Gossip-ES

Decentralized Evolution Strategies via Network Topologies

Cheng-Han Hsieh Chih-Chuan Huang
Team 19

Evolution Strategies

1. Evolution Strategies is a zero-order, black-box optimizer
2. Instead of computing gradients through the environment, ES samples many perturbed policies, evaluates their fitness independently, and uses the rewards to estimate an update direction.

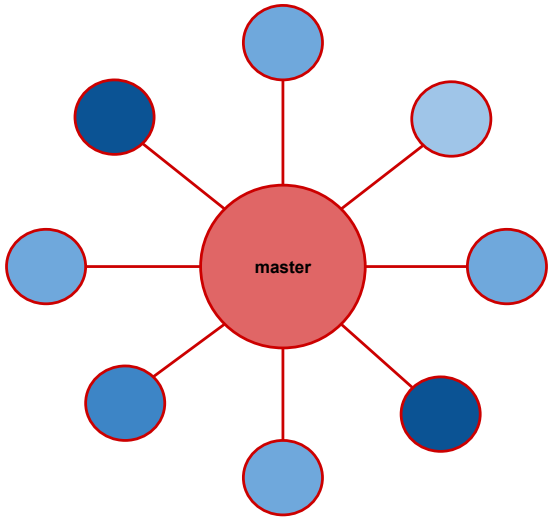
First-Order Optimization vs Evolution Strategies

1. First-order optimization uses calculus to precisely compute the gradient for descent
2. Evolution Strategies rely on massive random sampling to blindly estimate the best direction

	First-order (SGD, Adam)	Evolution Strategies (zero-order)
Information used	exact gradient $\nabla F(\theta)$	only scalar fitness $F(\theta)$
Needs differentiability?	yes	no — pure black box
"Gradient" comes from	analytic backprop	estimated from ± perturbations
Cost per step	1 forward + 1 backward	2m forward evals, no backward
Parallelism	limited (sequential backward)	evaluations embarrassingly parallel
Best suited to	differentiable losses, deep nets	RL, simulators, non-differentiable

Weakness of Evolution Strategies

1. Communication hotspot
    a. As N grows, the bottleneck is the master, not the computation
2. Single point of failure + barrier
    a. Fast nodes parked at the barrier ~99% of the time
3. One shared search center
    a. premature convergence

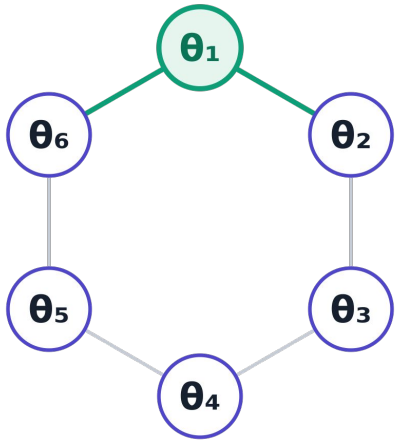


Core concept of Gossip-ES

1. No master; every node owns its own θ and runs its own ES
2. Nodes exchange with neighbors and take a weighted average
3. The topology becomes a new hyperparameter

Remove the master

1. No coordinator — every node owns its own θi and runs a full local ES step
2. Nodes mix parameters only with graph neighbors
3. N nodes become N independent search centers



θ1 mixes only with its two neighbors

One iteration, three stages

Every node repeats the same loop — no master, no central collection



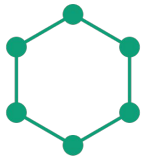
WHY IT WORKS

Metropolis weights make ***W*** doubly stochastic, so mixing is **mean-preserving** — no information is created or lost.

Topology is the knob

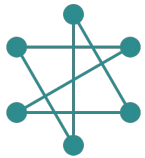
The shape of the graph sets the exploration–consensus trade-off

1. Topology is a new hyperparameter you choose, not a fixed wiring
2. Spectral gap $\gamma = 1 - |\lambda 2|$ sets how fast nodes agree



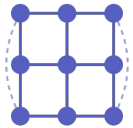
Ring

$\gamma \approx 0.05$



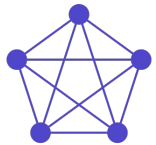
Random k-regular

$\gamma \approx 0.27$



2D-torus

$\gamma \approx 0.40$



Complete

$\gamma = 1.0$

diversity · exploration

consensus · exploitation

Every gossip topology beats both baselines ~15×

1. All four gossip graphs reach f ≈ 19–21 with 100% success; both baselines fail on every seed.
2. Centralized commits immediately (f ≈ 290), one search center locks onto the nearest basin.
3. Independent has diversity but can't exploit it (best 213, mean 289), no node shares its discovery.

Method	f_{best}	f_{mean}	$\text{Div}(T)$	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

Rastrigin-32, N=16, 1000 iterations, 20 seeds.
Final best objective (mean ± std over seeds) and mean objective across nodes, final population diversity, and success rate (f_best <= 40)

Topology trades exploration for consensus

1. Gossip converts more nodes into better optima (ring 61.5 → 11.6 for N=4 → 32); centralized stays flat at ≈ 279.
2. All connected graphs sit within ~15%; the spectral gap γ orders their diversity, not a strict quality ranking.

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

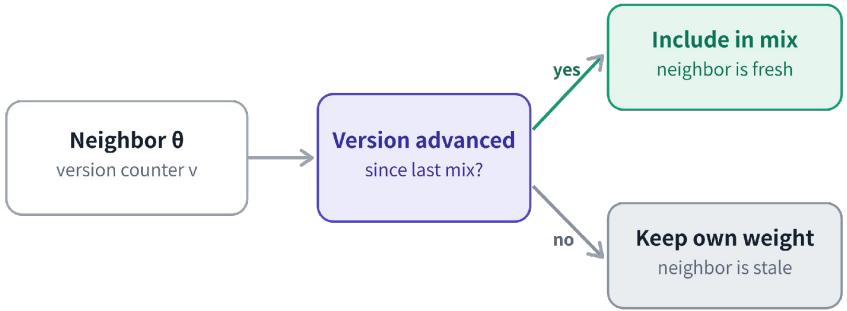
Scaling with node count (Rastrigin-32, 1000 iterations, 5 seeds): final best objective.

Hands-on challenges: Slow workers poison fast ones

1. Async removes the barrier throughput explodes:
   a. fast nodes sustain ≈110k updates, about 960× the gated sync rate.
2. Fast nodes keep re-mixing against unchanged neighbor parameters, repeatedly pulled back to the same point
3. A slow node updates ≈300× while neighbors mix against it ≈110k× — a near-frozen anchor in their averaging

Solution: A one-line freshness gate

1. **The rule**: a neighbor enters the mix only if its version advanced since the last exchange; otherwise its weight is redirected to the node itself.
2. **How it stays exact**: each parameter row carries a version counter, published atomically with the params, for free, and fully barrier-free.
3. Caps each neighbor's influence at once per real update, the frozen anchor can no longer be re-averaged in.



The barrier tax — in hard numbers

Synchronous modes are gated to the straggler; async sustains 2–3 orders of magnitude more updates

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

Layer-2 multiprocessing, N = 8 ring, D = 16. Barrier alone causes a ~170× (δ = 0.1 s: ~320×) throughput collapse — matching the analytic bound B/δ almost exactly.

~960× more updates than the gated synchronous rate (async)

99% of the 15 s budget fast nodes sit idle at the barrier (stragglers present)

4× sooner: async-sw hits synchronous quality in ~0.5 s wall-clock

Local updates per node within the 15 s budget (δ = 0.05 s). Async cells show fast / slow.

The freshness gate is what makes async work

1. Remove the gate (plain async): quality tanks to 54–78 — the stale-anchor effect
2. Add it back (async-sw): recovers to 21–27, matching synchronous gossip at full async speed
3. Centralized-sync stays trapped at 118–150 — the single-center pathology, in real processes

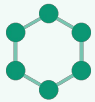
Mode	$r=0$	$r=0.25$	$r=0.5$
centralized sync	141.9 ± 35.4	150.2 ± 35.0	118.0 ± 26.2
gossip sync	$\mathbf{20.2 \pm 6.1}$	$\mathbf{19.3 \pm 7.4}$	$\mathbf{20.1 \pm 8.2}$
gossip async	53.9 ± 11.6	78.1 ± 22.5	36.4 ± 12.1
gossip async-sw	21.0 ± 6.9	23.6 ± 8.0	26.7 ± 7.6

Straggler study (Layer-2 multiprocessing, N=8 ring, D=16, fixed 15s budget, δ = 0.05s):
final best objective by straggler ratio r. 10 seeds (centralized: 3); results are insensitive to δ

Schedule the topology over time

Exploration matters early, consensus late — so switch the graph during the run instead of fixing it

PHASE 01 · EXPLORE



Ring

iters 0–50%

γ ≈ 0.05 · slow mixing keeps basins diverse

PHASE 02 · DENSIFY



Random 4-regular

iters 50–80%

γ ≈ 0.27 · good basins start sharing

PHASE 03 · CONVERGE



Complete

iters 80–100%

γ = 1.0 · fast consensus locks the optimum

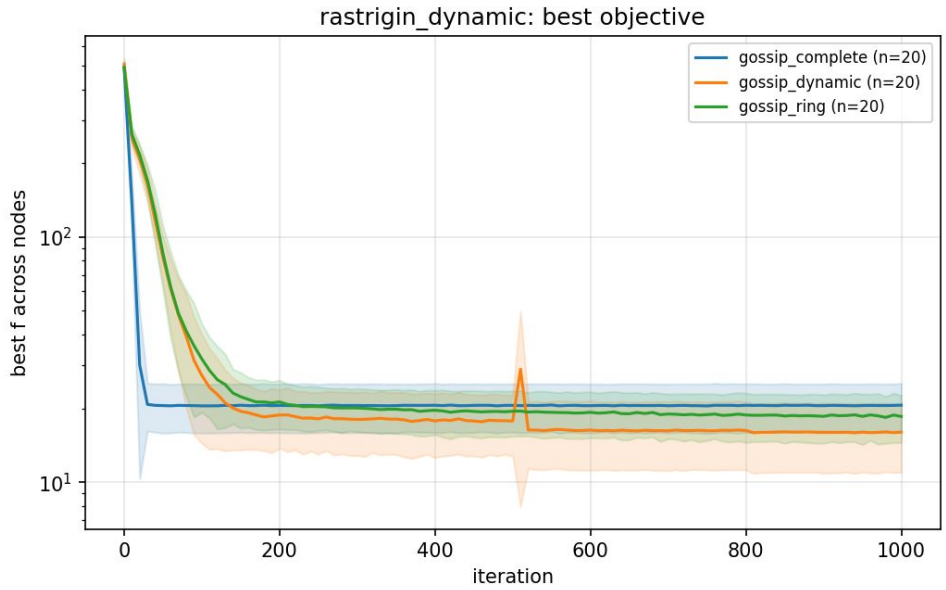
sparse · explore — dense · exploit

RESULT

Best mean final objective of every topology tested — **f = 16.0 ± 5.0**, beating both static endpoints ($p < 10^{-3}$ vs complete, $p = 0.03$ vs ring), with no topology to pick in advance.

The schedule finishes below every static graph

ring → random-regular → complete (Rastrigin-32, N=16, 20 seeds)



orange = dynamic schedule · the spike at t=500 is the ring→rr switch

Final best objective

Topology	f_best
ring (static)	18.5 ± 4.0
rr-k4 (static)	19.1 ± 5.6
complete (static)	20.6 ± 4.7
torus (static)	21.1 ± 4.8
ring→rr→complete	**16.0 ± 5.0**

All topologies: 100% success rate.

Ackley-32: the result generalizes

Secondary benchmark — one global basin instead of Rastrigin's lattice of local minima

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	11.47 ± 1.13
gossip ring	11.52 ± 0.96	gossip complete	12.24 ± 0.86

Ackley-32, N = 16, 1000 iterations, 20 seeds. Final best objective (lower is better; global optimum f = 0).

Every gossip topology reaches f ≈ 11.5–12.2 while both baselines stay above 19 **— a 1.7× margin.** The qualitative picture survives on a non-lattice landscape; the absolute gap shrinks because Ackley has a single basin to find.

CartPole: an honest negative control

1. Every method solves 5/5 (best regret 0), confirming the gossip advantage is specific to multimodal landscapes
2. The structural benefit shows in propagation: independent ES leaves unlucky nodes behind (mean regret 106.6)
3. Gossip recruits the whole population to zero regret: good basins reach every node

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

CartPole-v1 policy search (114-parameter tanh policy, N=8, 60 iterations, 5 seeds).
Regret = 500 - return (0 = perfect); solved = return >= 475.

Layer-3: it survives real TCP sockets

Engineering bonus — re-running the async straggler study over asyncio TCP confirms every Layer-2 conclusion

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	17.9 ± 5.7	45 017 / —	6.4
async-sw	0.25	21.6 ± 3.3	51 333 / 294	5.5

Per project scope, Layer 3 is a deployment validation of the Layer-2 findings, not primary evidence — but the same code would run across machines by changing the host list.

What transfers

Staleness gate is decisive over the network too: **async-sw 17.9–21.6 vs plain async 30.7–101.5.**

Cost of going to real sockets: **2.1× throughput vs shared memory** — still ~5–6 MB/s, fully barrier-free.

Version counter rides every payload, so the freshness test is exact and free.

N = 8 ring, D = 16, 15 s budget, δ = 0.05 s, 5 seeds. Traffic measured on the wire.

Contribution

- Decentralize the evolutionary strategies to improve the **robustness** and **efficiency**
- Introduce **async-sw** to prevent poisoning from the slow workers
- **Improving the performance** on function optimization tasks and a small control optimization task
- Schedule the topology from ring to complete to **beat every static topology**
- Compare **ring, torus, random-regular, and complete topologies** under a budget-matched setup
- Connect the spectral gap to the **consensus speed** and the **population diversity**

Reference

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. Nedi´c 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.

Thank You!

Have a good summer vacation ~~~