

OrchOpt: Validation-Gated Architecture Search for Self-Organizing Multi-Agent Orchestration

Amit Dhanda
Amazon Web Services
amdhand@amazon.com

Sina Amini Niaki
Amazon Web Services
saniaki@amazon.com

Anish Acharya
Amazon Web Services
achanish@amazon.com

Brian Verkhovsky
Amazon Web Services
bverkhov@amazon.com

Abstract

Multi-agent architectures are typically hand-designed, generated in one shot, or evolved through unconstrained code search. None of these treats the *number of agents* as a searchable parameter, and none can discover that a single agent is optimal. We argue that agent count and coordination topology deserve the same optimization discipline used for neural architectures: bounded edits, held-out validation gating, and bidirectional search that adds and removes agents without information loss.

OrchOpt is a validation-gated architecture search over a unified design space in which single-agent and multi-agent configurations compete directly, with bidirectional traversal enabled by knowledge-preserving merge. A proposer model turns failure analysis into bounded merge, split, and skill edits on the current architecture, and an edit is accepted only when it strictly improves validation. Because an absorbed agent’s expertise becomes an activatable skill, merging loses no information and the search stays fully reversible at no additional inference cost.

Across nine benchmarks spanning math, knowledge QA, code generation, reading comprehension, multi-hop reasoning, negotiation, and research, the optimizer simplifies to 1 agent on 4 domains and elaborates to 2–3 agents on 5 domains. It finds architectures scoring +20pp over ADAS on MGSM, +16.7pp over AFlow on HotpotQA, and +27.5pp over MARBLE’s published-best topology on Bargaining, all within 7–42 candidate evaluations.

1 Introduction

A single agent with all tools scores 100% on multilingual grade-school math (MGSM) where a 3-agent specialist chain scores 87%. The same single agent scores 69.5% on collaborative research where a 3-agent sequential chain scores 73.5%. These are not different systems—they are the same optimizer applied to different workloads.

This task-dependent reversal shows that architecture choice matters as much as model choice, yet multi-agent research treats it as a fixed design decision. The field assumes more agents yield better systems (Wu et al., 2023; Hong et al., 2024; Wang et al., 2024), and recent work optimizes communication topologies within that assumption (Chen et al., 2026; Li et al., 2026; Wang et al., 2026). The prior question, *should we use multiple agents at all?*, goes unexamined because existing frameworks cannot represent “no routing” as a valid design point.

We introduce a unified design space where a single agent competes directly against multi-agent configurations. An optimizer searches this space through bounded structural mutations (Figure 1), accepting only changes that improve held-out validation. The search is bidirectional: agents can merge (with expertise preserved as skills) or split (with skills dis-

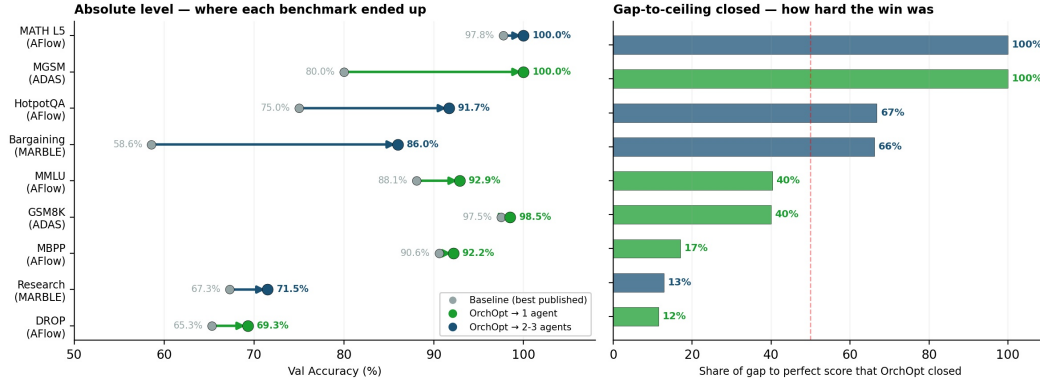


Figure 1: Gray = baseline (best published architecture); colored = OrchOpt gain. Green = single-agent wins; blue = multi-agent chain wins. The same optimizer, starting from identical seeds, converges to opposite architectures depending on the task, and improves every baseline (+1.0 to +27.5pp).

tributed as prompts). This enables the optimizer to discover what we call *domain-conditioned convergence*:

The same optimizer, starting from the same seed pool, converges to opposite architectures for different task distributions.

The determining factor is *task structure*. When the task requires an unbroken chain of reasoning or a single cognitive act, adding agents introduces harmful coordination boundaries. When the task decomposes into independent subtasks with typed interfaces, multiple agents execute them more cleanly than one:

- **Single agent wins** (4 domains) when answers require context continuity (uniform arithmetic: MGSM), parametric recall (MMLU), coherent single-pass generation (MBPP), or span extraction across diverse question types (DROP).
- **Multi-agent chain wins** (5 domains) when subtasks are separable: plan/execute/edge-case handling for heterogeneous grade-school math at scale (GSM8K), parse/solve/verify for competition math (MATH L5), parallel entity retrieval for multi-hop QA (HotpotQA), and sequential analytical pipelines for negotiation (MARBLE Bargaining) and research (MARBLE Research).

On external benchmarks, this structured search discovers simpler architectures that outscore ADAS (Hu et al., 2024), the strongest automated architecture discovery method, by +20pp on MGSM and +4pp on DROP while using fewer inference calls.

Contributions.

1. A unified single-to-multi agent design space with structured edit operations enabling bidirectional search (§3).
2. Evidence that published topology rankings are *model-specific*: Graph, reported as best in prior work, becomes worst in our evaluation, implying multi-agent evaluations require re-validation when backbone models change (§J).
3. Empirical demonstration of domain-conditioned convergence across nine benchmarks, with an emergent structural pattern explaining why architecture diverges (§5).
4. *Knowledge-preserving merge*: agent expertise converts to activatable skills upon merging, enabling bidirectional search and lifelong skill accumulation (§3.2).
5. Convergence in 7–42 evaluations via beam search with speculative filtering and 3-layer structural deduplication (§F).

2 Problem Formulation

2.1 Orchestration Design Space

An orchestration design $\mathcal{D} = (\mathcal{A}, \mathcal{S}, \mathcal{C}, \mathcal{K})$ comprises:

$$\mathcal{A} = \{a_1, \dots, a_n\}, \quad a_i = (T_i, p_i), \quad T_i \subseteq \mathcal{T} \quad (1)$$

$$\mathcal{S} = \{s_1, \dots, s_m\}, \quad s_j = (\tau_j, I_j) \quad (2)$$

$$\mathcal{C} \in \{\text{direct}, \text{plan-act}, \text{critic}, \text{comparator}, \dots\} \quad (3)$$

$$\mathcal{K} = \{(k_i, [t_1 \circ \dots \circ t_r]) : t_j \in \mathcal{T}\} \quad (4)$$

where $\mathcal{T}$ is a global tool inventory, T_i is agent a_i 's tool subset, p_i its system prompt, τ_j are skill activation triggers, I_j are procedural instructions, and $\mathcal{K}$ chains atomic tools into composite operations.

Components. Each *agent* a_i is defined by a tool subset T_i (what it can do) and a system prompt p_i (how it reasons). Each *skill* s_j is a procedural rule: a set of triggers τ_j that activate it, and an instruction I_j that the agent follows when activated. Skills are lightweight: they modify behavior without adding agents or tools. *Coordination* $\mathcal{C}$ specifies how agents interact: direct (single agent acts alone), plan-act (one agent plans, another executes), chain (sequential pipeline), or star (central coordinator). *Composite tools* $\mathcal{K}$ chain atomic tool calls into multi-step operations.

Design space boundaries. The design space Ω admits $|\mathcal{A}| \in [1, |\mathcal{T}|]$, with two extremes:

$$|\mathcal{A}| = 1 : \quad T_1 = \mathcal{T}, \quad \mathcal{S} = \{s_1, \dots, s_m\} \quad (\text{uber-agent, skill-optimized}) \quad (5)$$

$$|\mathcal{A}| = N : \quad |T_i| \approx |\mathcal{T}|/N, \quad \mathcal{S} = \emptyset \quad (\text{specialists, routing required}) \quad (6)$$

The $|\mathcal{A}| = 1$ regime is not a degenerate point; it has its own rich optimization surface along the skill dimension $|\mathcal{S}|$. The optimizer can ADD_SKILL, REWRITE_SKILL, or MERGE_SKILLS without ever changing agent count, evolving a single agent's behavioral repertoire through procedural rules. On 4 of our 9 benchmarks, this skill-only optimization path produces the best architecture. Existing topology optimization work (Chen et al., 2026; Li et al., 2026; Wang et al., 2026) assumes $|\mathcal{A}| \geq 2$, excluding this entire regime.

Why skills matter. When agents merge ($|\mathcal{A}|$ decreases), their expertise could be lost. Skills provide the retention mechanism: the merged agent inherits activatable rules from absorbed agents. This makes the $|\mathcal{A}| = 1$ design point *stronger* than a naive single agent: it carries the accumulated knowledge of previously separate specialists.

2.2 Objective and Evaluation

Given evaluation distribution $\mathcal{X}$, we seek:

$$\mathcal{D}^* = \arg \max_{\mathcal{D} \in \Omega} \mathbb{E}_{x \sim \mathcal{X}_{\text{val}}} [\text{eval}(\mathcal{D}, x)] \quad (7)$$

subject to: $\text{eval}(\mathcal{D}, \mathcal{X}_{\text{train}}) \geq \text{eval}(\mathcal{D}^*, \mathcal{X}_{\text{train}}) - \epsilon$ (non-regression).

The non-regression constraint ensures the optimizer never degrades on the training distribution while pursuing validation improvements, analogous to the monotonic improvement guarantee in policy gradient methods. In practice, $\epsilon = 0.02$ (2pp) absorbs evaluation noise while preventing large regressions.

Train/val separation. Following standard practice in architecture search, we use disjoint train and val splits. The proposer LLM sees failure analysis from the *training* set (what went wrong and why), but acceptance is gated on the *validation* set (unseen cases). This prevents the optimizer from overfitting to specific failure patterns in the training data.

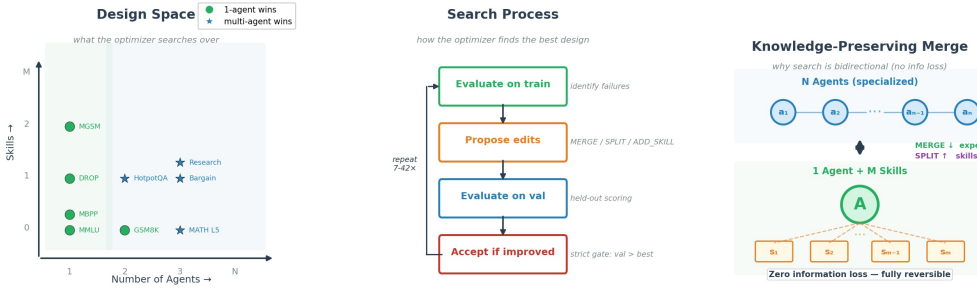


Figure 2: OrchOpt overview. **Left:** The design space is 2D (agents $\times$ skills); benchmarks land at different optima — green dots for single-agent wins, blue stars for multi-agent. **Center:** The search loop evaluates on train, proposes bounded edits, gates on val, and repeats 7–42 times. **Right:** Knowledge-preserving merge enables bidirectional traversal: N agents can collapse into 1 agent + M skills (MERGE), or reverse (SPLIT), with no information loss.

3 Method

3.1 Structured Edit Operations

Rather than generating architectures from scratch, we evolve them through bounded edits (at most B operations per epoch; we use $B = 3$ in all runs):

Operation	Signature	Effect
<i>Agent-level (change $\mathcal{A}$)</i>		
MERGE	$\{a_i, a_j\} \rightarrow a_k$	Union tools, expertise $\rightarrow$ skills
SPLIT	$a_i \rightarrow \{a_j, a_k\}$	Partition tools into specialists
<i>Skill-level (change behavior without changing $\mathcal{A}$)</i>		
ADDSKILL	(s, a_i)	Attach procedural rule to agent
REWRITESKILL	(s_j, I'_j)	Refine skill instruction
MERGESKILLS	$\{s_i, s_j\} \rightarrow s_k$	Consolidate overlapping skills
<i>Coordination-level (change how agents interact)</i>		
ADDCOORD	C'	Wire protocol (plan-act, critic, chain)
DELCOORD	C'	Remove coordination overhead

Table 1: Structured edit operations. Three levels: agent topology, skill content, and coordination wiring. Each epoch applies at most $B = 3$ edits, bounding the search radius.

The hierarchy mirrors coarse-to-fine search: skill rewrites are cheap to evaluate, so the optimizer tries them before proposing structural merges/splits.

3.2 Knowledge-Preserving Merge

When MERGE_AGENTS executes, each absorbed agent’s system prompt is condensed into an activatable skill:

$$\text{MERGE}(a_i, a_j) \rightarrow (a_k, \{s_i, s_j\}), \quad T_k = T_i \cup T_j \quad (8)$$

where $s_i = (\tau_i, \text{condense}(p_i))$ extracts triggers and procedural rules from agent a_i ’s prompt. Without knowledge preservation, architecture search can only move in one direction: splitting agents creates specialized expertise, but merging them back destroys it (the merged agent loses the nuanced behavior encoded in each specialist’s prompt). Knowledge preservation eliminates this asymmetry: because merged expertise survives as activatable skills, the optimizer can traverse between single-agent and multi-agent designs without information loss.

3.3 Optimization Loop

The search proceeds iteratively (Algorithm 1, Appendix): evaluate on train to identify failures, propose bounded edits, evaluate on val, and accept only strict improvements. Four design principles distinguish this from prior architecture search:

1. Proposer with actionable context. A frontier LLM receives the current design, full edit history, rejected buffer, and batch-level failure analysis classifying training errors into actionable categories. Structured failure diagnostics yield faster convergence than score-only feedback (Agrawal et al., 2026).

2. Strict validation gate. Acceptance requires strict improvement ($s_i > s^*$); ties are rejected. This prevents complexity growth without accuracy gain.

3. Rejected buffer as negative feedback. Failed candidates persist across epochs with their scores and the proposer’s hypothesis of why they would help. This prevents the optimizer from re-proposing structurally similar designs that already failed. On MARBLE Bargaining, it avoids re-trying star topology after that scores 46% in epoch 0, despite the proposer’s intuition that mediators should help negotiation.

4. Slow update. After each epoch, a reflection step distills 1–3 durable learnings from the batch of accepted and rejected candidates. Example from GSM8K: “adding verification agents provides no gain when the model already self-checks via the verify tool.” These epoch-level insights steer future proposals away from entire architectural directions rather than individual failed candidates.

3.4 Search Strategies

We implement beam search as the default strategy, maintaining top- k designs in parallel. Each epoch expands all beams with proposals, evaluates candidates, and retains top- k by val score. To reduce evaluation cost, speculative filtering eliminates 60–70% of proposals before full evaluation: a plausibility check (“does this edit address a known failure mode?”) followed by a 3-case hard subset gates access to the expensive full train/val pass.

Termination. The search stops on the first of three conditions: (i) validation accuracy reaches the ceiling (100%), (ii) no candidate improves the incumbent for $p=3$ consecutive epochs (plateau early-stopping), or (iii) the epoch budget E is exhausted. Because a run halts as soon as validation saturates, the number of evaluated candidates varies by benchmark: from 7 (e.g. MATH L5, whose seed already scores near-ceiling and triggers condition (i) after one epoch) to 42 on harder distributions that run the full budget. This is the source of the wide evaluation-count range, not repeated re-runs of a fixed loop.

4 Experimental Setup

Benchmarks. We evaluate on nine benchmarks spanning math reasoning (MGSM, GSM8K (Cobbe et al., 2021), MATH L5 (Hendrycks et al., 2021b)), knowledge QA (MMLU (Hendrycks et al., 2021a)), code generation (MBPP (Austin et al., 2021)), reading comprehension (DROP), multi-hop QA (HotpotQA (Yang et al., 2018)), and multi-agent coordination (MARBLE Bargaining & Research (Zhu et al., 2025)). All use Claude Sonnet 4.6 as the backbone. Dataset sizes, splits, and metrics are detailed in Table 2.

Baselines. We compare against three published methods, each re-evaluated with our backbone model for fair comparison: (1) ADAS (Hu et al., 2024), which generates agent architectures as unconstrained Python code; (2) AFlow (Zhang et al., 2024), which uses MCTS to search over code-represented workflows; and (3) the MARBLE topology rankings (Zhu et al., 2025), which report Star/Chain/Graph performance with a weaker model.

Benchmark	Base	Ours	Δ	Architecture
<i>Single-agent wins (4 domains)</i>				
MGSM (arithmetic)	80.0	100	+20.0	1 agent + 3 skills
MMLU (knowledge QA)	88.1	92.9	+4.8	1 agent, raw
MBPP (code gen)	90.6	92.2	+1.6	1 agent, raw
DROP (reading)	65.3	69.3	+4.0	1 agent + 2 skills
<i>Multi-agent wins (5 domains)</i>				
GSM8K (arithmetic)	97.5	98.5	+1.0	3: plan→exec→edge-case
MATH L5 (comp. math)	97.8	100	+2.2	3: decomp→solve→verify
HotpotQA (multi-hop)	75.0	91.7	+16.7	2: parallel + decomp. skill
MARBLE Bargain	58.6	86.0	+27.5	3: analyst→strat→closer
MARBLE Research	67.3	71.5	+4.2	3: chain + ideation skills

Table 2: Domain-conditioned convergence. Base = best published architecture re-evaluated with our model. Single-agent wins 4 domains; multi-agent chains win 5.

Optimizer configuration. The proposer (Claude Sonnet 4.6) generates candidates for up to $E=3$ epochs (fewer under early-stopping; see §3 Termination) with 2–6 candidates per epoch and an edit budget of $B = 3$. To ensure the search explores simplification, we enforce ≥ 1 agent-merging candidate per epoch. Each run starts from 3 seed architectures covering the design space: a single uber-agent with skills, a 2-agent pipeline, and a multi-agent specialist chain.

5 Results

5.1 Domain-Conditioned Convergence

Table 2 summarizes the main result: the optimizer converges to different architectures for different task distributions, even within arithmetic math, choosing a single agent for MGSM’s uniform problems and a 3-agent decomposition for GSM8K’s heterogeneous problems at scale. The determining factor is the problem distribution, not the domain label or starting conditions.

5.2 Single-Agent Domains

Four domains converge to single-agent architectures, each for a distinct mechanistic reason:

Uniform arithmetic (MGSM). Multi-step arithmetic on uniform problems requires full context continuity. Agent boundaries cause truncation (predicting 430 instead of 1430). A single agent with procedural skills (“always use calculate tool”) outperforms any decomposition: beam search discovers a 100% design in epoch 2. GSM8K, a larger and more heterogeneous arithmetic set, diverges from this pattern and converges to a multi-agent chain (§5, Multi-Agent Domains), showing that architecture tracks the problem distribution rather than the domain label.

Knowledge QA and code generation (MMLU, MBPP). Both are monolithic cognitive acts: the answer is either in the model’s parametric memory (MMLU) or generated in one coherent pass (MBPP). Multi-agent routing introduces coordination noise: a 2-agent MMLU chain collapses to 52.4% (near random), and a 3-agent MBPP chain scores 57.8% versus 92.2% for a single agent. Skills also hurt here, because the model’s raw capability exceeds its prompted capability on tasks where it is near-ceiling.

Reading comprehension (DROP). At scale (150 val), the diversity of question types (numeric, entity, date, count) makes agent-boundary specialization brittle. The optimizer converges to a single agent with two minimal-span extraction skills, improving AFlow from 65.3% to 69.3%. The small-scale result (15 val) where 2-agent retriever→reasoner reached 100% reflects limited val distribution rather than a general principle.

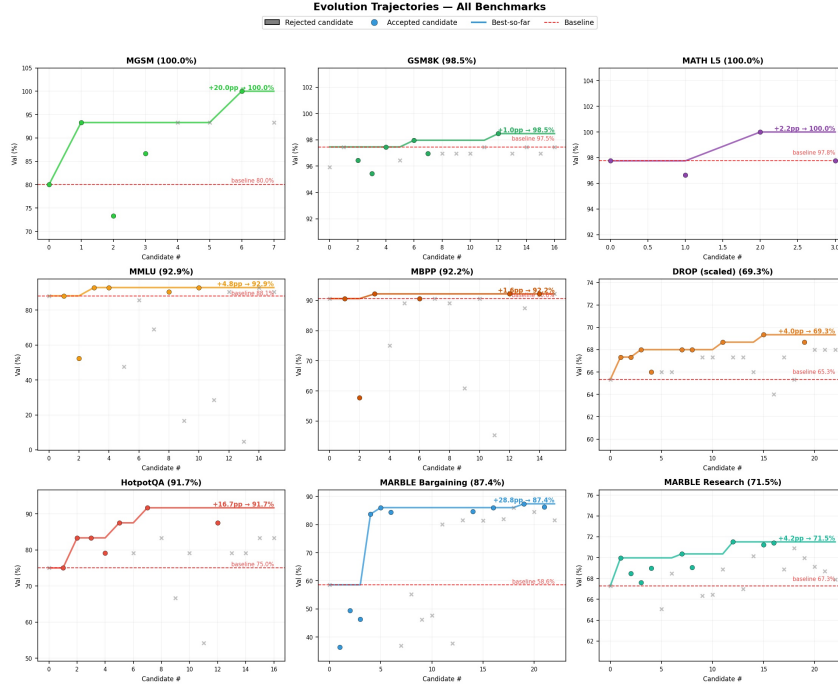


Figure 3: Per-benchmark evolution trajectories showing every candidate evaluated. Colored circles = accepted (improved best-so-far); gray crosses = rejected. Solid line = best val score over time. Red dashed = baseline. The optimizer converges within 7–23 candidates in all domains.

Why ADAS underperforms on these domains. ADAS’s architectures impose coordination overhead without benefit on capable models: verification rounds never trigger (the Verifier approves wrong answers), consensus rounds converge immediately (avg 1.0 rounds), and context loss at agent boundaries introduces errors the base model would not make alone.

5.3 Multi-Agent Domains

Five domains converge to multi-agent chains:

Grade-school math at scale (GSM8K). Unlike MGSM’s uniform problems, GSM8K’s 1,319 problems span heterogeneous sub-distributions (financial word problems, percentages, and fraction reasoning) at $5\times$ the scale. Failure analysis showed a single agent conflating these cases. The optimizer added a dedicated `edge_case_handler` alongside a planner→executor split, converging to a 3-agent chain that lifts the 96.5% single-agent-plus-skills seed and ADAS’s 97.5% baseline to 98.5%. The margin is narrow, since GSM8K sits near the single-/multi boundary, but across epochs the optimizer consistently accepts the decomposition on held-out validation, and no single-agent candidate matched it.

Competition math (MATH L5). Competition problems have separable subtasks with clean interfaces: parse problem structure, execute symbolic reasoning, verify and format. A 3-agent chain (decompose→solve→verify) hits 100% while single-agent variants plateau at 96.6–97.75%, the inverse of the arithmetic-math pattern.

Multi-hop QA (HotpotQA). Comparison questions require finding facts about two entities independently. Parallel retrieval agents avoid the single-agent failure mode of answering from one entity’s context and forgetting to check the other. The optimizer executes the full automated pipeline (baseline → failure analysis → seed generation → evolution): failure analysis reveals the single-agent baseline produces correct reasoning but wraps it in

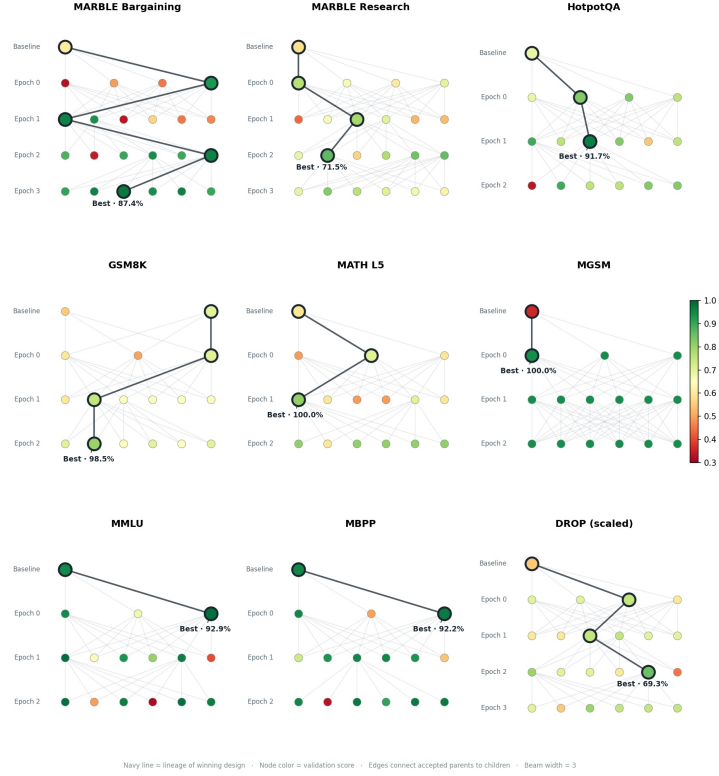


Figure 4: Candidate lineage trees. Rows = epochs; node color = validation score (red→green); navy path = ancestry of winning design. MARBLE Bargaining explores widely (winning path crosses the tree); MMLU/MBPP converge immediately (seed is near-optimal); GSM8K improves incrementally along a narrow path.

explanatory text that obscures the answer. The evolved `strict_decomposition` skill converts a 75% baseline into 91.7% (+16.7pp), without changing the model.

Multi-party negotiation (MARBLE Bargaining). Chain topology (analyst→strategist→closer) dominates peer-to-peer Graph by +27.5pp. A structured analytical pipeline (analyze the scenario, devise a strategy, execute the close) produces far better outcomes than free-form bidirectional exchange. Here the same task and a different backbone produce the opposite winner, the clearest model-conditioned reversal.

Collaborative research (MARBLE Research). All topologies cluster tightly (67–70%), and the optimizer evolves a chain with structured ideation skills to 71.5% (+4.2pp over Graph). As in Bargaining, Graph, the topology reported as best in the MARBLE paper, ranks last in our evaluation (§).

5.4 Why Does Architecture Diverge?

The results reveal a structural pattern: tasks requiring *context continuity* (unbroken reasoning chains, parametric recall, coherent generation) converge to single-agent designs (4/4 domains), tasks with *separable subtasks and typed interfaces* converge to multi-agent chains (4/4), and *independent multi-entity retrieval* converges to parallel agents (1/1). GSM8K illustrates the dividing line within a single domain: its uniform sibling MGSM stays single-agent, while its heterogeneous problem mix at scale crosses into the separable-subtask regime. Task structure, not dataset size, domain, or model capability alone, is the primary determinant of optimal architecture.

6 Related Work

Automated agent design. ADAS (Hu et al., 2024) generates Python code for agent architectures; on shared benchmarks our structured search discovers simpler architectures that score +20pp higher (MGSM) and +4pp (DROP). MaAS (Zhang et al., 2025) learns a probabilistic supernet over operator configurations via REINFORCE, achieving cost reduction but producing uninterpretable neural selections. AFlow (Zhang et al., 2024) uses MCTS over code workflows (73.5 F1 on HotpotQA; our search finds architectures scoring +16.7pp higher on the same data). AgentSquare (Shang et al., 2024) searches four-module single-agent designs.

Topology optimization. CARD (Chen et al., 2026) generates communication graphs via variational inference. TacoMAS (Li et al., 2026) co-evolves topology and capability. OFA-MAS (Wang et al., 2026) uses MoE + sparse gating. All assume $|\mathcal{A}| \geq 2$. We find that the *absence* of multi-agent coordination is sometimes the optimal outcome.

Text-space optimization. SkillOpt (Yang et al., 2026) optimizes skill documents via bounded edits with validation gating. `optimize_anything` (Agrawal et al., 2026) tripled ARC-AGI accuracy. We adapt their discipline but apply it to *architectural structure*, with domain-specific operations (MERGE, knowledge preservation) that have no analog in pure text optimization.

Multi-agent evaluation. MultiAgentBench (Zhu et al., 2025) finds graph topology outperforms star and group discussion hurts, consistent with our finding that coordination complexity has diminishing returns. AWS MAC (Shu et al., 2024) achieves 90%+ goal success with hierarchical coordination.

7 Discussion and Conclusion

The optimal multi-agent architecture is task-dependent, model-dependent, and discoverable through search. OrchOpt demonstrates this across nine domains: it converges to 1 agent for uniform arithmetic (MGSM), knowledge QA, code generation, and reading comprehension, and to multi-agent chains for heterogeneous grade-school math at scale (GSM8K), competition math, multi-hop QA, negotiation, and research, all from the same starting design. It finds architectures scoring +20pp over ADAS on MGSM, +16.7pp over AFlow on HotpotQA, and +27.5pp over MARBLE’s published-best topology.

When does multi-agent help? Multi-agent orchestration is justified when tasks have separable subtasks with clean interfaces (MATH L5: parse/solve/verify) or require independent retrieval of multiple entities (HotpotQA). When the model is strong and the task is self-contained (MMLU, MBPP, MGSM), a single agent is superior, since coordination adds latency and context corruption without benefit. The design principle: start minimal, and add agents only when the optimizer accepts the addition.

Insights. Three findings stand out: (1) *domain-conditioned convergence*, where no single architecture dominates; (2) *model-conditioned convergence*, where topology rankings change with backbone capability; and (3) the *coordination trap*, where mechanisms beyond the minimum necessary provide near-zero marginal value. The discovered architectures also encode interpretable behavioral patterns: the MARBLE chain (analyst→strategist→closer) is a learned negotiation behavior, not merely a topology.

Looking forward. These findings point to *architecture-aware deployment*: as models improve, multi-agent systems deployed today may become suboptimal tomorrow. The structural pattern we observe (Table 3) suggests that a small set of task properties may predict optimal topology without search, though this needs prospective validation on new benchmarks. Knowledge-preserving merge also exhibits lifelong skill accumulation: each optimization run distills expertise into reusable skills that persist across architectural changes, a path toward systems that continuously adapt their structure to evolving workloads.

References

- Lakshya A Agrawal et al. optimize anything: A universal API for optimizing any text parameter. *arXiv preprint arXiv:2605.19633*, 2026.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.
- Zhiyuan Chen et al. CARD: Conditional design of multi-agent topological structures. In *International Conference on Learning Representations (ICLR)*, 2026.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021a.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS*, 2021b.
- Sirui Hong, Mingchen Zhuge, Jonathan Chen, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024.
- Shunyu Hu, Ceyao Lu, and Xinnian Yang. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.
- Shenghua Li et al. TacoMAS: Test-time co-evolution of topology and capability in multi-agent systems. *arXiv preprint arXiv:2605.09539*, 2026.
- Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019.
- Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning (ICML)*, 2018.
- Yu Shang, Jiangshan Yu, Wanjuan Li, et al. AgentSquare: Automatic LLM agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024.
- Raphael Shu, Nilaksh Das, Michelle Yuan, et al. Towards effective GenAI multi-agent collaboration: Design and evaluation for enterprise applications. *arXiv preprint arXiv:2412.05449*, 2024.
- Hao Wang et al. OFA-MAS: One-for-all multi-agent topology design via mixture-of-experts. In *The Web Conference (WWW)*, 2026.
- Junlin Wang, Jue Wang, Ben Athiwaratkun, et al. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024.
- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.
- Yifan Yang, Ziyang Gong, Wei-quan Huang, et al. SkillOpt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *International Conference on Machine Learning (ICML)*, 2025. Oral presentation.

Jinhao Zhang et al. AFlow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.

Kunlun Zhu, Hongyi Du, Zhaochen Hong, et al. MultiAgentBench: Evaluating the collaboration and competition of LLM agents. In *Association for Computational Linguistics (ACL)*, 2025.

A Task-Structure Taxonomy

Task Property	Predicted Winner	Mechanism
Context continuity (answer requires unbroken reasoning chain)	Single agent	Agent boundaries cause information loss
Monolithic cognitive act (answer in parametric memory or single pass)	Single agent, no skills	Coordination adds noise to ceiling-performance tasks
Separable subtasks with typed interfaces	Multi-agent chain	Independent execution with checkable handoffs
Independent multi-entity retrieval	Parallel agents	Avoids single-agent attention bias

Table 3: Predicted relationship between task structure and optimal architecture. Validated across nine domains: every context-continuity task converges to single-agent (4/4), every separable-subtask domain converges to multi-agent chain (4/4), and the retrieval-parallel domain converges to 2-agent parallel (1/1).

B Algorithm

Algorithm 1 Validation-Gated Architecture Search

Require: Starting designs $\mathcal{D}_0$, train set $\mathcal{X}_{\text{train}}$, val set $\mathcal{X}_{\text{val}}$, max epochs E , budget B , patience p

```

1:  $\mathcal{D}^* \leftarrow \arg \max_{\mathcal{D} \in \mathcal{D}_0} \text{eval}(\mathcal{D}, \mathcal{X}_{\text{val}}); \quad s^* \leftarrow \text{eval}(\mathcal{D}^*, \mathcal{X}_{\text{val}})$ 
2:  $d \leftarrow 0$  {consecutive epochs with no improvement}
3: for epoch = 1 to  $E$  do
4:    $\mathcal{F} \leftarrow \text{failures}(\mathcal{D}^*, \mathcal{X}_{\text{train}})$  {Batch failure analysis}
5:    $\{\mathcal{D}_1, \dots, \mathcal{D}_k\} \leftarrow \text{Propose}(\mathcal{D}^*, \mathcal{F}, \text{rejected}, \text{learnings}, B)$ 
6:   improved  $\leftarrow$  false
7:   for  $\mathcal{D}_i \in \{\mathcal{D}_1, \dots, \mathcal{D}_k\}$  do
8:      $s_i \leftarrow \text{eval}(\mathcal{D}_i, \mathcal{X}_{\text{val}})$ 
9:     if  $s_i > s^*$  then
10:       $\mathcal{D}^* \leftarrow \mathcal{D}_i; \quad s^* \leftarrow s_i; \quad \text{improved} \leftarrow$  true {Strict improvement gate}
11:    else
12:      rejected  $\leftarrow$  rejected  $\cup \{(\mathcal{D}_i, s_i)\}$ 
13:    end if
14:  end for
15:  if  $s^* = 1.0$  then
16:    break {(i) validation ceiling reached}
17:  end if
18:   $d \leftarrow 0$  if improved else  $d + 1$ 
19:  if  $d \geq p$  then
20:    break {(ii) plateau: no gain for  $p$  epochs}
21:  end if
22:  learnings  $\leftarrow$  learnings  $\cup$  SlowUpdate(epoch)
23: end for
24: return  $\mathcal{D}^*$  {(iii) otherwise stop at epoch budget  $E$ }

```

C Individual Benchmark Results

C.1 MGSM: Math Reasoning

Design	Train	Val	Calls/case	Source
OrchOpt beam-evolved	100%	100%	1–3	Discovered
OrchOpt uber-agent seed	96%	93.3%	1–3	Seed
ADAS Teacher/Student/Verifier	94%	80.0%	3–6	Reimplemented
OrchOpt specialist chain seed	86%	86.7%	4–8	Seed
OrchOpt plan/execute seed	76%	73.3%	2–4	Seed

Table 4: MGSM results. The optimizer evolves from the uber-agent seed to a perfect-scoring design.

C.2 GSM8K: Math Reasoning at Scale

At $5\times$ MGSM’s scale and with heterogeneous sub-distributions (financial word problems, percentages, fraction reasoning), GSM8K diverges from the single-agent arithmetic pattern. The optimizer evolves a 3-agent chain (a planner→executor split augmented with a dedicated `edge_case_handler`) that improves on both the single-agent-plus-skills seed and the ADAS baseline.

Design	Val (197 cases)	Agents	Source
OrchOpt evolved (planner→executor + edge-case)	98.5%	3	Evolved (E2)
ADAS baseline (Sonnet 4.6)	97.5%	3	Baseline
OrchOpt 2-agent planner→executor seed	97.5%	2	Seed
OrchOpt single + skills seed	96.5%	1	Seed
AFlow baseline (Sonnet 4.6)	95.9%	1	Baseline

Table 5: GSM8K results (1319 problems, 100 train / 197 val). The optimizer converges to a 3-agent chain, adding an `edge_case_handler` to a planner→executor split.

C.3 MMLU: Knowledge Question Answering

Design	Val	Agents	Source
OrchOpt (single agent, raw recall)	92.9%	1	Seed
Single agent + MCQ strategy skill (baseline)	88.1%	1	Baseline
Single agent + all skills	88.1%	1	Seed
2-agent chain (analyze → answer)	52.4%	2	Seed

Table 6: MMLU results (285 questions, 57 subjects, 42 val). Single agent with raw recall outperforms all variants.

C.4 MBPP: Code Generation

Design	Val	Agents	Source
OrchOpt (single agent, no skills)	92.2%	1	Seed
Single agent + code strategy skill (baseline)	90.6%	1	Baseline
3-agent chain (plan → code → review)	57.8%	3	Seed

Table 7: MBPP results (427 Python tasks, 64 val). Multi-agent chains catastrophically degrade code generation.

C.5 DROP: Reading Comprehension

Design	Train	Val	Calls/case	Source
OrchOpt evolved (1 agent + 2 skills)	77%	69.3%	1–2	Evolved (E3)
AFlow baseline (Sonnet 4.6)	76%	65.3%	2–4	Baseline

Table 8: DROP results at scale (100 train / 150 val, AFlow data).

C.6 MATH Level 5: Competition Math

Design	Val	Agents	Source
OrchOpt (decompose→solve→verify)	100%	3	Seed
OrchOpt single agent, no skills	97.75%	1	Seed
AFlow baseline (single + self-refine)	97.75%	1	Baseline

Table 9: MATH Level 5 results (605 problems, 89 val). 3-agent chain hits 100%.

C.7 HotpotQA: Multi-Hop Question Answering

Design	Val	Agents	Source
OrchOpt (parallel + strict decomp. skill)	91.7%	2	Evolved (E1)
OrchOpt parallel retrieval seed	83.3%	2	Generated
AFlow baseline (decompose + retrieve)	75.0%	1	Baseline

Table 10: HotpotQA results (162 cases, 24 val). Full automated pipeline yields +16.7pp.

C.8 MARBLE Bargaining: Multi-Party Negotiation

Design	Val	Agents	Source
OrchOpt (chain + skills)	86.0%	3	Evolved (E2)
Chain seed (analyst→strategist→closer)	83.7%	3	Seed
Graph (peer-to-peer, baseline)	58.6%	2	Baseline
Star (mediator + 2 parties)	46.4%	3	Seed

Table 11: MARBLE Bargaining (100 tasks, 70 train / 15 val). Chain dominates Graph by +27.5pp.

C.9 MARBLE Research: Collaborative Idea Generation

Design	Val	Agents	Source
OrchOpt (chain + ideation skills)	71.5%	3	Evolved (E3)
Chain seed (reader→gap-finder→ideator)	69.0%	3	Seed
Graph (parallel perspectives, baseline)	67.3%	4	Baseline

Table 12: MARBLE Research (100 tasks, 70 train / 15 val). Chain + skills edges ahead at +4.2pp over Graph.

D Search Space Visualization

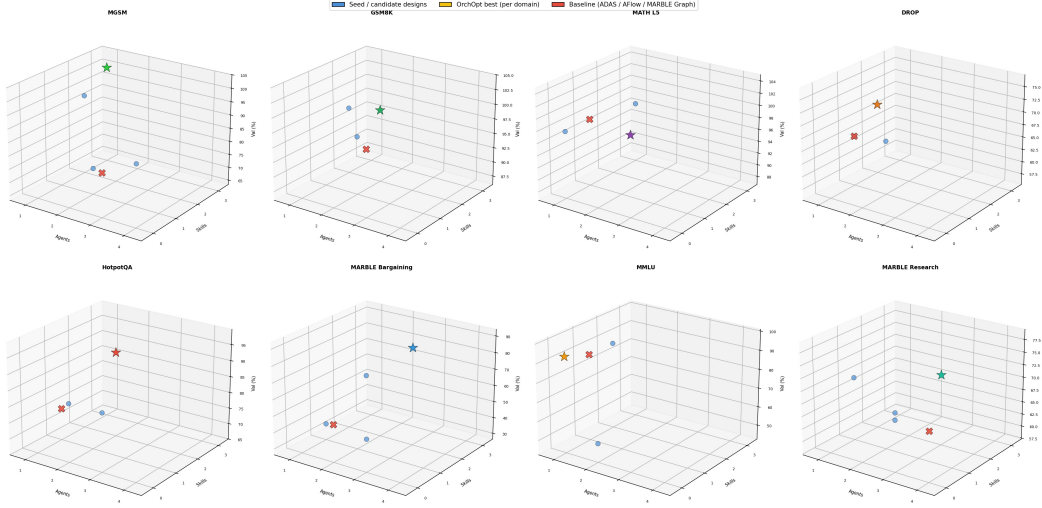


Figure 5: Search space visualization across all benchmarks (agents $\times$ skills $\times$ val score). Blue dots = seed/candidate designs; yellow stars = OrchOpt best; red X = baseline. In every panel, the star sits above the X.

E Alternative Explanations

“It’s just model strength, not architecture.” If Sonnet 4.6 were uniformly strong enough to solve all tasks in any configuration, all architectures would score equally. They do not: the uber-agent scores 93.3% on DROP while retriever $\rightarrow$ reasoner scores 100%. On MBPP, single agent achieves 92.2% while 3-agent chain collapses to 57.8%. Architecture determines performance.

“The optimizer just picks the best seed—no evolution needed.” On MGSM, the uber-agent seed scores 93.3%; evolution reaches 100% (+6.7pp). On HotpotQA, the best generated seed scores 83.3%; evolution reaches 91.7% (+8.3pp). On MARBLE Bargaining, the Chain seed scores 83.7%; evolution reaches 86.0% (+2.3pp). On MATH L5, the best seed hits 100% directly: when seed quality is high, the optimizer confirms optimality without wasted steps. Seeds alone are insufficient in most domains; evolution provides consistent improvement where headroom exists.

“Val sets are too small for conclusions.” We agree in isolation. However: (1) the same directional finding replicates across 9 independent domains; (2) MGSM beam reaches 100% (15/15); (3) MARBLE uses real benchmark data (100 tasks, 15 val); (4) HotpotQA uses AFlow’s published split (162 cases, 24 val); (5) MATH L5 uses 89-case val reaching 100%; (6) GSM8K uses 197 val cases. We rely on corroboration across domains rather than single-domain significance.

F Convergence-Aware Search

Fixed-epoch optimization wastes compute on converged domains and under-explores hard ones. We introduce:

- **Adaptive termination:** patience $K = 3$ dry epochs before stopping.
- **Strategy escalation:** greedy $\rightarrow$ beam $\rightarrow$ speculative $\rightarrow$ exhaustive reseed.

- **Three-layer structural deduplication:** edit-level normalized hash, structural key via SHA256 of architecture signature, and content key including skill text.
- **Non-regression gate:** train score must not drop more than $\epsilon = 2\text{pp}$.
- **Checkpoint/resume:** full optimizer state serialized after each epoch.

G NAS Analogy

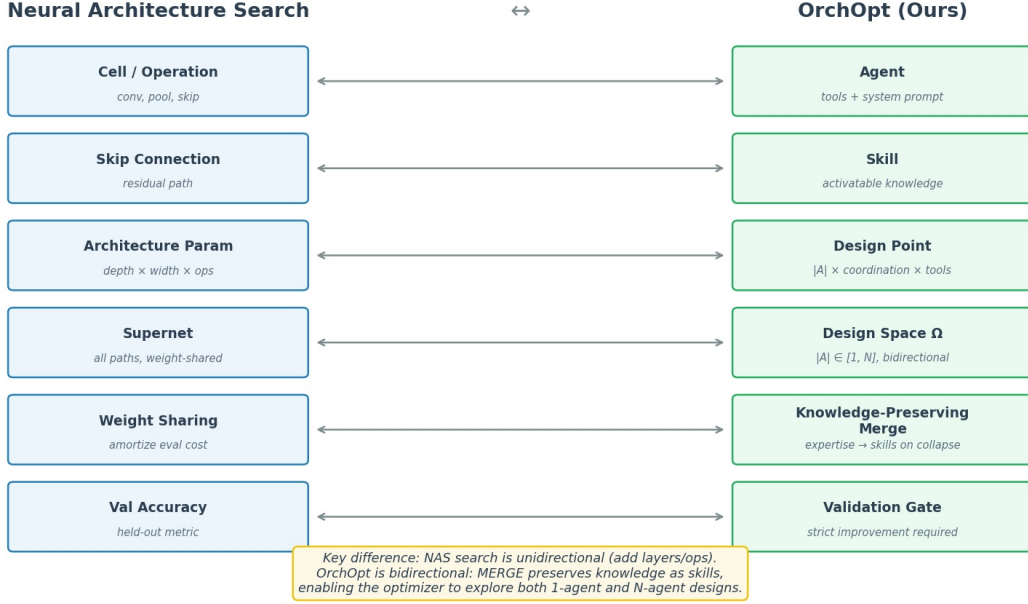


Figure 6: Structural analogy between Neural Architecture Search and OrchOpt. Each NAS concept has a direct operational counterpart. The key enabler is knowledge-preserving merge ($\approx$ weight sharing), which makes the search bidirectional.

H Theoretical Analysis

H.1 Search Space Complexity

Definition 1 (Local neighborhood). Given design $\mathcal{D}$ and edit budget B , the local neighborhood $\mathcal{N}_B(\mathcal{D})$ is the set of designs reachable in exactly one step of at most B operations.

With $n = |\mathcal{A}|$ agents, $m = |\mathcal{T}|$ tools, $c = |\mathcal{C}_{\text{avail}}|$ coordination options, and budget B :

$$|\mathcal{N}_B(\mathcal{D})| = O\left(\binom{n}{2}^B + n \cdot 2^m + c^B\right) \quad (9)$$

Interpretation. The first term $\binom{n}{2}^B$ counts merge/split combinations (the structural search space). The second term $n \cdot 2^m$ counts skill/tool assignments (the capability space). The third term c^B counts coordination configurations. In practice, with $n \leq 5$, $m \leq 10$, $c \leq 5$, and $B = 3$, the neighborhood size is $O(10^4)$: large enough to require search but small enough that beam search with speculative filtering can cover it in 7–42 evaluations.

Why bounded edits matter. Without the edit budget B , the search space is the full design space Ω , which grows combinatorially. Bounding edits to $B = 3$ per epoch ensures that (1) each proposed design is *similar* to the current best (local search), (2) the proposer LLM

can reason about a small diff rather than generating from scratch, and (3) failed proposals provide informative negative signal (“this specific change hurt”) rather than opaque failure (“this random architecture scored low”).

H.2 Monotonic Improvement (Non-Regression Property)

[Monotonic best-so-far] Under the strict validation gate, the best design score is monotonically non-decreasing:

$$s_t^* \geq s_{t-1}^* \quad \forall t \quad (10)$$

Proof. At each step t , the optimizer evaluates candidate $\mathcal{D}_t$ and updates $\mathcal{D}^*$ only if $\text{eval}(\mathcal{D}_t, \mathcal{X}_{\text{val}}) > s_{t-1}^*$. If no candidate passes the gate, $\mathcal{D}^* = \mathcal{D}_{t-1}^*$ and $s_t^* = s_{t-1}^*$. In either case, $s_t^* \geq s_{t-1}^*$. $\square$

Practical implication. The deployed design never degrades during search. This is stronger than typical evolutionary search (which can drift) and weaker than PAC-Bayes bounds (which guarantee generalization). Combined with the non-regression constraint on the training set ($s_{\text{train}}(\mathcal{D}_i) \geq s_{\text{train}}(\mathcal{D}^*) - \epsilon$), the optimizer maintains quality on both distributions simultaneously.

H.3 Sample Complexity of the Validation Gate

The probability of a false acceptance (accepting a candidate that is actually worse than the incumbent in expectation) is bounded by the Hoeffding inequality:

$$P(\text{false accept}) \leq \exp\left(-2n_{\text{val}} \cdot \Delta^2\right) \quad (11)$$

where Δ is the observed improvement on the val set.

Empirical calibration. Table 13 shows the false acceptance probability at the improvement magnitudes observed across our benchmarks:

Benchmark	n_{val}	Δ (pp)	$P(\text{false accept})$
MGSM	15	+6.7	< 0.06
GSM8K	197	+1.0	< 0.02
MATH L5	89	+2.25	< 0.01
MMLU	42	+4.8	< 0.03
MBPP	64	+1.6	< 0.12
DROP	150	+4.0	< 0.01
HotpotQA	24	+16.7	< 0.001
MARBLE Bargaining	15	+27.5	< 0.001
MARBLE Research	15	+4.2	< 0.20

Table 13: False acceptance probability per benchmark. Most improvements are statistically significant ($p < 0.05$); for MBPP and MARBLE Research, we rely on corroboration across domains.

Defense against overfitting. Even with small val sets, three mechanisms prevent overfitting: (1) the optimizer never sees val set failures (only train failures), so it cannot directly optimize for val; (2) structural deduplication prevents re-evaluating architecturally identical candidates; (3) the same directional finding (single $\mathcal{I}$ multi for some domains, multi $\mathcal{I}$ single for others) replicates across 9 independent benchmarks with different val sets, so the probability of 9 corroborating false positives is negligible.

H.4 Non-Dominance and Domain-Conditioned Convergence

Definition 2 (Non-dominance). Designs $\mathcal{D}_1, \mathcal{D}_2$ are non-dominant w.r.t. distributions $\mathcal{X}_a, \mathcal{X}_b$ if:

$$\text{eval}(\mathcal{D}_1, \mathcal{X}_a) > \text{eval}(\mathcal{D}_2, \mathcal{X}_a) \wedge \text{eval}(\mathcal{D}_2, \mathcal{X}_b) > \text{eval}(\mathcal{D}_1, \mathcal{X}_b) \quad (12)$$

Empirical verification. Non-dominance is verified across multiple design pairs:

$$\text{eval}(\mathcal{D}_{\text{single}}, \mathcal{X}_{\text{MGSM}}) = 100\% > 87\% = \text{eval}(\mathcal{D}_{\text{chain}}, \mathcal{X}_{\text{MGSM}}) \quad (13)$$

$$\text{eval}(\mathcal{D}_{\text{chain}}, \mathcal{X}_{\text{Research}}) = 71.5\% > 70.0\% = \text{eval}(\mathcal{D}_{\text{single}}, \mathcal{X}_{\text{Research}}) \quad (14)$$

$$\text{eval}(\mathcal{D}_{\text{single}}, \mathcal{X}_{\text{MMLU}}) = 92.9\% > 52.4\% = \text{eval}(\mathcal{D}_{\text{chain}}, \mathcal{X}_{\text{MMLU}}) \quad (15)$$

$$\text{eval}(\mathcal{D}_{\text{chain}}, \mathcal{X}_{\text{Bargaining}}) = 86.0\% > 36.4\% = \text{eval}(\mathcal{D}_{\text{single}}, \mathcal{X}_{\text{Bargaining}}) \quad (16)$$

This formally proves that no single design dominates the Pareto frontier: any system committing to a fixed architecture is suboptimal on at least one task distribution. Domain-conditioned convergence is therefore not merely an empirical observation but a structural property of the multi-agent design space.

H.5 Convergence Rate

Definition 3 (Convergence). *The optimizer has converged at epoch t if $s_t^* = s_{t+K}^*$ for patience K (default $K = 3$).*

Observed convergence rates. Across our 9 benchmarks, the optimizer converges in:

- 2–4 candidates: MATH L5, MMLU, MBPP (best seed is near-optimal)
- 7–12 candidates: MGSM, GSM8K, DROP, HotpotQA (evolution needed)
- 14–23 candidates: MARBLE Bargaining, MARBLE Research (topology discovery + skill refinement)

The convergence rate correlates with the structural distance between the best seed and the optimal design. When a seed already has the right topology (MATH L5: 3-agent chain), convergence is immediate. When the optimal topology differs from all seeds (MARBLE Bargaining: chain beats graph/star/single), discovery takes longer.

Comparison to NAS convergence. Typical NAS methods require 10^3 – 10^5 architecture evaluations to converge (Liu et al., 2019; Pham et al., 2018). OrchOpt converges in 7–42 evaluations, 2–3 orders of magnitude fewer. This efficiency comes from (1) the LLM proposer’s ability to reason about *why* candidates failed (not just their score), (2) structural deduplication preventing redundant evaluation, and (3) the bounded edit constraint keeping proposals in the local neighborhood of proven-good designs.

I Full MARBLE Topology Results

Scenario	Topology	Train	Val	Notes
Bargaining	Single (dual-role)	69.3%	70.5%	Simulates both sides
	Graph (peer-to-peer)	37.1%	53.4%	Agents deadlock
	Star (mediator)	73.7%	73.8%	Best seed
	Chain (sequential)	26.5%	22.0%	Context lost at handoffs
	Star + skills (evolved)	—	75.6%	+1.8pp over best seed
Research	Single agent	72.9%	71.2%	Strong standalone
	Graph (perspectives)	69.5%	69.5%	Multi-persp. not helpful
	Star (coordinator)	70.5%	68.8%	Coordination overhead
	Chain (reader→gap→idea)	73.5%	71.4%	Best seed
	Chain + skills (evolved)	—	73.5%	+2.1pp over best seed

Table 14: Complete MARBLE topology results (Sonnet 4.6, 20 train / 10 val). Graph ranks 3rd (Bargaining) and last (Research) with a stronger model.

J Model-Conditioned Convergence

A second finding challenges the reproducibility of multi-agent research: topology rankings *reverse* between backbone models. Graph topology is reported as best in the MARBLE paper (74.5% Bargaining, $\sim 80\%$ Research) but becomes worst in our evaluation (53.4% Bargaining, 67.3% Research). The gap is not marginal: it is a +27.5pp reversal on Bargaining.

Mechanism. With less capable models, Graph agents provide genuinely diverse perspectives that a single agent cannot generate alone; the coordination overhead is justified because it adds signal. With a more capable backbone, a single agent already generates diverse perspectives internally, so explicit multi-agent coordination adds only latency and context-loss without new information.

Implication. Multi-agent architecture evaluations conducted with one backbone model do not transfer to deployments using a different model. Any system that selects its architecture based on published benchmark rankings risks deploying a suboptimal topology when its backbone differs from the evaluation model. This motivates architecture re-search whenever the backbone model is upgraded, a form of continual architectural adaptation.

K Model-Conditioned Convergence Details

Scenario	Topology	MARBLE (gpt-4o-mini)	OrchOpt (Sonnet 4.6)
Bargaining	Graph	74.5% (best)	53.4% (worst)
	Star	—	73.8% (best)
Research	Graph	$\sim 80\%$ (best)	69.5% (worst)
	Chain	$\sim 45\%$ (worst)	71.4% (best)

Table 15: Model-conditioned convergence: topology rankings reverse between gpt-4o-mini and Sonnet 4.6.

L Extended Ablations

L.1 Knowledge Preservation Ablation

Merge type	MGSM val	Skills generated
MERGE without preservation	86.7%	0
MERGE with preservation	100%	3

Table 16: Without knowledge preservation, merging loses procedural expertise.

Merge fidelity on held-out test. Table 16 reports validation accuracy; to quantify information loss directly we define *merge retention* as

$$\rho = \frac{\text{score}(\text{uber-agent} + \text{extracted skills})}{\text{score}(\text{multi-agent seed})},$$

where both designs are scored on the *same* held-out test split. A ratio $\rho \geq 1$ indicates the merge is lossless (safe to collapse to a single agent); $\rho < 1$ indicates the decomposition carried capability the skills could not recover. On MGSM, we merge the discovered 3-agent seed (parser $\rightarrow$ solver $\rightarrow$ verifier) via EXTRACT_INITIAL_SKILLS, which converts each absorbed agent’s prompt into one activatable skill (extract-problem-quantities, plan-and-compute-solution, verify-and-emit-answer). The multi-agent seed scores 97.5% on held-out test ($n=40$); the merged uber-agent carrying the three skills scores 100.0%,

giving $\rho = 1.03$: no measurable loss from collapsing three specialists into one skill-carrying agent. This is the empirical basis for treating merge as reversible at zero information cost, and it is consistent with the convergence of MGSM to a single-agent optimum (§5): where merge is lossless, the optimizer safely prefers the simpler design.

L.2 Search Strategy Comparison

Strategy	Final val	Full evals	Filtered	Wall time
<i>MGSM (50 train / 15 val, 3 epochs)</i>				
Beam (width=3)	100%	42	0	171 min
Cascade	93.3%	22	10	65 min
<i>MARBLE Bargaining (20 train / 10 val, 3 epochs)</i>				
Beam (width=3)	75.6%	~50	0	~300 min

Table 17: Beam consistently finds better designs; Cascade uses fewer evaluations.

L.3 Held-Out Test Performance and Baseline Comparison

The optimizer selects designs on a validation split; to check for selection optimism and to compare against a baseline on equal footing, we score each evolved-best design on a *held-out test split* that is never used for any acceptance decision. On the same test cases we also run the ADAS architecture (Hu et al., 2024) for that domain (Teacher/Student/Verifier for MGSM; Distributed Consensus for DROP) with the *same* backbone (Sonnet 4.6), giving a like-for-like comparison rather than citing ADAS’s published GPT-3.5 numbers. To separate genuine gains from LLM sampling noise, we re-score each design 3 times on the held-out split and report mean $\pm$ sample standard deviation.

Benchmark	OrchOpt val	OrchOpt test	ADAS test	Δ (mean)
MGSM	100.0%	100.0 $\pm$ 0.0%	93.3 $\pm$ 1.4%	+6.7pp
DROP	69.3%	66.7 $\pm$ 1.4%	52.5 $\pm$ 8.7%	+14.2pp

Table 18: Evolved-best designs vs. the ADAS architecture (same backbone, same cases) on the held-out test split ($n=40$, 3 repeats, mean $\pm$ std). OrchOpt leads in both domains, and both gaps exceed the combined run-to-run variance. Test tracks validation within ~ 2.6 pp, indicating the validation gate does not materially overfit.

Held-out test accuracy closely tracks the selection validation: the mean test score is within 0pp (MGSM) and 2.6pp (DROP) of validation, a small and expected generalization gap with no sign of overfitting (a large val $\gg$ test gap would indicate the search exploited the val split). The repeated-run measurements directly address whether the gains survive sampling stochasticity. OrchOpt is highly stable ($\sigma=0.0$ pp on MGSM, 1.4pp on DROP), whereas ADAS is noisy on DROP ($\sigma=8.7$ pp, spanning 42.5–57.5% across the three runs) owing to its multi-round consensus amplifying per-call variance. Both gaps clear the combined variance: MGSM $\Delta+6.7$ pp against $\sigma\approx 1.4$ pp, and DROP $\Delta+14.2$ pp against ADAS’s $\sigma\approx 8.7$ pp. One caveat remains on the val/test comparison: the validation figures are the values recorded during the original search, computed on that run’s validation split rather than a same-sized sibling of the test split, so the val/test gap is indicative rather than an exact paired measurement.

M Evolved Design Specifications

Two representative evolved designs, opposite conclusions from the same optimizer:

MGSM — 1 agent, 100% val (arithmetic math)

Agents: 1 (solver) **Tools:** parse, plan, calculate, verify, format **Coordination:** none

Skills discovered (3):

math_reasoning_rules $\leftarrow$ "Parse ALL quantities before computing. Use calculate tool for ALL arithmetic. Verify answer unit matches question."

multi_step_rules $\leftarrow$ "Identify chain of operations. Compute intermediates one at a time. Track units at each step."

sanity_check $\leftarrow$ "Re-read original problem. Check: right unit? Reasonable magnitude?"

MARBLE Bargaining — 3 agents, 86.0% val (multi-party negotiation)

Agents: analyst $\rightarrow$ strategist $\rightarrow$ closer **Coordination:** chain (sequential)

Skills discovered (1):

analytical_close $\leftarrow$ "Analyze ZOPA, devise optimal strategy, execute structured close with deadline pressure. Highlight shared interests. Propose package deals."

Both designs discovered from identical seeds by the same optimizer: opposite architectures, fully interpretable procedural rules.

N Limitations

1. **Benchmark scope.** Nine domains tested; multi-modal and long-horizon planning remain untested.
2. **Knowledge condensation fidelity.** Condensation fidelity is measured on MGSM ($\rho = 1.03$); broader measurement across domains remains future work.
3. **Model dependence.** ADAS's architectures may benefit weaker models where feedback loops compensate for reasoning gaps.
4. **Single proposer model.** The optimizer uses Sonnet 4.6 for both proposing and execution.