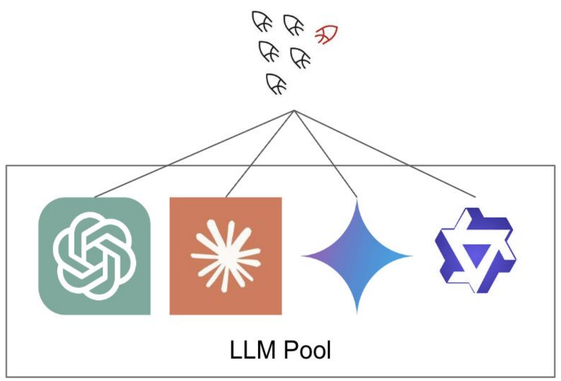
Learning to Orchestrate Agents in Natural Language with the Conductor

Stefan Nielsen[1], Edoardo Cetin[1], Peter Schwendeman[2], Qi Sun[1 3], Jinglue Xu[1], Yujin Tang[1]
[1]Sakana AI, Japan [2]University of Michigan, US [3]Institute of Science Tokyo, Japan

We propose the Conductor, a small LM which designs adaptive agentic scaffolds over powerful workers to solve complex tasks

Background

Multi-Agent Orchestration



Reinforcement Learning & Reasoning

The **elicitation of "reasoning"** in LLMs, broadly understood as the **capability to understand and decompose highly complex tasks into methodically solvable steps**, with approrpiate **exploration, self-refinement, backtracking, and aggregation** of intermediate reasoning pathways, is increasingly accredited to **reinforcement learning**

Multi-agent orchestration ought to resemble this same reasoning process, but at a macro-level over agents. **This naturally invites RL as our training paradigm**

The RL Conductor

The **Conductor objective is to solve tasks indirectly** by **designing agentic workflows** to input questions.

Each workflow is made up of
1) The **participating agents**
2) The **subtasks** assigned to each agent
3) The **communication topology** between agents

Pure **end-to-end reward maximization** with GRPO:

$$J(\theta) = \mathbb{E}_{q\sim D,\,\{o\}_1^G\sim\pi_\theta(\cdot|q)}\left[\frac{1}{G}\sum_{i=1}^{G}\left(\min\left(r_i A_i,\ \mathrm{clip}\left(r_i,\,1-\epsilon,\,1+\epsilon\right)A_i\right)-\beta\,\mathbb{D}_{\mathrm{KL}}\left(\pi_\theta\,\|\,\pi_{\mathrm{ref}}\right)\right)\right]$$

over the **format reward**, which **enforces the parseable Conductor strucured response**, and the **final downstream task reward**

User

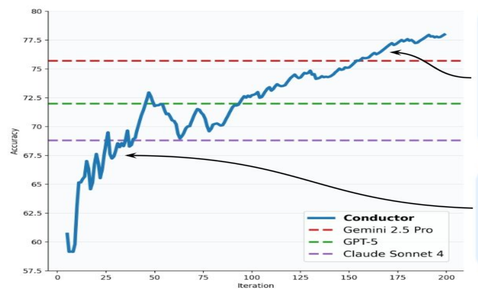
We call a subarray of an array complete if the number of distinct elements in the subarray is equal to the number of distinct elements in the whole array. Return the number of complete subarrays.

Conductor

Here's the plan: 1) Model 2 will develop an algorithm to efficiently count all possible complete subarrays for a given array, 2) Model 0 will implement the function to solve the problem.

model_id = [2, 0]
subtasks = ["Develop an efficient algorithm to count the number of complete subarrays of an array", "Implement the algorithm described by the previous agent in Python"]
access_list = [[], ["all"]]

Training



. . .

subtasks = ["Understand the problem and find a strategy to calculate the punishment number. Identify the integers whose squares can be partitioned into substrings that sum to the integer itself. Then, calculate the sum of the squares of these integers.",

"Verify the proposed strategy and find any gaps or errors in the understanding. Provide a step-by-step verification process in <idea>tags.",

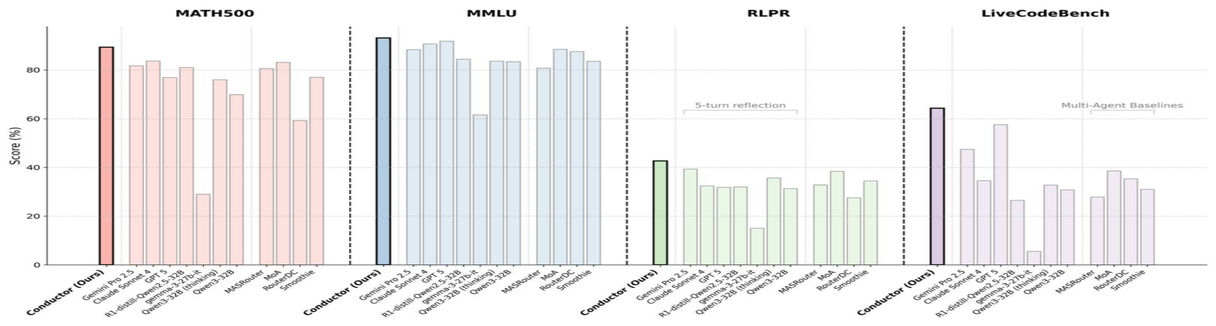
"Refine the proposed strategy if necessary and then implement the punishmentNumber function in Python according to the given constraints. Provide a step-by-step solution in <idea>tags and return the final code in <answer>tags as specified."]

. . .

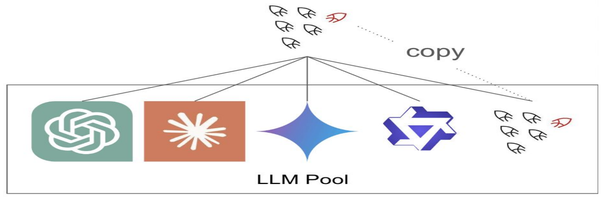
subtasks = ["Analyze the given parameters of the simple harmonic motion (period and amplitude) and the initial position of the object. Identify the mathematical equation that describes the position of the object as a function of time.",

"Calculate the time it takes for the object to go from x = 5.70 cm to x = -1.60 cm using the equation derived in the previous step and the given parameters."]

Empirical Results



Recursion



Model	AIME25	BigCodeBench	GPQA-D	Average score
gemma-3-27b-it	6.67	10.8	33.33	16.93
Qwen3-32B	23.33	23.0	54.05	33.46
Qwen3-32B (thinking)	23.33	20.9	59.09	34.44
R1-Distill-Qwen-32B	30.00	24.3	51.01	35.10
Gemini Pro 2.5	46.67	35.1	75.25	52.34
Claude Sonnet 4	35.33	35.8	67.30	46.14
GPT 5	46.67	33.8	72.73	51.73
Conductor **(Ours)**	**66.67**	37.8	81.31	**61.93**
Conductor-Recursive **(Ours)**	**66.67**	**40.0**	**82.32**	**63.00**

Efficiency