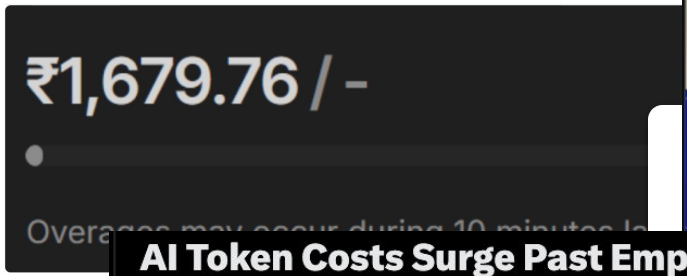
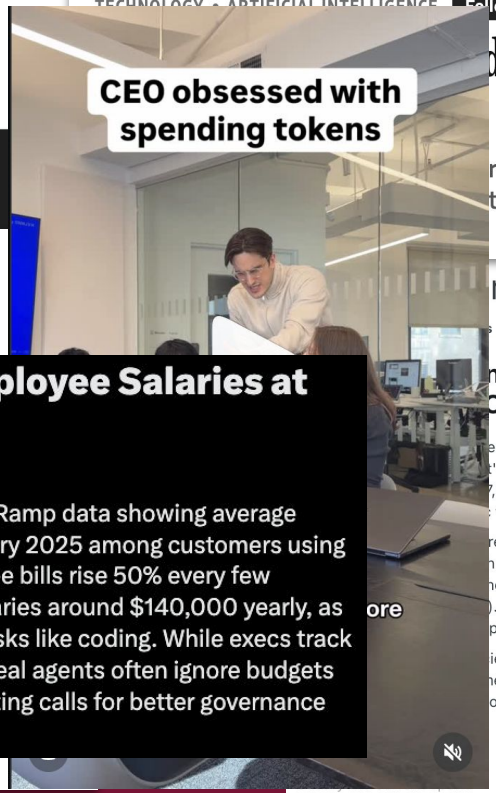
Harshith @HarshithLucky3 · Apr 21

₹1,679.76 = ~$18 for one **Deep** Research 💀

i never use it again 😭





AI Token Costs Surge Past Employee Salaries at Tech Firms

Last updated Apr 22

Tech companies face skyrocketing AI costs, with Ramp data showing average monthly token spend jumping 13-fold since January 2025 among customers using tools like Claude and GPT models. Heavy users see bills rise 50% every few months, sometimes matching junior engineer salaries around $140,000 yearly, as autonomous AI agents burn through tokens on tasks like coding. While execs track high usage as a productivity win, experiments reveal agents often ignore budgets and produce output needing human fixes, prompting calls for better governance like team alerts and invoice matching.

r/claude · 7d ag
Mammoth_Docto

The Problem

Discussion

Uber recently annou
"token-maxxing".

If someone told you
machine, and saying

Why do we treat AI usage differently?

THE WALL STREET JOURNAL.

SUBSCRIBE SIGN IN

TECHNOLOGY · ARTIFICIAL INTELLIGENCE Follow

l Out AI at Work—Now

rtificial intelligence are starting to track
t of measurement

reckless by

· 20d ago

nance just got real, our token bill hit six figures
O cares

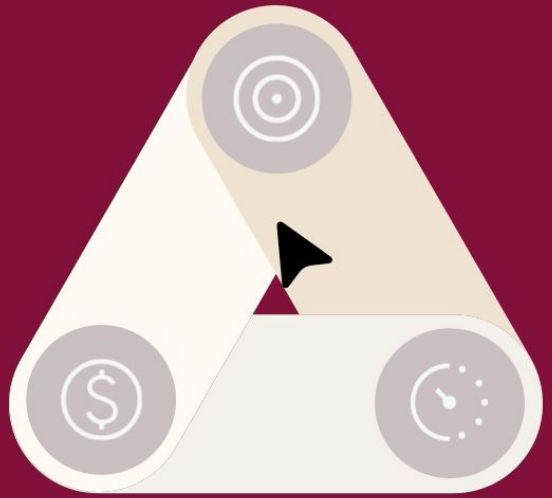
ech company with about 500 developers. Last year leadership said "give every
'll pay for itself in productivity." So we did and fast forward 8 months, our AI too
7,000. Projected annual cost was $340,000+. And that's before the engineering teams
workflows which will increase token consumption significantly.

reakdown of ROI. The conversation has shifted from "everyone needs AI tools" to
what we're paying." The awkward truth is we can't prove it. We can show adoption
e tools daily), satisfaction scores (developers like the tools), and proxy metrics (PR
). But connecting $340k in AI tooling costs to actual revenue impact or a specific
productivity gained? Nobody can do that cleanly.

iency. Our initial analysis suggests we're burning a massive amount of tokens in a
e codebase context gets sent with every inference request. There's no caching, no
o efficiency optimization. It's like if every Google search had to re-index the internet

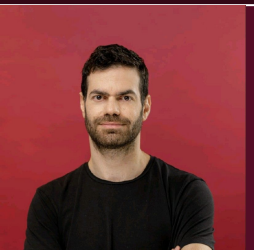
A vicious cycle...

Optimizing **accuracy**, **cost** and **latency**



Mind the Gap

The Four Gaps Between Demo Agents and Production Systems



Yuval Belfer

Sr. Developer Advocate
AI21 Labs

AI21

Agenda

Mind the Gap

The four gaps that separate demo agents from production systems

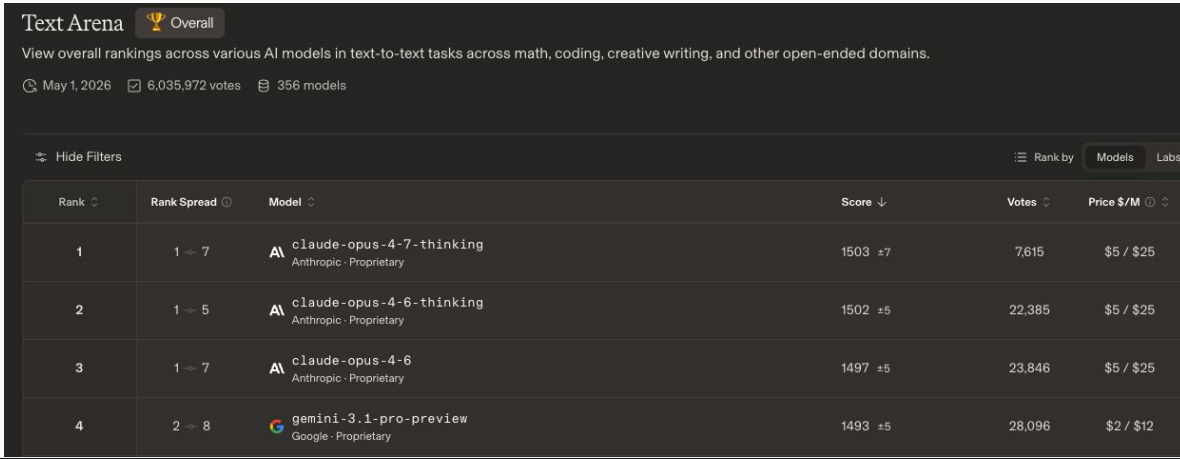
Close the Gap

Optimization methods and the MAESTRO approach

What leaderboards don't tell you

When a new model is released, benchmarks report one number: the average score across all test problems, and the average dollar cost.

But in production, you don't run averages. You run specific tasks, with specific budgets, under specific time constraints.



The Four Gaps

Validation

Validation

`Why your agent doesn't know when it got it right`

What it means:
Every time an AI model runs, it draws from a distribution — it doesn't always produce the same answer. Run it once, you get one sample. Run it ten times, and the right answer is probably in there somewhere.

The Validation Gap is the difference between:

- Success@1 — did it get the right answer on its one try?
- Success@N — did it get the right answer at least once across N tries?

For hard reasoning tasks, Success@10 can be 15–20 percentage points higher than Success@1. That's a lot of performance sitting on the table.

Validation

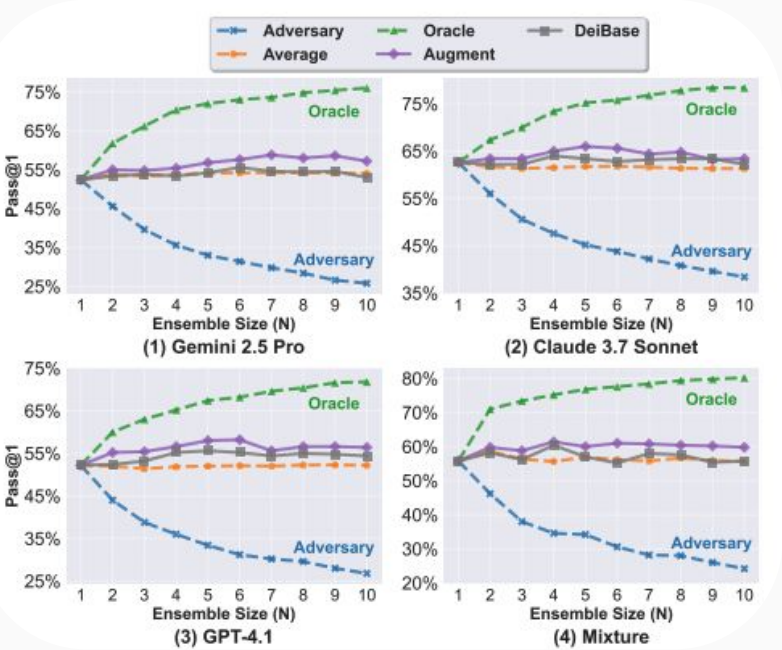
`Why your agent doesn't know when it got it right`

The catch:
Getting the right answer is only half the problem. The system also needs to know which answer is right — that's what a validator does.

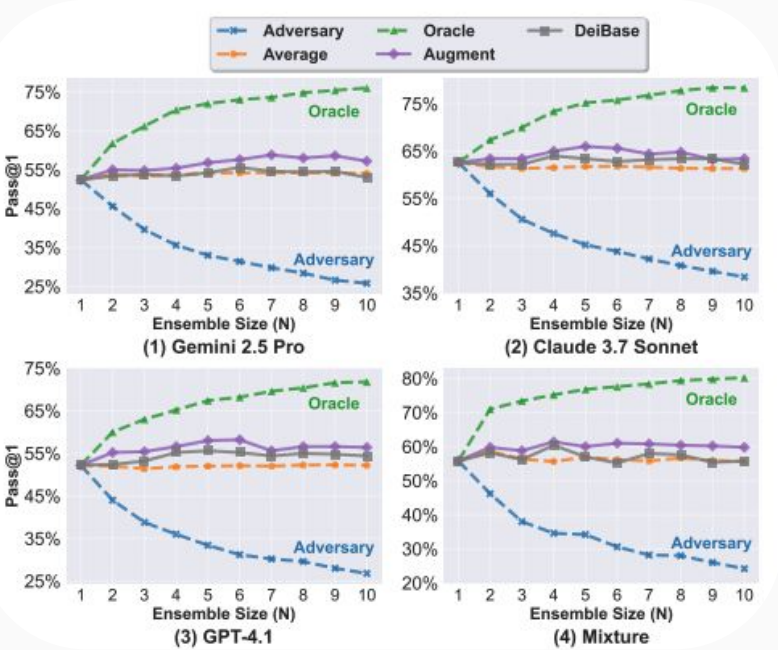
Imagine you're building a coding agent. It generates 5 candidate fixes for a bug. Which one do you ship?

- If you can run the test suite → the validator is the test suite. It's essentially perfect. You just pick whichever fix passes.
- If you're generating marketing copy → there's no test suite. Your "validator" might be an LLM judge scoring each version. It's imperfect, but it still beats picking blindly.

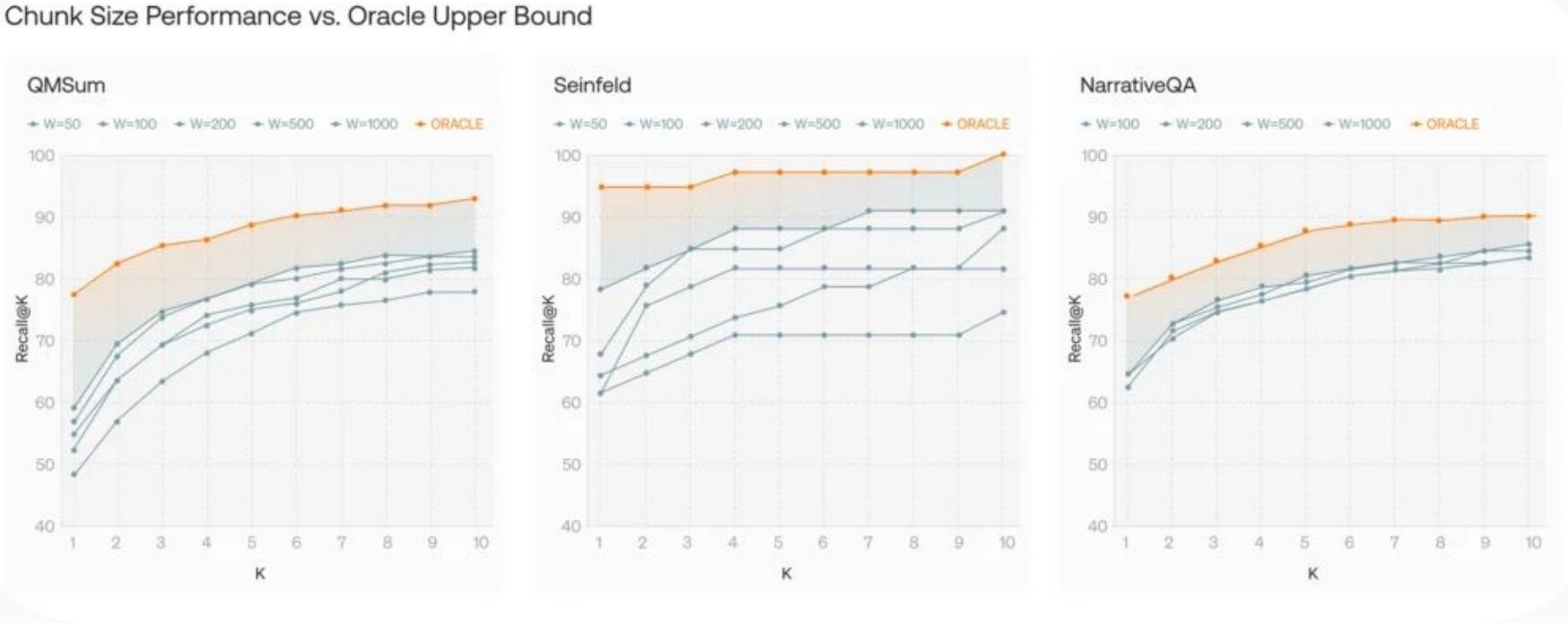
Validation – “Oracle”



Validation – “Oracle”



Validation – “Oracle”

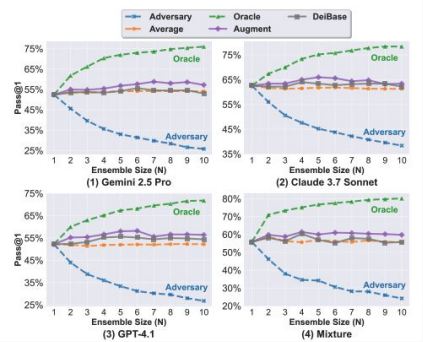


https://www.ai21.com/blog/query-dependent-chunking/

The Four Gaps

Validation

Contextualization



Contextualization

`The best model on average isn't the best model for *your* task`

What it means:
Cheap small models solve easy problems almost as well as expensive frontier models.
Expensive models only pull ahead on genuinely hard tasks.

Contextualization

`The best model on average isn't the best model for *your* task`

What it means:
Cheap small models solve easy problems almost as well as expensive frontier models.
Expensive models only pull ahead on genuinely hard tasks.

If you used the cheap model for the easy 60% of tasks, you'd get the same answer — at a fraction of the cost.

Contextualization

`The best model on average isn't the best model for *your* task`

The Contextualization Gap is the difference between what's best on average (one model for everything) and what's best per input (the right model for each specific task).

Concrete example:
Your support team uses an AI agent to handle customer queries.

- "What's your refund policy?" → a small, fast, cheap model handles this perfectly
- "Explain why my enterprise contract pricing changed" → frontier model with strong reasoning.

Routing every query through the frontier model means paying frontier prices for FAQ answers.

Contextualization

`The best model on average isn't the best model for *your* task`

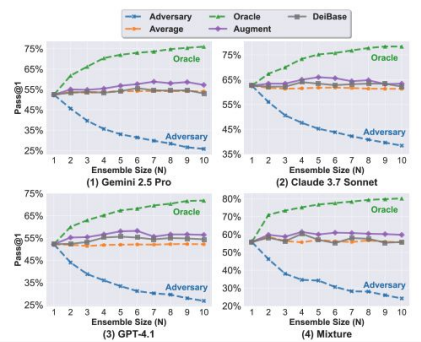
Solution: **portfolio**

Instead of picking one model, maintain a portfolio and route each input to the right one.

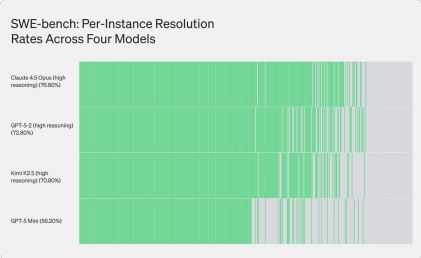
Think of it like staffing: you don't use your most senior engineer to book meeting rooms.

The Four Gaps

Validation



Contextualization



Latency

Latency

`Dollar cost and time cost are not the same problem`

What it means:
Most AI systems execute **sequentially**: do step 1, wait, do step 2, wait. Simple to build, but slow.

The Latency Gap is the difference between how long the system actually takes and how long it could take if it found the shortest successful path.

On hard reasoning tasks, the shortest successful trajectory is often 3x faster than the average one.

Latency

`Dollar cost and time cost are not the same problem`

Optimizing for latency and optimizing for cost require **opposite strategies**.

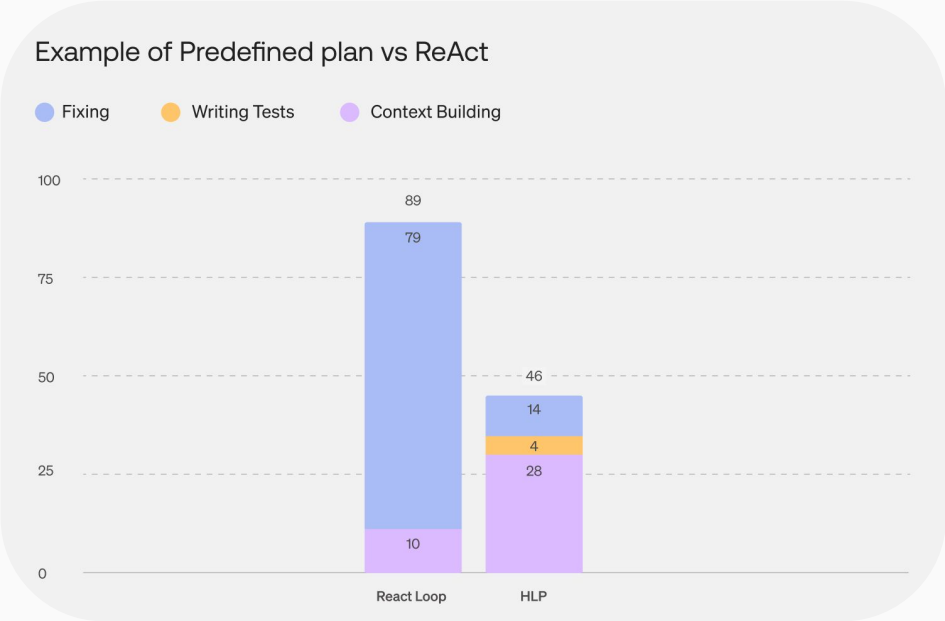
- Minimize cost → run things sequentially, validate, try the next thing only if needed
- Minimize latency → run multiple things in parallel, terminate everything the moment one succeeds

Neither is wrong — a fraud detection system needs speed, an overnight batch job needs efficiency. A production system should let you choose.

Latency

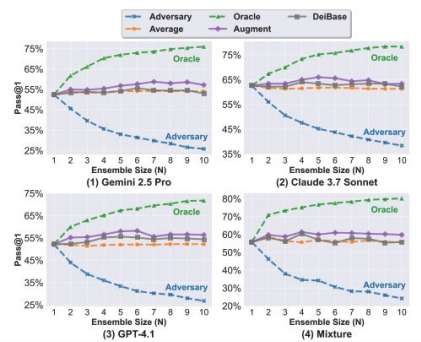
`Dollar cost and time cost are not the same problem`

Also, it’s a big picture thing:

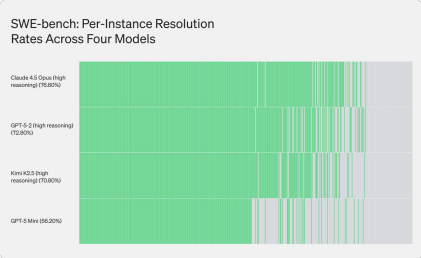


The Four Gaps

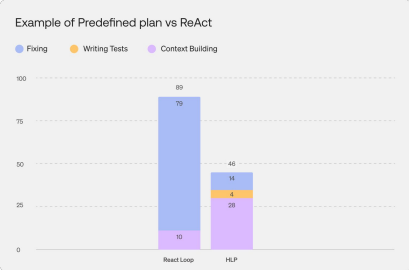
Validation



Contextualization



Latency



Decomposition

Decomposition

`What if you applied all three gaps at every single step?`

What it means:
The first three gaps were applied externally — around the agent. Decomposition asks:
What if you applied all of them inside the agent, at every decision point?

Most agents work like a straight line: step → step → step → done, committing to a path even if it's going badly.

A system that closes the Decomposition Gap works like a search tree — branching, evaluating, switching models mid-task, and abandoning bad paths early.

Decomposition

`What if you applied all three gaps at every single step?`

Concrete example:
A coding agent is asked to fix a bug with two independent sub-problems: one in the database layer, one in the API layer.

- Today's approach: tackle them sequentially. Total time: **8 minutes.**
- Decomposition approach: spawn two parallel branches, validate each independently, merge the results. Total time: **~3 minutes, lower cost.**

Decomposition

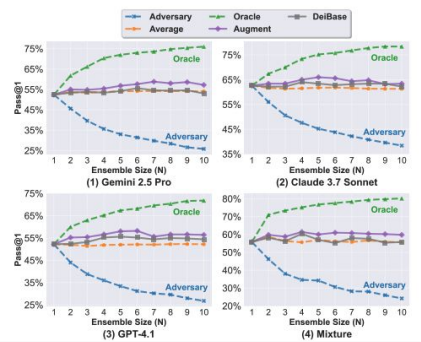
What if you applied all three gaps at every single step?

Closing gaps 1–3 at the system level gives you gains. Closing them at every step inside the task gives you those gains compounded across every node in the tree.

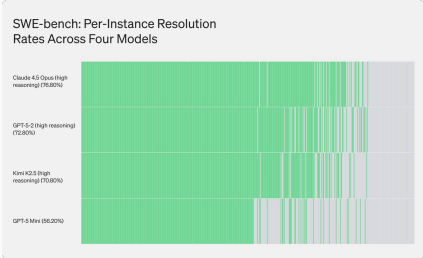
It's the hardest gap to close — you need both the execution engine to understand task structure, and models trained to think in search trees rather than linear chains.

The Four Gaps

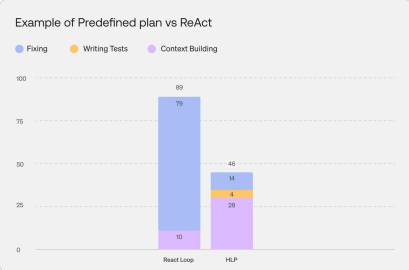
Validation



Contextualization

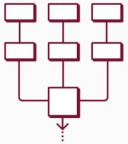
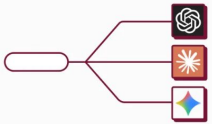
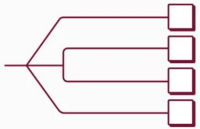
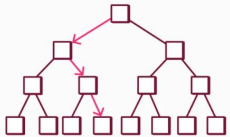


Latency



Decomposition

The Four Gaps Between Demo Agents and Production Systems

Gap	Demo Agent	VS.	Production System
Validation	 Single run		 Multi-run + validator ↑ Quality
Contextualization	 One model		 Portfolio routing ↓ Cost
Latency	 Sequential		 Parallel + early stop ↓ Time
Decomposition	 Linear		 Multi-run + validator ↑ Capability

Closing — what this buys you

Back to the vicious cycle

We started with the triangle: you can't optimize accuracy, cost, and latency all at once.
The four gaps explain why - and they also show the path out.

- The Validation Gap tells you there's more accuracy available without changing your model — you just need a validator.
- The Contextualization Gap tells you there's massive cost savings available without sacrificing accuracy — you just need smarter routing.
- The Latency Gap tells you speed is achievable — but it requires a different execution strategy, not a faster model.
- The Decomposition Gap tells you the gains compound when you apply this thinking throughout execution, not just at the edges.

The way to close the gaps

Optimization

Double click on agent optimization

Common methods today:

Agent configuration

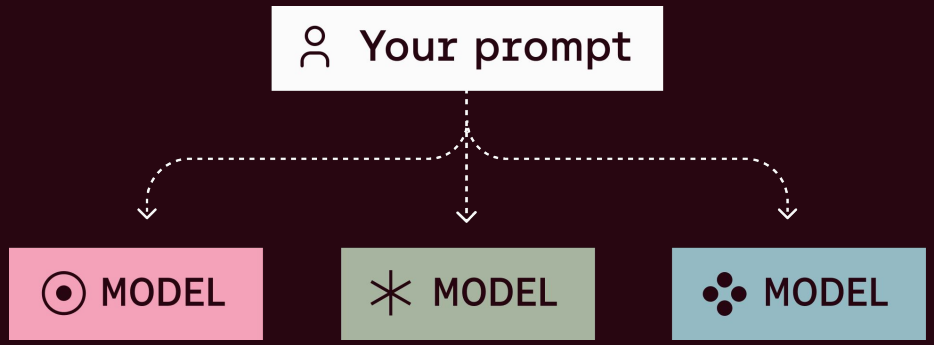
Models, prompts, tools, harnesses, orchestration layer

Scaling

Best-of-n sampling, heterogeneous model ensembles, execution strategies

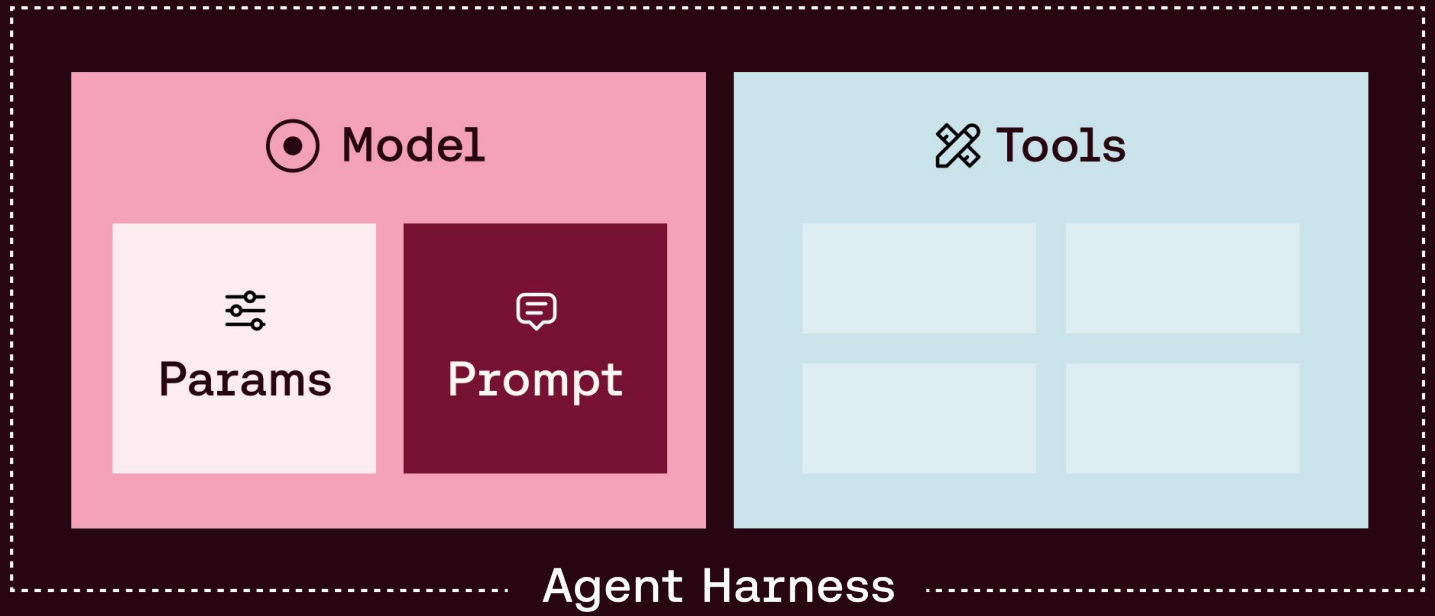
Agent optimization methods

Naive model testing



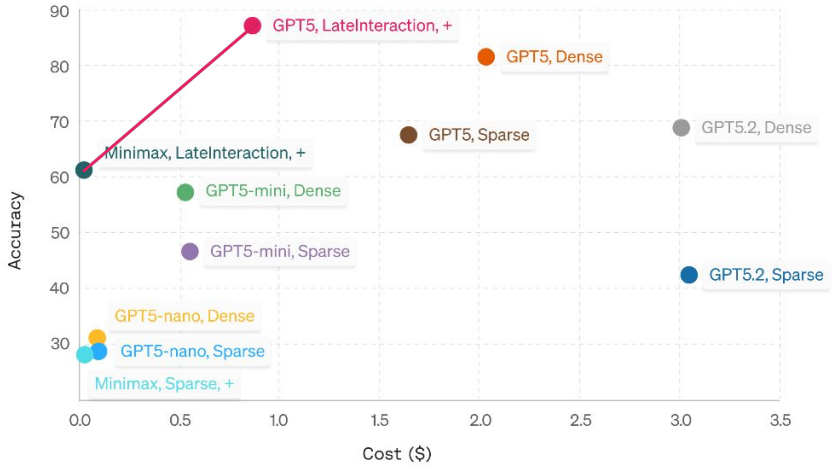
Agent optimization methods

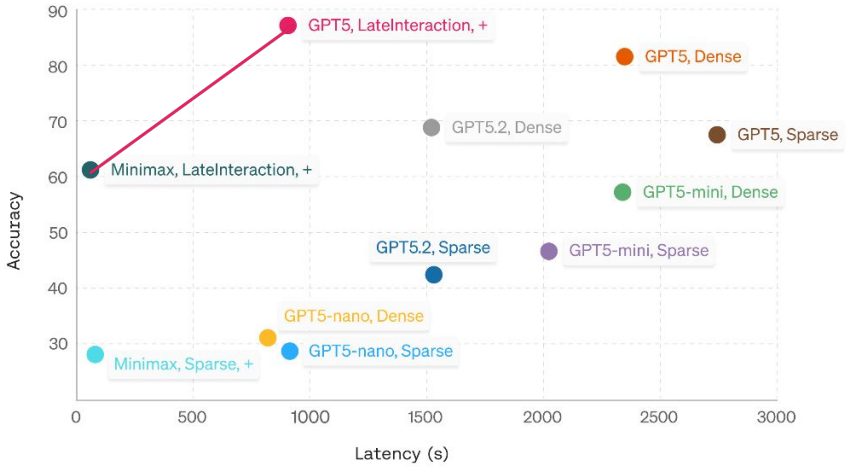
Agent configuration



Mapping the Pareto frontier:

Model & agent tuning





- **GPT5-nano, Dense** – cost=$0.09, latency=820.55s, acc=31.10%
- **GPT5-nano, Sparse** – cost=$0.10, latency=914.49s, acc=28.70%
- **GPT5-mini, Dense** – cost=$0.53, latency=2337.99s, acc=57.20%
- **GPT5-mini, Sparse** – cost=$0.56, latency=2022.53s, acc=46.60%
- **GPT5, Sparse** – cost=$1.65, latency=2742.45s, acc=67.50%
- **GPT5, Dense** – cost=$2.03, latency=2347.04s, acc=81.50%
- **GPT5, LateInteraction, +** – cost=$0.87, latency=906.46s, acc=87.10%
- **GPT5.2, Dense** – cost=$3.01, latency=1520.50s, acc=68.80%
- **GPT5.2, Sparse** – cost=$3.05, latency=1530.51s, acc=42.40%
- **Minimax, Sparse, +** – cost=$0.03, latency=81.98s, acc=28.10%
- **Minimax, LateInteraction, +** – cost=$0.03, latency=61.63s, acc=61.20%

Agent optimization methods

Scaling (test-time compute)

↕ Vertical scaling

- Longer reasoning/thinking tokens
- Longer ReAct loops
- Critique-and-repair loops

↔ Horizontal scaling

- Best-of-N sampling (single model)
- Heterogeneous ensemble (multiple models)
- Execution strategies at runtime

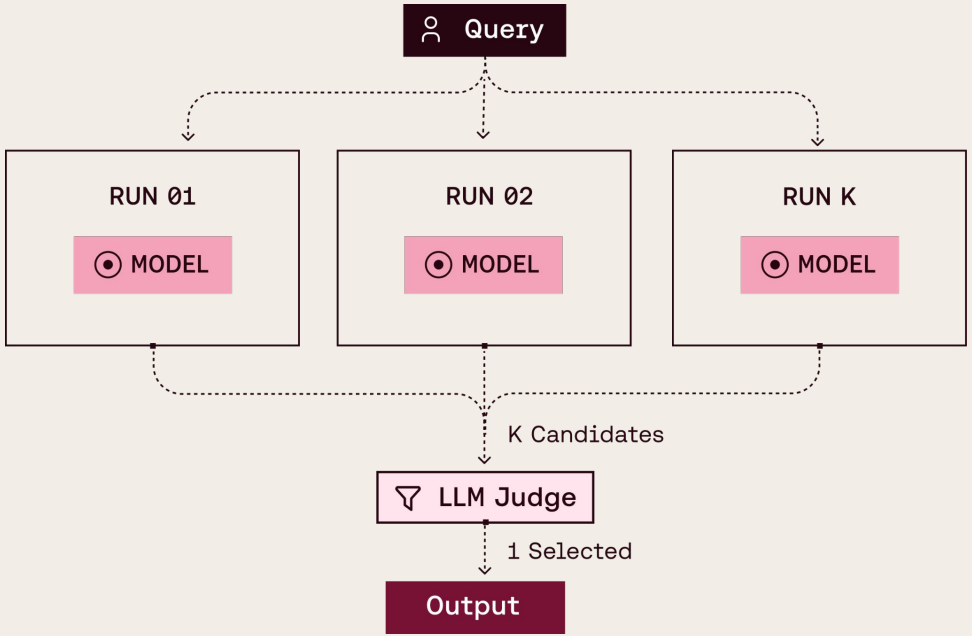
Scaling

Best-of-N sampling

→ One model

→ Many parallel runs

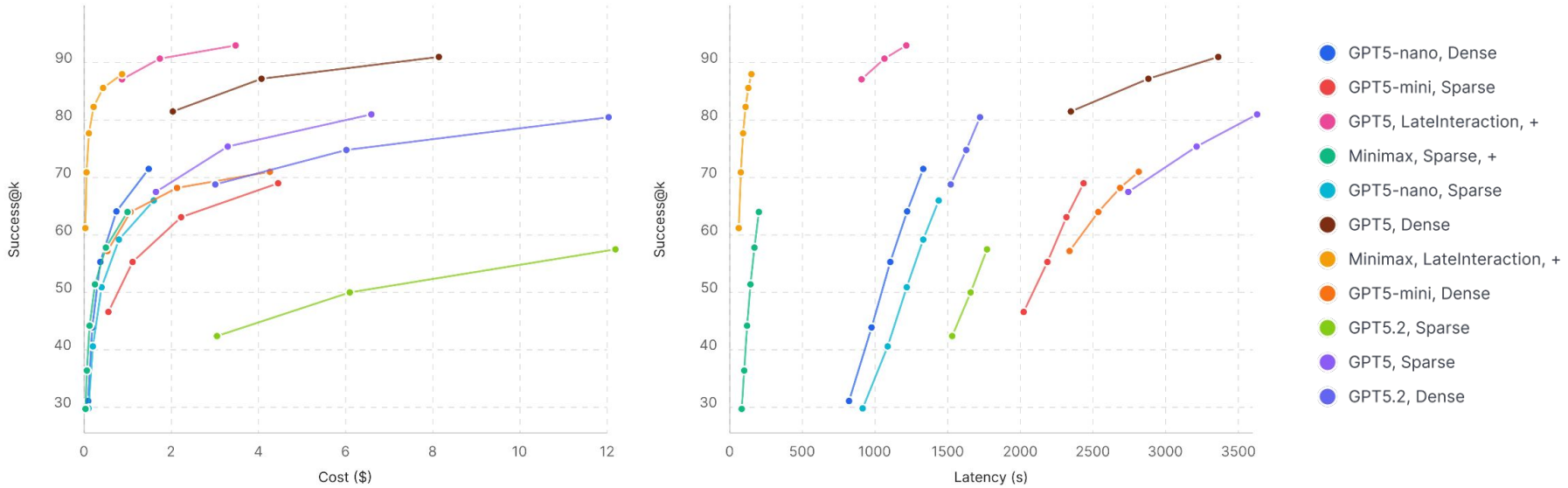
→ Compare & select the best



Mapping the Pareto frontier:

Best-of-N sampling

Single-Variant Scaling (Oracle)

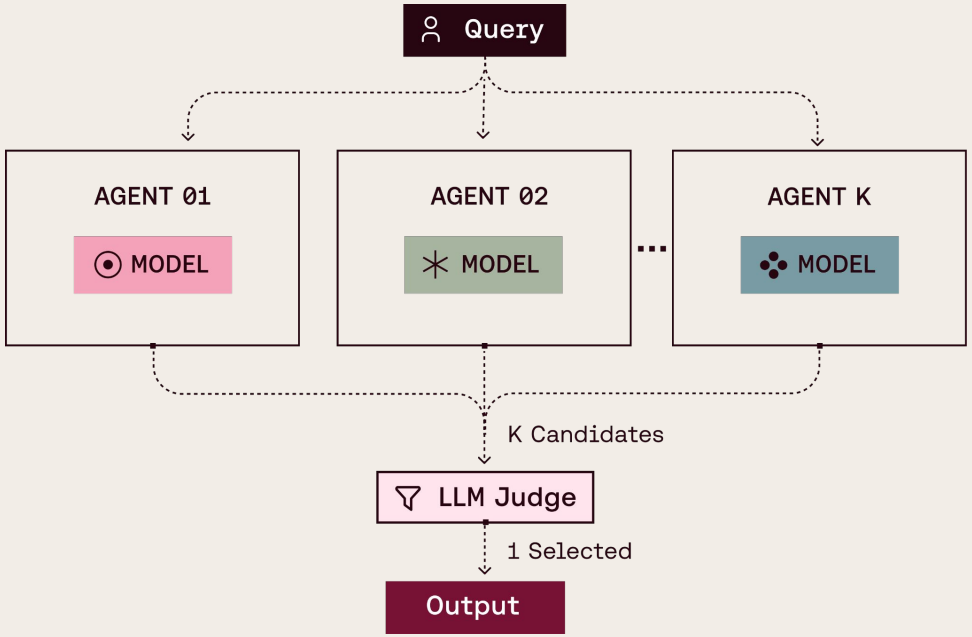


Heterogeneous ensemble

→ Many models

→ Sometimes also different prompts, tools, etc.

→ Combine or compare their outputs

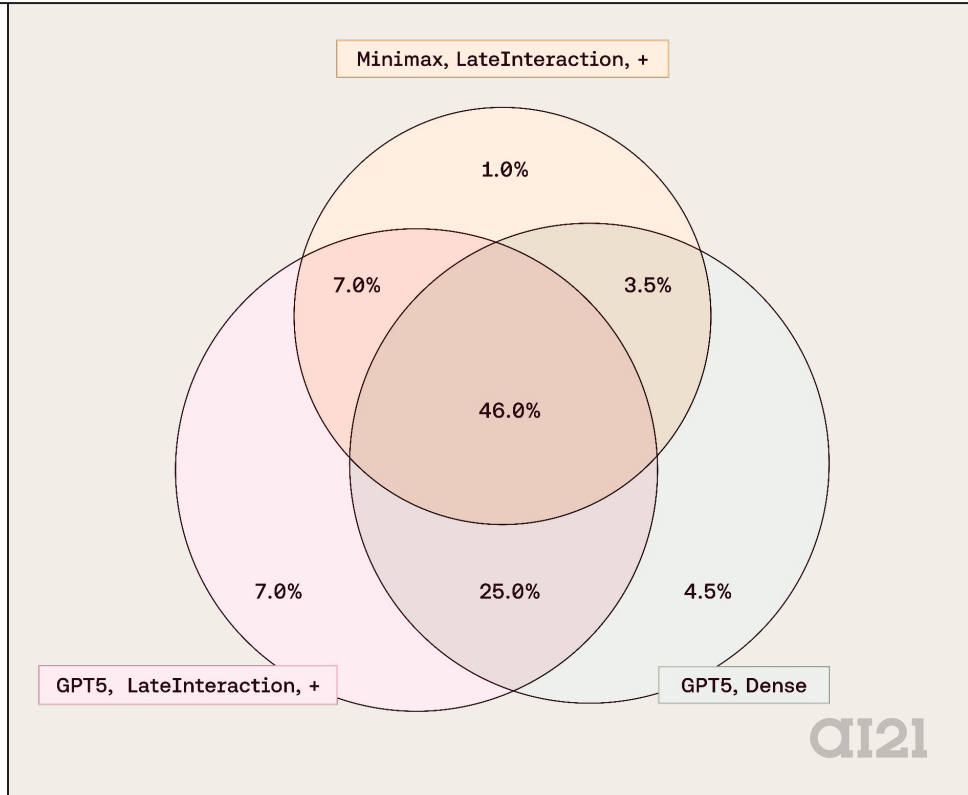


Scaling

Heterogeneous ensemble

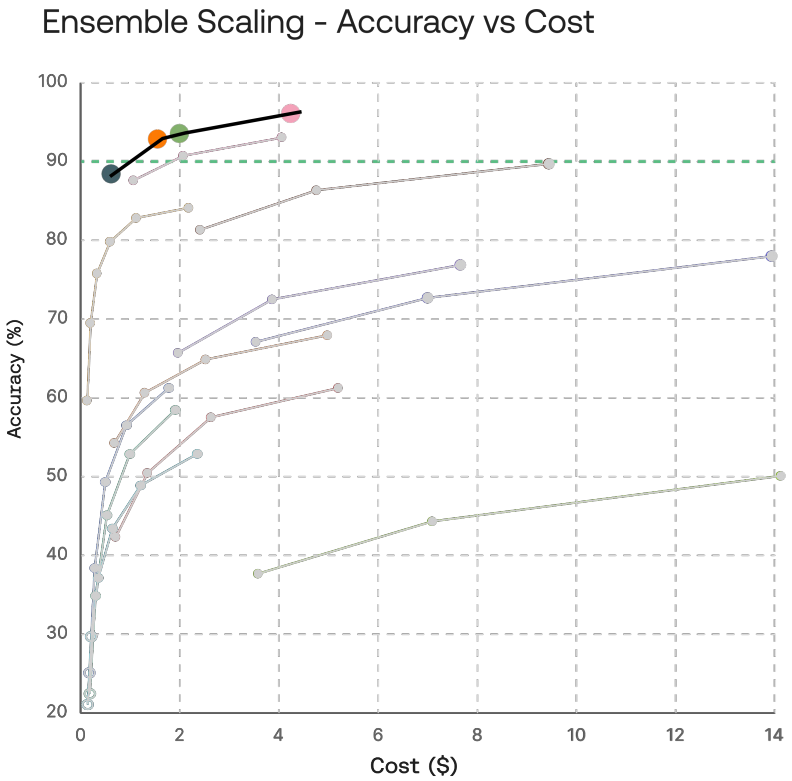
→ Measuring success covariance: Degree to which different configurations complement each other

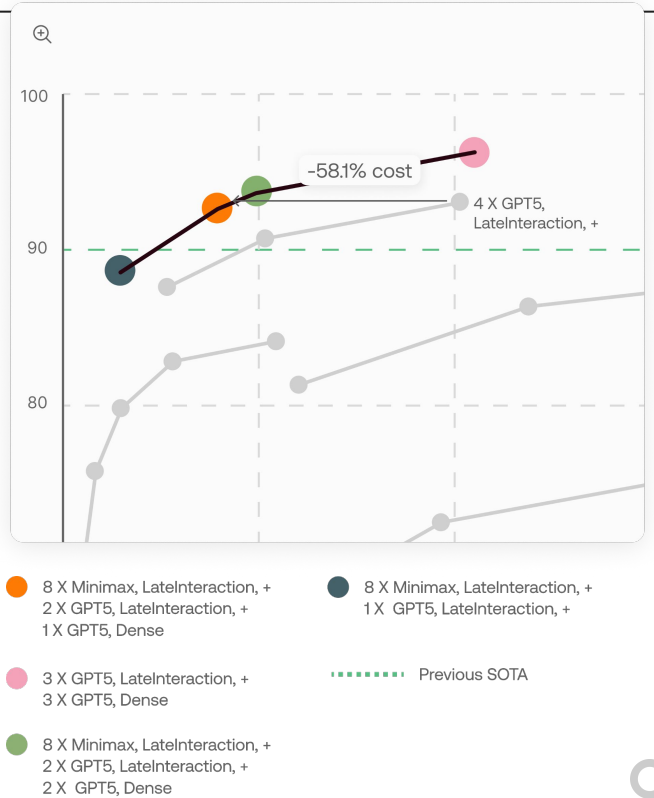
→ Success covariance of the whole > the parts



Mapping the Pareto frontier:

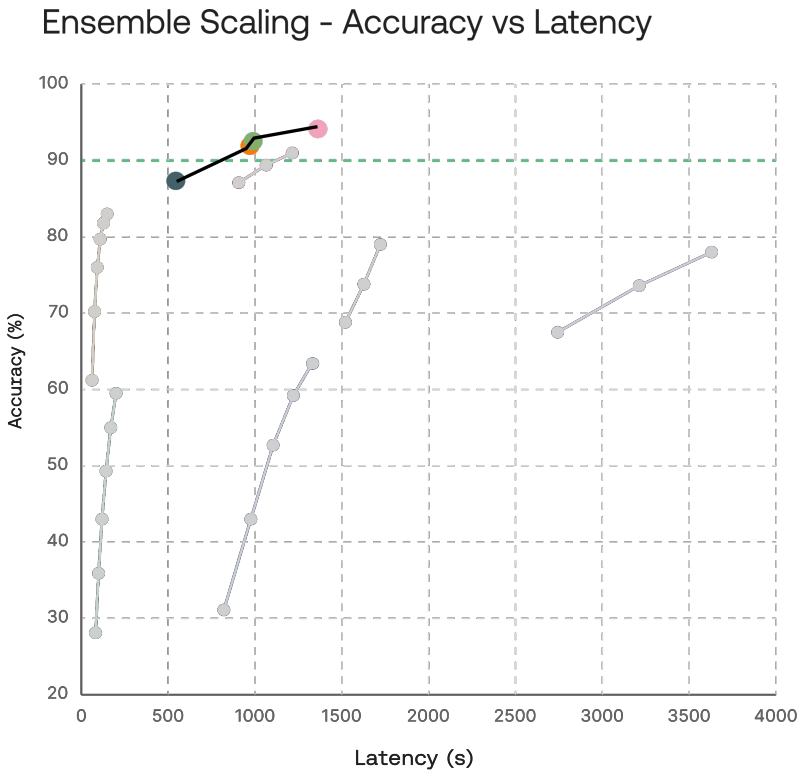
Heterogeneous ensemble

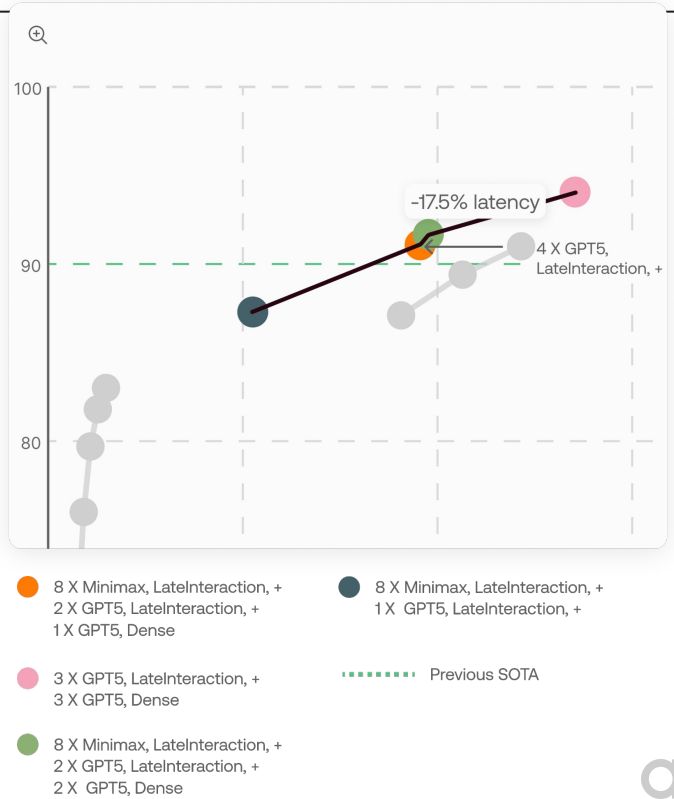




Mapping the Pareto frontier:

Heterogeneous ensemble





Scaling

Execution strategies

`Runtime agent decision checklist:`

→ Which candidates to try first?

→ Start cheap and escalate?

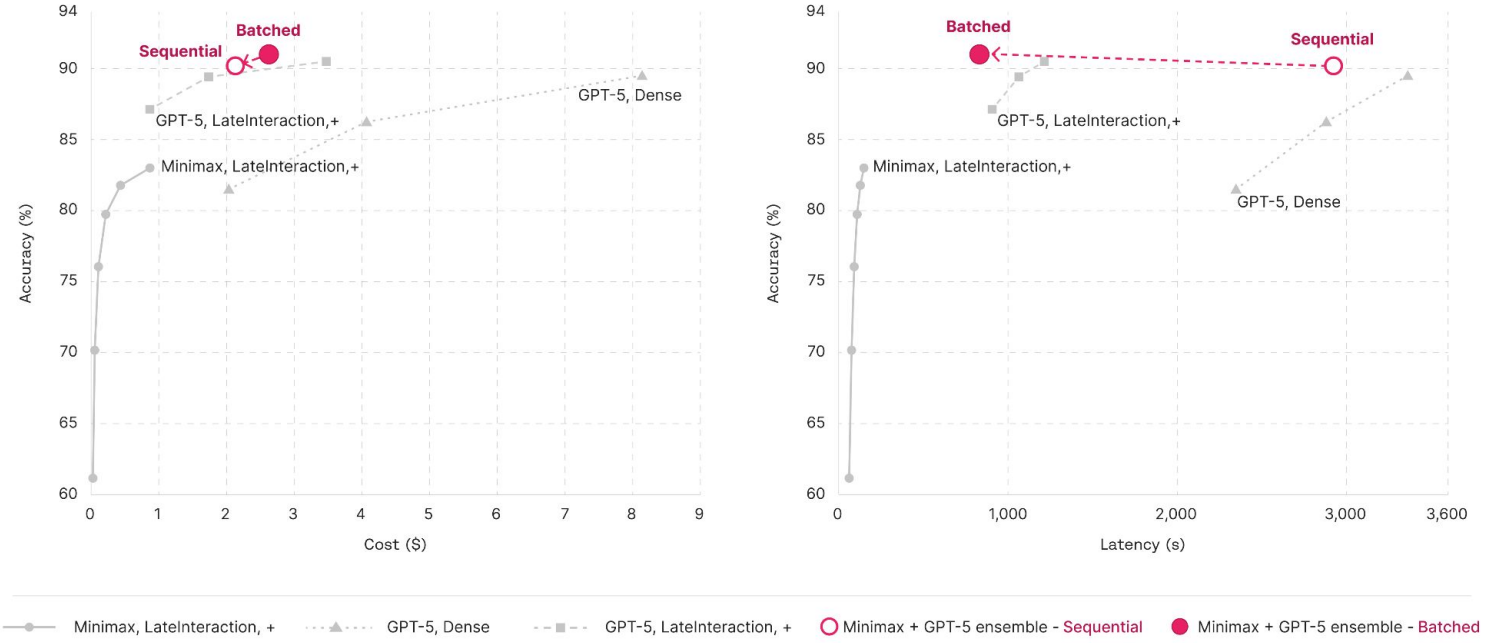
Or full blown parallelize?

→ Prioritize the ones most likely to succeed?

→ When is the stop threshold?

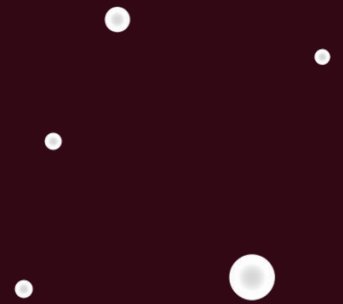
Mapping the Pareto frontier:

Execution strategies

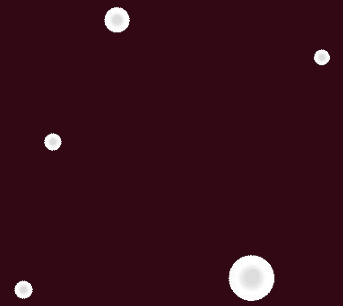


Inside the Lab | BrowseCompPlus benchmark experiments conducted by researchers **AI21 Labs**

Searching across an infinite space



Searching across an infinite space



Searching across an infinite space

Production constrains

Accuracy

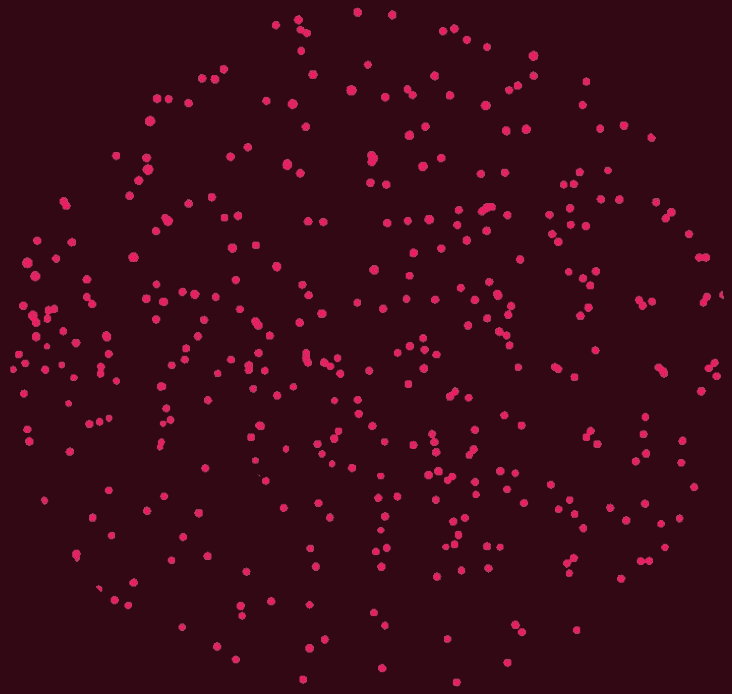
0% 100%

Cost

0$ 10$

Latency

0 sec 1,000 sec

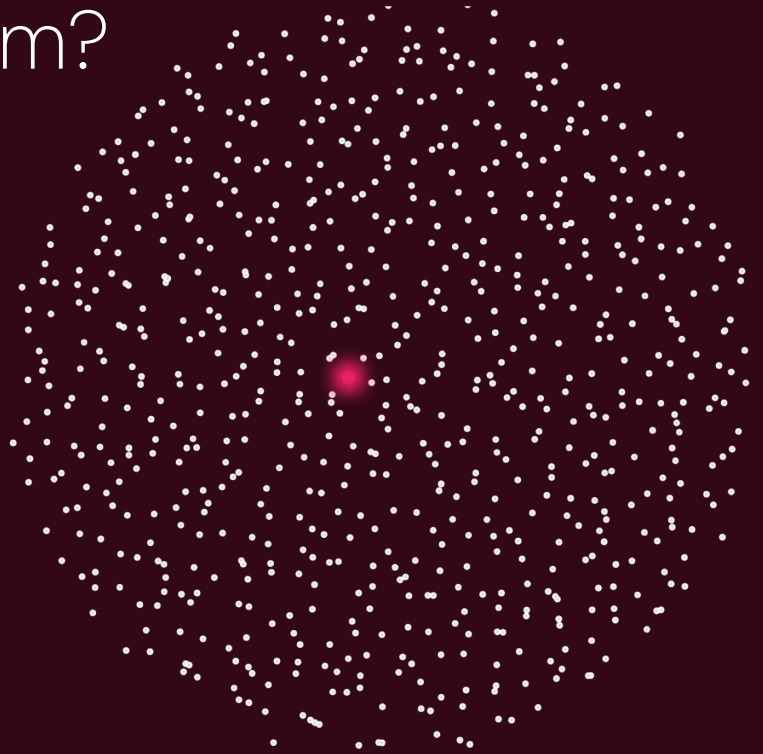


Adding complexity

Navigating infinity

How do we spot the optimum?

Searching for the best outcome
within my cost and latency constraints



Adding complexity

The manual optimization paradox

What happens if...

→ A new frontier model is released?
→ Pricing changes?
→ The task or data changes?

Manual optimization doesn't scale.

Manual → automatic optimization

We need a **systematic method**
for finding and maintaining the optimal
cost/latency/accuracy operating point
for any production agent.

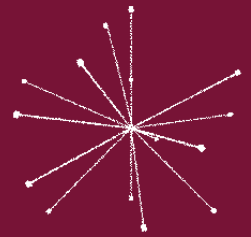
How we're automating agent optimization

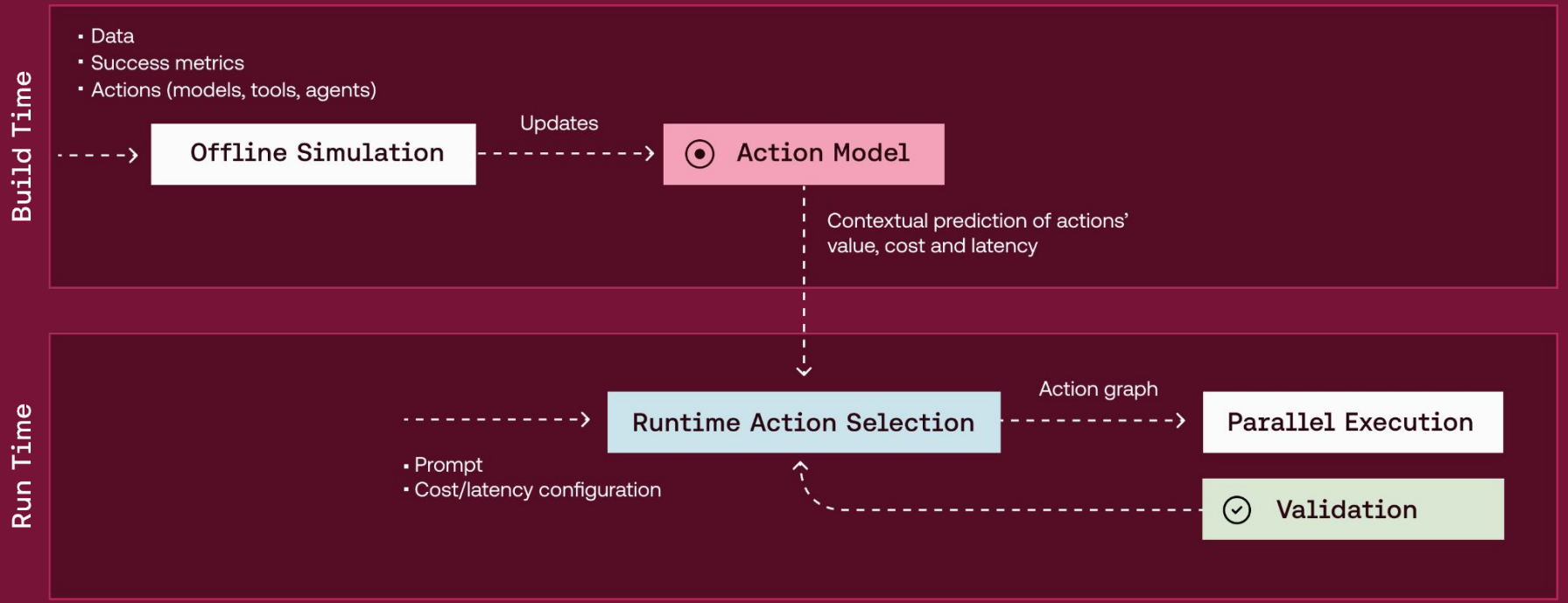
Our systematic method should be:

1. **Automatic:** Removes the need for manual experimentation.
2. **Efficient:** Explores the solutions space at minimal computational cost
3. **Observable:** Reveals the full Pareto frontier across quality, cost, and latency for a specific workload.
4. **Future-proof:** Can easily re-calibrate when query distributions shift or new models are released.

Agent optimization

The MAESTRO approach



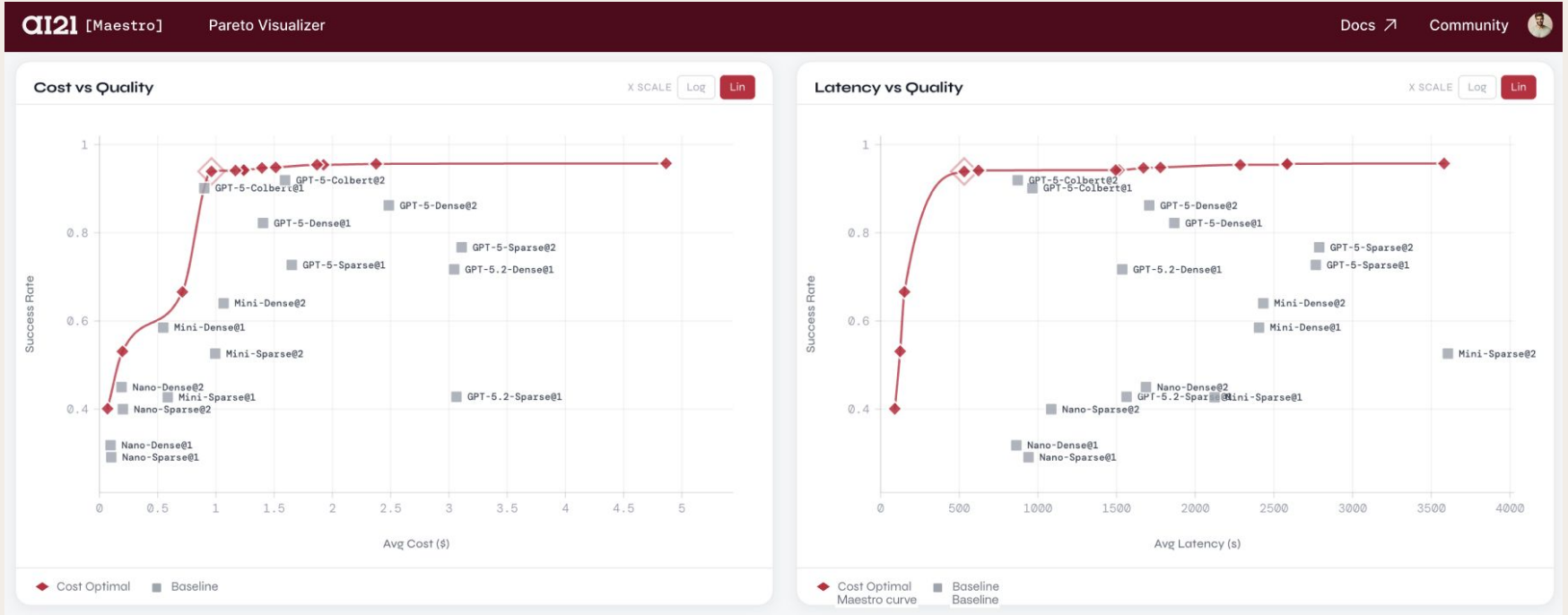


The MAESTRO approach in action:

Optimizing agents for BrowseComp-Plus

Maestro automatically identifies the **optimal model portfolios** and **execution strategies** for BCP by:

Reaching SOTA on BrowseComp-Plus



[Inside the Lab] BrowseCompPlus benchmark experiments conducted by researchers **AI21 Labs**

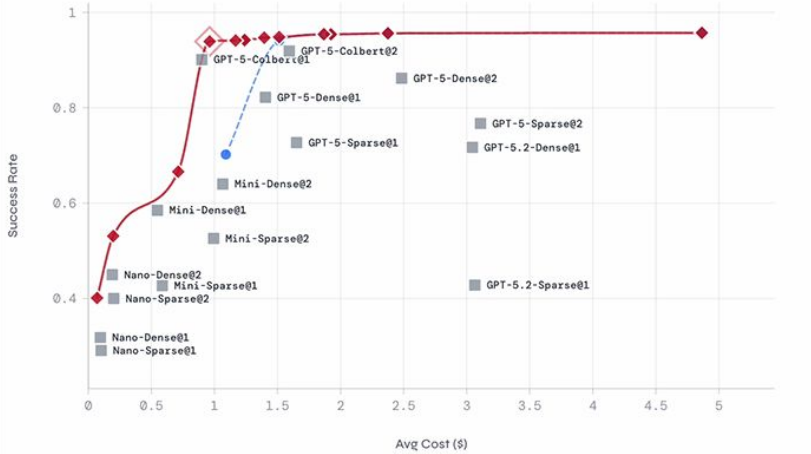
default

Load file

Cost Optimal | Both | Latency Optimal

Cost vs Quality

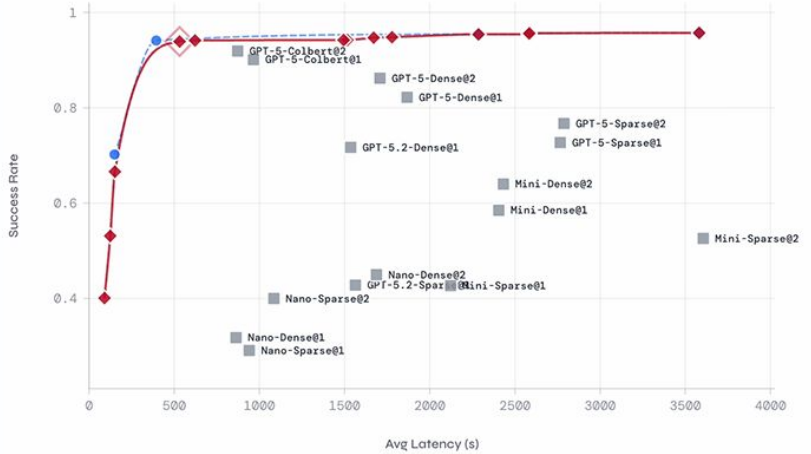
X SCALE Log Lin



Cost Optimal | Latency Optimal | Baseline

Latency vs Quality

X SCALE Log Lin



Cost Optimal | Latency Optimal | Baseline

greedy_lookup_6_6_confidence

Cost Optimal

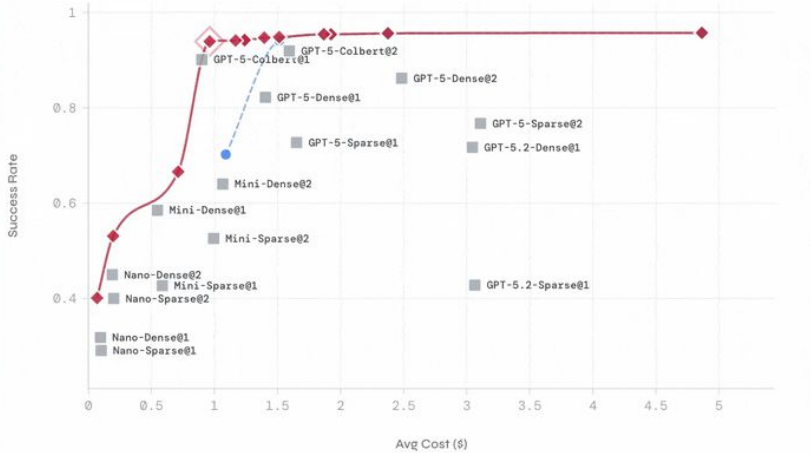
default

Load file

Cost Optimal | Both | Latency Optimal

Cost vs Quality

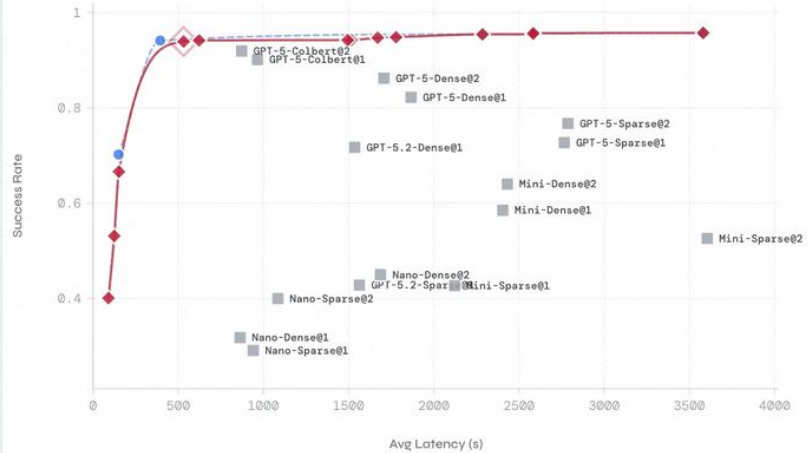
X SCALE Log Lin



Cost Optimal | Latency Optimal | Baseline

Latency vs Quality

X SCALE Log Lin



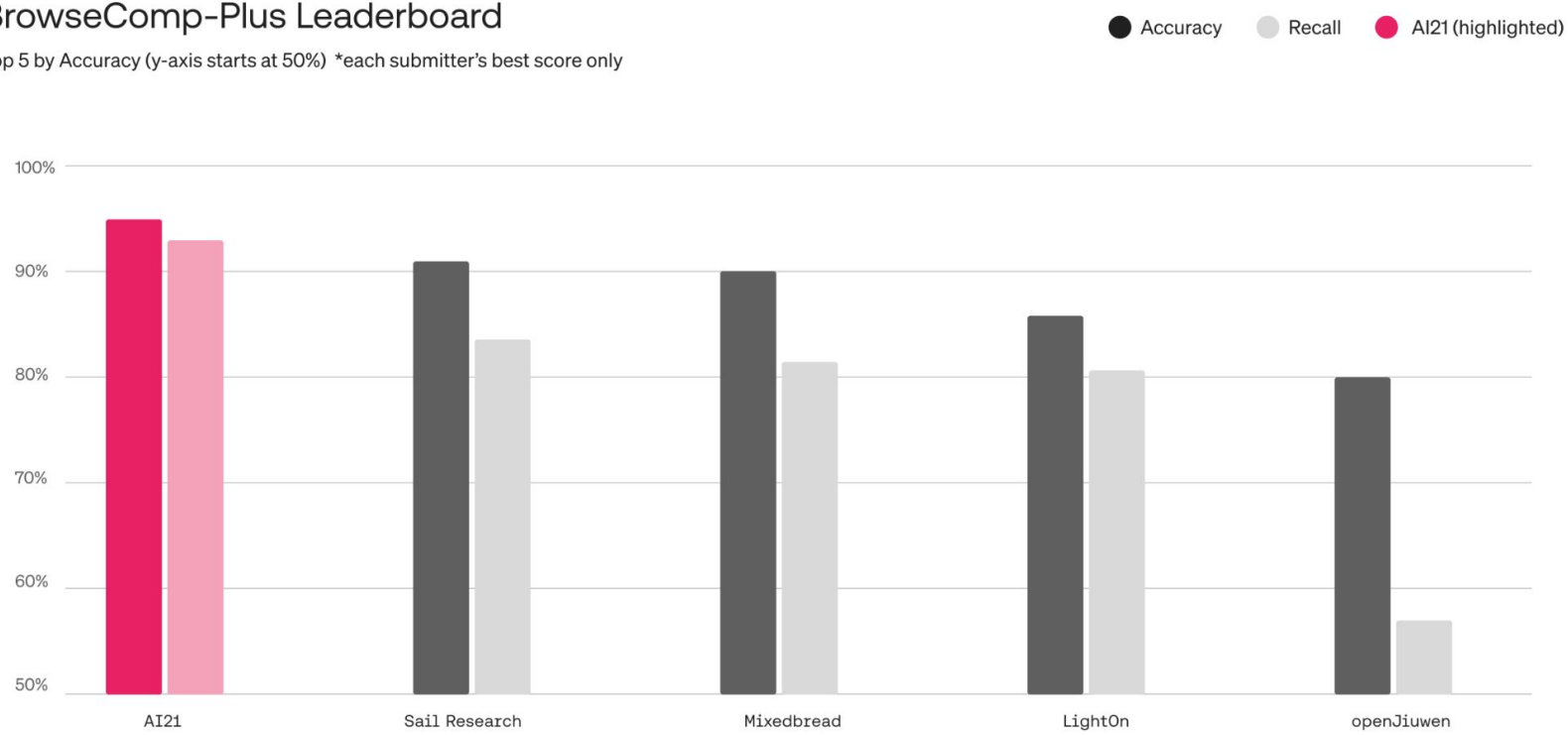
Cost Optimal | Latency Optimal | Baseline

greedy_lookup_6_6_confidence

Cost Optimal

BrowseComp-Plus Leaderboard

Top 5 by Accuracy (y-axis starts at 50%) *each submitter's best score only



[Inside the Lab] BrowseCompPlus benchmark experiments conducted by researchers **AI21 Labs**

The MAESTRO approach in action:

Optimizing Accuracy-Cost-Latency on Deep Research Bench 1 (DRB1)

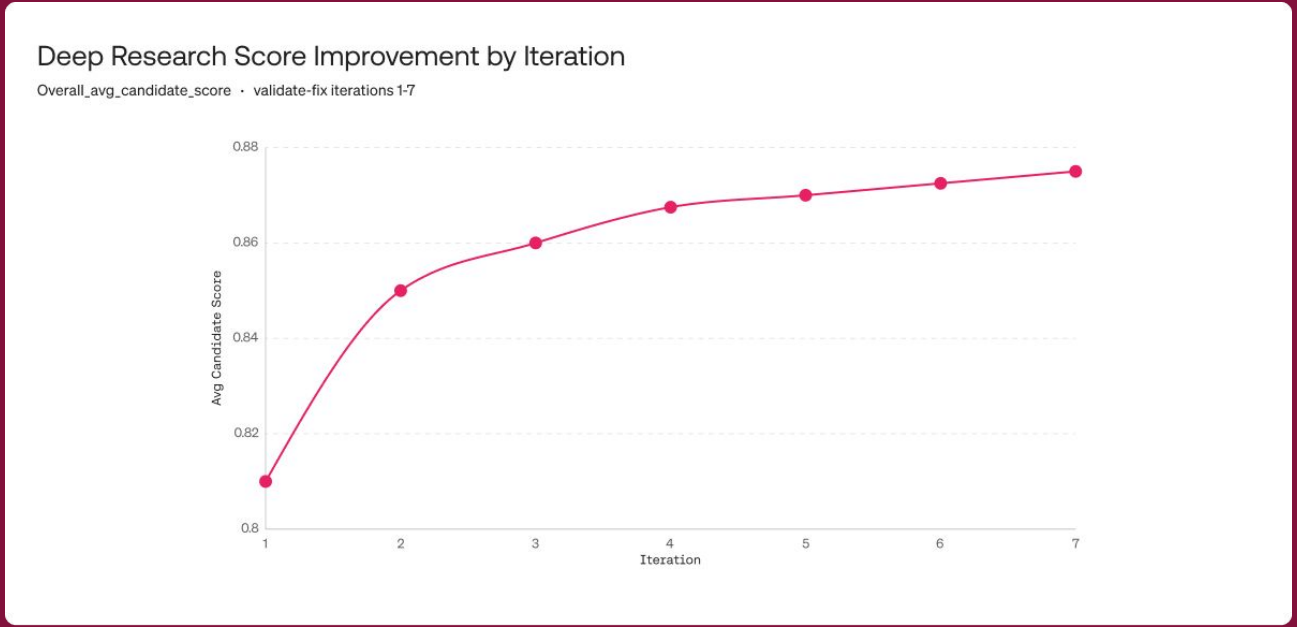
Applying **critique-and-repair loops** to provide a continuous score and structured feedback during runtime.

Step 01	Step 02	Step 03	Step 04	Step 05
Generate query-specific rubric	Generate a research report	Validate against rubrics	Fix	Repeat

The MAESTRO approach in action:

Optimizing Accuracy-Cost-Latency on Deep Research Bench 1 (DRB1)

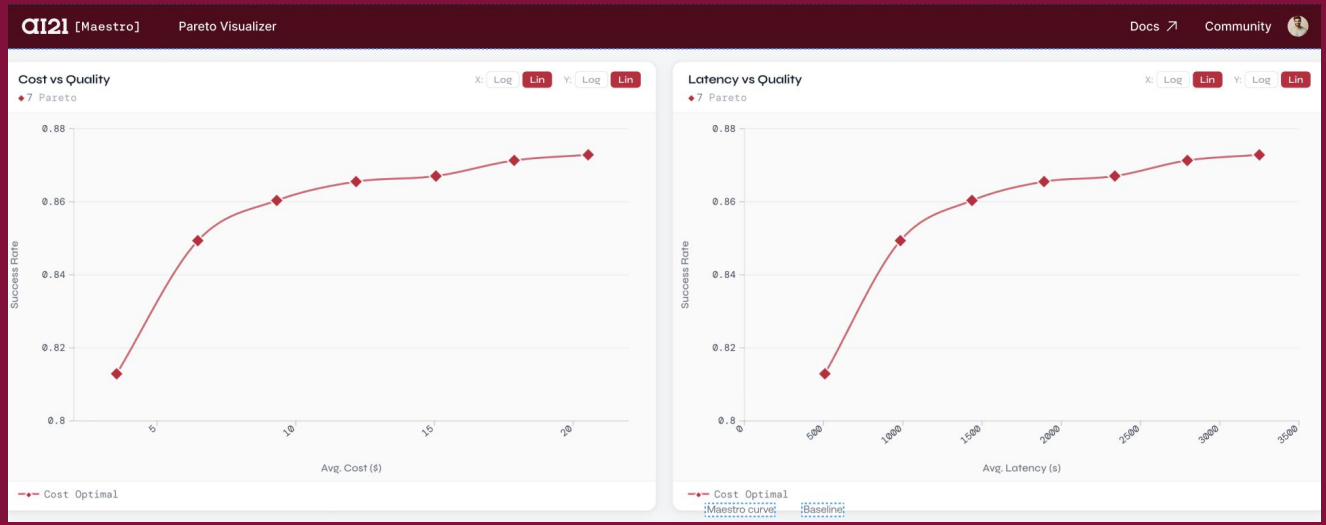
Critique-and-repair improves performance over iterations



The MAESTRO approach in action:

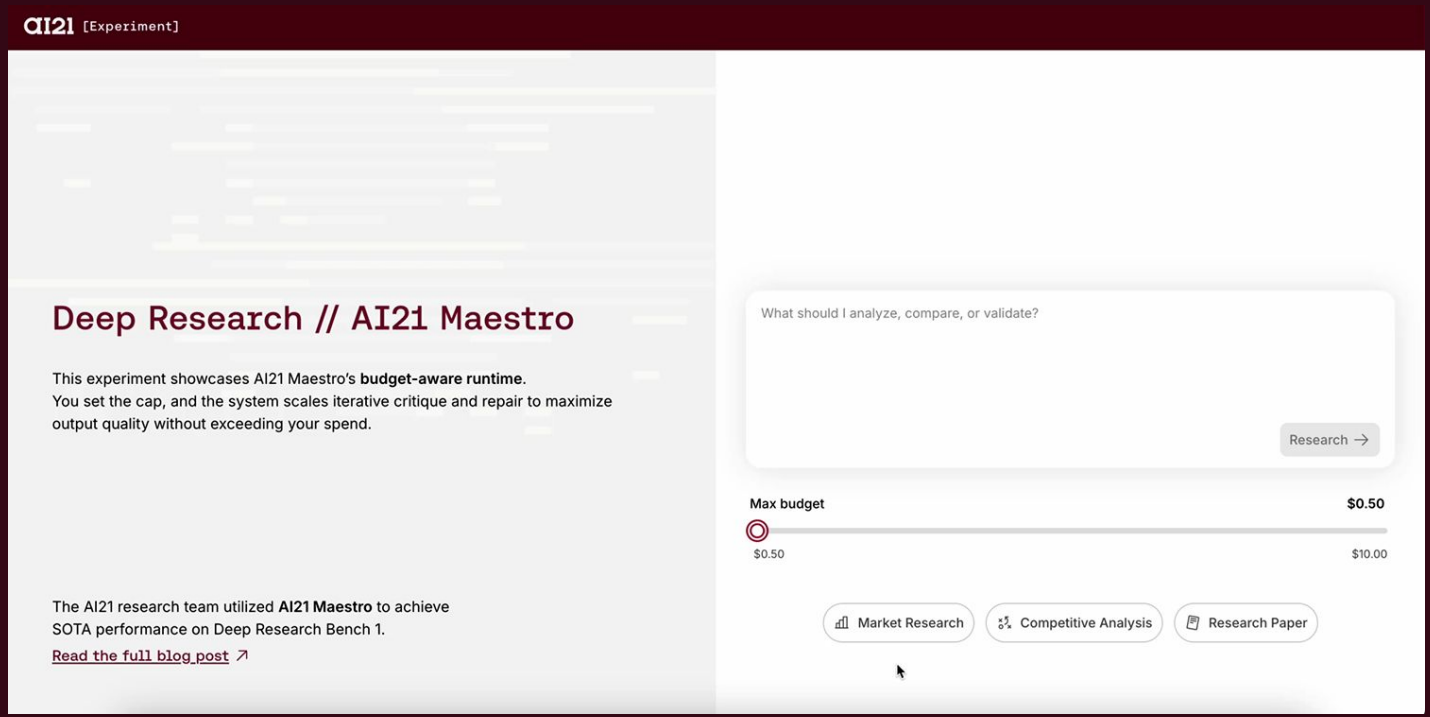
Optimizing Accuracy-Cost-Latency on Deep Research Bench 1 (DRB1)

Managing critique-and-repair loops using Maestro's budget-aware runtime, so you can build a cost-sensitive deployment or premium research product.



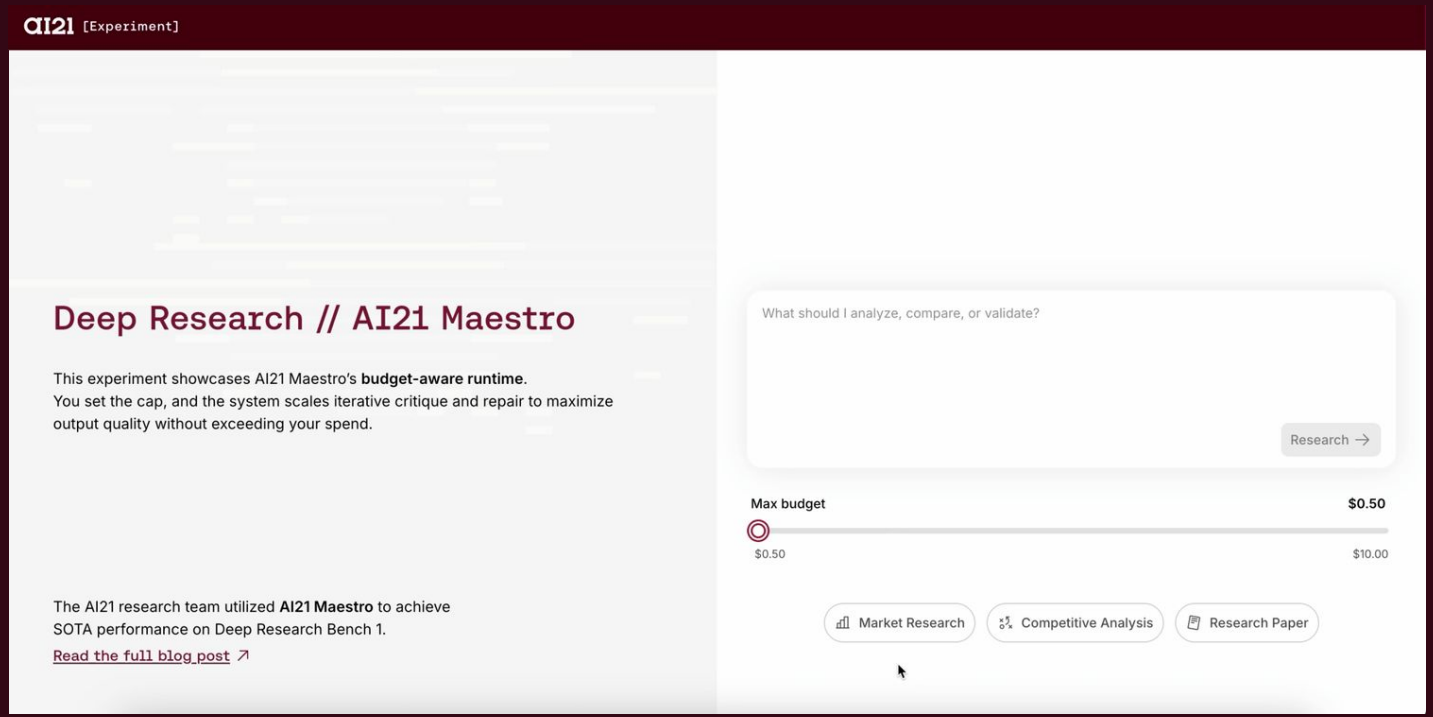
The MAESTRO approach

DR agent in action



The MAESTRO approach

DR agent in action



So finally, we can optimize our agents

Automatic

Removes the need for manual experimentation.

Efficient

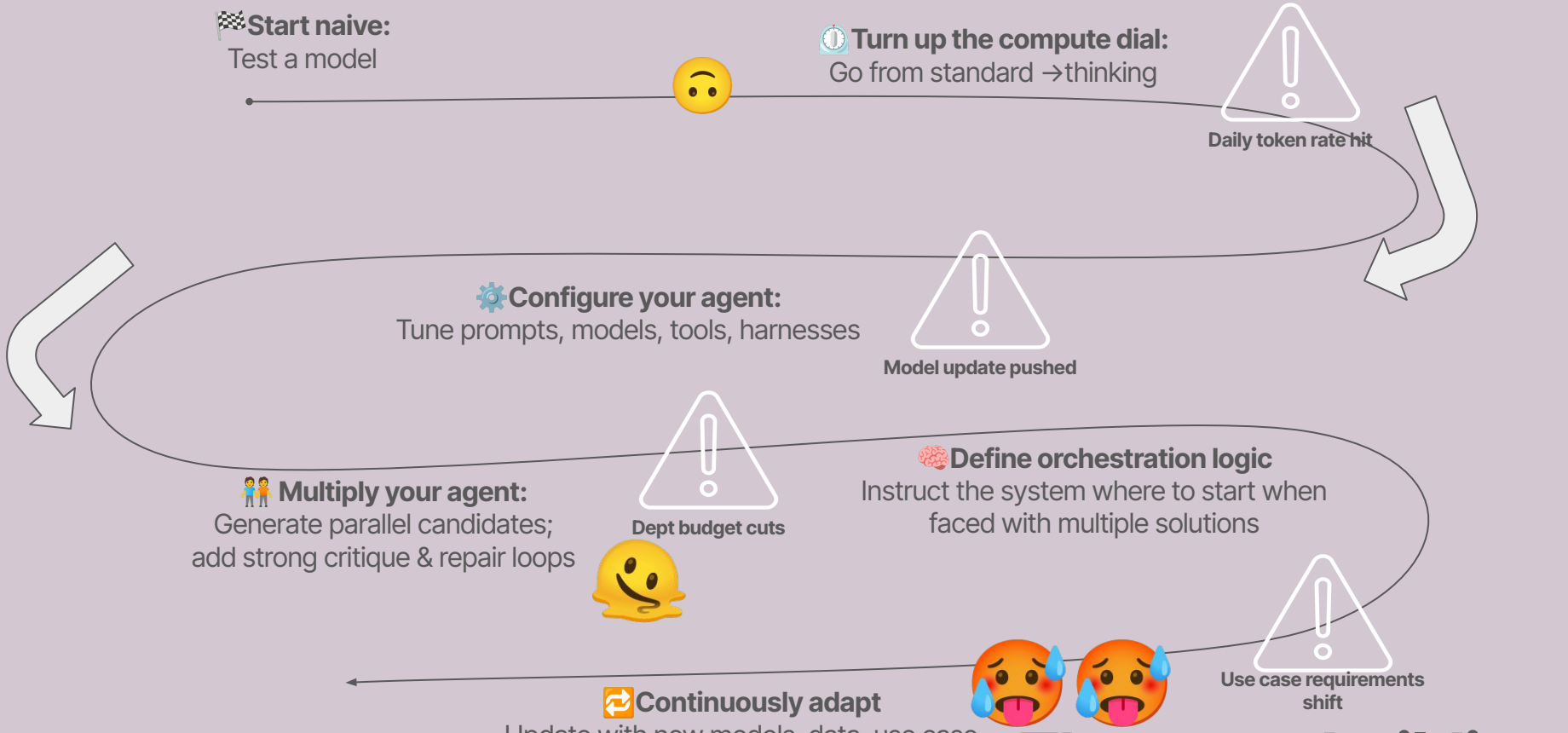
Explores the solutions space at minimal computational cost.

Observable

Reveals the full Pareto frontier across quality, cost, and latency for a specific workload.

Future-proof

Can easily re-calibrate when query distributions shift or new models are released.

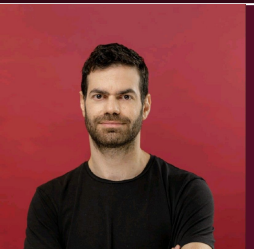


The agent building game of life

Learn more:

The Four Gaps
(Podcast)

SOTA on BPC
(Blog)



Yuval Belfer

Sr. Developer Advocate
AI21 Labs

ai21