Huxley-Gödel Machine

Human-Level Coding Agent Development by an Approximation of the Optimal Self-Improving Machine

KAUST, 2025

The Challenge

🎯

Central Question

Which self-modifications should we accept?

❌ Prior Approach

DGM & SICA use benchmark scores

Assume: High score → Better lineage

⚠️ The Problem

High scores ≠ Good descendants

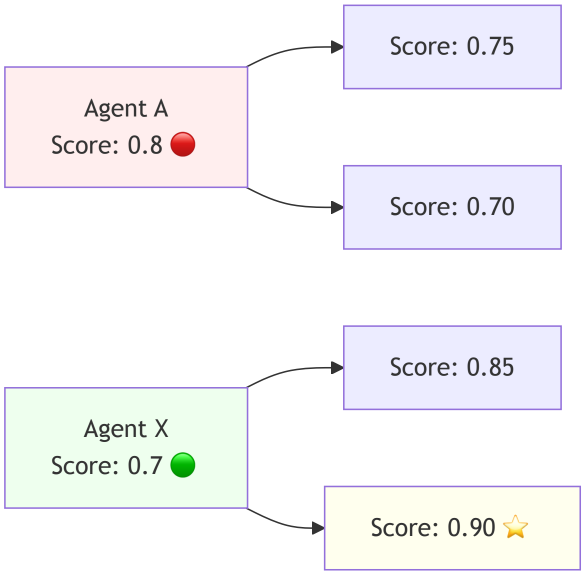
Metaproductivity-Performance Mismatch

Performance

Immediate scores

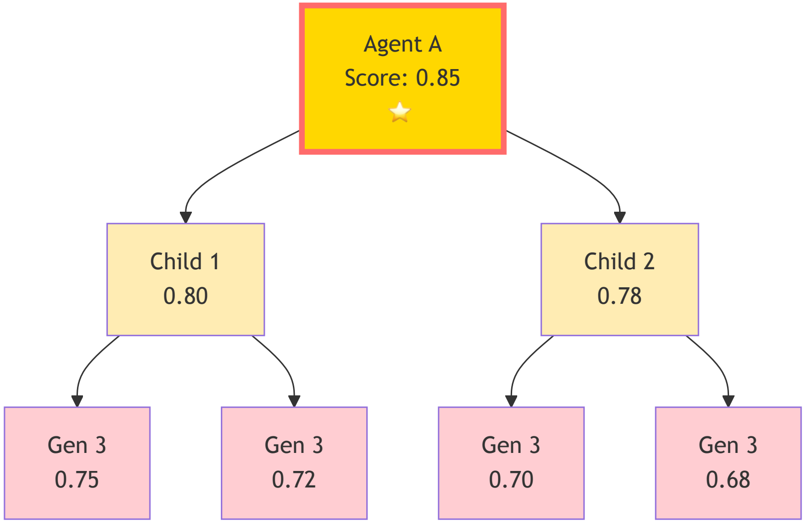
Metaproductivity

Long-term potential



Visualizing the Mismatch

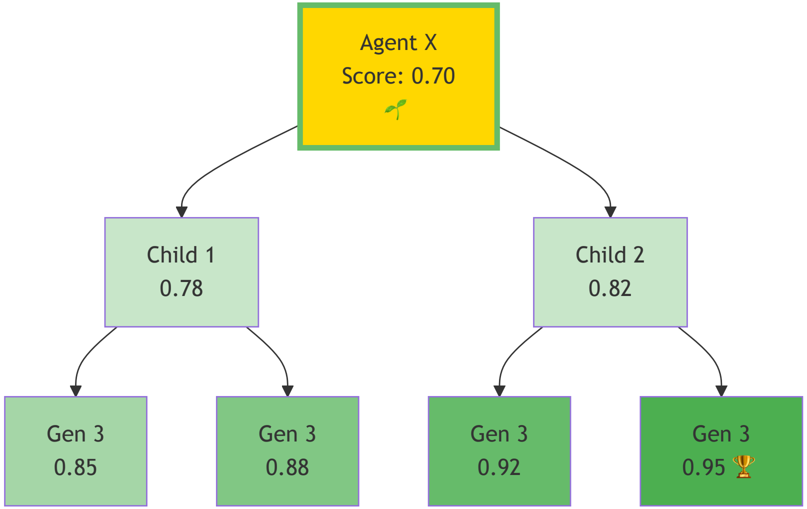
❌ High Score, Poor Lineage



Trend: ⬇️ Declining

CMP ≈ 0.70 (max descendant)

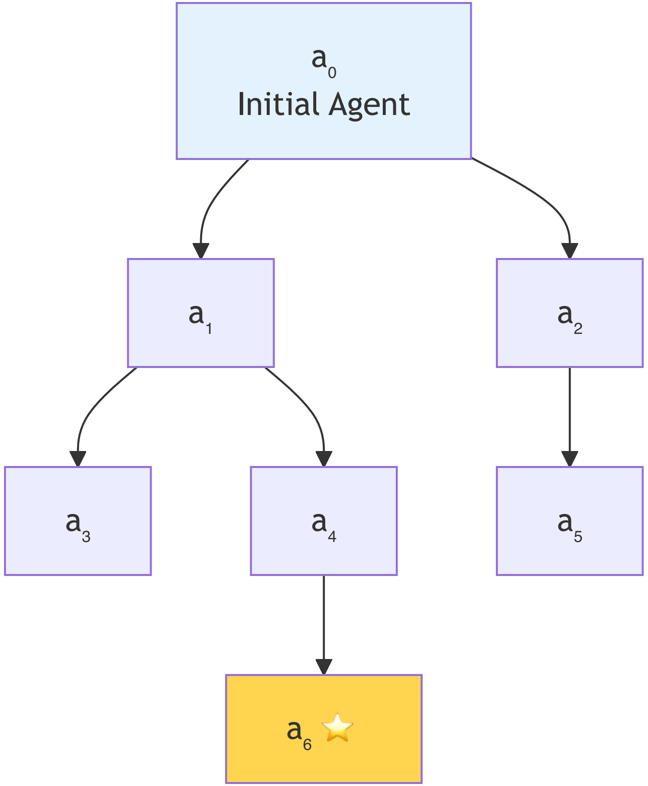
✅ Lower Score, Great Lineage



Trend: ⬆️ Improving

CMP ≈ 0.95 (max descendant)

Self-Improvement as Tree-Search



At each step:

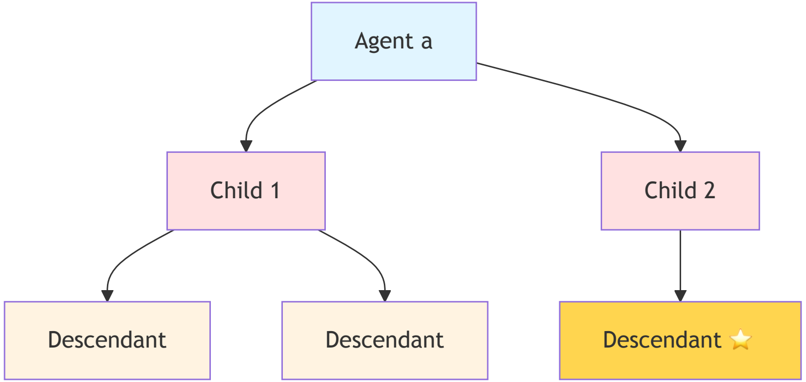
- 🔧 **Modify** agent → create child
- 📊 **Evaluate** agent on task
- 🎯 **Goal:** Find best final agent

From Gödel Machine to HGM

🧮		⚡
Gödel Machine	→	**HGM**
Formal proofs		Estimate CMP
Theoretically optimal		Practical approximation

Clade-Metaproductivity (CMP)

The Idea: Focus on **Lineages**



🌿 **Clade** = Agent + all descendants

Why CMP?

✅ Modest ancestor can have great descendants

✅ More statistically robust

✅ Inspired by biology (Huxley)

✅ Captures long-term potential

Theorem 1: CMP Oracle = Gödel Machine

CMP oracle is sufficient to implement the Gödel Machine

🎯 CMP ≡ Q-value

✓ Provably optimal

⚡ No proofs needed

💡 **Insight:** Estimate CMP → Approximate Gödel Machine

Deep Dive: What is a Gödel Machine?

Self-referential universal problem solver that makes provably optimal self-improvements

🔍 Core Mechanism

Runs a **proof searcher** looking for formal proofs that a code modification will increase expected utility

✅ When proof found

The modification is **executed** and permanently alters the machine

🎯 Guarantee

Provably optimal with respect to given objective

⚠️ Challenge

Finding proofs is **computationally expensive** and impractical

📚 *Schmidhuber (2003)*

Assumption 1: Our Specialized Setting

How does our **coding agent development** setting differ from general Gödel Machine?

1️⃣ **Final-only objective**

Only care about the **final agent**, not intermediate rewards

3️⃣ **No proof cost**

Proofs don't consume budget (unlike original GM)

2️⃣ **Repeatable trials**

Can **reset environment** for each test; evaluations are independent

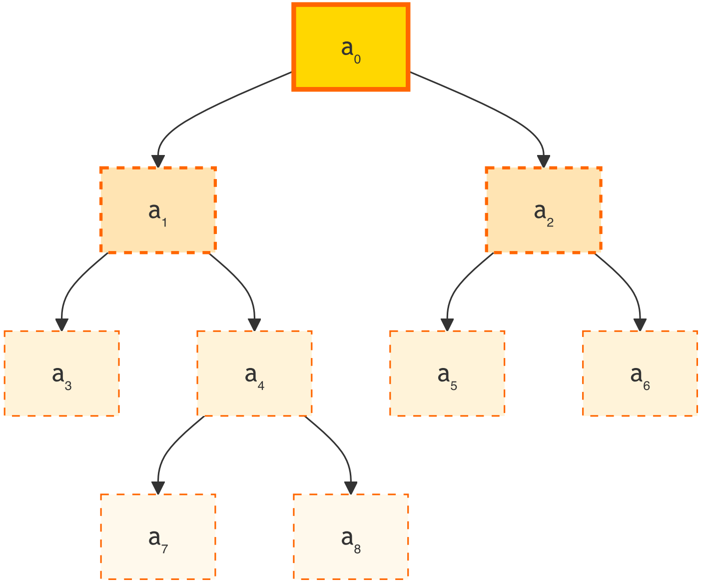
4️⃣ **Uniform costs**

Each self-modification costs **exactly 1 budget unit**

💡 These simplifications make optimal policy **tractable** via CMP!

Global Metaproductivity (GMP)

How does evolving agent *a* affect the **entire tree's** final performance?



⚠️ **Global Impact**

Expanding a_0 affects ALL future nodes

$$\mathrm{GMP}_\pi(\mathcal{T}, a) = \mathbb{E}\left[U\left(\max_{a' \in \mathcal{T}_B} a'\right)\right]$$

Expected utility of best agent in ENTIRE final tree

🌳 **Full tree**

🎲 **Rollout**

🏆 **Best**

⚡ **Key insight:**

GMP = Q-value in RL

(state = tree, action = expand agent)

From GMP to CMP: Why Clades?

❌ Problem with GMP

- Too global - considers entire tree
- Self-modifications can affect ancestors
- Hard to conceptualize
- Difficult to estimate

✅ Solution: CMP

- Localized to **subtree** (clade)
- Only descendants matter
- Conceptually clear
- Practically estimable

🧮 Gödel Machine Focus

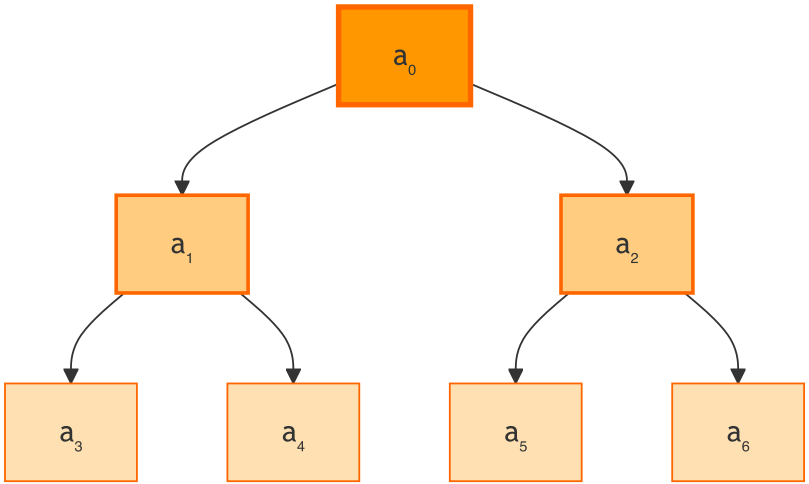
Decides whether to **accept or reject** based on provable potential of **subsequent** self-improvements

🌿 Biological Inspiration

Clade (Huxley, 1957): lineages of common ancestry

GMP vs CMP: Visual Comparison

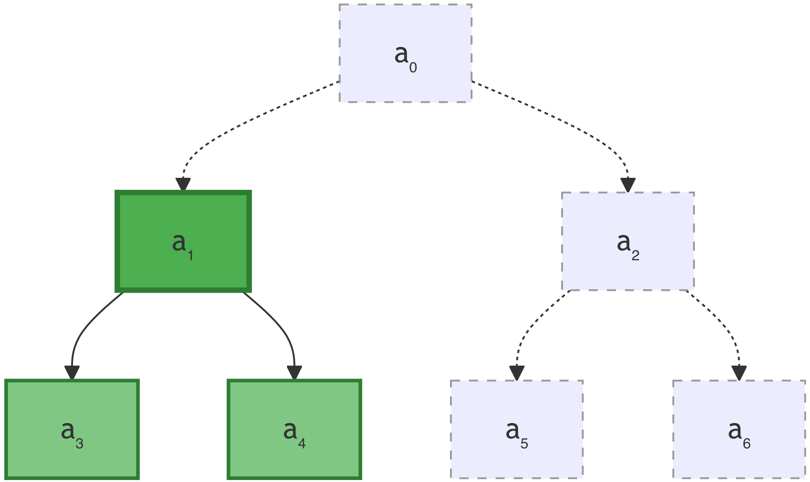
GMP: Global Scope



Considers ALL nodes

Max over $\mathcal{T}_B$ (entire tree)

CMP: Clade Scope



Only descendants of a_1

Max over $C(\mathcal{T}_B, a_1)$ (clade of a_1)

The Mathematics of CMP

Clade-Metaproductivity: Expected utility of best agent in **clade**

$$\mathrm{CMP}_\pi(\mathcal{T}, a) = \mathbb{E}_{\mathcal{T}_B \sim p_\pi(\cdot|\mathcal{T},a)}\left[U\left(\underset{a' \in C(\mathcal{T}_B,a)}{\arg\max}\,\mathrm{Score}_\pi(a')\right)\right]$$

$$= \mathbb{E}_{\mathcal{T}_B \sim p_\pi(\cdot|\mathcal{T},a)}\left[\max_{a' \in C(\mathcal{T}_B,a)} U(a')\right] \quad (\text{if Score} = \text{U})$$

where $C(\mathcal{T}_B, a)$ is the clade (subtree rooted at a) in final tree $\mathcal{T}_B$

📊 Estimator

🔢 Where

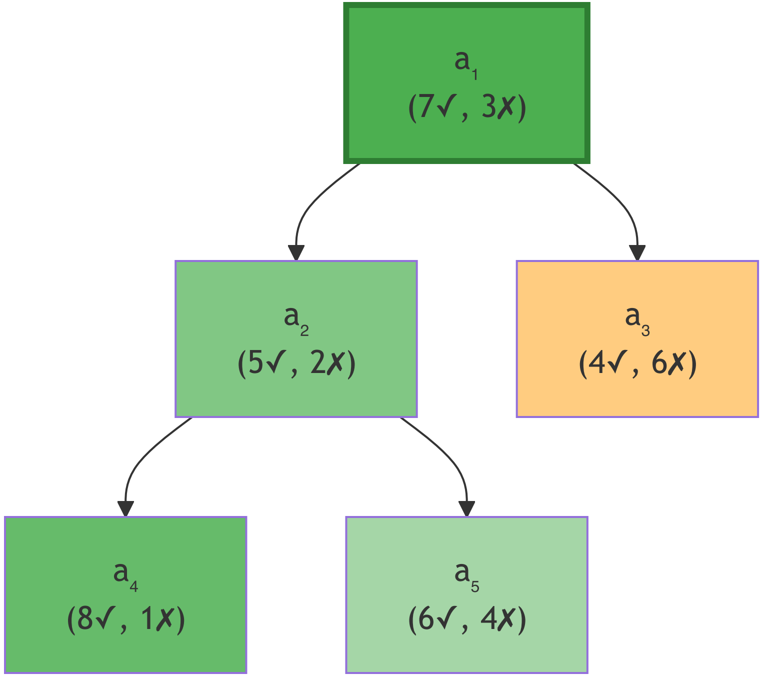
$$n^C_{\text{success}}(a) = \sum_{a' \in C(a)} n_{\text{success}}(a')$$

⚖️ Weighted

More evaluated agents get higher weight

CMP Estimator: Step-by-Step

How to estimate CMP: Aggregate test results across the **clade**



Tree showing test results (successes ✓, failures ✗)

Step 1: Collect in Clade

Gather all results from a_1 and descendants:

a_1: 7✓, 3✗ | a_2: 5✓, 2✗ | a_3: 4✓, 6✗ | a_4: 8✓, 1✗ | a_5: 6✓, 4✗

Step 2: Sum Totals

$$n^C_{\text{success}}(a_1) = 7 + 5 + 4 + 8 + 6 = 30$$
$$n^C_{\text{failure}}(a_1) = 3 + 2 + 6 + 1 + 4 = 16$$

Step 3: Compute CMP

$$\widehat{\text{CMP}}(a_1) = \frac{30}{30 + 16} = \frac{30}{46} \approx 0.652$$

Theorem 1: The Main Result

Theorem 1

Under **Assumption 1**, access to the **CMP oracle** is *sufficient* to implement the **Gödel Machine**.

🔑 Key Equality

$$\mathrm{CMP}_\pi(\mathcal{T}, a) = Q_\pi(\mathcal{T}, a)$$

CMP is exactly the Q-value in the Gödel POMDP!

🎯 POMDP Setup

- **State:** Tree + parent + child
- **Action:** Accept or reject
- **Observe:** Only parent, child, budget

✓ Optimality

Selecting max CMP = selecting max Q-value = **Bellman optimal**

💡 **Implication:** No need for expensive proofs! Just estimate CMP well.

Proof Sketch: Why CMP = Q-value

1️⃣ Gödel Machine Policy

At each step: observe $(a_{\text{parent}}, a_{\text{child}}, b)$ and choose to accept or reject child

2️⃣ Final Selection

Score_π is indicator function: picks either final parent *or* final child

3️⃣ Clade Structure

Only descendants reachable; ancestors above clade are **not accessible**

4️⃣ Expected Return

$$\begin{aligned}
&\text{CMP}_\pi((\mathcal{T}, a_p, a_c, b), a) \\
&= \mathbb{E}[U(\arg\max_{a' \in C} \text{Score}(a'))] \\
&= \mathbb{E}[U(\arg\max_{a' \in \{a_p, a_c\}} \text{Score}(a'))] \\
&= Q_\pi((\mathcal{T}, a_p, a_c, b), a)
\end{aligned}$$

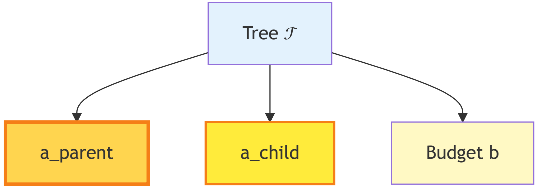
✅ Conclusion

CMP oracle knows the true Q-values → can make optimal accept/reject decisions

POMDP: The Gödel Machine's Decision Process

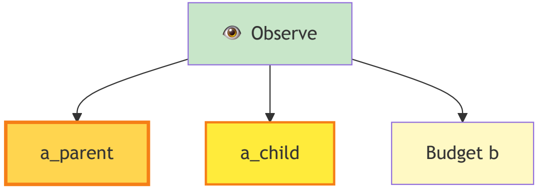
Gödel Machine operates as a POMDP: Partial observability, Accept/Reject decisions

State



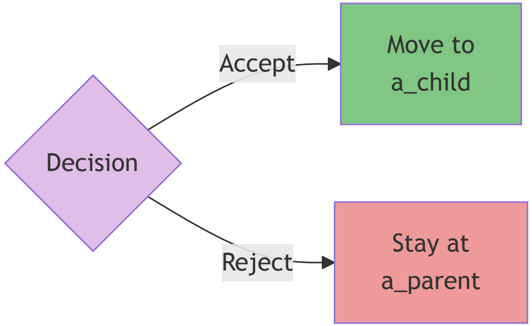
Full state includes entire tree + special nodes

Observation



Only see parent, child, budget (not full tree!)

Action



Binary choice: Accept or Reject

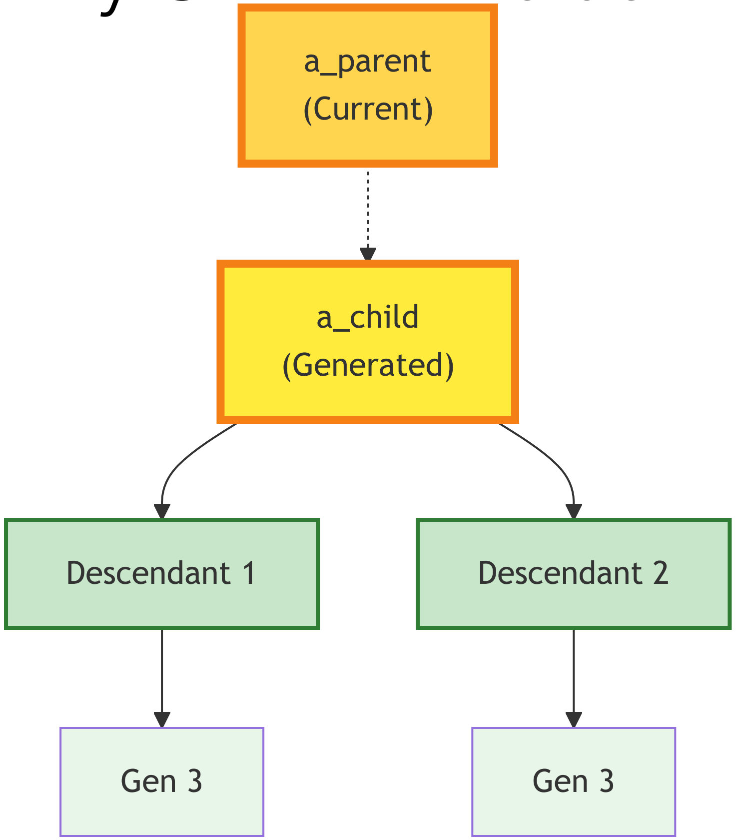
🔄 Transition

1. Generate child via self-modification 2. New parent = accepted agent 3. Budget decreases by 1

🎯 Reward

Final reward = $U(\text{Score}_\pi)$ picks best from final (parent, child) pair

Why CMP = Q-value: Visual Proof



Clade of a_child
Only these nodes are reachable!

1️⃣ Final Selection Constraint

Score_π is an **indicator function**: Picks either a_{parent} OR a_{child}. All other agents get score 0

2️⃣ Clade = Reachable Set

If we accept a_{child}, only its **descendants** are reachable. $C(\mathcal{T}_B, a_{\text{child}})$ = all nodes below a_{child}

3️⃣ The Equality

$$\text{CMP} = \mathbb{E}[\max_{a' \in C} U(a')] = \mathbb{E}[\max_{a' \in \{a_p, a_c\}} U(a')] = Q_\pi$$

Thompson Sampling for Exploration-Exploitation

How to select which agent to expand when we only have **estimates** of CMP?

🎲 Thompson Sampling

1. Model CMP as Beta distribution:

$$\mathrm{Beta}(\tau(1+n_s^C), \tau(1+n_f^C))$$

2. Sample from each posterior

3. Select agent with highest sample

⚖️ Exploration Scheduler

Temperature parameter $\tau(t)$ increases over time:

- **Early:** Low τ → more exploration
- **Late:** High τ → more exploitation

Automatically balances discovery vs refinement

🎯 Unlike greedy baselines, TS **probabilistically** explores promising clades

Temperature Schedule: τ(t) Over Time

Temperature $\tau(t) = \frac{B}{b}$ increases over time: more **exploitation** as budget runs out

t=0 τ≈1 🔍 → t=B/2 τ≈2 ⚖️ → t=B τ→∞ 🎯

🔍 Early (low τ)

Wide Beta distributions → more random sampling → **exploration**

⚖️ Middle (medium τ)

Moderate distributions → balanced → **mixed strategy**

🎯 Late (high τ)

Narrow distributions → pick best → **exploitation**

Beta Distributions Evolution

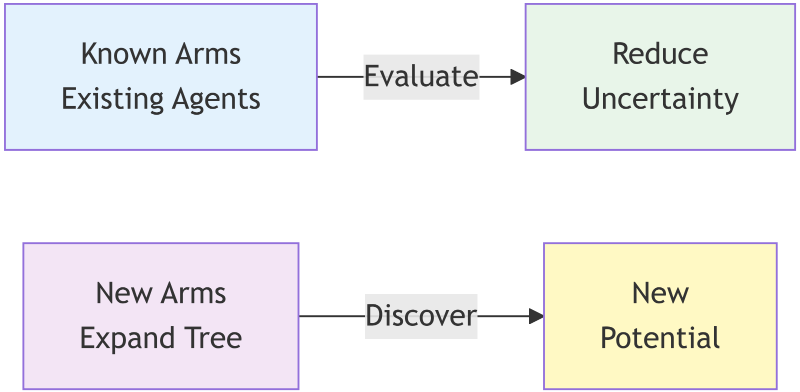
Low τ = 1	**Med τ = 5**	**High τ = 20**
$\mathrm{Beta}(1+7, 1+3)$	$\mathrm{Beta}(5+35, 5+15)$	$\mathrm{Beta}(20+140, 20+60)$
Wide curve 📊	Medium 📈	Peaked 📉

📊 Effect on Selection

- **Low τ:** Sample values spread out → try different agents
- **High τ:** Sample values concentrated → pick highest CMP

Connection to Infinite-Armed Bandits

When to **expand** (create new agent) vs **evaluate** (test existing)?



📊 UCB-Air Strategy

Expand when:

$$N_t^{\alpha} \geq |\mathcal{T}_t|$$

Number of evaluations vs number of agents

⚡ Decoupled Policy

- DGM/SICA: Always expand then evaluate
- **HGM:** Adaptive choice each step

HGM: How to Estimate CMP?

Aggregate success across **entire clade** (lineage)

$$\widehat{\text{CMP}}(a) = \frac{\textbf{successes in clade}}{\textbf{total tests in clade}}$$

Clade-level
aggregation

Weighted by
evaluations

Thompson
Sampling

HGM Algorithm: Complete Walkthrough

Step-by-step execution of one HGM iteration

Step 1: Compute CMP Estimates

For each agent a in tree:

$$\widehat{\mathrm{CMP}}(a) = \frac{n^C_{\text{success}}(a)}{n^C_{\text{success}}(a) + n^C_{\text{failure}}(a)}$$

Step 2: Thompson Sample

For each a, sample:

$$S_C(a) \sim \mathrm{Beta}(\tau(1 + n^C_s), \tau(1 + n^C_f))$$

Select $a^* = \arg\max_a S_C(a)$

Step 3: Decide Action

If $N^\alpha_t \geq |\mathcal{T}_t|$: **Expand** (create child)

Step 4a: If Expanding

1. Agent a^* self-modifies → creates child c
2. Add c to tree $\mathcal{T}$
3. Initialize counters for c

Step 4b: If Evaluating

1. Sample agent to test (by individual stats)
2. Run on one task → get result
3. Update n_{success} or n_{failure}
4. Bubble up to ancestors (update n^C)

Step 5: Repeat

Continue until budget exhausted. Return best-belief agent:

HGM: Three Adaptive Policies

🔧

Expansion

Which agent to modify?

Use **clade CMP**

📊

Evaluation

Which agent to test?

Use **individual stats**

⚖️

Selection

Expand or evaluate?

Adaptive scheduling

🚀 **Key Innovation:** Decoupled expansion from evaluation!

HGM in Practice

⚡

Asynchronous

Run on all CPUs

Early stopping

🏆

Best-Belief

Select final agent

by posterior

Experimental Setup

Benchmarks

SWE-bench Verified (500)
SWE-bench Lite (300)
Polyglot

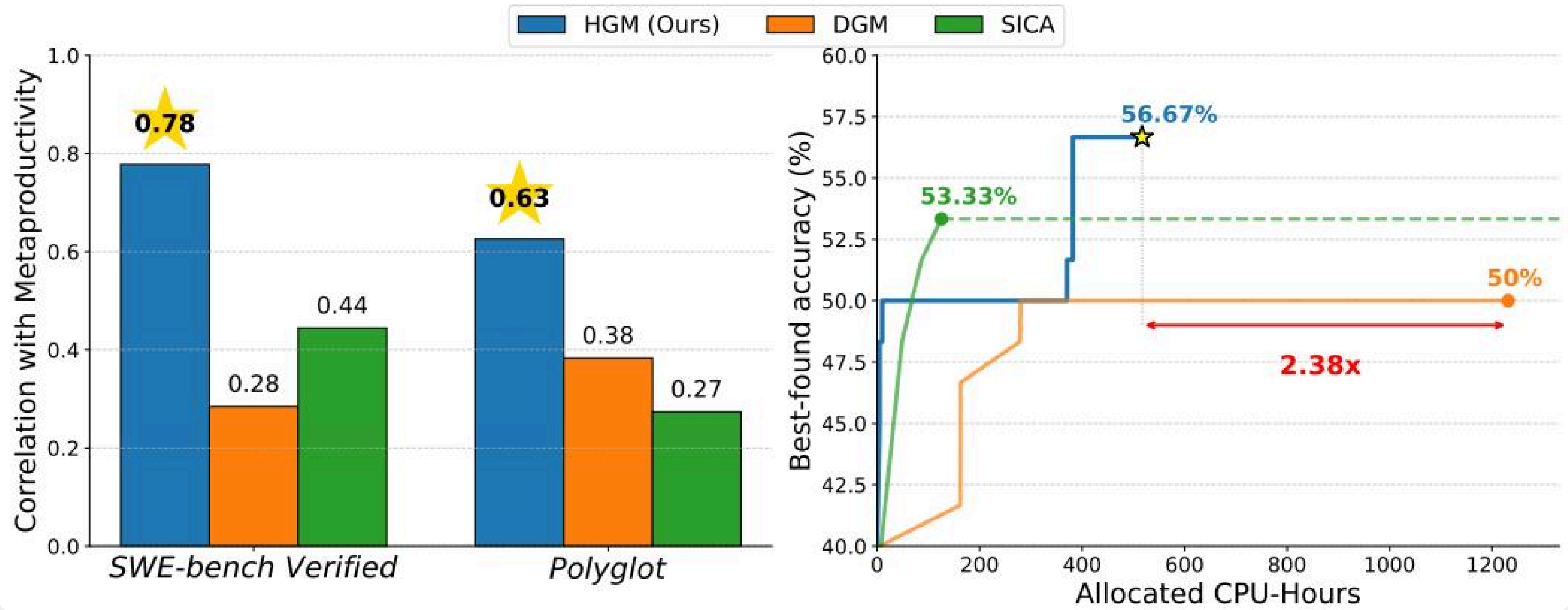
Baselines

DGM
SICA
SWE-agent

Models

GPT-5 / GPT-5-mini
Qwen3-Coder

Main Results from Paper



Result 1: CMP Correlation

Method	Weighted Correlation	Unweighted Correlation
SICA	0.444 / 0.274	0.444 / 0.274
DGM	0.285 / 0.383	0.406 / 0.357
HGM	**0.778 / 0.626**	**0.512 / 0.873**

SWE-Verified-60 / Polyglot

📈 HGM: **2-3× better correlation** with true metaproductivity

Result 2: Performance & Efficiency

🎯 Accuracy

SWE-Verified-60

56.7%

+16.7% improvement

Polyglot

30.5%

+10.2% improvement

⚡ Speed

vs DGM

2.38×

faster

Polyglot speedup

6.86×

faster

Result 3: vs. Human Designers

HGM on SWE-bench Verified

53.2%
Initial agent

61.4%
HGM discovered ⭐

🏆 **Top-10** on leaderboard (all models)
#1 GPT-5-mini based system

Result 4: Human-Level Performance! 🎉

HGM optimized with **GPT-5-mini** → Evaluated with **GPT-5**

Agent	SWE-Lite Standard (%)
SWE-agent (Best human design)	56.7
HGM + GPT-5	**57.0** 🏆

✅ Transfers across **model sizes**

✅ Optimized on **different dataset**

✅ Not overfitting, **genuine ability**

Emergent Behaviors

🔄

Self-Motivated Iteration

Agents perform **multiple self-modifications** per instruction

→ *Arbitrary levels of meta-improvement!*

🌀

Nested Diff Structures

Diff patches of diff patches
Multiple levels of changes

"Mind-bending to understand manually"

Key Contributions

Identified Metaproductivity-Performance Mismatch

Developed HGM Algorithm

Introduced Clade-Metaproductivity (CMP)

Validated 2-3× better CMP estimation

Proved Theorem 1 (CMP = Gödel Machine)

Achieved Human-level performance

HGM vs. Baselines

	SICA	DGM	**HGM**
Guidance	Performance	Performance	🌿 **CMP**
Expansion	Greedy	Probabilistic	**Thompson Sampling**
Decoupled?	❌	❌	✅
Theory	❌	❌	✅ **Gödel Machine**
Correlation	0.44	0.29	**0.78**
Speed	1×	1×	**2-7×**

Key Takeaways

❌

Performance ≠ Metaproductivity

✅

Lineages > Individuals

🧮

CMP Oracle = Gödel Machine

🏆

Human-Level Performance

💡 **Paradigm Shift:** Focus on capacity to **keep improving**