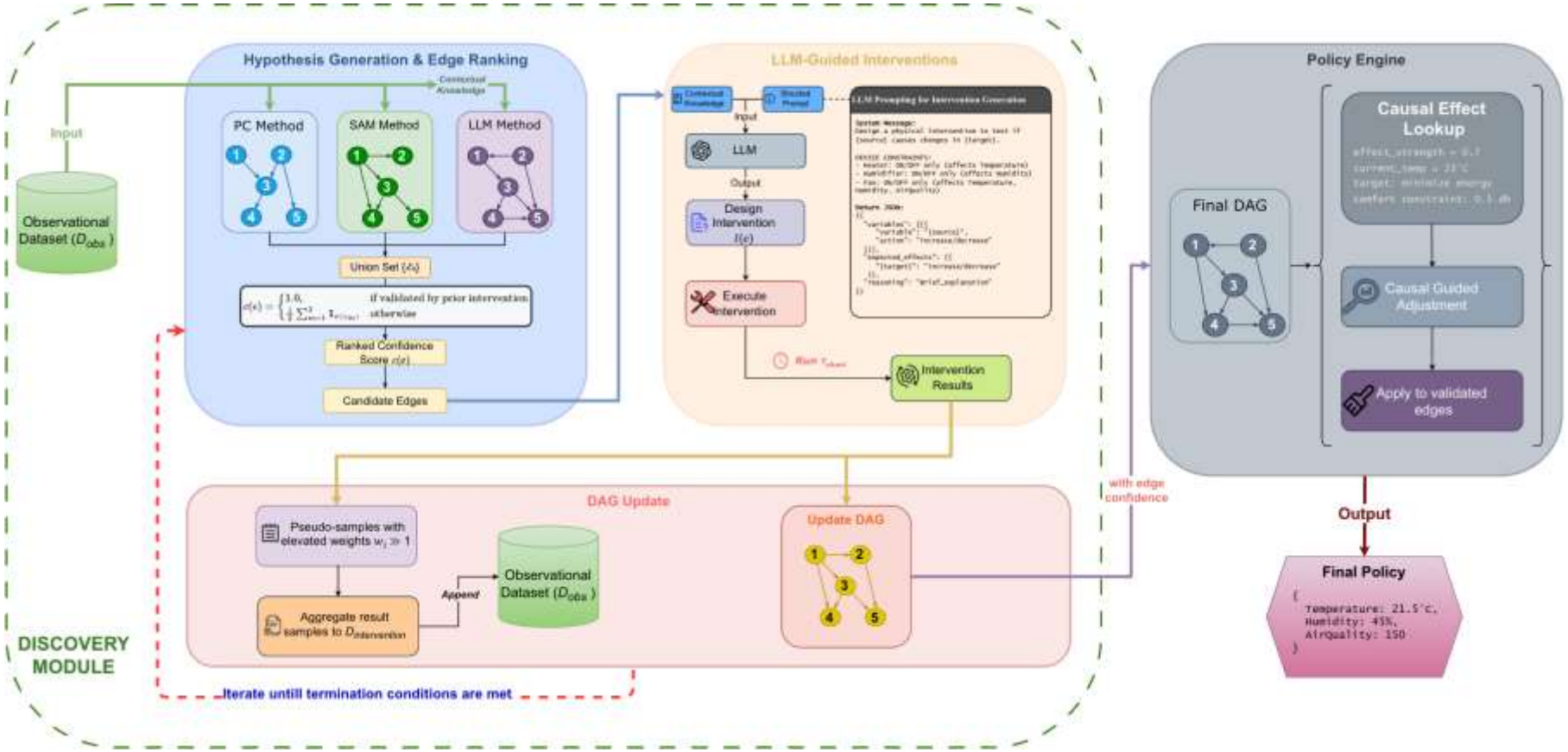
Why PolicyGRID

Bridging Understanding and Action

- Existing agents → *ignore causality* or treat it as *offline pre-processing*
- PolicyGRID:
  → *discovers* causal structure
  → *validates* interventions
  → *translates* causal knowledge into adaptive policies

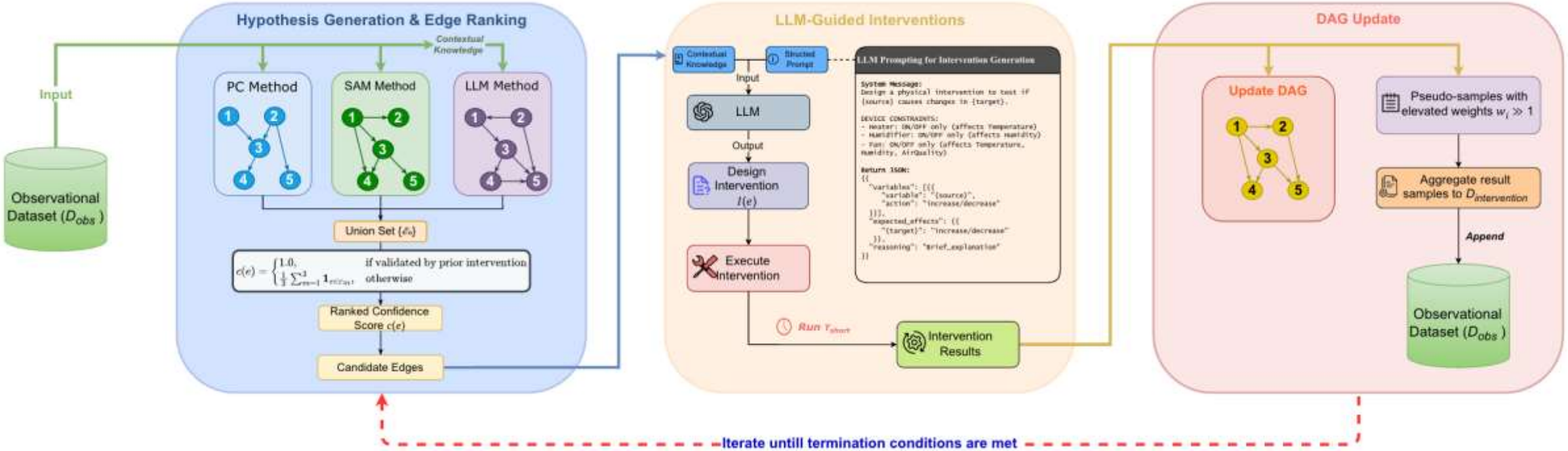
Acting to Understand, Understanding to Act

PolicyGRID

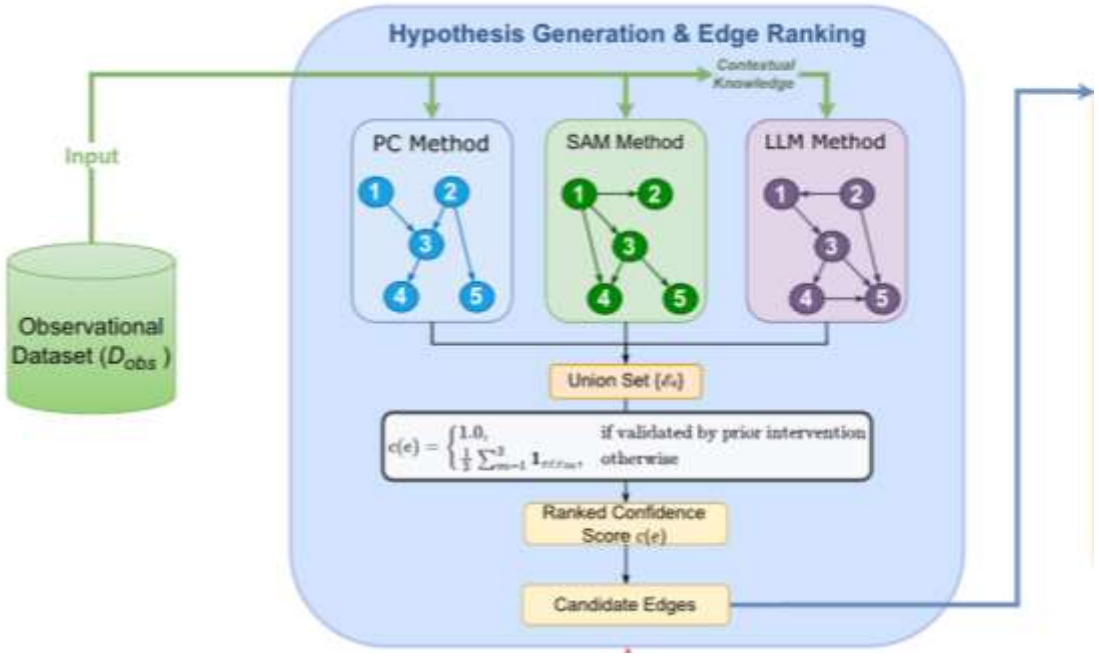


1. Discovering causal structure from data
2. Validating causal relationships
3. Translating causal knowledge into policies.

Discovery Module

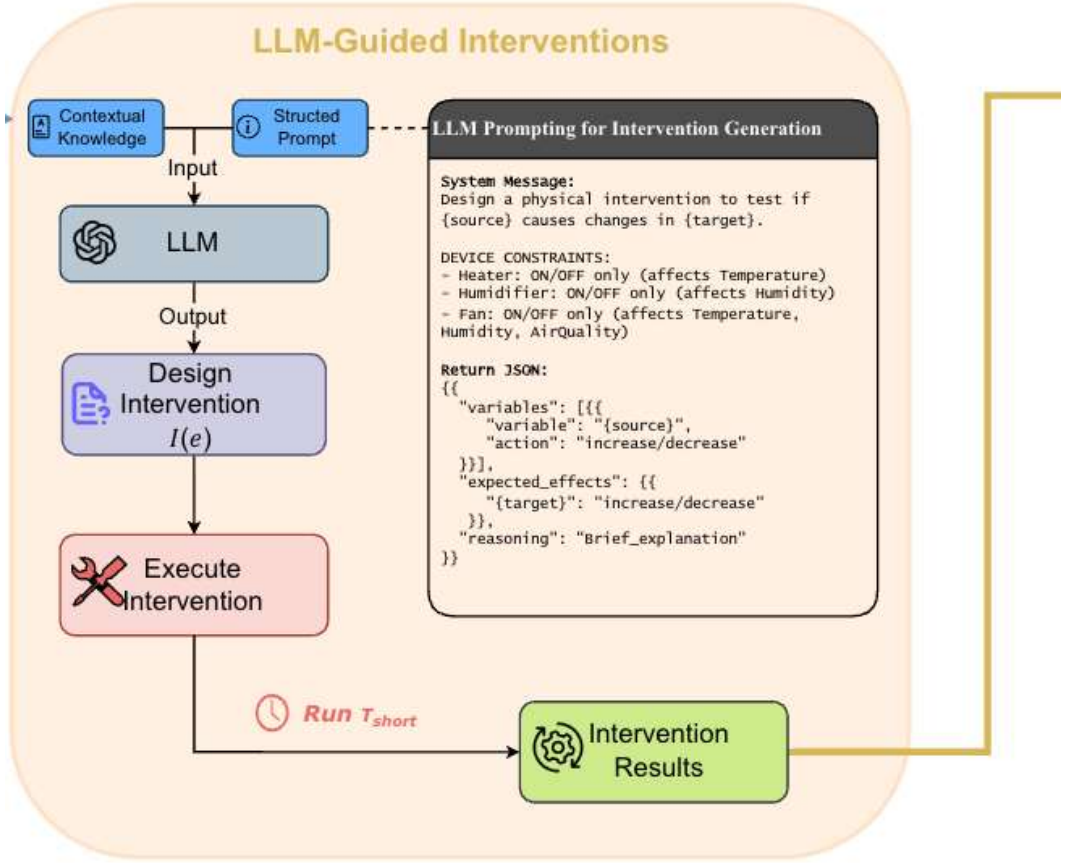


Discovery Module: **Hypothesis Generation**



- 3 generators: PC, SAM, LLM
- Hypothesis edge union + scored ranking
- Start iterative testing in ascending order of confidence scores

Discovery Module: **Interventions**



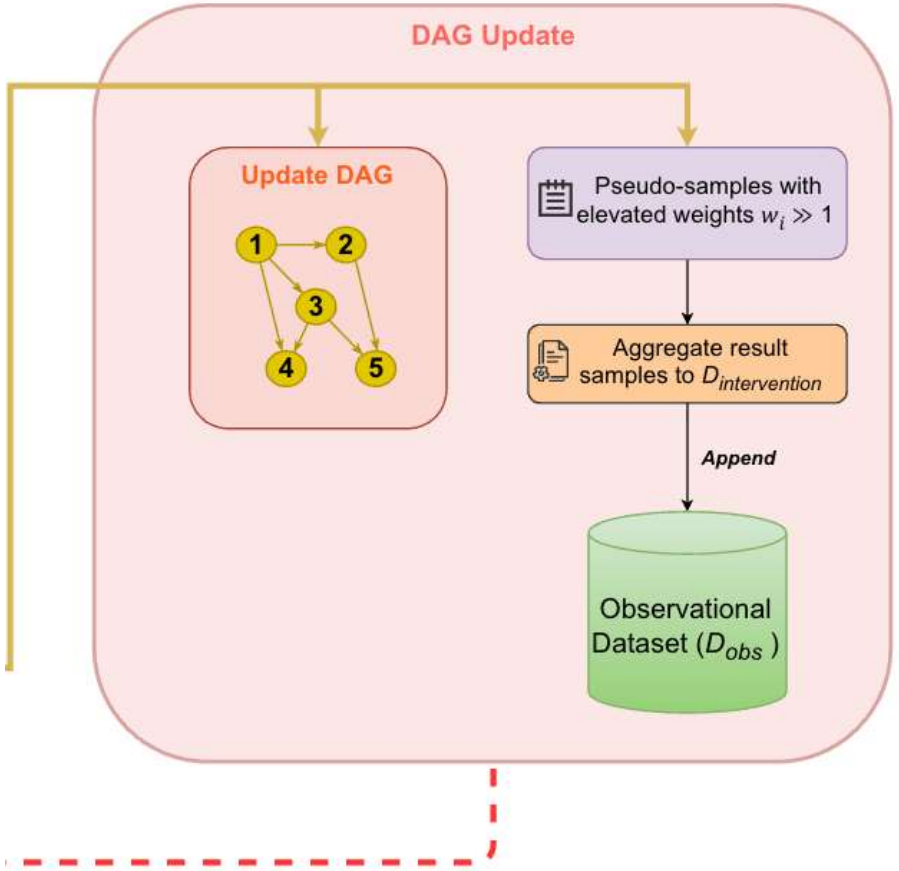
- Intervention on each edge
- LLM designs the intervention to be executed

Stage 2: Iterative Testing Loop (repeat until max iterations or all validated)
For each edge in ranked order (starting with lowest confidence):

```
# Initial Testing Phase
FOR i = 1 to 3:
    intervention_value = design_intervention(source_variable)
    result = perform_intervention(source, intervention_value)

    # Measure effect
    source_change = |result.post[source] - result.pre[source]|
    target_change = |result.post[target] - result.pre[target]|

    IF target_change > 0.001:
        effect_size[i] = target_change / source_change
    ELSE:
        effect_size[i] = 0

# Decision Point
IF no effect detected in 3 consecutive tests:
    DISCARD edge permanently
    CONTINUE to next edge
```

```
# Extended Testing Phase (if initial effect detected)
ELSE:
    FOR i = 4 to 10:  # Collect 5-10 additional data points
        perform_intervention()
        calculate_effect_size[i]

    # Calculate validation metrics
    avg_effect = mean(effect_sizes)
    variation = std(effect_sizes)
    consistency_factor = 1.0 / (1.0 + variation/avg_effect)
    confidence = min(1.0, avg_effect × consistency_factor)

    # Validation Decision
    IF confidence ≥ 0.5 AND variation < 2×avg_effect:
        VALIDATE edge
        ADD intervention data to training dataset
        validated_edges.add(edge)
    ELSE:
        DISCARD edge
```

Discovery Module: **DAG Update**



- If intervention effects > threshold:
  - Continue testing and append intervention data to dataset
  - Else, discard edge
- Update final DAG with validated edge
- Repeat until:
  - all candidate edges are evaluated
  - Tmax interventions is reached
  - learned DAG converges

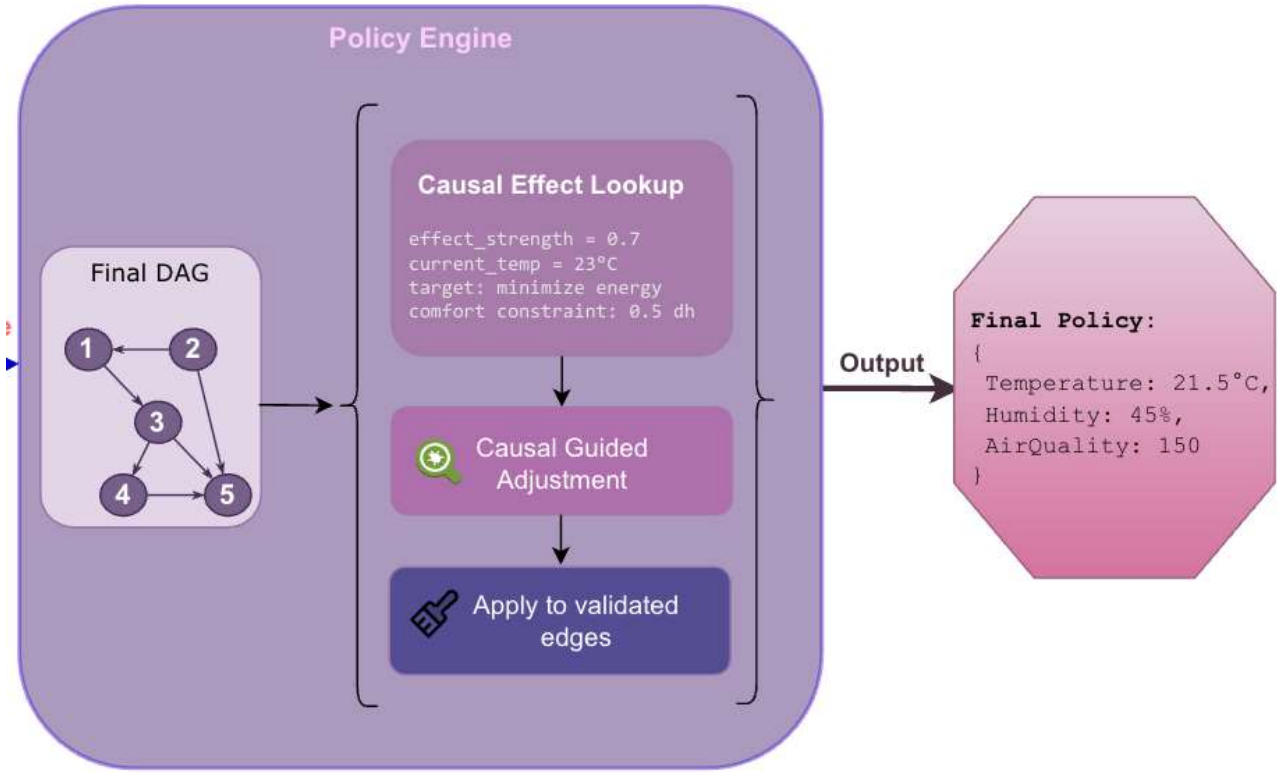
Stage 3: DAG Regeneration (after each validated edge)

```
IF any edges validated in iteration:
    # Update training data
    training_data = original_data + intervention_results

    # Regenerate hypotheses with new data
    PC_dag = run_PC_algorithm(training_data)
    SAM_dag = run_SAM_algorithm(training_data)
    LLM_dag = run_LLM_algorithm(training_data)

    # Re-rank remaining edges
    union_edges = (PC ∪ SAM ∪ LLM) - validated_edges
    re_rank_by_method_agreement(union_edges)

    # Restart testing from lowest confidence edge
    GOTO Stage 2
```

Stage 4: Final DAG Construction

```
# Build DAG from validated edges only
G = DirectedGraph()
G.add_nodes(all_variables)

# Add edges in confidence order, maintaining acyclicity
FOR edge IN sorted(validated_edges, by=confidence, descending):
    G.add_edge(edge)

    IF creates_cycle(G):
        G.remove_edge(edge)
        LOG("Edge discarded due to cycle")

RETURN G
```

Policy Engine



- Validated DAG → world model
- For each edge:
  - Get effect_strength from past intervention data
  - Adjust controls accordingly
  - Apply necessary constraints to all adjustments
  - Return final policy

Mock Example: Causal Optimization

Validated DAG

Temperature → EnergyConsumption (correlation: 0.82)

Temperature → OverallSatisfaction (correlation: 0.68)

Humidity → OverallSatisfaction (correlation: 0.45)

Input Parameters

```
objectives = {
    'energyconsumption': {'target': 20.0, 'weight': 0.4},
    'overallsatisfaction': {'target': 80.0, 'weight': 0.6}
}
constraints = {
    'Temperature': (18.0, 30.0),    #°C
    'Humidity': (30.0, 70.0),       #%
    'comfort_limit': 0.7
}
current_state = {
    'Temperature': 23.4,            #°C
    'Humidity': 46.0,               #%
    'EnergyConsumption': 28.0,      #kWh
    'OverallSatisfaction': 72.0     #%
}
```

Step 1: Initialize (current setpoints)

policy = {'Temperature': 23.5, 'Humidity': 46.0}

Step 2: Edge – *Temperature → EnergyConsumption*

effect_strength = 0.82

reduction = 0.4 × 0.82 × 0.4 × 0.7 = 0.092

 # Convert to temperature change: 0.092 × (30 - 18) = 1.1°C

policy['Temperature'] = clip(23.4 - 1.1, 18.0, 30.0) = 22.3 #°C

Step 3: Edge – *Temperature → OverallSatisfaction*

effect_strength = 0.68

boost = 0.3 × 0.68 × 0.6 × (2.0 - 0.7) = 0.159

 # Convert to temperature change: 0.159 × 12 = 1.9°C

policy['Temperature'] = clip(22.3 + 1.9, 18.0, 30.0) = 24.2 #°C

Step 4: Edge – *Humidity → OverallSatisfaction*

effect_strength = 0.45

boost = 0.3 × 0.45 × 0.6 × 1.3 = 0.105

 # Convert to humidity change: 0.105 × (70 - 30) = 4.2%

policy['Humidity'] = clip(46.0 + 4.2, 30.0, 70.0) = 50.2 #%

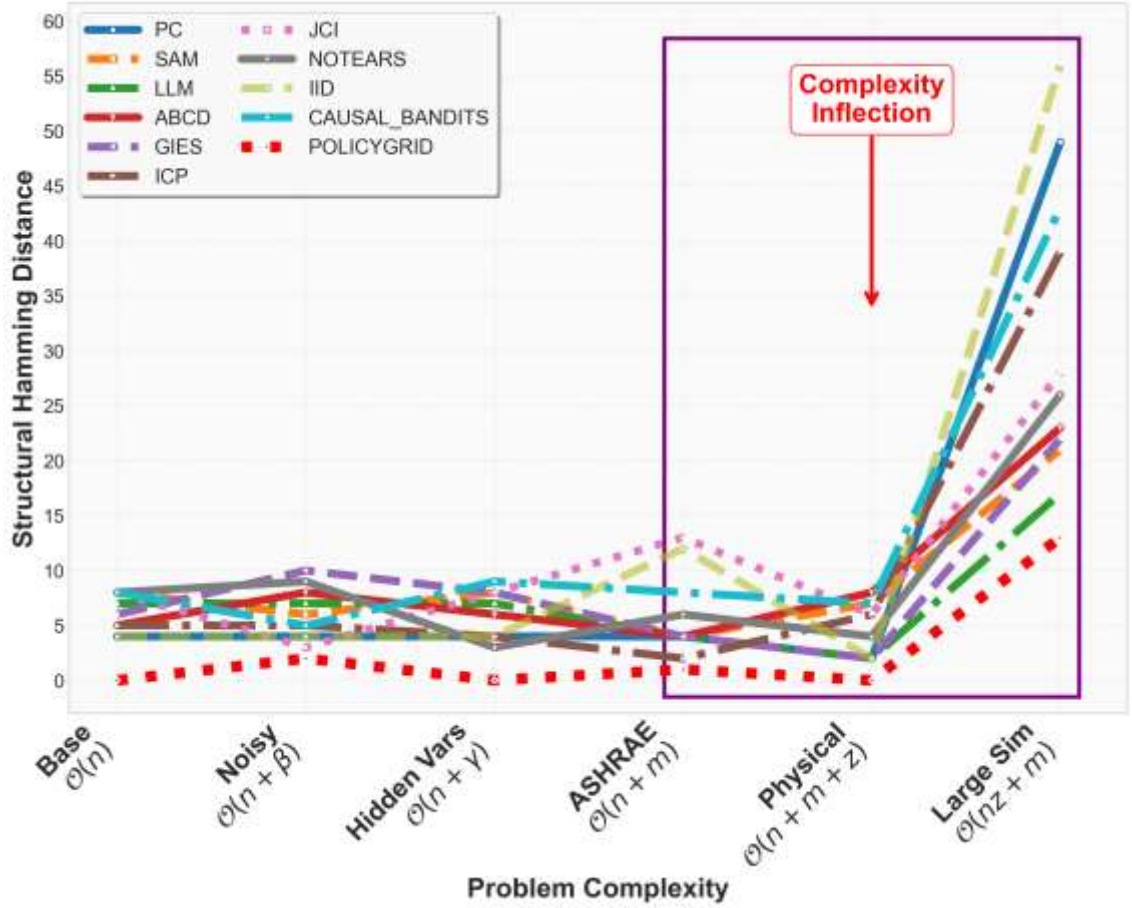
Return Optimized Policy {'Temperature': 24.2, 'Humidity': 50.2}

Experiments

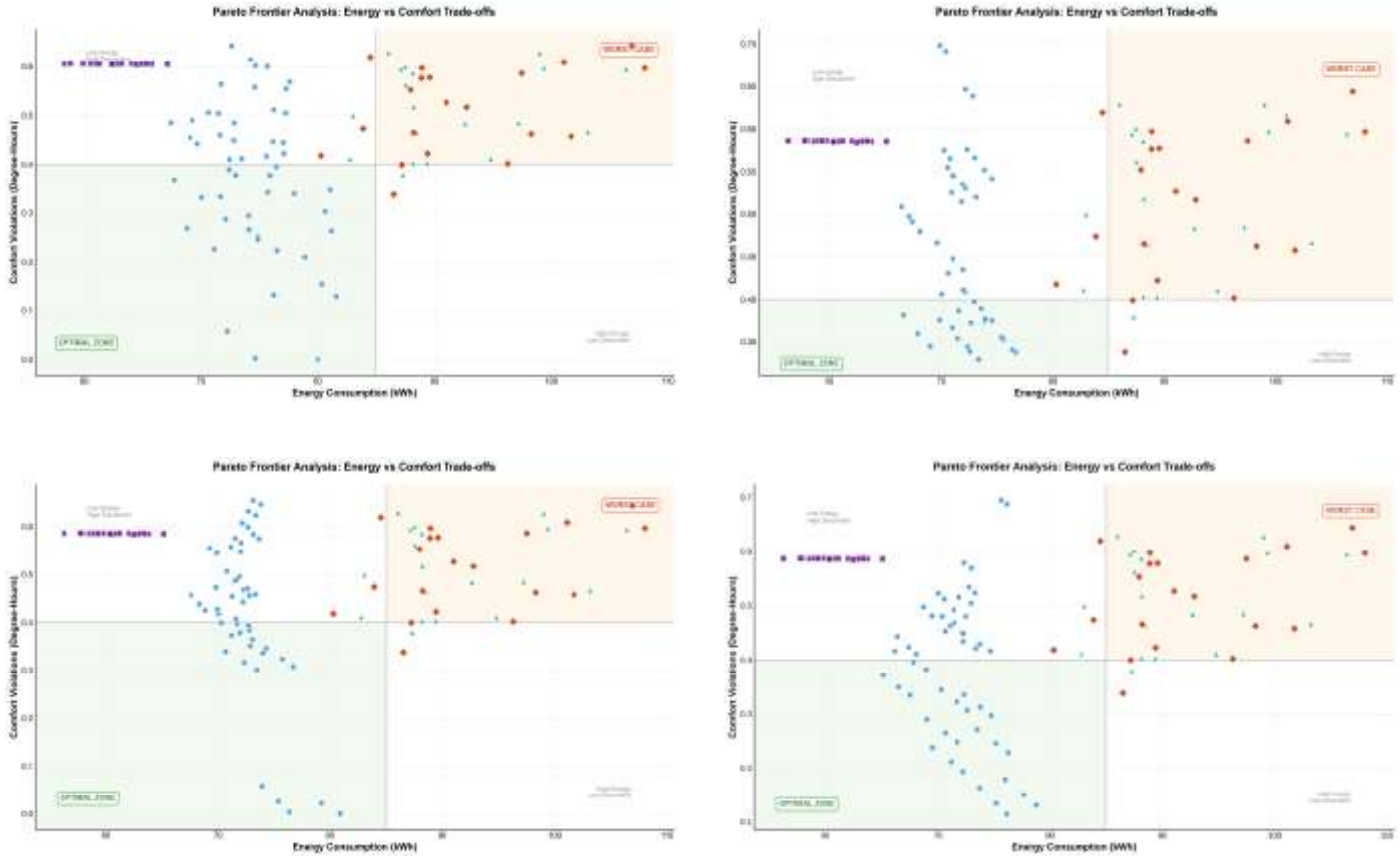
1. Causal Structure Recovery
   i. tested on 6 progressively
   ii. compared against 10 representative methods
   iii. SHD, F1, precision-recall, cost/risk
2. Embodied Control Performance
   i. tested in 4 embodied control scenarios
   ii. compared against 3 baselines
   iii. 2 complimentary evaluation perspectives

Results: **Causal World Model Fidelity**

Method	Base	Noisy	Hidden	ASHRAE	Physical	Large-Scale
PC	4	4	4	4	2	49
SAM	8	6	8	4	7	21
LLM	7	7	7	4	2	17
GIES	6	10	8	4	2	22
JCI	8	3	8	13	6	28
ABCD	5	8	6	4	8	23
Causal Bandits	8	5	9	8	7	43
ICP	5	5	4	2	6	39
IID	4	4	4	12	2	56
NOTEARS	8	9	3	6	4	26
PolicyGRID	**0**	**2**	**0**	**1**	**0**	**13**



- PolicyGRID has lowest SHD across all setups
- Baselines degrade with complexity



Results: **Policy Performance**

- Highest hypervolume and lowest violation rates with PolicyGRID
- Causal DAG critical for policy quality
- Baselines (PID, Correlation) show low hv and high violations
- PolicyGRID dominates Pareto front with low energy and low discomfort

Policy	Base		Noisy		Hidden-Vars		Large-Sim	
	hv↑	V↓	hv↑	V↓	hv↑	V↓	hv↑	V↓
ASHRAE	8.81	8.85	8.87	8.86	9.12	9.34	8.93	8.95
Correlation	8.81	19.87	8.76	20.81	8.79	21.13	8.82	20.78
PolicyGRID (w/o DAG)	18.72	24.13	20.42	24.21	19.87	23.98	20.41	24.24
PolicyGRID	**24.55**	**6.82**	**21.90**	**7.37**	**20.91**	**7.41**	**24.06**	**7.53**

hv=Hypervolume, V=Violation %

Looking Ahead

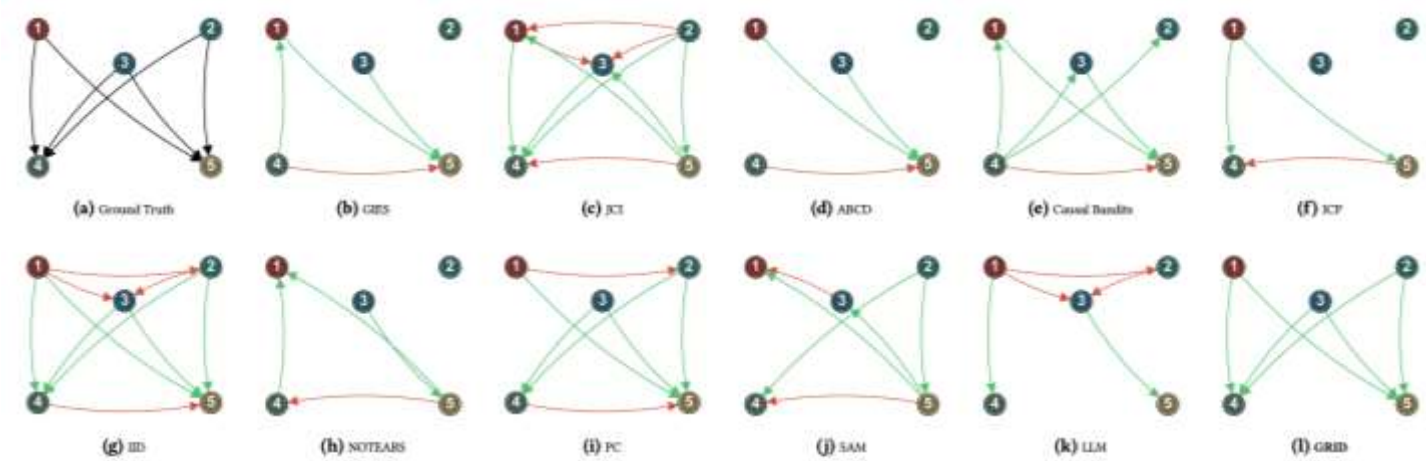


Figure 1: Discovered causal DAGs by different methods in the *Base Simulation* setup. The ground truth DAG (a) illustrates the true causal structure, while the other panels show outputs from benchmark methods in Section 5.3 and GRID. Nodes are indexed as follows: 1. temperature, 2. humidity, 3. air quality, 4. energy consumption, 5. overall satisfaction. Green solid edges indicate correct (true positive) causal relationships; red dashed edges indicate incorrect (false positive) ones. While individual node connections may be visually dense, the key takeaway lies in the structural fidelity of each method. GRID achieves perfect reconstruction, matching the ground truth without spurious edges.

- Cycles in causal structures like BMS
- Scalability of causal discovery
- Reliability via uncertainty-aware inference and real-world deployments
- Adaptation with online updates and continual (temporal) causal learning
- Broader applications in robotics, transport, assistive tech, and other multi-objective control domains