Gossip-ES: Decentralized Evolution Strategies via Network Topologies

Final Project — Parallel Programming, Spring 2026

Cheng-Han Hsieh
R14922068
Computer Science and Information Engineering
r14922068@ntu.edu.tw

Chih-Chuan Huang
R14944078
Graduate Institute of Networking and Multimedia
r14944078@ntu.edu.tw

***Abstract*—Distributed Evolution Strategies (ES) are usually implemented with a parameter-server architecture: a master maintains a single global search center, collects rewards from all workers, and broadcasts updates. This creates a communication hotspot, a single point of failure, and—because the entire population explores around one center—a tendency toward premature convergence on multimodal objectives. We present *Gossip-ES*, a fully decentralized ES framework in which each of N nodes maintains its own parameter vector and communicates only with its neighbors in a configurable graph topology (ring, 2D-torus, random k-regular, complete), using doubly stochastic Metropolis–Hastings mixing. On the multimodal Rastrigin benchmark (D=32, N=16, 20 seeds) under a strictly budget-matched comparison, every gossip topology reaches a final best objective of $f \approx 19$–21 with a 100% success rate, while budget-matched centralized ES ($f \approx 290$) and communication-free independent ES ($f \approx 213$) remain trapped, a $\sim 15\times$ quality gap. Gossip quality improves monotonically with N (ring: $61.5 \to 11.6$ for N=4 $\to$ 32) while centralized ES is flat in N, and sparse topologies bound per-node communication by the node degree instead of $O(N)$. Under injected stragglers in a real multiprocessing deployment, asynchronous gossip sustains two to three orders of magnitude more updates per second than any synchronous mode, but we identify a *stale-anchor effect* through which asynchrony degrades solution quality—and show that a simple staleness-weighted mixing rule (mix with a neighbor only if it has updated since the last exchange) recovers synchronous-level quality at asynchronous speed, reaching the synchronous quality level $4\times$ sooner in wall-clock time. Finally, a dynamic ring$\to$random-regular$\to$complete schedule achieves the best mean final objective of all tested topologies ($f = 16.0 \pm 5.0$; unpaired rank tests: $p < 10^{-3}$ against the dense endpoint, $p = 0.03$ against the sparse one), evidence that the exploration–consensus trade-off can be scheduled over time.**

***Index Terms*—evolution strategies, gossip protocols, decentralized optimization, network topology, spectral gap, asynchronous algorithms**

Project Category: ■ **A** (Self-chosen topic) □ **B** (AI-agent code optimization)
Tick exactly one. This tells reviewers which lens to apply.

I. Introduction

Evolution Strategies (ES) are zero-order black-box optimizers that require only scalar fitness evaluations, which makes them attractive for reinforcement learning, simulator-based optimization, and other non-differentiable settings [1]. Because the $2m$ fitness evaluations per update are embarrassingly parallel, ES is almost always distributed. The standard architecture is a synchronous parameter server: a master holds one global parameter vector θ, workers evaluate perturbations of it, and the master aggregates a pseudo-gradient and broadcasts the new θ.

This design has three structural weaknesses. First, the master processes $O(N)$ messages per iteration ($O(ND)$ bytes if parameter vectors are shipped), an asymptotic bottleneck as the worker count N grows. Second, the master is a single point of failure and a synchronization barrier: one straggling worker stalls the entire iteration. Third—and specific to ES as a *population* method—all workers explore around the same search center, so if that center descends into a poor basin of attraction the whole population collapses together.

Gossip-ES removes the coordinator. Each node i keeps a local parameter θ_i, performs local ES updates, and averages parameters only with its graph neighbors using a doubly stochastic mixing matrix. The communication topology then becomes an explicit algorithmic knob: sparse graphs (large mixing time) preserve inter-node diversity and exploration; dense graphs (small mixing time) accelerate consensus and exploitation.

This project makes the following contributions:

1) a fully decentralized ES framework with no parameter server, implemented both as a deterministic single-process simulator and as a real shared-memory multiprocessing system;
2) a budget-matched comparison of centralized ES, independent ES, and Gossip-ES over ring, 2D-torus, random 4-regular, and complete topologies on multimodal benchmarks;
3) an analysis connecting the spectral gap of the mixing matrix to consensus speed and population-diversity decay;
4) a straggler study comparing synchronous and asynchronous gossip under heterogeneous worker speeds, identifying a *stale-anchor* failure mode of asynchronous mixing and a barrier-free staleness-weighted mixing rule that repairs it;

5) a dynamic topology schedule (ring→random-regular→complete) that beats every static topology.

II. Background and Related Work

A. Evolution Strategies

ES estimates a search gradient from function values only. With mirrored (antithetic) sampling, node i draws $\epsilon_{i,k} \sim \mathcal{N}(0, I_D)$, $k = 1, \ldots, m$, evaluates rewards $R^{\pm}_{i,k} = F(\theta_i \pm \sigma\epsilon_{i,k})$, and forms the pseudo-gradient

$$g_i \;=\; \frac{1}{2m\sigma} \sum_{k=1}^{m} \big(\tilde{R}^{+}_{i,k} - \tilde{R}^{-}_{i,k}\big)\, \epsilon_{i,k}, \tag{1}$$

where $\tilde{R}$ denotes fitness-shaped rewards (the reward differences are standardized to zero mean and unit variance for scale invariance, as in [1]). The local update is $\theta_i \leftarrow \theta_i + \alpha g_i$. Our test suite verifies on a D=16 quadratic that the estimator in (1) achieves cosine similarity $>$0.9 to the true gradient at m=512 mirrored pairs.

B. Gossip Averaging and Decentralized Optimization

Distributed averaging by repeated local mixing is classical: Xiao and Boyd [2] analyze the convergence rate of linear iterations $x \leftarrow Wx$ in terms of the second-largest eigenvalue modulus of W, and randomized gossip protocols were analyzed in [3]. Decentralized first-order methods built on the same mixing primitive include distributed subgradient descent [4], decentralized parallel SGD, which can outperform centralized SGD when communication is the bottleneck [5], and stochastic gradient push for time-varying directed graphs [6]. Gossip-ES transplants this template from SGD to ES, where the population aspect adds a second role for slow mixing: preserving *search diversity*, not just tolerating it.

III. Methodology

A. Gossip-ES

N nodes form an undirected connected graph $G = (V, E)$ with neighbor sets $\mathcal{N}(i)$. Every iteration has three stages:

1) Local ES step: each node computes g_i by (1) from its own RNG stream and applies $\theta_i^{\text{tmp}} = \theta_i + \alpha g_i$.

2) Gossip exchange: node i sends θ_i^{tmp} to its neighbors and receives theirs; there is no broadcast and no central collection.

3) Consensus mixing: parameters are averaged with mixing weights:

$$\theta_i \;\leftarrow \sum_{j \in \mathcal{N}(i) \cup \{i\}} W_{ij}\, \theta_j^{\text{tmp}}. \tag{2}$$

B. Metropolis–Hastings Mixing Matrix

For neighbors $i \neq j$ we use

$$W_{ij} = \frac{1}{1 + \max(d_i, d_j)}, \qquad W_{ii} = 1 - \sum_{j \in \mathcal{N}(i)} W_{ij}, \tag{3}$$

with $d_i = |\mathcal{N}(i)|$. On undirected graphs this W is symmetric and doubly stochastic ($W\mathbf{1} = \mathbf{1}$, $\mathbf{1}^{\top}W = \mathbf{1}^{\top}$), which our test suite asserts to within 10^{-12} for every topology and size used in the experiments.

C. Virtual Mean Model

Let $\bar{\theta} = \frac{1}{N}\sum_i \theta_i$. Double stochasticity makes the mixing step average-preserving, so

$$\bar{\theta}_{t+1} = \bar{\theta}_t + \frac{\alpha}{N} \sum_{i=1}^{N} g_i. \tag{4}$$

The *average* system behaves like a decentralized approximation of centralized ES, but the individual nodes are allowed to disagree transiently. Gossip-ES therefore balances two forces: consensus (nodes must not diverge indefinitely) and diversity (nodes must not become identical too early, or the method degenerates to centralized ES). The topology controls the balance.

D. Spectral Gap

Order the eigenvalues of the symmetric W as $1 = \lambda_1 \geq |\lambda_2| \geq \cdots \geq |\lambda_N|$. The spectral gap

$$\gamma = 1 - |\lambda_2| \tag{5}$$

governs the consensus rate: disagreement contracts by a factor $|\lambda_2| = 1 - \gamma$ per mixing round. Measured gaps for $N = 16$ are: ring 0.051, random 4-regular $\approx$0.27 (seed-dependent, 0.11–0.40), 4×4 torus 0.40, complete 1.0 (exact consensus in one round). Our hypothesis is that small-γ topologies preserve diversity longer and escape poor basins more reliably.

E. Population Diversity

We track

$$\mathrm{Div}(t) = \frac{1}{N} \sum_{i=1}^{N} \big\|\theta_i^{(t)} - \bar{\theta}^{(t)}\big\|_2^2, \tag{6}$$

plus mean pairwise distance, and report final best/mean objective values in the original (minimization) scale, where the global optimum is $f = 0$. Note that the consensus error $\mathrm{CE}(t) = \frac{1}{N}\|\Theta_t - \mathbf{1}\bar{\theta}_t^{\top}\|_F^2$ coincides identically with $\mathrm{Div}(t)$ under this normalization, so we track a single quantity for both roles.

F. Dynamic Topology Schedule

Because exploration matters early and consensus matters late, we also evaluate a schedule that switches the topology during training: ring for iterations 0–50%, random 4-regular for 50–80%, complete for 80–100%. The mixing matrix is rebuilt at each switch; the update rule is otherwise unchanged.

G. Implementation

We implement three layers, all in Python/NumPy with no GPU requirement.

Layer 1 (single-process simulator). All nodes live in one process; mixing is a matrix product $\Theta \leftarrow W\Theta$. This layer gives deterministic, seed-reproducible algorithmic comparisons and logs analytic communication costs (messages and bytes per iteration as a function of topology, N, and D).

Layer 2 (multiprocessing). Each node is a separate OS process; parameters live in POSIX shared memory. Four

modes are implemented: `centralized_sync` (barrier per iteration, master aggregates), `gossip_sync` (barrier per iteration, neighbor mixing), `gossip_async` (no barriers: each node loops evaluate–update–mix against whatever its neighbors' current parameters are), and `gossip_async_sw` (staleness-weighted async, Sec. V-D: parameter rows carry version counters published atomically with the parameters, and a neighbor enters the mix only if its version advanced since the last exchange). Stragglers are injected by sleeping a configurable `delay_seconds` inside designated slow nodes each local iteration. This layer measures real wall-clock behavior under heterogeneity.

Layer 3 (TCP, engineering bonus). Each node is a separate process running an asyncio event loop with a TCP server (`asyncio.start_server`) that publishes the node's parameters, and a trainer coroutine that pulls neighbor parameters over persistent TCP connections (`asyncio.open_connection`), mixes, and publishes. The wire protocol is a one-byte command followed by a fixed-shape binary response (version, dimension, raw `float64` payload); the version counter ships with every payload, giving the staleness-weighted mode an exact freshness test for free. Nothing is shared between processes except sockets—the monitor is a plain TCP client—so the same code would run across machines by changing the host list. Per the project scope, Layer 3 is a deployment validation of the Layer-2 findings, not primary evidence.

Budget matching. All Layer-1 methods consume identical function evaluation budgets per iteration: $2Nm$ evaluations. Centralized ES uses all Nm mirrored pairs on its single center per iteration, so it receives a pseudo-gradient that is N times less noisy than each gossip node's—the comparison isolates the effect of *having one search center versus N communicating centers*, not raw evaluation counts. Independent ES is the $W = I$ ablation.

IV. Experimental Setup

Environment. All experiments ran on one node of a SLURM cluster (partition `gp2d`): 2× Intel Xeon Gold 6154 (36 cores @ 3.0 GHz, no SMT), 754 GiB RAM, Rocky Linux 8.4, Python 3.13.13, NumPy 2.4.6. The job was allocated 8 cores—one per process for the N=8 Layer-2/3 runs—and no GPU is used. Layer-1 results are deterministic given the seed; Layers 2–3 wall-clock numbers depend on the runtime and machine load.

Benchmark. Rastrigin in $D = 32$: $f(x) = 10D + \sum_i [x_i^2 - 10\cos(2\pi x_i)]$, global minimum $f(\mathbf{0}) = 0$, with nodes initialized uniformly in $[-5.12, 5.12]^D$ (initial $f \approx 500$; initial Div ≈ 280). Ackley-32 is used as a secondary check. ES maximizes $F = -f$; all reported numbers are raw f (lower is better).

Defaults (Experiments 1, 3, 5). $N = 16$ nodes, $m = 8$ mirrored perturbations per node, $\sigma = 0.1$, $\alpha = 0.02$, 1000 iterations, 20 seeds. Success is defined as final $\min_i f(\theta_i) \leq 40$. Experiment 2 varies $N \in \{4, 8, 16, 32\}$ with 5 seeds. Experiment 4 uses Layer-2 with $N = 8$, $D = 16$, ring topology, a fixed 15 s wall-clock budget, 10 seeds (3 for the centralized baseline), straggler ratio $r \in \{0, 0.25, 0.5\}$ and per-iteration delay $\delta \in \{0.01, 0.05, 0.1\}$ s.

Methods. *centralized* (budget-matched parameter-server ES), *independent* ($W = I$), and gossip over *ring*, *2D-torus*, *random 4-regular* (self-implemented pairing-model sampler with connectivity check), and *complete* graphs.

TABLE I
Rastrigin-32, N=16, 1000 iterations, 20 seeds. Final best/mean objective across nodes (mean ± std over seeds), final population diversity, and success rate ($f_{\text{best}} \leq 40$).

Method	f_{best}	f_{mean}	Div(T)	Succ.
centralized	290.4 ± 41.3	290.4	0	0.00
independent	212.6 ± 28.9	289.4	255.8	0.00
gossip ring	$\mathbf{18.5 \pm 4.0}$	28.6	0.259	1.00
gossip torus	21.1 ± 4.8	21.6	0.005	1.00
gossip rr-k4	19.1 ± 5.6	19.6	0.005	1.00
gossip complete	20.6 ± 4.7	20.6	10^{-30}	1.00

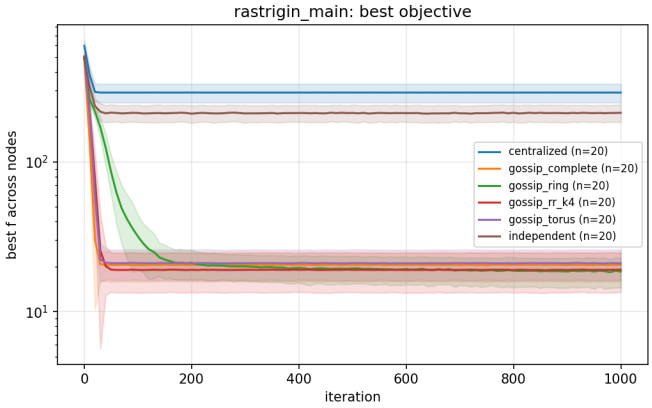


Fig. 1. Best objective across nodes vs. iteration (Rastrigin-32, N=16, mean over 20 seeds, shaded ±1 std). All gossip topologies escape to $f \approx 19$–21; centralized and independent ES flatline above 200 within ∼50 iterations.

V. Results

A. Experiment 1: Topology vs. Final Quality

Table I and Figs. 1–2 contain the central result. Every gossip topology reaches $f \approx 19$–21 with a 100% success rate, while both baselines fail on all 20 seeds—a ∼15× gap. The failure modes of the two baselines are instructive and different:

Centralized ES commits immediately. Despite receiving the least noisy gradient estimate per iteration ($Nm = 128$ mirrored pairs on one center), centralized ES flatlines at $f \approx 290$ by iteration ∼50 and never improves again (Fig. 1). With $\sigma = 0.1$, ES descends a Gaussian-smoothed Rastrigin whose local structure is preserved; one search center commits to the nearest basin and stays. More accurate gradients only make the commitment faster.

Independent ES has diversity but cannot use it. With $W = I$, diversity stays at the initialization level (Div = 255.8, Fig. 2), so some node always lands in a better basin than

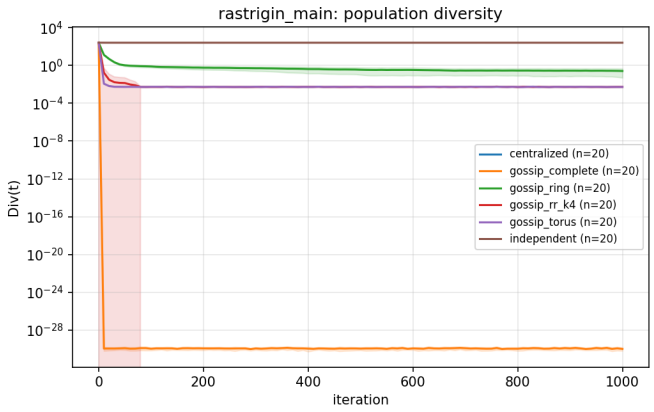


Fig. 2. Population diversity $\mathrm{Div}(t)$ (log scale). Independent ES never mixes and stays at the initialization level (≈280); ring sustains Div ≈ 0.26 for the whole run; the complete graph collapses to numerical zero ($\sim 10^{-30}$) almost immediately.

TABLE II
SCALING WITH NODE COUNT (RASTRIGIN-32, 1000 ITERATIONS, 5 SEEDS): FINAL BEST OBJECTIVE.

N	centralized	ring	rr-$k4$	complete
4	278.8 ± 24.1	61.5 ± 9.2	—	61.3 ± 9.6
8	277.9 ± 23.7	33.4 ± 5.7	43.9 ± 4.3	32.2 ± 5.7
16	279.7 ± 24.0	17.0 ± 3.9	21.9 ± 7.6	19.5 ± 3.1
32	281.2 ± 24.3	$\mathbf{11.6 \pm 1.5}$	12.3 ± 1.8	11.8 ± 1.0

centralized ES (212.6 vs. 290.4 best-of-N)—but without communication no node can exploit a neighbor's discovery, and the population mean (289.4) is as bad as centralized.

Gossip combines both. Mixing lets good basins recruit neighbors while inter-node disagreement keeps multiple basins under exploration. The ring—the slowest-mixing topology—achieves the best final value (18.5) *and* retains three orders of magnitude more diversity at $t = 1000$ than any denser gossip graph. Note also the gap between ring's $f_{\text{best}} = 18.5$ and $f_{\text{mean}} = 28.6$: slow consensus means ring nodes still disagree at the end, whereas torus/rr/complete have essentially converged to a single point ($f_{\text{best}} \approx f_{\text{mean}}$).

The differences *among* gossip topologies (18.5–21.1) overlap within one standard deviation, so the honest claim is not a strict quality ordering by spectral gap, but rather: (i) any connected gossip graph beats both no-mixing and single-center extremes by an order of magnitude, and (ii) slow mixing buys sustained diversity at no quality cost.

Secondary benchmark. The same ablation on Ackley-32 (Table VI, Appendix) reproduces the qualitative picture: every gossip topology reaches $f \approx 11.5$–12.2 while both baselines remain above 19—a $1.7\times$ margin on a landscape with one global basin rather than Rastrigin's lattice of local minima.

B. Experiment 2: Scalability

Table II shows the sharpest scalability contrast we measured: *gossip converts additional nodes into better optimization; centralized ES converts them into nothing.* Ring improves monotonically from 61.5 (N=4) to 11.6 (N=32); centralized ES is flat at $f \approx 279$ for every N.

Communication makes the systems argument. At N=32 the ring sends 64 messages per iteration (16.4 KB)—the same count as centralized—but spreads them uniformly: 2 per node ($O(1)$) vs. all 64 at the master ($O(N)$). The complete graph pays $O(N^2)$ (992 messages, 254 KB) for quality the ring matches at $1/15$ the byte cost.

At $N = 32$ the ring run also exposes the cost of slow mixing: $f_{\text{mean}} = 81.9$ against $f_{\text{best}} = 11.6$, with $\mathrm{Div}(T) = 3.5$. The diameter-16 ring has not finished agreeing after 1000 rounds. This is exactly the regime where a dynamic schedule (Sec. V-E) is attractive.

TABLE III
STRAGGLER STUDY (LAYER-2 MULTIPROCESSING, N=8 RING, D=16, FIXED 15 S BUDGET, $\delta = 0.05$ S): FINAL BEST OBJECTIVE BY STRAGGLER RATIO r. 10 SEEDS (CENTRALIZED: 3); RESULTS ARE INSENSITIVE TO δ.

Mode	r=0	r=0.25	r=0.5
centralized sync	141.9 ± 35.4	150.2 ± 35.0	118.0 ± 26.2
gossip sync	$\mathbf{20.2 \pm 6.1}$	$\mathbf{19.3 \pm 7.4}$	$\mathbf{20.1 \pm 8.2}$
gossip async	53.9 ± 11.6	78.1 ± 22.5	36.4 ± 12.1
gossip async-sw	21.0 ± 6.9	23.6 ± 8.0	26.7 ± 7.6

TABLE IV
LOCAL UPDATES PER NODE WITHIN THE 15 S BUDGET ($\delta = 0.05$ S). FOR THE ASYNC MODES, FAST/SLOW NODE MEANS ARE SHOWN.

Mode	r=0	r=0.25	r=0.5
centralized sync	48 108	295	295
gossip sync	52 417	295	295
gossip async	92 752	114 931 / 299	140 767 / 299
gossip async-sw	82 400	106 272 / 299	127 102 / 299

C. Experiment 3: Diversity and Consensus Dynamics

Equilibrium diversity decreases monotonically with γ, spanning nearly thirty orders of magnitude: ring (γ=0.051, Div=0.26), random-regular/torus ($\gamma \approx 0.27$–0.40, Div$\approx 5 \times 10^{-3}$), complete (γ=1, Div$\sim 10^{-30}$). This confirms the prediction of Sec. III-D. The observed diversity is an equilibrium between ES noise injection and mixing contraction by $1-\gamma$, which is why Div plateaus (Fig. 2) rather than decaying to zero for $\gamma < 1$.

D. Experiment 4: Heterogeneous Worker Speed

Throughput: barriers behave exactly as predicted. With any stragglers present ($r > 0$), both synchronous modes are gated to the straggler rate: measured updates per node are ≈1390, 295, and 148 for $\delta = 0.01, 0.05, 0.1$ s—matching the analytic bound $B/\delta = 1500, 300, 150$ for budget $B = 15$ s almost exactly (Table IV). Relative to the straggler-free rate ($\approx 5 \times 10^4$), this is a $\sim 170\times$ (δ=0.1: $\sim 320\times$) throughput collapse caused by the barrier alone. Directly measured idle time tells the same story from the other side: with stragglers present, fast nodes sit parked at the barrier for 14.90 of the 15 s budget (99% idle), and even with no stragglers the fine-grained synchronization leaves nodes idle 57–62% of the time

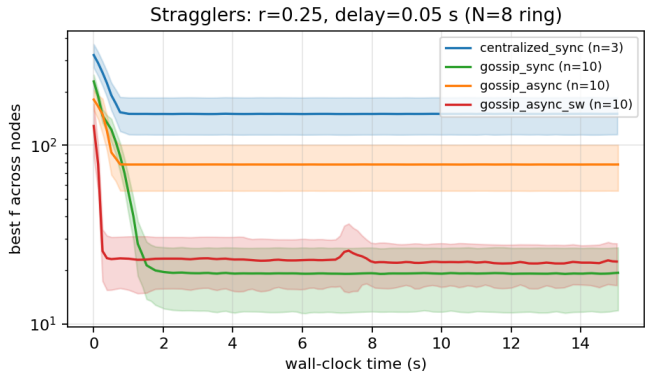


Fig. 3. Best objective vs. wall-clock time under stragglers (r=0.25, δ=0.05 s). Plain async gossip saturates at $f \approx 78$ within $\sim$1 s and never improves despite $\sim 10^5$ further updates per fast node. Staleness-weighted async (async-sw) reaches the synchronous quality level in $\sim$0.5 s, roughly $4\times$ sooner than synchronous gossip itself.

at this evaluation granularity (cluster-wide idle fraction 0.74 at r=0.25, 0.50 at r=0.5, where the sleeping slow nodes are counted as busy). The asynchronous modes are immune: fast nodes sustain 0.8–1.4 $\times$ 10^5 updates regardless of δ—up to $\sim 960\times$ the gated synchronous rate—while slow nodes still contribute only B/δ.

Quality: plain asynchrony pays a staleness tax. Table III: gossip-sync reaches $f \approx$ 19–20 in *every* condition—even gated to 290 (or 147) iterations, that many well-mixed synchronous rounds suffice to converge on this benchmark, so the throughput collapse costs it nothing here. Centralized-sync is trapped at $f \approx$ 118–150 everywhere—the single-center pathology of Experiment 1 reproduced in real processes, dwarfing any straggler effect. Plain async gossip pays a quality penalty in every condition, including with *no* stragglers (53.9 vs. 20.2 at $r = 0$): mixing against neighbors' parameters at uncontrolled phases means most mixing events re-read parameters that have not changed since the previous event, repeatedly pulling the node back toward the same point (Fig. 3). Stragglers make this extreme—a slow node's parameters change $\sim$300 times in 15 s while its fast neighbors mix against them $\sim 1.1\times 10^5$ times, acting as a nearly frozen *anchor* in their averaging—and async is worst at $r = 0.25$ (78.1). The partial recovery at $r = 0.5$ (36.4) tracks a throughput rise ($1.1 \rightarrow 1.4 \times 10^5$ updates per fast node) consistent with reduced contention on the shared mixing lock when half the nodes spend most of their time sleeping, but we do not consider the non-monotonicity fully explained.

Mitigation: staleness-weighted mixing. The anchor diagnosis suggests a one-line protocol fix, which we implement as `gossip_async_sw`: a neighbor's parameters enter the mix only if that neighbor has completed at least one update since we last mixed with it; otherwise its Metropolis weight is redirected to our own parameters (the parameter row and its version counter are published atomically, so the freshness test is exact under concurrency). This caps each neighbor's influence at once per actual update while keeping the protocol fully barrier-free. The effect (Table III) is decisive: async-sw reaches $f = 21.0$–26.7 across all nine conditions, statistically matching synchronous gossip at $r \leq 0.25$ while retaining the full asynchronous update rate ($\sim 1.1 \times 10^5$ updates per fast node at r=0.25, $\sim 360\times$ the gated synchronous rate). It also repairs the straggler-free case (21.0 vs. 53.9), confirming that re-mixing against unchanged parameters—not heterogeneity per se—is the root cause. And because it needs no barriers, async-sw reaches the synchronous quality level in $\sim$0.5 s of wall-clock, about $4\times$ sooner than synchronous gossip itself (Fig. 3).

TABLE V
DYNAMIC SCHEDULE VS. STATIC TOPOLOGIES (RASTRIGIN-32, N=16, 1000 ITERATIONS, 20 SEEDS). TORUS AND RANDOM-REGULAR VALUES ARE THE EXPERIMENT-1 RUNS (IDENTICAL CONFIGURATION AND SEEDS).

Method	f_{best}	Succ.
ring (static)	18.5 ± 4.0	1.00
rr-$k4$ (static)	19.1 ± 5.6	1.00
complete (static)	20.6 ± 4.7	1.00
torus (static)	21.1 ± 4.8	1.00
ring→rr→complete	$\mathbf{16.0 \pm 5.0}$	1.00

Takeaway. Asynchrony converts straggler stalls into raw update throughput, but uncontrolled staleness converts that throughput into noise; freshness-gated mixing recovers synchronous solution quality at asynchronous speed. In this extreme delay-to-compute regime (10^3–$10^4\times$), staleness-weighted async gossip is the best wall-clock method we tested, and the simple barrier-free freshness rule is the reason why.

Layer-3 TCP validation. Re-running the asynchronous straggler comparison over real TCP sockets (Table VII, Appendix) confirms that all Layer-2 conclusions transfer: staleness-weighted async reaches $f = 17.9$–21.6 vs. 30.7–101.5 for plain async, at a $2.1\times$ throughput cost relative to shared memory.

E. Experiment 5: Dynamic Topology

The schedule of Sec. III-F achieves the best mean final objective of every topology we tested (Table V): 16.0 ± 5.0 vs. 18.5–21.1 for the four static graphs. A methodological caveat: although all runs share seed indices, they are *not* initialization-matched—the simulator draws initial parameters after building the topology, and the schedule's random-regular phase consumes the seed RNG while static ring/complete do not—so a paired per-seed test would be invalid, and we use an unpaired rank test instead. A two-sided Mann–Whitney U test (permutation p-values, n=20 per group) favors the schedule against both of its endpoints: $U = 80/400$, $p < 10^{-3}$ vs. static complete, and $U = 119/400$, $p = 0.028$ vs. static ring (0.056 after Bonferroni correction over the two endpoint comparisons). We therefore claim the schedule at least matches the best static topology while removing the need to choose it in advance. Fig. 4 shows the mechanism. During the ring phase the population explores as in Experiment 1. At $t = 500$ the switch to random-regular produces a visible

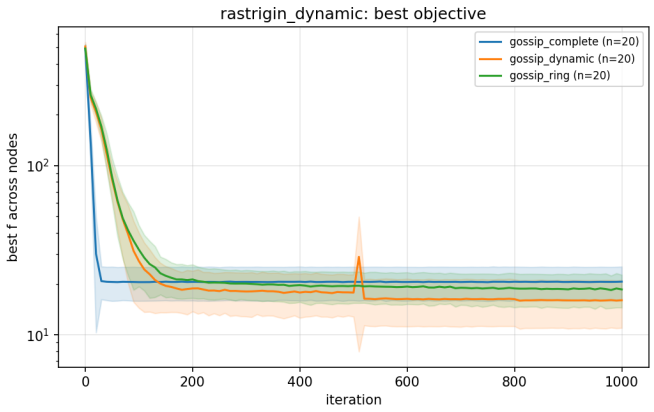


Fig. 4. Dynamic topology schedule. The dynamic run tracks ring during the exploration phase, shows a visible transient at the t=500 ring→random-regular switch, then settles below both static topologies.

quality *spike*: nodes that had settled into different basins are suddenly averaged with distant neighbors, dragging parameters off their basin floors. The population then re-converges—but now informed by every basin the ring phase found—and finishes below either static topology. The transient is the price of consensus; paying it late, after exploration has accumulated, is better than paying it from iteration zero (complete) or never completing consensus at all (ring at large N).

CartPole negative control. On the nearly unimodal CartPole-v1 task (Table VIII, Appendix), every method solves every seed, confirming that the gossip advantage is specific to multimodal landscapes. Gossip's structural benefit persists in *propagation*: independent ES leaves unlucky nodes behind (mean regret 106.6), while gossip recruits the entire population to zero regret.

VI. Discussion and Analysis

Topology is an exploration–consensus dial. The experiments support a simple reading of the spectral gap: γ sets the equilibrium between ES noise injection and mixing contraction, and through it the population's capacity for parallel basin exploration. The extremes are both pathological on multimodal landscapes—$\gamma = 1$ reproduces centralized ES's premature commitment with extra steps, $\gamma = 0$ (independent) wastes the population—while everything connected-but-sparse in between works, with at most $\sim$15% differences among gossip variants at $N = 16$.

When does sparse beat dense? At fixed iteration count, sparser is better the larger the population (Table II) and the more multimodal the objective—but ring's incomplete consensus at $N = 32$ shows the limit: pure exploration without a consensus phase leaves value on the table ($f_{\text{mean}} \gg f_{\text{best}}$). The dynamic schedule resolves exactly this tension and achieved the best mean final objective of the topologies we tested (Sec. V-E).

Asynchrony is a throughput–staleness trade—but a cheap one to fix. The stale-anchor effect (Sec. V-D) is the ES analogue of stale-gradient pathologies in asynchronous SGD, transmitted through the *mixing* step: a rarely-updated node acts as a fixed-point attractor in its neighbors' averaging. Unlike SGD, the repair needs no learning-rate correction [6]—a per-neighbor version gate suffices—and recovers synchronous quality at asynchronous speed.

Limitations. All layers run on one machine; cross-machine latency is not modeled. Results are on synthetic benchmarks in $D \leq 32$; higher-dimensional policy search may differ. We do *not* claim Gossip-ES always outperforms centralized ES—the CartPole control (Appendix) shows the advantage is specific to multimodal landscapes.

Future work. Cross-machine deployment; adaptive topology schedules reacting to measured diversity; communication-efficient seed-plus-reward payloads ($O(d)$ vs. $O(dD)$); and larger RL environments where evaluation cost dominates.

VII. Conclusion

Gossip-ES replaces the parameter server with neighbor-only averaging over a configurable graph. On budget-matched multimodal benchmarks every gossip topology beats centralized and independent ES by $\sim 15\times$, with per-node communication bounded by degree. Asynchronous gossip eliminates straggler stalls; a freshness gate on each exchange recovers synchronous quality at asynchronous speed. A sparse-to-dense topology schedule captures both ends of the exploration–consensus trade-off. The practical recipe: start sparse, densify late, gate on freshness, and treat the mixing graph as a first-class hyperparameter.

References

[1] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[2] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[3] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[4] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, 2009.

[5] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2017.

[6] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

APPENDIX A
SUPPLEMENTARY TABLES

TABLE VI
ACKLEY-32, N=16, 1000 ITERATIONS, 20 SEEDS: FINAL BEST OBJECTIVE.

Method	f_{best}	Method	f_{best}
centralized	19.55 ± 0.12	gossip torus	12.18 ± 0.85
independent	19.28 ± 0.19	gossip rr-$k4$	$\mathbf{11.47 \pm 1.13}$
gossip ring	11.52 ± 0.96	gossip complete	12.24 ± 0.86

TABLE VII
LAYER-3 VALIDATION: ASYNC GOSSIP OVER TCP/ASYNCIO (N=8 RING, D=16, 15 S BUDGET, δ=0.05 S, 5 SEEDS). TRAFFIC COLUMNS ARE *measured* ON THE WIRE, NOT ANALYTIC.

Mode	r	f_{best}	Upd. fast/slow	MB/s
async	0	101.5 ± 19.9	44 794 / —	6.4
async	0.25	30.7 ± 16.8	51 490 / 295	5.5
async-sw	0	$\mathbf{17.9 \pm 5.7}$	45 017 / —	6.4
async-sw	0.25	$\mathbf{21.6 \pm 3.3}$	51 333 / 294	5.5

TABLE VIII
CARTPOLE-V1 POLICY SEARCH (114-PARAMETER TANH POLICY, N=8, 60 ITERATIONS, 5 SEEDS). REGRET = 500 − RETURN (0 = PERFECT); SOLVED = RETURN $\geq$ 475.

Method	Best regret	Mean regret	Solved	Iters to solve
centralized	0.0 ± 0.0	**0.0**	5/5	5–35
independent	0.0 ± 0.0	106.6	5/5	10–15
gossip ring	0.0 ± 0.0	**0.0**	5/5	10–25
gossip complete	0.0 ± 0.0	**0.0**	5/5	5–35

APPENDIX B
REPRODUCIBILITY

All code, configs, and scripts are in https://github.com/HuangMason320/Gossip-ES#. Every experiment is one script:

```
pip install -r requirements.txt
python3 -m tests.test_core   # correctness suite
scripts/run_topology_ablation.sh  # Exp. 1+3
OBJECTIVE=ackley \
  scripts/run_topology_ablation.sh  # Ackley check
scripts/run_scaling.sh            # Exp. 2
scripts/run_delay.sh              # Exp. 4 (+ Fig.
    3)
scripts/run_idle.sh               # idle-time
    numbers
scripts/run_dynamic.sh            # Exp. 5 (torus/
    rr
    # rows reuse Exp. 1; run topology
    # ablation first)
scripts/significance_analysis.py  # rank tests (VI-
    F)
scripts/run_cartpole.sh           # Phase 2 (RL)
scripts/run_tcp.sh                # Layer-3 TCP
    validation
scripts/plot_all.sh               # all figures
```

Runs are seeded; Layer-1 results are bit-reproducible (Layers 2–3 are wall-clock measurements, so statistical conclusions reproduce while exact numbers vary with machine load). No GPU is required; the full Experiment-1 suite (20 seeds $\times$ 6 methods $\times$ 1000 iterations) completes in minutes on a laptop-class CPU.