rStar-Math: Small LLMs Can Master Math Reasoning with Self-Evolved Deep Thinking

rStar-Math: 小模型自进化数学推理

ICML 2025

Xinyu Guan* Li Lyna Zhang*◇ Yifei Liu
Ning Shang Youran Sun Yi Zhu Fan Yang Mao Yang

Microsoft Research Asia

王兆年

2025.9.20

提纲

- 研究背景
- 三项方法创新
- 实验评估
- 总结与讨论

背景：从System–1到System–2

- System-1：一次性整解（快但易错）
- System-2：多步生成 + 评估 + 选择（慢但稳）
- 数学推理更需要“中间步骤质量”而非仅最终答案

System-1

System-2

背景：核心挑战及目标

核心挑战

- 高质量数学 CoT (Chain of Thought) 数据稀缺，蒸馏存在上限
- 过程奖励模型步级标注昂贵且噪声大
- 大模型结合 MCTS 成本高且难题覆盖不足

目标

研究如何让小模型（小于 10B 级别）在数学推理方面达到甚至超越现有的较大模型或商业化模型（如OpenAI o1）。

背景：MCTS 与 UTC

MCTS 通常分为四个循环步骤：

1. 选择（Selection）：基于 UCT(Upper Confidence bounds for Trees) 分数，选择某条当前最优路径。
   - UCT的公式是：

$$\mathrm{UCT}(s) = Q(s) + c\sqrt{\frac{\ln N_{parent}(s)}{N(s)}}; \quad \text{where} \quad Q(s) = \frac{q(s)}{N(s)}$$

参数	含义
$Q(s)$	平均质量分
$N_{parent}(s)$	父节点访问次数
c	平衡探索与开发的常数
$q(s)$	PPM对该节点 reward 的累计预测

2. 扩展（Expansion）：从该路径处扩展新节点（即：生成新的下一步推理动作）
3. 模拟（Rollout）：让策略模型继续向前 rollout 若干步至最终答案
4. 回传（Backpropagation）：将模拟结果沿着路径回传到根节点，更新各节点的统计数据（如胜率、访问次数）。

方法总览：两模型 + 一搜索

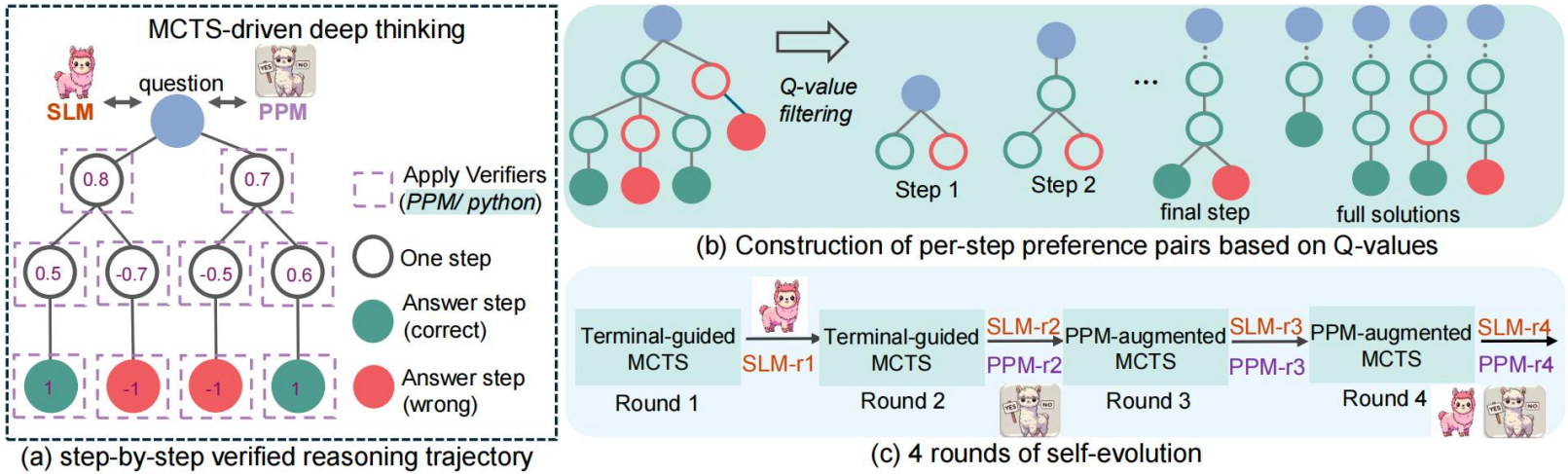


Figure 1: The overview of rStar-Math

方法总览：

- 策略 SLM (Small Language Model)：逐步生成候选步骤
- PPM（Process Preference Model）：对步骤打分（作为过程奖励）
- MCTS (Monte Carlo Tree Search)：在步骤空间里深思考 + 选优

方法设计

方法目标： 训练两个关键模型并将其嵌入 MCTS 中

1. Math Policy SLM（小型策略语言模型，负责逐步生成解题动作）
2. Process Reward Model（PRM，即过程奖励模型 / PPM，用于评估过程步骤的优劣）

方法机制 (Why MCTS)：

- 拆解复杂任务 → 降维生成压力
- 自然产生 step-level 训练数据

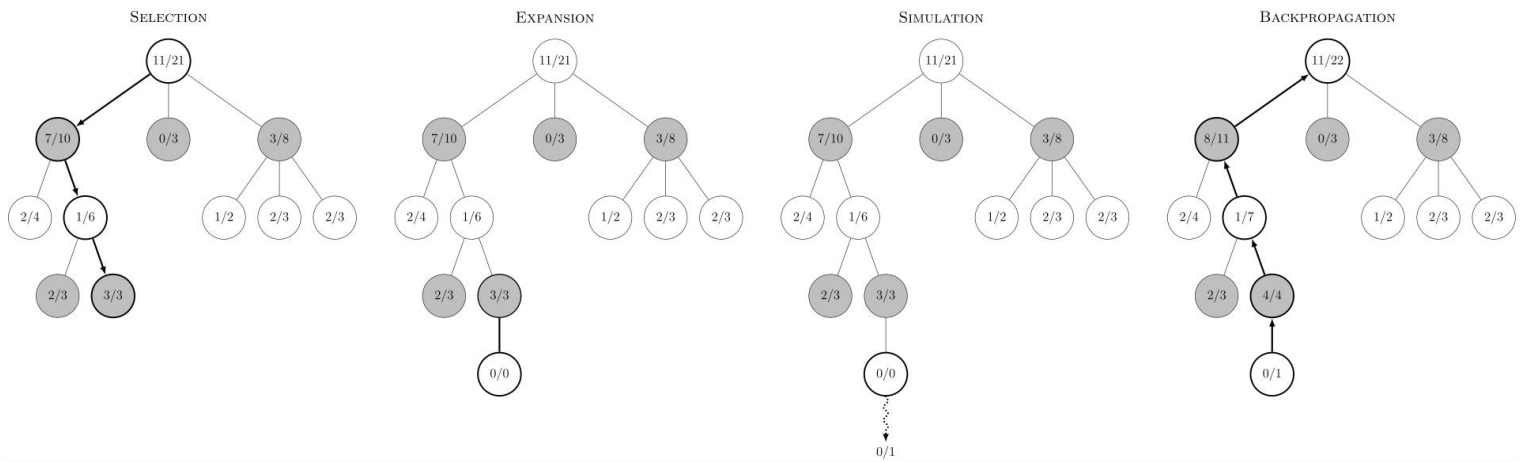


Figure 2: The overview of MCTS

方法 1：Code–augmented CoT 的“逐步验证”轨迹

技术点1： CoT + Python代码

- 策略模型生成步骤时同时生成自然语言描述 + Python 代码
- CoT 作为注释，嵌入生成的代码中
- 若代码能成功运行，步逻辑视为可验证、可信的步骤

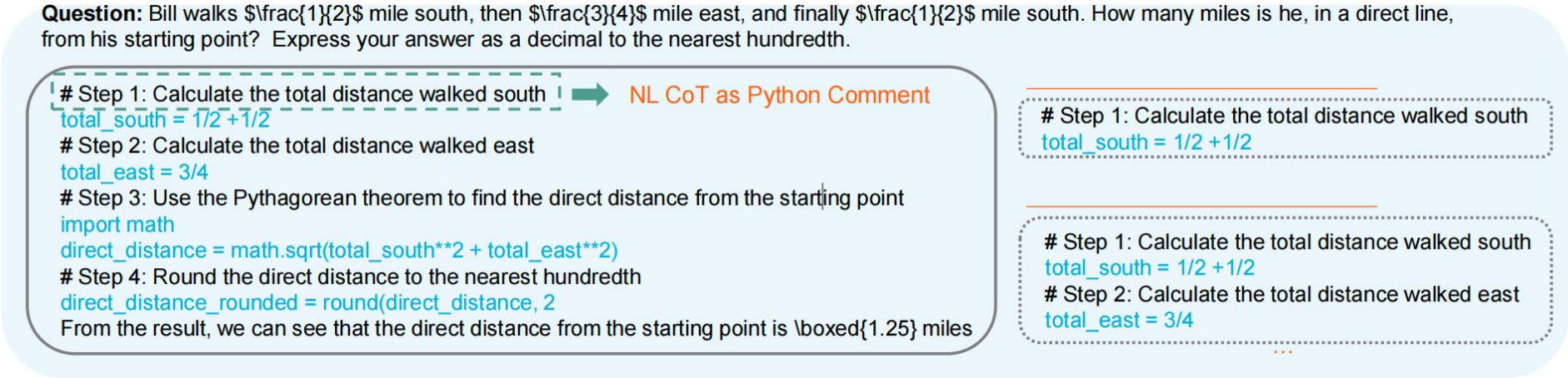


Figure 3： An example of Code–augmented CoT

方法 1：Code-augmented CoT 的“逐步验证”轨迹

技术点2： 准确且稳定的 Q-value 注释机制

- 阶段一：Terminal-Guided Annotation（前两轮）

$$q(s_i)^k = q(s_i)^{k-1} + q(s_d)^k$$

参数	意义
$q(s_i)^k$	第 k 次模拟后，step s_i 的 q 值
$q(s_d)^k$	本次 rollout 的终点节点得分，正确为 +1，错误为 -1

方法 1：Code–augmented CoT 的“逐步验证”轨迹

技术点2： 准确且稳定的 Q–value 注释机制

例子： 题目： 求解方程 $x^2 - 5x + 6 = 0$

- s_1 ：使用求根公式， $x^2 - 5x + 6 = 0$
- s_2 ：a=1, b=-5, c=6
- s_3 ：$x = [5 \pm \sqrt{(25-24)}] / 2$
- s_4 ：$x = [5 \pm 1] / 2$
- s_5 ：x = 2 或 x = 3
- 终局： 正确 → q(sd) = +1

- s_1 ：使用因式分解法， $x^2 - 5x + 6 = 0$
- s_2 ：寻找两个数，乘积为6，和为-5
- s_3 ：这两个数是 -2 和 -3
- s_4 ：(x - 2)(x - 3) = 0
- s_5 ：x = 2 或 x = 3
- 终局： 正确 → q(sd) = +1

- s_1 ：使用因式分解法， $x^2 - 5x + 6 = 0$
- s_2 ：寻找两个数，乘积为6，和为-5
- s_3 ：这两个数是 -1 和 -6
- s_4 ：(x - 1)(x - 6) = 0
- s_5 ：x = 1 或 x = 6
- 终局： 错误 → q(sd) = -1

- s_1 ：使用求根公式， $x^2 - 5x + 6 = 0$
- s_2 ：a=1, b=-5, c=6
- s_3 ：$x = [5 \pm \sqrt{(25-24)}] / 2$
- s_4 ：$x = [5 \pm 5] / 2$ （计算错误： $\sqrt{1} \neq 5$）
- s_5 ：x = 5 或 x = 0
- 终局： 错误 → q(sd) = -1

- 进行 8 次 MCTS Rollout 的 Q 值累积过程：
- 初始化：所有步骤 $q(s_i)^0 = 0$

Rollout	步骤内容	选择轨迹	出现在轨迹	终局结果	Q值更新	累积 Q 值
第1次	"使用因式分解法"	A	A(+3), B(–2)	+1	A的所有步: 0→1	+1
第2次	"寻找两数，积6和–5"	B	A(+3), B(–2)	-1	B的所有步: 0→-1	-1
第3次	"两数是–2和–3"	A	A(+3)	+1	A的所有步: 1→2	+3
第4次	"两数是–1和–6"	C	B(–2)	+1	C的所有步: 0→1	-2
第5次	"使用求根公式"	D	C(+2), D(–1)	-1	D的所有步: 0→-1	+1
第6次	"x = 2 或 x = 3"	A	A(+3), C(+2)	+1	A的所有步: 2→3	+5
第7次		C		+1	C的所有步: 1→2	
第8次		B		-1	B的所有步: -1→-2	

- 最终 Q 值分布

方法 1：Code–augmented CoT 的“逐步验证”轨迹

技术点2： 准确且稳定的 Q–value 注释机制

- **阶段二：PPM–Augmented Annotation（从第三轮起）**
- 第二阶段，不再依赖多次 backpropagation，而是直接一次判断当前 step 对最终质量的“正面贡献”

$$q(s_i)^0 = PPM(x \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_i)$$

  - x 为任务输入，$s_1 \ldots s_i$ 为当前推理路径，PPM 对其拼接做评估，输出为 q 值初值

方法 1：Code-augmented CoT 的“逐步验证”轨迹

技术点3： Step-by-Step Trajectory

- **使用MCTS构建推理轨迹（Step-wise）：**
  - 当前状态为：$t = x \oplus s_1 \oplus s_2 \oplus \ldots \oplus s_{i-1}$
    - 根节点 = 问题 x
    - 子节点 = 每一步的中间状态（step s）
  - 策略模型 M 生成多个候选的 Step − i：$s_{i,0}, s_{i,1}, s_{i,2}, \ldots, s_{i,n-1}$
  - 每个新 step 会被与前面所有步骤进行组合，形成新状态：$s_1 \oplus s_2 \oplus \ldots \oplus s_{i-1} \oplus s_{i,j}$
- **执行校验 & Q 值评估（代码增强）：**
  - 每条生成路径都会被转换为附带 Python 代码的版本（Code-augmented CoT）；
  - 仅当 Python 代码能够正常运行通过时，才视为“有效路径”；
  - 然后被PPM打分生成 Q 值 q(s)，被用于下一步 UCT 压缩挑选。

方法2：PPM（过程偏好模型）

与其给每个 step 一个具体分值，不如构造一对“谁更好（哪个 step 更优）”的比较任务（preference pairwise training）

构造偏好样本对：

- 每个 step i，选择：
  - 两个 Q 值最高的轨迹（作为 positive）
  - 两个 Q 值最低的轨迹（作为 negative）
- 要求：
  - 正向轨迹能得出正确结果
  - 负向轨迹必须得出错误答案

偏好训练损失函数：

$$\mathcal{L}_{\text{ppm}}(\theta) = -\frac{1}{2 \times |E|} \mathbb{E}_{(x, y_i^{\text{pos}}, y_i^{\text{neg}}) \in \mathcal{D}} \left[\log(\sigma(r_\theta(x, y_i^{\text{pos}}) - r_\theta(x, y_i^{\text{neg}}))) \right]$$

损失的目标：希望模型学出一个打分函数 $r_\theta(x, y)$ ，使得每组样本中：

$$r(x, y_i^{\text{pos}}) > r(x, y_i^{\text{neg}})$$

- 也就是说：让模型“更喜欢”正路径，赋予更高评分

方法3：四轮自我进化(Self-Evolved Deep Thinking)

第一轮： Bootstrapping an SLM policy (SLM-r1)

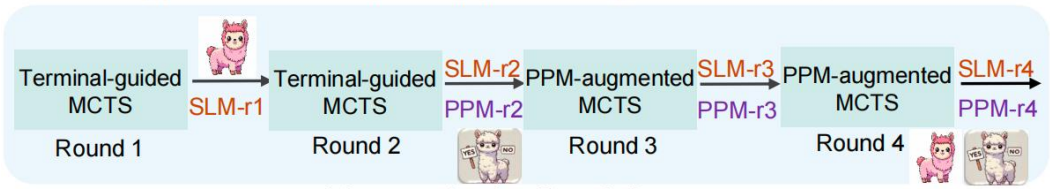
- 使用 DeepSeek-Coder-V2-Instruct 模型
- 只进行 8 次 MCTS rollout
- 得到最好的 top-2 轨迹作为 SFT 数据

第二轮： 训练第一个可靠的 PPM 模型 (PPM-r2)

- 使用 Round1 得到的 SLM-r1 用于 roll 出高质量轨迹
- 每题进行 16 次 MCTS rollout
- 使用这些轨迹训练一个更稳健的 PPM-r2，提升 Q-value 打分精度
- 同时得到 SLM-r2

第三轮： PPM-Augmented MCTS 输出来提升轨迹质量

第四轮： 针对难题加强采样，重点解决 Olympiad 级题目



(c) 4 rounds of self-evolution

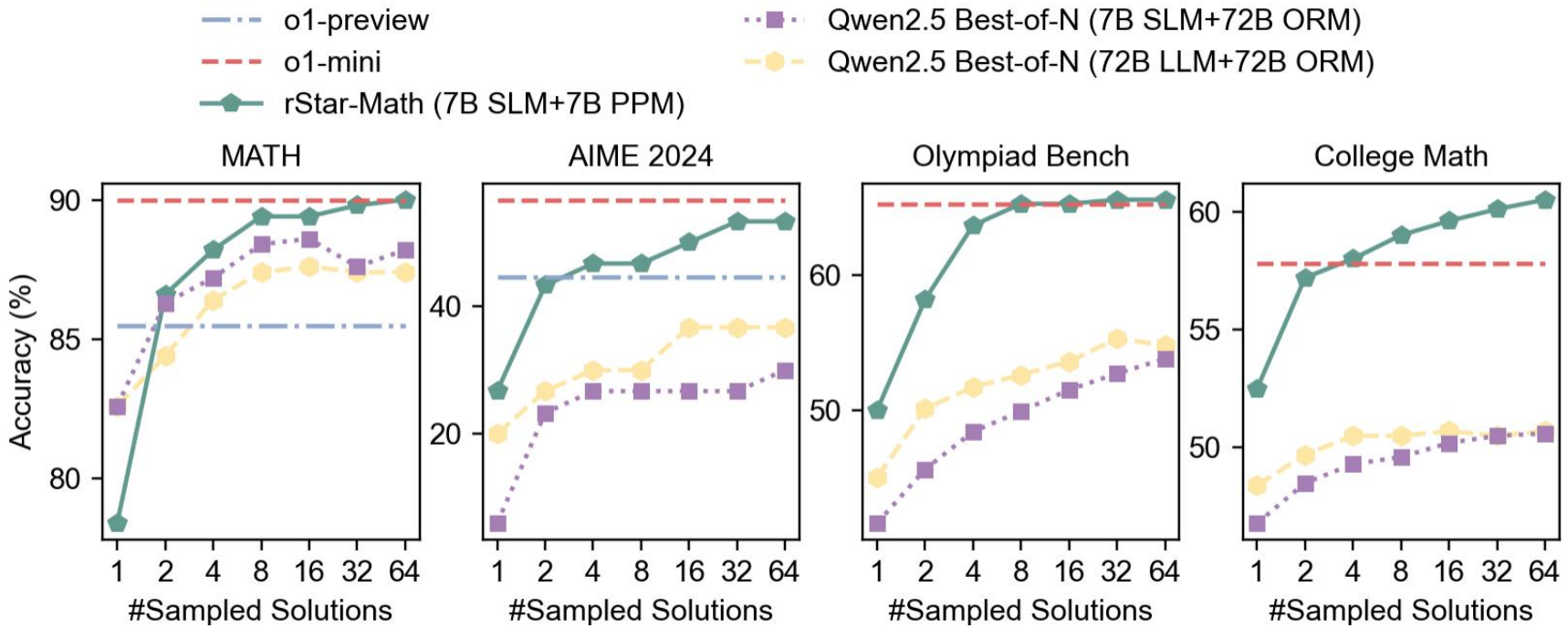
Table 2: Percentage of the 747k math problems correctly solved in each round

#	models in MCTS	GSM-level	MATH-level	Olympiad-level	All
Round 1	DeepSeek-Coder-V2-Instruct	96.61%	67.36%	20.99%	60.17%
Round 2	policy SLM-r1	97.88%	67.40%	56.04%	66.60%
Round 3	policy SLM-r2, PPM-r2	98.15%	88.69%	62.16%	77.86%
Round 4	policy SLM-r3, PPM-r3	98.15%	94.53%	80.58%	90.25%

实验结果

实验设置简述：

- Base Model：Qwen2/2.5-Math 1.5B/7B、Phi-3 mini（3.8B）等；PPM 使用 7B
- 评测：MATH、AIME 2024、Olympiad Bench、College Math、GSM8K、Gaokao 等
- 测试时：默认 8—16 条 rollout，扩展到 64 进一步提升



实验结果

- 在 MATH 上，Qwen2.5-Math-7B 提升到 90.0%（64 轨迹），对标/超越 o1-preview
- AIME 2024：平均 53.3%（8/15），达全美高中前 20%水平
- 多个基座（1.5B/3.8B/7B）一致提升，优于 Best-of-N + 超大 RM

Task (pass@1 Acc)	rStar-Math (Qwen-7B)	rStar-Math (Qwen-1.5B)	rStar-Math (Phi3-mini)	OpenAI o1-preview	OpenAI o1-mini	QWQ 32B-preview	GPT-4o	DeepSeek-V3
MATH	90.0	88.6	86.4	85.5	90.0	90.6	76.6	90.2
AIME 2024	53.3	46.7	43.3	44.6	56.7	50.0	9.3	39.2
Olympiad Bench	65.6	64.6	60.3	-	65.3	61.2	43.3	55.4
College Math	60.5	59.3	59.1	-	57.8	55.8	48.5	58.9
Omni-Math	50.5	48.5	46.0	52.5	60.5	49.6	30.5	35.9

总结及不足

总结:

本论文提出了面向小模型的自我进化思路：以往大多使用 GPT-4 之类的大模型提供教师示例，本研究则证明小模型也可「自给自足」，不依赖更强模型进行蒸馏。并且，实现了基于 MCTS 的步骤级别验证，针对数学推理的特点，放弃了精确打分，改为偏好对比，使得模型无需人类细粒度标注就能获得细步监督。

不足:

- 对极高难度或含图像信息的题目仍存在局限
- 大量算力和推理时间消耗
- 依赖能执行的 Python 代码

Thanks

2025年9月20日

王兆年