Auto-Formulating Dynamic Programming Problems with Large Language Models

Chenyu Zhou

Shanghai Jiao Tong University

Joint work with *Jingyuan Yang*§, *Linwei Xin*†, *Yitian Chen*‡, *Ziyan He*⋆, *Dongdong Ge**

* Shanghai Jiao Tong University
§ Chicago Booth
† Cornell ORIE
‡ Cardinal Operations
⋆ Shanghai University of Finance and Economics

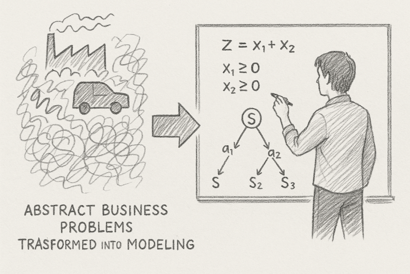
July, 2025

When Solvers Are Powerful but Modeling Is Manual

GUROBI OPTIMIZATION

IBM CPLEX

COPT Cardinal Optimizer



- Solvers can handle optimization problems with millions of variables in seconds.
- Translating real-world problems into decision-making models still requires manual effort from domain experts.

LLM Performance on Auto-Formulation

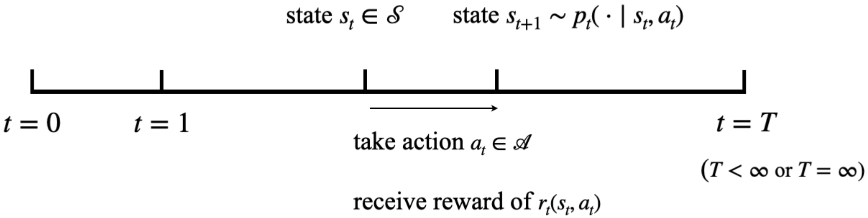
- LLMs perform well on deterministic optimization problems (e.g., LP, MILP, NLP), but struggle with textbook-level DP exercises.

Type	Model	NL4OPT	MAMO-E	MAMO-C	OptMATH	Micro
Baseline	GPT-4	89.0	87.3	49.3	16.6	70.9
	DeepSeek-V3	95.9	88.3	51.1	32.6	75.3
Agent-based	OptiMUS	78.8	-	-	-	-
Fine-tuned	ORLM-Llama3-8B	85.7	82.3	37.4	-	-
	OptMATH-Qwen2.5-7B	94.7	86.5	51.2	24.4	73.5

Source: Lu et al. (2025)

Type	Model	Param	Easy	Hard	Micro
Baseline	o1	*300B	57.8	31.0	50.0
	GPT-4o	*200B	45.6	19.0	37.1
	DeepSeek-V3	671B	51.1	26.2	43.2
Fine-tuned	ORLM-Llama3-8B (pass@1)	8B	1.1	0.0	0.8
	ORLM-Llama3-8B (pass@10)	8B	11.1	2.4	8.3

Formulating a Dynamic Programming Problem



- Determine the action to take at each state s_t in period t to maximize the expected reward at the end of the horizon:

$$\text{(Finite)} \quad v_t(s_t) = \sup_{a \in A_{s_t, t}} \left[r_t(s_t, a) + \sum_{j \in S_{t+1}} p_t(j|s_t, a) v_{t+1}(j) \right], \quad \forall s_t, t$$

- Infinite-horizon settings: discounted and average reward.

Why is DP Challenging to Auto-Formulate?

An Inventory Example

A warehouse has an **end-of-period** capacity of 3 units. During a production period, a \$4 setup cost is incurred, and a holding cost of \$1 per unit applies to the period's ending inventory. Variable production costs are \$1 per unit. Demand each period is equally likely to be either 1 or 2 units, and all demand must be fulfilled immediately. The discount factor is $\beta = 0.8$. The objective is to minimize the expected discounted costs over an infinite horizon, starting with zero initial inventory.

- A toy inventory problem with a capacity constraint.

All Advanced Commercial LLMs Failed

GPT-4o

ChatGPT 4o ⌄ 临时聊天

Problem Overview

- **Inventory system:**
  - Capacity: 3 units
  - Setup cost: $4 if any production occurs in a period
  - Holding cost: $1 per unit of ending inventory
  - Variable production cost: $1 per unit produced
  - Demand: 1 or 2 units with equal probability (i.e., 0.5 each)
  - All demand must be fulfilled immediately (no backordering)
  - Discount factor: β = 0.8
  - Starting inventory: 0

Key Definitions

- Let the **state** be the current inventory level $x \in \{0, 1, 2, 3\}$
- Let the **action** be the production quantity $a \in \{0, 1, 2, 3 - x\}$ to not exceed capacity

o4-mini-high

We've set up a classic infinite-horizon, discounted dynamic-programming (DP) model with these pieces:

1. **State**
$s \in \{0, 1, 2, 3\}$ is the inventory on hand *at the start* of a period (after last period's demand).

2. **Action**
$x \geq 0$ is the number of units you produce at the *start* of the period. We restrict $s + x \leq 3$ (capacity) and $s + x \geq 2$ so you can always meet the worst-case demand of 2.

o3

What the script does

1. **Enumerates states** $i=0,1,2,3$ (opening inventory).
2. **Enumerates feasible actions** q obeying
   - capacity: $i+q\le 3$
   - service guarantee: $i+q\ge 2$ (enough to cover the worst-case demand 2).
3. Uses **value iteration** to solve the Bellman optimality equations
$V(i) = \min_{q \in \mathcal{A}(i)} \mathbf{E}\left[c(i, q, D) + \beta V(i')\right]$,
with
   - immediate cost $c(i,q,D)=\mathbf1_{{q>0}},4 + q + \text{holding}=1\cdot(i+q-D)$,
   - random demand $D\in{1,2}$ with $\Pr(D=1)=\Pr(D=2)=0.5$.

DeepSeek-R1

So possible actions a are:

max(0, d - x) ≤ a ≤ 3 - x.

But demand is random when we choose a, so we can't base a on d directly. Maybe the interpretation is that we choose a, then observe d, and must have x + a ≥ d (i.e., a ≥ d - x), otherwise it's infeasible. But since d is random when choosing a, we must choose a such that for all possible d, x + a ≥ d. So a ≥ max(0, d_max - x). Here d_max is 2, so a ≥ max(0, 2 - x).

But also a ≥ 0, and x + a ≤ 3 ⇒ a ≤ 3 - x.

So possible a is from max(0, 2 - x) to 3 - x.

LLMs Misinterpret Implicit Constraints

An Inventory Example

A warehouse has an **end-of-period** capacity of 3 units. ... Demand each period is equally likely to be either 1 or 2 units, and all demand must be fulfilled immediately. ...

GPT-4o's Response

...
Capacity: 3 units
...
$x \in \{0, 1, 2, 3\}$
...
$a \in \{0, \ldots, 3 - x\}$
...

- Interprets 3 as an absolute capacity.
- Production is limited so that $x + a \leq 3$.

DP vs. LP/MILP/NLP

- **Order of events** matters: when decisions occur, costs incur, and states evolve.
- **Complex transitions with uncertainty** are common to most DP problems.
- **Implicit constraints** often buried in narratives rather than explicit equations.
- **For LLMs**: there is no testing dataset (we provide the first one!)

Literature Comparison: Auto-Formulation

Reference: ORLM (Huang et al. 2025) *vs.* **Ours**

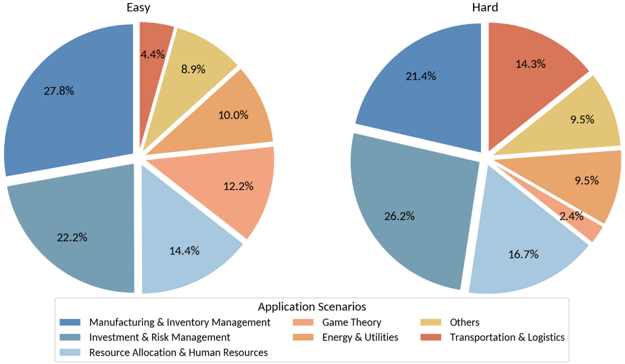
- **Problem Focus:** Mainly LP/MILP *vs.* DP (more complex)
- **Data Source:**
  - Private seed data *vs.* Curated from textbooks
  - Well-established benchmarks *vs.* No existing benchmark
- **Synthetic Data Generation:**
  - Forward only *vs.* Forward + Backward (for accuracy and diversity)
  - Zero-shot or static prompting *vs.* RAG-based few-shot prompting (improves output quality)
- **Training Pipeline:**
  - SFT only *vs.* SFT + RL (for better solution quality and robustness)

Literature Comparison: Data Synthesis (Backward)

Reference: Lyapunov function discovery (Alfarano et al. 2024)

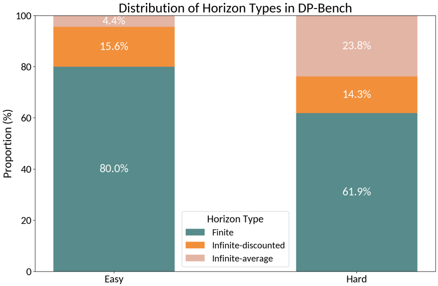
- **Forward**: Start from a dynamic system → find Lyapunov via SOS solvers.
- **Backward**: Start from Lyapunov → construct a dynamic system.
- **In optimization:**
  - Forward generation has no guarantee of correctness.
  - Recent works rely on backward only: Yang et al. (2024b), Lu et al. (2025)
- We show both forward and backward generation are necessary:
  - Forward: diversity
  - Backward: correctness

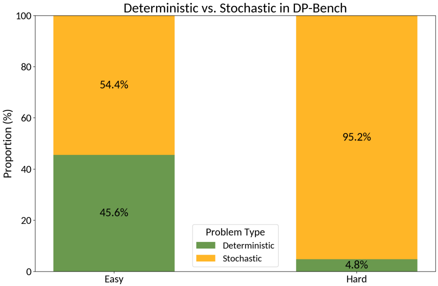
DP-Bench: The First DP Benchmark



- 132 DP problems: 90 Easy + 42 Hard, all with numeric ground-truth answers.
- All problems were manually curated and enriched from textbooks (Winston 2004, Puterman 2005).

DP-Bench: The First DP Benchmark

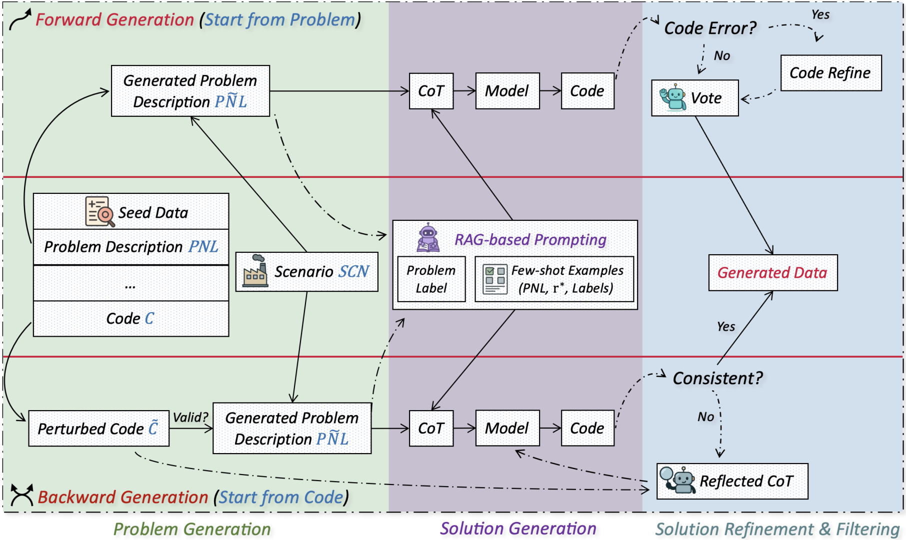




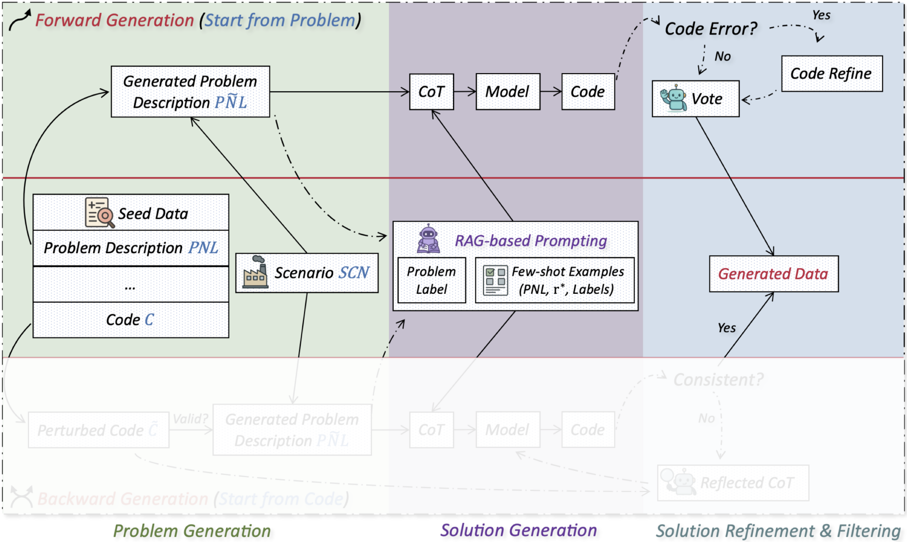
Label	Easy (%)	Hard (%)
action-dependent transition probability	22.22	66.67
optimal stopping problem	4.44	14.29
truncation-required state space	4.44	11.90
time-dependent state space	4.44	7.14
continuous or non-integer state space	2.22	7.14

- Cover deterministic & stochastic, finite & infinite horizon settings.
- Hard subset is a real stress-test for model auto-formulation.

Data Synthesis Pipeline



Data Synthesis Pipeline - Forward Generation



Examples for Forward Generation

Original Problem (PNL)

A repairman who services $Q = 4$ facilities moves between location s and location j in any period according to the stationary transition probability $\boldsymbol{p}(j \mid s)$ (omitted here). An equipment trailer which carries spare parts and tools may be located at any one of $M = 3$ sites. If the trailer is at site m and the repairman is at facility j, the cost of obtaining material from the trailer is $c(m, j)$, where:

$$\begin{array}{llll} c(1,1) = 2, & c(1,2) = 5, & c(1,3) = 6, & c(1,4) = 8 \\ c(2,1) = 3, & c(2,2) = 4, & c(2,3) = 7, & c(2,4) = 9 \\ c(3,1) = 4, & c(3,2) = 6, & c(3,3) = 5, & c(3,4) = 7 \end{array}$$

The cost of moving the trailer from site m to site j is $d(m, j)$, where:

$$\begin{array}{lll} d(1,1) = 0, & d(1,2) = 3, & d(1,3) = 4 \\ d(2,1) = 2, & d(2,2) = 0, & d(2,3) = 5 \\ d(3,1) = 3, & d(3,2) = 4, & d(3,3) = 0 \end{array}$$

The decision maker's objective is to dynamically relocate the trailer so as to minimize expected costs. Assume that the decision maker observes the location of the repairman and trailer, relocates the trailer, and then the repairman moves and services a facility. Given the costs above, determine the minimum expected cost incurred over a decision-making horizon of 5 periods when trailer and repairman are both at site 1.

Generated Problem ($\widetilde{\text{PNL}}$)

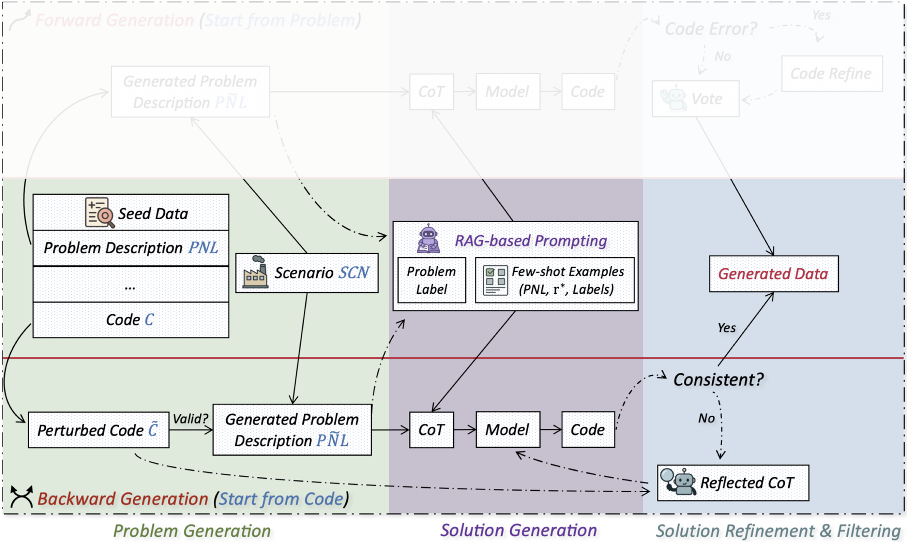
A thermal power plant operator manages a coal plant and aims to minimize operational costs while meeting fluctuating energy demands over a 5-day period. The plant can be operated at one of three output levels each day: low, medium, or high, each with associated operational costs and energy outputs.
The operational cost per day, depending on the output level, is given by:

- Low output: \$50 with energy output = 20 units
- Medium output: \$100 with energy output = 40 units
- High output: \$150 with energy output = 60 units

Daily energy demand fluctuates with probabilities: 20 units w.p. 30, 40 units w.p. 50, and 60 units w.p. 20. If the energy output exceeds the demand, the excess energy incurs a storage cost of \$2 per unit. If the demand exceeds the output, the plant incurs a penalty cost of \$5 per unit of unmet demand. The objective is to minimize the total expected operational and penalty costs over the 5-day period, starting with a high output level on day 1. Using dynamic programming, determine the minimum expected cost after 5 days.

Scenario
- **Category:** *Thermal Plant Operational Scheduling*
- **Characteristics:** *Focuses on minimizing operational costs while meeting energy demands*

Data Synthesis Pipeline - Backward Generation



Examples for Backward Generation

```python
T = 6                  # Increased the time horizon
Q = 5                  # Increased the number of states
M = 4                  # Increased the number of maintenance sites

# Adjusted transition probabilities to sum to 1 for each state
p = {
    1: {1: 0.2, 2: 0.3, 3: 0.2, 4: 0.2, 5: 0.1},
    2: {1: 0.15, 2: 0.2, 3: 0.25, 4: 0.25, 5: 0.15},
    3: {1: 0.25, 2: 0.25, 3: 0.2, 4: 0.15, 5: 0.15},
    4: {1: 0.3, 2: 0.2, 3: 0.2, 4: 0.1, 5: 0.2},
    5: {1: 0.2, 2: 0.2, 3: 0.25, 4: 0.15, 5: 0.2},
}

# Adjusted costs
c = {
  1: {1: 3, 2: 6, 3: 5, 4: 7, 5: 4},
  2: {1: 4, 2: 5, 3: 8, 4: 10, 5: 6},
  3: {1: 5, 2: 7, 3: 6, 4: 8, 5: 7},
  4: {1: 6, 2: 8, 3: 7, 4: 9, 5: 8},
}

# Adjusted movement costs
d = {
  1: {1: 0, 2: 4, 3: 5, 4: 2},
  2: {1: 3, 2: 0, 3: 6, 4: 3},
  3: {1: 4, 2: 5, 3: 0, 4: 4},
  4: {1: 2, 2: 3, 3: 4, 4: 0},
}

V = [[[0.0 for _ in range(M+1)] for _ in range(Q+1)] for _ in range(T+1)]
Policy = [[[0 for _ in range(M+1)] for _ in range(Q+1)] for _ in range(T)]
for t in range(T-1, -1, -1):
    for s in range(1, Q+1):
        for m_old in range(1, M+1):
            best_cost = float('inf')
            best_site = -1
            for a in range(1, M+1):
                move_cost = d[m_old][a]
                expected_servicing = 0.0
                for j in range(1, Q+1):
                    prob = p[s][j]
                    cost_service = c[a][j]

                    future_cost = V[t+1][j][a]

                    expected_servicing += prob * (cost_service + future_cost)

                total_cost = move_cost + expected_servicing

                if total_cost < best_cost:
                    best_cost = total_cost
                    best_site = a
            V[t][s][m_old] = best_cost
            Policy[t][s][m_old] = best_site

s0 = 1
m0 = 1
optimal_value = V[0][s0][m0]
```

Figure: Perturbed Code $\tilde{C}$

Generated Problem ($\tilde{\text{PNL}}$)

In India, a thermal plant manager faces the challenge of scheduling operations for a coal plant over a 6-day period. The plant operates under variable energy demand and seeks to minimize operational costs while ensuring demand fulfillment. The plant manager begins with one plant site operational on Day 1 and must decide whether to continue or shift operations to one of four different plant sites each day. Each site incurs a specific movement cost depending on the current site, and daily operational costs vary based on energy demand.The daily energy demand is represented by 5 discrete states. The transition probability matrix, P, for energy demand states from Day t to Day $t+1$ is given as follows:

$$P = \begin{bmatrix} 0.2 & 0.3 & 0.2 & 0.2 & 0.1 \\ 0.15 & 0.2 & 0.25 & 0.25 & 0.15 \\ 0.25 & 0.25 & 0.2 & 0.15 & 0.15 \\ 0.3 & 0.2 & 0.2 & 0.1 & 0.2 \\ 0.2 & 0.2 & 0.25 & 0.15 & 0.2 \end{bmatrix}$$

Operational costs at each plant site based on the energy demand state are defined by matrix C (omitted here). Furthermore, transitioning between plant sites incurs movement costs specified by matrix D (omitted here). The goal is to determine the minimum expected total operational cost over the 6-day period starting with energy demand stat[...] expected op[...]

Scenario

- **Category:** *Thermal Plant Operational Scheduling*
- **Characteristics:** *Focuses on minimizing operational costs while meeting energy demands*

Reflected CoT: Learning from Correct Solution

- **Motivation:** Hard problems are often discarded due to incorrect outputs
  - LLMs struggle to solve DP problems (only 47.22 pass@5)

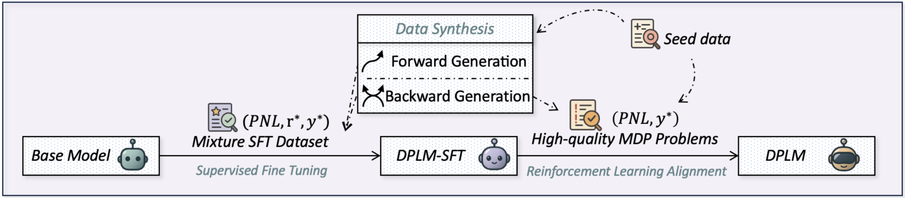
- **Reflected CoT enables:**
  - Retaining hard questions with verified correct solutions
  - Generating traceable reasoning: initial attempt → self-reflection → revision until correct

- **Why it works:** Backward generation provides the ground-truth solution for reflection and correction

- **Results:** Recovers **20.3%** of problems that would otherwise be discarded

High-Quality Data for RL Fine-Tuning!

Training Recipes



- Two Stage Training: Supervised Fine-Tuning (SFT) + RL
- SFT (cold-start the base):
  - 113K paired trajectories (PNL, CoT, M, C)
    (70K forward, 34K backward, 8K reflection)
- RL (exploration & debiasing):
  - $\mathcal{D}_{\mathrm{RL}}$ 8K verifiable problems
  - GRPO / DPO

Main Result

Type	Model	Parameters	Easy(%)	Hard(%)	Micro(%)	Macro(%)
Baseline Large-Scale	o1	*300B	57.8	31.0	50.0	44.4
	GPT-4o	*200B	45.6	19.0	37.1	32.3
	DeepSeek-R1	671B	**73.3**	28.6	**59.1**	51.0
	DeepSeek-V3	671B	51.1	26.2	43.2	38.7
	Qwen-2.5-72B-Instruct	72B	41.1	19.0	34.1	30.1
	Qwen-2.5-32B-Instruct	32B	35.6	19.0	30.3	27.3
Baseline Small-Scale	Gemma-2-9B-It	9B	4.4	2.4	3.8	3.4
	LLama-3.1-8B-Instruct	8B	7.8	2.4	6.1	5.2
	Qwen-2.5-7B-Instruct	7B	10.0	2.4	7.6	6.2
Ours	DPLM-7B-SFT	7B	38.9	21.4	33.3	30.2
	DPLM-7B-SFT-GRPO	7B	65.6	**38.1**	56.8	**51.9**

* These are estimations (e.g., Ben Abacha et al. 2025), given that OpenAI has not publicly disclosed the information.

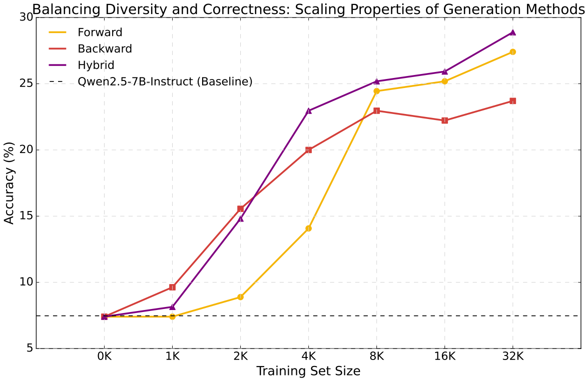
- DPLM achieves better performance than its teacher model GPT-4o.
- DPLM outperforms DeepSeek-R1 on hard problems, despite using 100× fewer parameters.

Ablation Study

Model	Easy (%)	Hard (%)	Micro (%)	Macro (%)
Base-Model-7B (no further training)	10.0	2.4	7.6	6.2
DPLM-7B (SFT only)	38.9	21.4	33.3	30.2
DPLM-7B (DPO only)	23.3	9.5	18.9	16.4
DPLM-7B (GRPO only)	27.8	14.3	23.5	21.1
DPLM-7B (SFT $\rightarrow$ DPO)	48.9	21.4	40.2	35.2
DPLM-7B (SFT $\rightarrow$ GRPO)	**65.6**	**38.1**	**56.8**	**51.9**

- SFT is necessary!

Insights: Forward vs. Backward



Pros vs. Cons

Forward	**Backward**
+ Breadth / Diversity	+ Depth / Correctness
− Noisy labels	− Limited diversity

Our Contributions

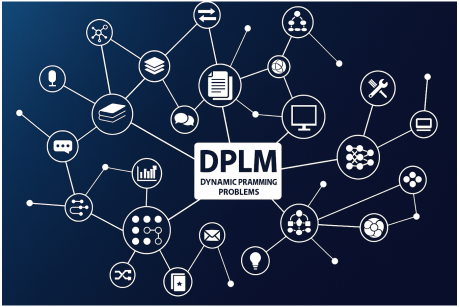
- **DP-Bench:** 132 textbook-style DP problems, first public benchmark for DP
- **Lightweight yet strong model:** our **DPLM-7B** attains high accuracy while using **100**× fewer parameters
- **Scalable data synthesis pipeline from scratch:**
  - Forward and backward are both needed!
    - Forward generation for diversity
    - Backward generation for depth & correctness

Thanks for Your Attention!

Auto-Formulating Dynamic Programming Problems with Large Language Models

Chenyu Zhou, Jingyuan Yang, Linwei Xin, Yitian Chen, Ziyan He, Dongdong Ge

Draft Available Online!



Literature Comparison: Learning from Answer

Reference: Self-Taught Reasoner (STaR) (Zelikman et al. 2022)

- **Problem Focus:** Short math/commonsense QA with known answers *vs.* DP (possibly flawed problem description)
- STaR generates rationales from answers
- For DP, even given full solutions, generating valid CoT is challenging, especially when the problem itself may be flawed
- **Our Approach (Reflected CoT):** Compare with ground-truth solution, identify errors, and retry
- **Data Improvement:**
  - Boosts the proportion of high-quality problems for training data
  - Mitigate a common issue in datasets that are mostly accurate but overly simplistic

Examples from DP-Bench

Easy

An electronics firm has a contract to deliver the following number of radios during the next three months; month 1, 200 radios; month 2, 300 radios; month 3, 300 radios. For each radio produced during months 1 and 2, a $10 variable cost is incurred; for each radio produced during month 3, a $12 variable cost is incurred. The inventory cost is $1.50 for each radio in stock at the end of a month. The cost of setting up for production during a month is $250. Radios made during a month may be used to meet demand for that month or any future month. Assume that production during each month must be a multiple of 100. Given that the initial inventory level is 0 units, use dynamic programming to determine the minimum total cost of three months.

- Finite-horizon
- Deterministic demand

Hard

Daily demand for paint brushes at a particular store follows the demand distribution: *Demand:* 0, 1, 2, 3, 4, *Probability:* 0.7, 0.15, 0.1, 0.04, 0.01. The stock level is reviewed in the evening every four days and when warranted an order is placed at the central warehouse to augment stock. Orders arrive two days later (a two day lead time) and are available to meet demand on the morning of the third day following the review. Demand not satisfed from stock on hand is never filled. Management imposes a penalty to account for this. Find the minimizes expected total ordering, holding and shortage costs under the assumption that the fixed cost for placing an order is $0.20, the daily per unit holding cost is $0.01 and the per unit penalty cost for unfilled orders is $0.50. Daily costs are incurred after the demand for the day is realized. Determine the minimum long-run average cost, rounded to four decimal places.

- Avg. cost over infinite-horizon
- Review every 4 days
- Positive lead time (2 days)

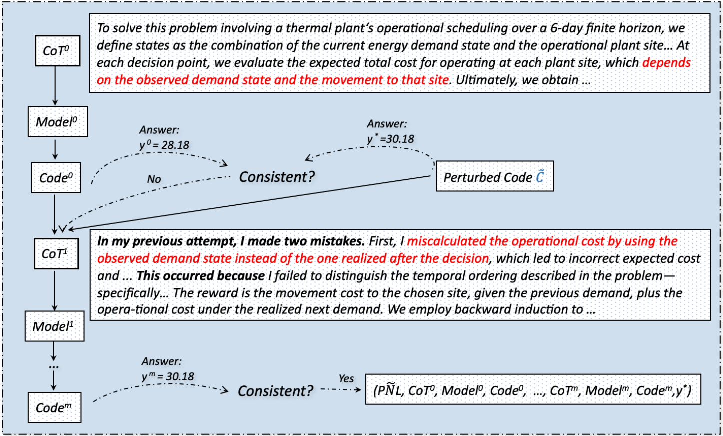
Why We Need Both Forward and Backward

- Limited seed data (only 91 problems).
- **Forward (diverse formulation)**:
  - Enables diverse problem generation.
  - Lacks performance guarantees.
- **Backward (high-quality solution)**:
  - Guarantees correctness with self-reflective reasoning.
  - Variety is constrained by seed coverage.

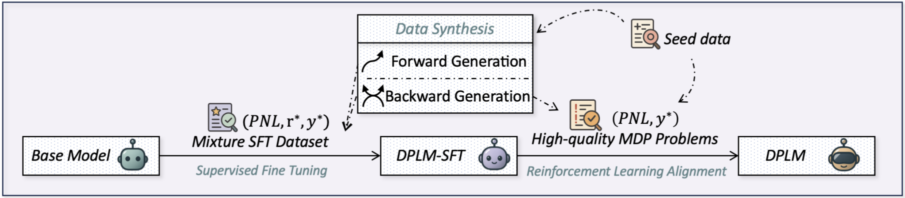
DP-specific adaptation:

- **Few-shot by problem type**:
  - Similar problem description, different models or algorithms (e.g., inventory problem with finite vs. infinite horizon).
  - Use type labels to select relevant examples.

Examples for Reflected CoT



Training Recipes - SFT



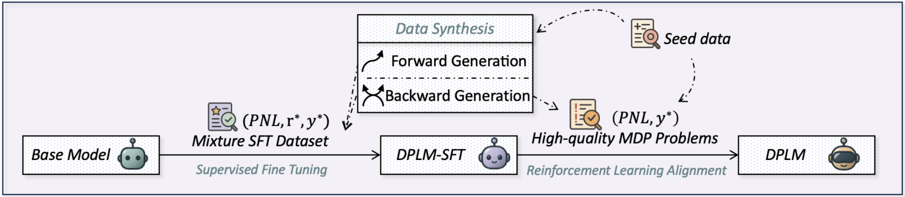
- **Objective.** Maximum-likelihood on teacher traces:

$$\mathcal{L}_{\text{SFT}} = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{SFT}}}\big[\log \pi_\theta(y \mid x)\big]$$

- **Data.** 113K paired trajectories (PNL, CoT, M, C)
  (70K forward, 34K backward, 8K reflection)

- **Intuition.** Teach domain reasoning knowledge & formatting, narrow RL search space.

Training Recipes - RL



- **Data.** $\mathcal{D}_{\text{RL}}$ 8K verifiable problems.
- **GRPO** (online, higher performance potential):

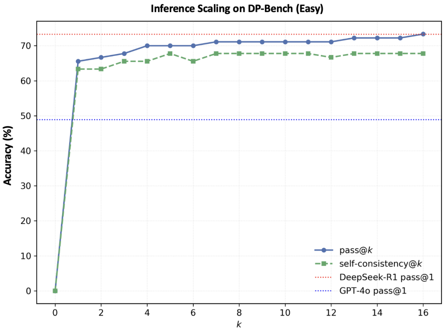
$$\max_{\theta} \mathbb{E}_{x,y_i}\Big[\min\big(\rho_i A_i, \text{clip}(\rho_i, 1\pm\varepsilon)A_i\big)\Big] - \beta D_{\text{KL}}\big(\pi_\theta \parallel \pi_{\text{ref}}\big),$$

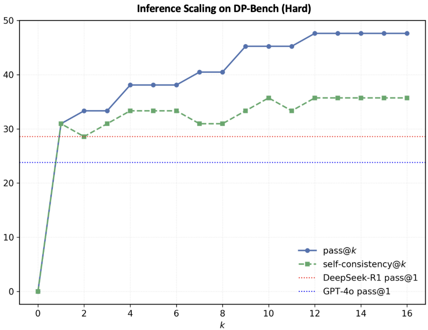
$$A_i = \frac{r_i - \bar{r}}{\sigma_r}, \quad \rho_i = \frac{\pi_\theta(y_i|x)}{\pi_{\text{old}}(y_i|x)}$$

- **DPO** (offline, more stable and computationally cheaper): minimize preference loss:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\big[\log \sigma\big(\beta(\log \pi_\theta(y_w) - \log \pi_\theta(y_l))\big)\big]$$

Inference Scaling Analysis





- Hard problem errors arise from reasoning
- Easy problem errors arise from missing domain knowledge

Model Size Scaling Analysis

Parameters	Easy (%)			Hard (%)			Micro (%)			Macro (%)		
	Base	+SFT	Δ	Base	+SFT	Δ	Base	+SFT	Δ	Base	+SFT	Δ
0.5 B	0.0	8.9	+8.9	0.0	0.0	0.0	0.0	6.1	+6.1	0.0	4.5	+4.5
1.5 B	3.3	15.6	+12.3	0.0	2.4	+2.4	2.3	11.4	+9.1	1.7	9.0	+7.3
3 B	4.4	25.6	+21.2	0.0	4.8	+4.8	3.0	18.9	+15.9	2.2	15.2	+13.0
7 B	10.0	38.9	**+28.9**	2.4	21.4	**+19.0**	7.6	33.3	**+25.7**	6.2	30.2	**+24.0**
14 B	24.4	48.9	+24.5	9.5	23.8	+14.3	19.7	40.9	+21.2	17.0	36.4	+19.4
32 B	35.6	55.6	+20.0	19.0	28.6	+9.6	30.3	47.0	+16.7	27.3	42.1	+14.8

- Below 7B, model size is the bottleneck; above 7B, data is.