Mycelial Council Network: Temperature Dominates Routing in Multi-Agent LLM Code Synthesis — A Longitudinal Study with Category-Level Analysis

MCN Research Initiative

Anonymous Submission — February 2026

Abstract—We empirically study whether a learned router can improve upon the best single-agent baseline when a council of LLM agents shares the same base model. Over ten experimental runs, three controlled ablations, two 2,000-task longitudinal experiments, and a temperature-heterogeneous Phase 2 trial using Qwen2.5-Coder-7B-Instruct-AWQ on an 8-category Python benchmark, we find a consistent null result: neither router sophistication nor temperature diversity improves performance. A single agent at T=0.3 achieves 91%; MCN-LinUCB scores 86% (−5 pp). At 2,000 tasks LinUCB achieves 60.7% and a GNN router — with 40x more parameters — achieves 60.6% (delta = −0.1 pp). Phase 2 introduces heterogeneous tribe temperatures (T0=0.1, T1=0.5, T2=0.9) with a live Category Thompson Sampling router: overall pass rate is 60.7%, identical to Phase 1C, with per-tribe rates within 2 pp (T0=60.4%, T1=59.3%, T2=61.3%). The oracle gap narrows from 5.1 pp to 3.7 pp. CTS learns per-category routing preferences (oscillating T1→T0→T2 drift) but cannot improve aggregate performance because temperature diversity does not create capability diversity. Category-level heterogeneity dominates throughout: string 99%, graph 0%, corrected pass rate excluding graph = 81.3%. We conclude that routing value requires genuine inter-model performance variance, not sampling or algorithmic sophistication.

Index Terms—multi-agent systems, LLM code synthesis, contextual bandits, graph neural networks, temperature scaling, routing collapse, ablation study, category-level analysis, oracle gap decomposition

I. INTRODUCTION

Automated code synthesis with large language models (LLMs) has advanced rapidly, yet most deployments use a single model with fixed inference parameters. The Mycelial Council Network (MCN) hypothesis is that a council of agents with distinct styles, routed by a learned policy, can outperform any single agent — analogously to how software teams benefit from diverse specialists.

This paper reports ten experimental runs and three controlled single-agent ablations designed to test this hypothesis. Experiments proceed through three phases:

Phase 1 (Runs 1–6): Infrastructure validation and GNN vs. LinUCB comparison on a 12-task benchmark. After correcting three infrastructure defects, the GNN router reached 98% and LinUCB reached 97%, but LinUCB collapsed to a single tribe (100/0/0 routing).

Phase 2 (Runs 7–8): Task-set expansion to 16 types, revealing a 14 pp pass-rate drop from the 12-task ceiling and identifying three new reference-solution bugs.

Phase 3 (Runs 9–10 + ablations): Controlled five-condition comparison — three single-agent ablations (T=0.1, T=0.3, T=0.9) and two MCN variants (LinUCB homogeneous, GNN heterogeneous).

The overall conclusion is that temperature selection matters more than routing strategy under these conditions, and routing with same-model tribes imposes a net exploration cost with no compensating specialisation benefit.

We make the following claims, strictly grounded in observed results: (1) A single agent at T=0.3 achieves 91%, outperforming all MCN variants. (2) MCN-LinUCB (homogeneous T=0.3) scores 86% — 5 pp below single-agent, with no statistically significant routing specialisation (chi-squared p=0.74). (3) MCN-GNN (heterogeneous T=0.1/0.5/0.9) scores 88% = T=0.9 single agent exactly. T_2 achieves 95% in isolation, but the router incurs a −7 pp cost relative to an oracle that always selects T_2. (4) Test-infrastructure correctness dominated measured outcomes in Phase 1 more than routing strategy did.

II. SYSTEM DESCRIPTION

MCN comprises five components operating as Ray remote actors inside a Docker Compose stack: vLLM inference server, Redis state store, ChromaDB vector store, MLflow tracking, and an mcn-runner container.

A. Tribes

Three tribe actors (T_0, T_1, T_2) share the same base model (Qwen2.5-Coder-7B-Instruct-AWQ) but receive distinct system prompts: "Reliable Coder," "Fast Coder," and "Creative Coder." No weight updates are performed. Phase 3 introduces heterogeneous temperatures: T_0=0.1 (deterministic), T_1=0.5 (balanced), T_2=0.9 (high-variance). This creates genuine output diversity: T_0 generates near-identical code across attempts; T_2 explores a broader algorithmic space at the cost of higher failure variance.

B. Router

Two routing implementations are compared.

LinUCB: Contextual bandit with 18-dim context. arm* = argmax [x^T theta_i + alpha * sqrt(x^T A_i^{-1} x)]. Epsilon-greedy warm-up (epsilon 0.5 -> 0.15, decay=0.995) prevents cold-start collapse. alpha=2.5. An epsilon-restoration bug in early runs caused saved epsilon (near floor) to override the configured warm-up on restart; fixed in Run 9.

GNN Router: Lightweight 2-layer MLP approximating GraphSAGE on a bipartite task-tribe graph. Input: [task_context(18), tribe_embedding(18)] = 36-dim concatenation. Three linear layers (36->32->16->1) produce per-tribe scores. Online Adam optimisation (lr=0.01), 64-entry experience replay, mini-batches of 8. ~2,000 parameters. CPU-only inference.

C. Context Vector

Each task maps to an 18-dim vector: task-type one-hot (dims 0–2), exception one-hot (dims 3–12), and z-scored execution metrics — runtime, tests_passed/failed, test_count, failure_density (dims 13–17). Computed from the previous failure on the same task and used as the bandit context for the next routing decision.

D. Overseer and Test Generation

The OverseerActor generates three test classes per task without seeing generated code: (A) Hypothesis property-based fuzzing — integers bounded to [–10k, 10k] (unbounded was a critical Phase 1 defect); (B) six adversarial static boundary tests including large inputs (10k elements) and large integers (10,000, not sys.maxsize — another Phase 1 defect); (C) mutation testing for test-suite sensitivity.

E. Patch Registry

Verified solutions are stored as context-vector embeddings. Top-K most similar prior solutions are retrieved and injected as few-shot hints into the selected tribe's prompt. When a reference solution is already present, the patch hint is suppressed to prevent conflicting code suggestions.

III. EXPERIMENTAL SETUP

Hardware: Single GPU workstation (Docker Compose). Model: Qwen/Qwen2.5-Coder-7B-Instruct-AWQ (4-bit AWQ). Inference: vLLM, temperature per condition, max_tokens=2048, sandbox timeout 10 s.

Phase 1 task set (Runs 1–6): 12 Python functions — sort_list, deduplicate, flatten, partition, reverse_string, is_palindrome, word_count, fibonacci, is_prime, gcd, invert_dict, running_sum. ~8–9 attempts per type per 100-task run.

Phase 2–3 task set (Runs 7–10, ablations): Extended to 16 types by adding has_cycle (graph cycle via Kahn's topological sort), permutations (combinatorial enumeration), search_insert (binary search insertion position), and unique_paths (DP grid path counting). These harder tasks reduce the achievable ceiling and stress-test model capability limits.

Five conditions in Phase 3: (A) T=0.1 ablation: Single tribe, T_0 prompt, temperature=0.1. (B) T=0.3 ablation: Single tribe, T_0 prompt, temperature=0.3. (C) T=0.9 ablation: Single tribe, T_2 prompt, temperature=0.9. (D) MCN-LinUCB (Run 9): Three homogeneous T=0.3 tribes, LinUCB router. (E) MCN-GNN (Run 10): Three heterogeneous tribes (T_0=0.1, T_1=0.5, T_2=0.9), GNN router.

Phase 1B (stratified live evaluation): Following Phase 3, a separate 400-task experiment evaluates MCN-LinUCB (homogeneous T=0.3, alpha=2.5) on a stratified sample of 8 algorithmic categories — string, data_structures, math, dynamic_programming, parsing, iterative, recursive, graph — with 50 tasks per category. All tasks executed via live vLLM inference (Qwen2.5-Coder-7B-Instruct-AWQ). Stratified sampling eliminates task-frequency confounds present in Phase 3's uniform task set, enabling direct category-level pass-rate comparison and oracle gap decomposition.

IV. INFRASTRUCTURE DEFECTS

Six defects across the three phases are documented here because they illustrate a systematic challenge in automated LLM evaluation: test infrastructure errors produce misleading metrics that corrupt router training signals. The pass-rate improvement from 83% (Run 1) to 98% (Run 4) was driven primarily by fixing these defects, not by router learning.

Defect 1 — sys.maxsize in adversarial test B6 (Run 1): fibonacci(sys.maxsize) hangs indefinitely. Corrected to fibonacci(10,000) with a 5-second limit.

Defect 2 — Unbounded st.integers() in Hypothesis (Runs 1–3): The strategy generator produced st.integers() (unbounded) for integer parameters. Hypothesis probes sys.maxsize; even a correct fibonacci loops forever, producing tests_passed=0, tests_failed=0 false negatives and corrupting router rewards for three runs. Corrected to st.integers(min_value=-10000, max_value=10000).

Defect 3 — PATCH_MIN_ATTEMPTS=2 (Run 1): Patch registration required two attempts, leaving the few-shot hint mechanism entirely inactive (0 patches stored). Corrected to PATCH_MIN_ATTEMPTS=1.

Defect 4 — permutations reference returns tuples (Phase 2): itertools.permutations yields tuples; tests expect list[list[int]]. Corrected: [list(p) for p in sorted(set(itertools.permutations(xs)))].

Defect 5 — unique_paths crash on edge cases (Phase 2): math.comb(m+n-2, m-1) raises ValueError when m=0 or n=0. Corrected with guard: if m<=0 or n<=0: return 0.

Defect 6 — Epsilon restoration overwrites warm-up (Phase 2): LinUCB.from_state_dict() restored saved epsilon (~0.05, floor), silently overriding the current .env warm-up schedule. On restart, the router skipped its warm-up phase, causing 97/1/2 degenerate routing. Corrected by overriding epsilon, epsilon_min, and epsilon_decay from MCNConfig after from_state_dict().

V. RESULTS

Table I presents the five-condition Phase 3 comparison. Table II gives the complete ten-run history. The primary finding is that the single-agent T=0.3 ablation outperforms all MCN variants.

TABLE I FIVE-CONDITION PHASE 3 COMPARISON (100 tasks, 16-task benchmark)

Condition	T	Pass	Fib	Search Ins.	Routing
(A) T=0.1 solo	0.1	86%	40%	25%	—
(B) T=0.3 solo	0.3	91%	80%	100%	—
(C) T=0.9 solo	0.9	88%	40%	n/m	—
(D) MCN-LinUCB	0.3×3	86%	40%	n/m	71/13/16
(E) MCN-GNN	0.1/.5/.9	88%	60%	n/m	66/13/21

n/m = not measured for that condition. T=0.9 unique_paths=100%. Best result (B) highlighted.

TABLE II COMPLETE RUN HISTORY (RUNS 1–10)

Run	Router	Tasks	Pass %	Routing	Notes
1	GNN	12	83%	10/69/21	Infra bugs 1–3
2	GNN	12	90%	33/48/19	Partial fix
3	GNN	12	91%	33/48/19	Partial fix
4	GNN	12	98%	39/13/48	All bugs fixed
5	GNN*	12	98%	10/8/82	Inherited state
6	LinUCB	12	97%	100/0/0	Cold-start collapse
7-8	LinUCB	16	84.5%	mixed	Exploratory; ε-bug
9	LinUCB	16	86%	71/13/16	Post-fix; –5pp vs solo
10	GNN	16	88%	66/13/21	Hetero T=0.1/.5/.9

** Run 5 inherits GNN state from Run 4 (no --fresh). Runs 7–8 combined exploratory dataset.*

A. Temperature Ablation — The Dominant Variable

Among the three single-agent ablations, T=0.3 achieves the highest pass rate (91%), outperforming both T=0.9 (88%) and T=0.1 (86%). This 5 pp spread across temperatures exceeds the improvement attributable to any routing strategy tested, establishing temperature as the dominant performance variable for Qwen2.5-Coder-7B on this task set.

T=0.1 failure mode: search_insert collapses to 25%. At very low temperature, the model locks into an incorrect algorithm (linear scan instead of binary search) and fails to self-correct across attempts.

T=0.9 failure mode: fibonacci and permutations show higher failure rates than T=0.3, despite T=0.9 succeeding more on unique_paths. High variance occasionally finds solutions for tricky DP problems but increases failures on tasks with a uniquely correct iterative pattern.

B. MCN-LinUCB vs. Single Agent T=0.3 (Run 9)

MCN-LinUCB with three homogeneous T=0.3 tribes scored 86% — 5 pp below the single-agent T=0.3 ablation (91%). Routing distribution: 71/13/16 (Gini=0.387). Chi-squared test for routing independence from task type: p=0.737. No statistically significant specialisation was detected.

The 5 pp loss arises from exploration overhead. Epsilon-greedy routing sends approximately 29% of tasks to non-dominant tribes during warm-up. Because all tribes use the same model at the same temperature, these explorations produce no useful diversity signal: the model generates statistically identical code regardless of which tribe receives the task. Variance introduced by routing is a pure cost with no compensating specialisation benefit.

Task-level losses (MCN vs. single-agent): fibonacci –40 pp, search_insert –25 pp, unique_paths –50 pp. These represent cases where routing variance accumulated failures during unlucky epsilon windows that a single-agent run would have resolved deterministically.

C. MCN-GNN (Heterogeneous) vs. Single Agents (Run 10)

MCN-GNN with heterogeneous tribes (T_0=0.1, T_1=0.5, T_2=0.9) scored 88% — exactly matching the T=0.9 single-agent ablation and 3 pp below T=0.3. Routing: 66/13/21 (Gini=0.353). Chi-squared p=0.762. No significant specialisation detected.

Critically, T_2 (T=0.9) achieved 95% pass rate in isolation within Run 10 — the highest single-tribe result in the entire study. Yet the full MCN system scored only 88%, quantifying a –7 pp routing cost relative to an oracle that always selects T_2. The GNN's inability to reliably route all tasks to T_2 stems from two failure modes: (i) search_insert is handled worse by T_2 (0%) than by T_0, so the oracle would need to route that task type away from T_2; (ii) unique_paths is handled poorly by T_1 (0%). The GNN must implicitly learn a per-task-type routing policy from ~6 attempts per type — insufficient signal.

Concretely, on fibonacci: T_0 (T=0.1) fails repeatedly with deterministic naive recursion; T_2 (T=0.9) succeeds through high-variance exploration of the search space. MCN-GNN fibonacci improved from 40% (Run 9, homogeneous) to 60% (Run 10), showing that T_2's variance does help — but the GNN must also avoid routing fibonacci to T_0, which it only partially achieves.

TABLE III MCN VARIANTS vs. BEST SINGLE AGENT

Metric	T=0.3 Solo	MCN-LinUCB	MCN-GNN
Pass rate	91%	86% (–5pp)	88% (–3pp)
fibonacci	80%	40% (–40pp)	60% (–20pp)
Routing dist.	—	71/13/16	66/13/21
Chi-sq. p	—	0.737	0.762
Specialisation?	—	No	No
T_2 isolation	88% (T=0.9)	n/a	95% (within-run)
Oracle gap	—	n/a	–7pp vs T_2 solo
Verdict	BEST	–5pp vs best	=T=0.9 solo

T_2 isolation = T_2 tribe pass rate computed within Run 10, not a standalone ablation. Oracle gap = 95%–88% = 7 pp routing cost.

D. Model Capability Limits

Two of the four new tasks reveal hard limits of the 7B model, independent of routing.

has_cycle achieves 0% across all conditions. A Kahn's topological sort reference solution was provided but the model consistently generates syntactically valid yet algorithmically incorrect implementations (missing in-degree initialisation, incorrect BFS termination). The overseer's adversarial tests — directed cycles, self-loops, disconnected components — catch all errors. This is a model capability limit, not a routing failure.

permutations achieves 0% before the type-annotation fix (Defect 4). The critical insight: itertools.permutations returns tuples, but tests expect list[list[int]]. With the corrected reference solution, small models reliably copy the wrapping pattern. This is a reference-solution quality issue, not a routing issue.

E. Phase 1B: Category-Level Analysis (400 Tasks)

Phase 1B evaluates MCN-LinUCB on 400 stratified live tasks. Overall pass rate: 245/400 = 61.2%, significantly below Phase 3's 86% on the 16-task benchmark. This drop reflects the harder category mix: graph tasks (has_cycle, topological_sort, count_components, num_islands, is_bipartite) achieve 0%, and recursive/iterative categories average 42–43%.

Category heterogeneity is extreme (Table IV). String tasks — palindromes, anagrams, compression, case conversion — achieve 100%, consistent with their dominance in pre-training corpora. Graph algorithms achieve 0% in all conditions, confirming the 7B model capability limit established in Phase 3. The 100 pp spread across categories dwarfs the 5 pp MCN-vs.-single-agent gap from Phase 3, establishing category as a far stronger predictor of outcome than routing strategy.

Routing specialisation: chi-squared test for routing independence from task category: p=0.396. No statistically significant specialisation was detected. Routing distribution shows dramatic T0→T1 drift across the 400 tasks, consistent with a bandit that has not converged. Analysis of UCB upper confidence bounds shows that the bandit requires approximately 2,000+ tasks for reliable per-category specialisation at this task diversity.

Oracle gap decomposition (Table V): The total oracle gap of 12.8 pp — MCN 61.2% vs. oracle 74.0% — decomposes into three roughly equal components: exploration cost 4.1 pp (ε-greedy routing to non-optimal arms during warm-up), tie-breaking noise 4.7 pp (near-equal UCB scores trigger random arm selection), and exploitation error 4.0 pp (insufficient per-category signal causes wrong arm selection even during exploitation). No single failure mode dominates, indicating the bandit is operating in a data-limited regime throughout.

TABLE IV PHASE 1B CATEGORY-LEVEL RESULTS (400 tasks, 8×50 stratified)

Category	Pass%	vs. Overall	Capability
string	100%	+38.8 pp	Full
data_structures	84%	+22.8 pp	High
math	86%	+24.8 pp	High
dynamic_prog.	76%	+14.8 pp	Moderate
parsing	66%	+4.8 pp	Moderate
iterative	43%	−18.2 pp	Low
recursive	42%	−19.2 pp	Low
graph	0%	−61.2 pp	None
Overall	61.2%	—	—

vs. Overall = category pass rate minus overall mean (61.2%). Capability = qualitative model-performance tier for Qwen2.5-Coder-7B. Graph 0% is a hard capability limit, not a routing failure.

TABLE V ORACLE GAP DECOMPOSITION (Phase 1B, 400 tasks)

Component	Gap (pp)	Mechanism
Exploration cost (Q1)	4.1	ε-greedy routes ~15% of tasks to non-optimal arms
Tie-breaking noise (Q2+Q3)	4.7	Near-equal UCB scores → random arm selection
Exploitation error (Q4)	4.0	Insufficient per-category signal → wrong arm
Total oracle gap	12.8	MCN 61.2% vs. oracle 74.0%

Oracle = per-task best-tribe selection with perfect knowledge. Q1–Q4 computed via counterfactual routing simulation on recorded decisions. All three components are roughly equal; no single source dominates.

VI. DISCUSSION

The central question — can a learned router improve upon the best single agent when tribes share the same base model? — receives a consistent negative answer across all three routing experiments and the Phase 1B stratified evaluation.

A. Why Routing Fails With Same-Model Tribes

The multi-armed bandit framing assumes that different arms have genuinely different expected rewards for different contexts. When tribes are functionally similar (same model, same temperature), all arms converge to the same expected reward, and exploration cost is never recovered.

Heterogeneous temperatures (0.1/0.5/0.9) create genuine output diversity but do not solve the routing problem, because no single temperature is uniformly best across all task types. T=0.3 beats T=0.9 on fibonacci and search_insert; T=0.9 beats T=0.3 on unique_paths. An oracle router would outperform any single temperature — but the GNN achieves p=0.76 routing independence from task type at 100 tasks, far from oracle behaviour. The −7 pp gap between T_2-in-isolation (95%) and MCN-GNN (88%) directly measures the cost of imperfect routing even when a clearly superior tribe exists.

B. What Would It Take for Routing to Add Value?

For routing to add aggregate value, the inter-tribe performance variance on each task type must exceed the exploration cost. Formally, the expected routing gain equals E[max_i R_i(task)] - E[R_routing(task)], where R_routing is the reward received under the routing policy. When sigma^2(performance | task) is small — as here, where tribes are functionally similar — this gain is negligible and the exploration cost dominates.

The boundary conditions for positive routing value are: (1) True model diversity: Routing between qualitatively different models (e.g., 7B code-specialist vs. 13B reasoning-specialist) would create large inter-tribe performance variance on specific task classes. (2) Sufficient data: At 6–8 attempts per task type, the GNN lacks the signal to learn per-type preferences reliably. Hundreds of attempts per type would provide the learning signal needed for genuine specialisation. (3) Adaptive temperature: Treating temperature as a continuous router output rather than a fixed per-tribe hyperparameter could achieve better task-type coverage than discrete temperature choices.

The negative result here is therefore a boundary condition, not a universal truth: routing is not beneficial with same-model, same-architecture tribes on a 100-task dataset, but may be beneficial with genuinely diverse tribes at larger scale.

C. Category-Level Heterogeneity and Bandit Convergence

Phase 1B establishes that algorithmic category is the dominant predictor of task outcome, with a 100 pp spread from string (100%) to graph (0%). This heterogeneity creates an apparent opportunity for category-adaptive routing: a converged bandit that identifies graph tasks and halts exploration on those categories would reduce the 4.1 pp exploration cost.

However, the critical limiting factor is tribe homogeneity. At T=0.3, all three tribes generate statistically identical code for any given task. The bandit cannot learn a meaningful per-category preference because the arms do not differ. The 12.8 pp oracle gap is therefore irreducible under the current tribe configuration regardless of how many tasks are collected: the oracle for homogeneous tribes is the same as any single tribe, making the gap an artefact of the exploration protocol rather than a recoverable routing signal.

The 2,000-task convergence estimate applies to the case where tribes are genuinely diverse. With qualitatively different tribes (different base models, architectures, or fine-tuning), the per-category reward signal would differ across arms, and the bandit could learn a meaningful routing policy. Under those conditions, the category-level heterogeneity documented in Phase 1B — specifically the mid-tier cluster (parsing 66%, iterative 43%, recursive 42%, dynamic_programming 76%) — represents the most actionable routing opportunity: categories where model capability is partial and tribe-level variance could in principle be exploited.

D. Router Sophistication Is Scale-Invariant

The canonical assumption in contextual bandit literature is that more expressive models should achieve lower regret on complex reward landscapes. The LinUCB vs. GNN comparison at 2,000 tasks directly refutes this assumption in the MCN setting: −0.1 pp delta, with the GNN performing marginally worse on convergence stability.

This is not a model expressiveness failure — the GNN is correctly trained and shows appropriate exploration-exploitation behaviour. The failure is conceptual: the reward landscape is flat. When E[R | task, arm_i] ≈ E[R | task, arm_j] for all i, j, no routing algorithm can learn a meaningful policy from finite data, regardless of its expressiveness.

The practical implication is that "which router is better?" is ill-posed for homogeneous tribes. The correct question is: what is the minimum inter-tribe performance variance required for routing to recover its exploration cost? The retrospective Category TS simulation identifies math as the category where this threshold is closest to being reached (T1=98.4% vs T0=75.7%, a 22.7 pp spread) — but this spread arises from sampling noise in 63 vs 144 tasks per tribe, not from genuine model diversity. With heterogeneous base models or fine-tuning targets, category-level performance splits of this magnitude could be reliably exploited by a category-aware bandit.

E. Temperature Diversity Does Not Create Routing Signal

Phase 2 tests the weakest form of tribe diversity achievable without changing the base model: different sampling temperatures. The result is a complete null: 60.7% with CTS + temperature-diverse tribes equals 60.7% with LinUCB + homogeneous tribes.

The information-theoretic explanation is as follows. Sampling temperature shifts the output distribution of the same underlying model: high temperature increases entropy over the token distribution but does not change which concepts or algorithms the model knows. For any given task, the conditional probability of a correct solution P(correct | task, model) is fixed by the model's weights; temperature scales the variance of outputs around this mean but leaves the mean unchanged. Concretely, a 7B model that does not know how to represent a graph as a boolean adjacency matrix will fail at temperature 0.1 and temperature 0.9 alike.

The oracle gap reduction (5.1 pp to 3.7 pp) confirms this directly: reducing inter-tribe variance reduces the maximum achievable routing gain, which is bounded by E[max_i R_i(task)] - E[R_mean(task)]. Temperature diversity moves this bound in the wrong direction.

This result closes the temperature-diversity hypothesis with a decisive null and sharpens the minimal condition for routing value: tribes must differ in their underlying model weights, fine-tuning targets, or capability profiles — not merely in their sampling hyperparameters. Future experiments should therefore use genuinely different models (e.g., a code-specialist, a reasoning-specialist, and a general-purpose model) rather than temperature variants of the same checkpoint.

F. Phase 1C: 2000-Task Longitudinal Experiment (LinUCB)

Phase 1C scales the stratified evaluation to 2,000 tasks (8x250 per category) using the LinUCB router (alpha=2.5, epsilon=0.3, decay=0.99). Overall pass rate: 1,214/2,000 = 60.7%, unchanged from Phase 1B (61.2%), confirming that additional tasks provide no benefit when tribes are homogeneous.

Spurious convergence is confirmed at scale. T0 routing share by 500-task window: 27% to 75% to 94% to 96%. The bandit reaches nominal lock-in by task ~1,700 despite T0=60.8%, T1=59.9%, T2=61.8% — all within 2 pp. The locked arm (T0) is not the best arm (T2 is marginally higher at 61.8%), confirming the convergence is driven by early random variance, not learned preference.

Parsing regression analysis (Phase 1B 65.5% to Phase 1C 58.8%) was traced to two task types: roman_to_int (0%, KeyError in format mapping) and decode_run_length (0%, ValueError in string split). Three other parsing tasks (title_case=100%, count_vowels=100%, camel_to_snake=94%) were unaffected. This is a format-compliance failure, not a routing problem, and not a systematic regression.

G. Phase 1D: GNN vs. LinUCB Router Comparison at Scale

Phase 1D repeats the 2,000-task experiment with the GNN router (hidden_dim=32, lr=0.01, buffer=64, batch=8). Table VI presents the head-to-head comparison (see also Table VII for strategy breakdown).

Overall: 1,212/2,000 = 60.6% — a -0.1 pp delta vs. LinUCB. All per-category deltas lie within +-3.6 pp, consistent with sampling noise for 250-task buckets (sigma ~3.2 pp). No systematic GNN advantage or disadvantage is detected in any category.

The GNN locks onto T0 faster (72% total routing share vs. 55% for LinUCB) and converges earlier (~task 800 vs. ~task 1,700). The GNN's mini-batch updates on an 8-entry replay buffer push it to commit earlier to the same spurious fixed point. Faster lock-in is a net disadvantage: the GNN sacrifices exploratory signal that LinUCB retains longer but equally fails to exploit.

A retrospective simulation on Phase 1C data tests four routing strategies (Table VII). Category Thompson Sampling (CTS) achieves 63.2% (+2.5 pp vs. LinUCB), with the largest gain in math: +12.0 pp (empirical T1=98.4% vs. T0=75.7%). Notably, random routing (62.4%) outperforms LinUCB (60.7%), confirming the bandit's exploration schedule imposes a net cost relative to the uniform baseline. The oracle ceiling is 65.4%.

TABLE VI ROUTER COMPARISON AT SCALE (2,000 tasks, 8×250 stratified)

Metric	LinUCB (Phase 1C)	GNN (Phase 1D)	Delta
Overall pass rate	60.7%	60.6%	−0.1 pp
T0 routing share	55%	72%	GNN locks harder
T1 routing share	28%	8%	GNN drops T1 early
T2 routing share	16%	19%	+3 pp
T0 pass rate	60.8%	60.9%	+0.1 pp
T1 pass rate	59.9%	59.6%	−0.3 pp
T2 pass rate	61.8%	60.1%	−1.7 pp
Convergence to T0	~task 1700	~task 800	GNN 2x faster
Router parameters	~54	~2,000	40x more complex

Convergence = task index where T0 share first exceeds 90% and stays there. Per-tribe pass rates within ±2 pp across both routers — statistically indistinguishable. GNN parameter count: 3 tribe embeddings (18-dim) + 2-layer MLP (36→32→16→1).

TABLE VII ROUTING STRATEGY COMPARISON (Phase 1C data, 2,000 tasks)

Strategy	Pass Rate	vs. LinUCB	Key Observation
LinUCB (actual)	60.7%	baseline	Locks to T0; hurts math
Random routing	62.4%	+1.7 pp	Uniform > trained bandit
Category TS (simulated)	63.2%	+2.5 pp	Math +12 pp; no lock-in
Oracle per-category	65.4%	+4.7 pp	Ceiling; requires hindsight

Category TS simulation uses imputed rewards from empirical per-(category, arm) rates for counterfactual arm selections. Random > LinUCB confirms the bandit's exploration schedule imposes a net cost vs. uniform baseline with homogeneous tribes. Math +12 pp for CTS arises from empirical T1=98.4% vs. T0=75.7% (250 tasks each).

H. Phase 2: Temperature-Heterogeneous Tribes + Live CTS Router

Phase 2 extends the homogeneity hypothesis test to temperature-diverse tribes. Three tribes are assigned temperatures T0=0.1 (deterministic), T1=0.5 (balanced), T2=0.9 (exploratory), with the live CategoryThompsonSampling (CTS) router. The experiment runs 1,502 tasks (8x~188 stratified). Table VIII presents the comparison with Phase 1C.

Overall pass rate: 911/1,502 = 60.7% — identical to Phase 1C LinUCB. Per-tribe routing converged to T2 (52.3% share), but per-tribe pass rates (T0=60.4%, T1=59.3%, T2=61.3%) differ by at most 2.0 pp, indistinguishable from sampling noise (sigma ~3.2 pp for 188-task buckets).

The oracle gap narrows from 5.1 pp (Phase 1C) to 3.7 pp (Phase 2). A smaller oracle gap indicates that tribe outputs are more similar, not that the router is better: with lower inter-tribe variance, even perfect hindsight routing achieves less improvement. This is the opposite of the desired outcome.

CTS routing drift reveals non-trivial learning dynamics absent in LinUCB/GNN. The 500-task windows show T1=41%, T0=42%, T2=68% routing shares — CTS oscillates between tribes before settling on T2, whereas LinUCB commits to T0 irreversibly by task ~1,700. Despite correct Bayesian updating, the learned routing preference produces no aggregate gain because the per-(category, arm) Beta posterior means remain within noise.

Category pass rates are stable across phases: string 96.2%, math 91.8%, data_structures 85.4%, dynamic_programming 72.6%, parsing 56.8%, iterative 48.4%, recursive 38.1%, graph 0.0%. The graph ceiling and the parsing failure cluster (roman_to_int=0%, decode_run_length=0%) persist independently of tribe temperature, confirming these are model-capability and prompt-compliance failures, not routing artefacts.

TABLE VIII PHASE 1C vs. PHASE 2: EFFECT OF TEMPERATURE DIVERSITY (CTS ROUTER)

Metric	Phase 1C (Lin UCB+Homo)	Phase 2 (CTS+Hetero)	Delta
Tasks	2,000	1,502	—
Overall pass rate	60.7%	60.7%	0.0 pp
Oracle (per-cat)	65.8%	64.3%	–1.5 pp
Oracle gap	5.1 pp	3.7 pp	–1.4 pp
Dominant tribe	T0 (55%)	T2 (52%)	shifted
T0 pass rate	60.8%	60.4%	–0.4 pp
T1 pass rate	59.9%	59.3%	–0.6 pp
T2 pass rate	61.8%	61.3%	–0.5 pp
Max inter-tribe gap	1.9 pp	2.0 pp	≈ same
Convergence pattern	Lock T0 (~t=1700)	Oscillate T1->T0->T2	CTS avoids lock
Graph pass rate	0%	0%	persists
String pass rate	99%	96.2%	≈ same

Oracle gap decrease indicates tribes became more similar under temperature diversity, not that routing improved. CTS routing drift (T1->T0->T2 oscillation) reflects correct Bayesian updating; the reward signal is insufficient because temperature diversity does not create capability diversity. Phase 2: 1,502 tasks (OOM restart at t=1,490).

VII. CONCLUSIONS

We have conducted ten experimental runs, three controlled ablations, and two 2,000-task longitudinal evaluations evaluating the Mycelial Council Network on Python code synthesis benchmarks. Primary conclusions:

(1) Temperature dominates routing. T=0.3 achieves 91% (single agent); all MCN variants score 86–88%. The optimal multi-agent configuration does not exceed the optimal single-agent temperature.

(2) Same-model routing imposes a net cost. MCN-LinUCB (homogeneous T=0.3) scores –5 pp vs. single-agent, with p=0.74 routing independence.

(3) Heterogeneous temperatures do not enable specialisation. MCN-GNN matches T=0.9 single agent (both 88%). The –7 pp gap between T_2-in-isolation (95%) and MCN-GNN (88%) quantifies the routing cost when a clearly superior tribe exists but cannot be selected reliably.

(4) Infrastructure correctness dominated early measured outcomes. Six defects across the three phases caused systematic performance distortions unrelated to routing quality.

(5) Model capability limits, not routing, explain the hardest failures. has_cycle achieves 0% in all conditions; this is a 7B model limit.

(6) Category-level heterogeneity dwarfs routing effects. Phase 1B/1C reveals a 100 pp spread from string (99%) to graph (0%). Corrected pass rate excluding the graph capability limit: 81.3%. The oracle gap of 12.8 pp splits equally among exploration cost (4.1 pp), tie-breaking noise (4.7 pp), and exploitation error (4.0 pp); the bandit is data-limited throughout.

(7) Router sophistication is scale-invariant. At 2,000 tasks, LinUCB (60.7%) and GNN (60.6%) produce a –0.1 pp delta — a null result robust to a 40x increase in router parameter count. The GNN locks onto T0 faster (task ~800 vs. ~1,700) and routes 72% vs. 55% to T0 — a worse outcome. No routing algorithm can learn from a flat reward landscape.

(8) Category Thompson Sampling is the correct next router. Retrospective simulation yields +2.5 pp (+12 pp on math) over LinUCB. CTS maintains per-(category, arm) Beta posteriors, preventing the cross-category signal contamination that causes spurious LinUCB/GNN convergence.

(9) Temperature diversity does not create routing signal. Phase 2 deploys live CTS with temperature-heterogeneous tribes (T0=0.1, T1=0.5, T2=0.9): overall pass rate is 60.7% —

identical to Phase 1C. Per-tribe rates span only 2.0 pp (T0=60.4%, T1=59.3%, T2=61.3%). The oracle gap narrows from 5.1 pp to 3.7 pp, moving in the wrong direction. CTS learns routing preferences (oscillating T1→T0→T2 drift) but cannot improve aggregate performance because temperature shifts output variance, not model capability. The minimal condition for routing value is inter-model capability diversity, not sampling or algorithmic sophistication.

Future directions: (A) heterogeneous base models — routing between a code-specialist, reasoning-specialist, and general-purpose 7B model to create genuine inter-tribe performance variance; (B) graph-fixed benchmark with output-format-constrained prompts to measure the 81.3% corrected ceiling under better task design; (C) per-category fine-tuning — train separate LoRA adapters per category to create specialised tribes on the same base checkpoint.

Reproducibility: all code, Docker configuration, and MLflow artifacts available in the MCN repository. Raw data: mcn-results:/results/runs.jsonl.