How to Organize your Deep Reinforcement Learning Agents: The Importance of Communication Topology

Dhaval Adjodah
MIT Media Lab
dval@mit.edu

Dan Calacci
MIT Media Lab
dcalacci@mit.edu

Abhimanyu Dubey
MIT Media Lab
dubeya@mit.edu

Peter Krafft
MIT Media Lab
pkrafft@mit.edu

Esteban Moro
Universidad Carlos III de Madrid
emoro@mit.edu

Alex 'Sandy' Pentland
MIT Media Lab
pentland@mit.edu

Abstract

In this empirical paper, we investigate how learning agents can be arranged in more efficient communication topologies for improved learning. This is an important problem because a common technique to improve speed and robustness of learning in deep reinforcement learning (DRL) and many other machine learning algorithms is to run multiple learning agents in parallel. The standard communication architecture typically involves all agents intermittently communicating with each other (fully connected topology) or with a centralized server (star topology). Unfortunately, optimizing the topology of communication over the space of all possible graphs is a hard problem, so we borrow results from the networked optimization and collective intelligence literatures which suggest that certain families of network topologies can lead to strong improvements over fully-connected networks. We start by introducing alternative network topologies to DRL benchmark tasks under the Evolution Strategies (ES) paradigm which we call Network Evolution Strategies (NetES). We explore the relative performance of the four main graph families and observe that one such family (Erdos-Renyi random graphs) empirically outperforms all other families, including the de facto fully-connected communication topologies. Additionally, the use of alternative network topologies has a multiplicative performance effect: we observe that when 1000 learning agents are arranged in a carefully designed communication topology, they can compete with 3000 agents arranged in the de facto fully-connected topology. Overall, our work suggests that distributed machine learning algorithms would learn more efficiently if the communication topology between learning agents was optimized.

1 Introduction

In distributed algorithms there is an implicit communication network between processing units. This network passes information such as data, parameters, or rewards between processors. The two network structures that are almost invariably used in modern distributed machine learning are either a complete network—in which all processors communicate with each other—or a star network—in which all processors communicate with a single hub server (in effect, a more efficient, centralized implementation of the complete network).

In this work, we empirically investigate whether using alternative communication topologies between processors could lead to improving learning performance in the context of deep reinforcement learning (DRL). Optimizing the communication topology between agents is a hard problem as

Preprint. Work in progress.

it involves searching over the space of all possible graphs to find a communication network that performs optimally for the learning objective under consideration. We therefore borrow results from the literatures of networked optimization (optimization over networks of agents with local rewards) [16] and collective intelligence (the study of how agents learn, influence and collaborate with each other) [27] which suggest that certain families of network topologies can lead to strong improvements over fully-connected networks.

To the best of our knowledge, almost no prior work has investigated how the topology of communication between agents affects learning performance in distributed DRL. Given that network effects tend to be significant only with large numbers of agents, we choose to build upon one of the DRL algorithms most oriented towards parallelizability and scalability: Evolution Strategies [19, 22, 25, 21]. We introduce Network Evolution Strategies (NetES), a networked decentralized variant of ES. NetES, like many DRL algorithms and evolutionary methods, relies on aggregating the rewards from a population of processors that search in parameter space to optimize a single global parameter set. Using NetES, we explore how the communication topology of a population of processors affects learning performance.

Our key findings and contributions are as follows: (1) We introduce the notion of communication network topologies to the ES paradigm for DRL tasks. (2) We run controls on all modifications to the ES algorithms to make sure that any improvements we see come exclusively from using alternative topologies. (3) We compare the learning performance of the main topological families of communication graphs, and observe that one family (Erdos-Renyi graphs) does best. (4) Using an optimized Erdos-Renyi graph, we evaluate NetES on five difficult DRL benchmarks and find large improvements compared to using a fully-connected communication topology. We observe that our 1000-agent Erdos-Renyi graph can compete with 3000 fully-connected agents. (5) We provide some theoretical insights into why alternative topologies might outperform a fully-connected communication topology.

2 Related Work

Running parallel (and sometimes asynchronous) agents is very common in modern deep reinforcement learning (DRL). For example, the Gorila framework [14] collects experiences in parallel from many agents and pools them into a global memory store on a distributed database. A3C [12] runs many agents asynchronously on several environment instances while varying their exploration policies. This effectively increases exploration diversity in parameter space and de-correlates agent learning.

Until recently, this distributed approach caused heavy communication bottlenecks and limited the number of agents that are able to be run in parallel. Black box optimization algorithms such as Evolution Strategies (ES) [19, 22, 25] are able to overcome such communication bottlenecks, and are capable of running thousands of parallel agents, while simultaneously achieving competitive performance [21]. The baseline ES algorithm has been extended by many various subsequent works, such as CMA-ES [1] that updates the covariance matrix of the gaussian distribution as well.

However, in all the approaches described above, agents are organized in a de-facto fully-connected centralized network topology: the algorithm uses and updates only one global-level parameter set using information available from all agents at every step.

There is significant evidence that the network structure of communication between nodes significantly affects the convergence rate and accuracy of learning from the literatures of decentralized optimization [16, 17, 15]. Similarly, in the collective intelligence literature, alternative network structures have been shown to result in increased exploration, higher overall maximum reward, and higher diversity of solutions in both simulated high-dimensional optimization [10] and human experiments [3]. We know of only one piece of prior work that has examined network topology in distributed machine learning [11], but network topology was only an aside in this work, and this prior work therefore presented little understanding or motivation for their brief investigation into the effect. Another recent piece of work examines the use of periodic broadcasting of successful parameter settings in deep learning but does not leverage complex network topologies [9].

3 Approach

3.1 Evolution Strategies for Deep RL

We begin with a brief overview of the application of the Evolution Strategies (ES) [22] approach to deep reinforcement learning, as done in [21]. Evolution Strategies is a class of techniques to solve optimization problems by utilizing a derivative-free parameter update approach. The algorithm proceeds by selecting a fixed model, initialized with a set of weights θ (whose distribution p_ϕ is parameterized by ϕ), and an objective (reward) function $R(\cdot)$. The ES algorithm then maximizes the average objective value $\mathbb{E}_{\theta \sim p_\phi} R(\theta)$, which is optimized with stochastic gradient ascent. The score function estimator for $\nabla_\phi \mathbb{E}_{\theta \sim p_\phi} R(\theta)$ is similar to REINFORCE [26], given by $\nabla_\phi \mathbb{E}_{\theta \sim p_\phi} R(\theta) = \mathbb{E}_{\theta \sim p_\phi}[R(\theta) \nabla_\phi \log p_\phi(\theta)]$.

The update equation used in this algorithm for the parameter θ at any iteration $t + 1$, for an appropriately chosen learning rate α and noise standar deviation σ, is a discrete approximation to the gradient:

$$\theta^{(t+1)} = \theta^{(t)} + \frac{\alpha}{N\sigma^2} \sum_{i=1}^{N} \left(R(\theta^{(t)} + \sigma\epsilon_i^{(t)}) \cdot \sigma\epsilon_i^{(t)} \right) \tag{1}$$

This update rule is normally implemented by spawning a collection of N agents at every iteration t, with perturbed versions of $\theta^{(t)}$, i.e. $\{(\theta^{(t)} + \sigma\epsilon_1^{(t)}), ..., (\theta^{(t)} + \sigma\epsilon_N^{(t)})\}$ where $\epsilon \sim \mathcal{N}(0, I)$. The algorithm then calculates $\theta^{(t+1)}$ which is broadcast again to all agents, and the process is repeated.

3.2 NetES : Networked Evolution Strategies

To maximize parameter exploration diversity, each agent can hold their own parameter $\theta_i^{(t)}$ instead of the global (noised) parameter $\theta^{(t)}$ given in the equation 1 above. At each time-step, an agent would look at the rewards and parameters of its neighbors, which we control using matrix $A = \{a_{ij}\}$, where $a_{ij} = 1$, if agents i and j communicate with each other, and 0 otherwise. A represents the *adjacency matrix* of connectivity if the networks were connected in a graph-like structure, and therefore characterizes fully the communication topology between agents. Each agent then calculates a gradient by computing a weighted average of the difference vector between its parameter and that of each of its neighbors, $((\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) - (\theta_i^{(t)}))$, using its neighbors' normalized rewards $R(\theta_j^{(t)} + \sigma\epsilon_j^{(t)})$ as weights. This leads to the update rule:

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \frac{\alpha}{N\sigma^2} \sum_{j=1}^{N} a_{ij} \cdot \left(R(\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \right) \tag{2}$$

Consequently, when agents have the same parameter (i.e. $\theta_i^{(t)} = \theta_j^{(t)}$), and the network is fully-connected (i.e. $a_{ij} = 1$), our update rule reduces to Equation 1.

In summary, we can interpret the form of Equation 1 as involving an average of the perturbations $\sigma\epsilon_i^{(t)}$ weighted by reward, such that ES corresponds to a kind of consensus-by-averaging algorithm [20]. Equation 2 is motivated by extension as corresponding to exactly the same weighted average, but averaging the differences between the agent i's neighbors' perturbed positions, $(\theta_j^{(t)} + \sigma\epsilon_j^{(t)})$, from the agent i's starting position, $\theta_i^{(t)}$.

Given the above modifications to Equation 1 to obtain Equation 2 , it is important to note that previous work has shown that the exact form of the update rule does not matter much and that sparser networks are better as long as the distributed strategy is to find and aggregate the parameters with the highest reward (as opposed to, for example, finding the most common parameters many agents hold) [3]. Therefore, although our update rule is a straightforward extension, we expect that our primary insight—that network topology can affect deep reinforcement learning—to still be useful with alternative update rules. Additionally, although Equation 2 is a biased gradient estimate, at least in the short term, it is unclear whether in practice we achieve a biased or an unbiased gradient

estimate, marginalizing over time steps between broadcasts. This is because in our full algorithm (see Appendix 1 in the supplementary material) we combine this update rule with a periodic parameter broadcast (as is common in distributed learning algorithms - we will address this in detail in a later section), and that every broadcast returns the agents to a consensus position.

Empirically, we find that NetES achieves large performance improvements. Future work can better characterize the theoretical properties of NetES and similar networked DRL algorithms using the recently developed tools of calculus on networks.

Selecting a network topology (or *adjacency matrix* A) in this context is a difficult problem – in addition to the credit assignment and exploration-exploitation dilemmas, directly optimizing for the adjacency matrix that provides highest expected rewards is non-convex, and would require substantially more computational power as the number of agents N increases.

Because almost no prior work has investigated how the topology of communication between agents affects learning performance in DRL, we believe that a starting contribution would be the empirical exploration of well-studied network topologies that are prevalent in modeling how humans and animals learn collectively. We focus on these main families of network topologies (in addition to the conventional fully-connected de facto topology): 1) Erdos-Renyi Networks: Networks where each edge between any two nodes has a fixed independent probability of being present [6]. They are among the most common graphs in social networks [8] used to define properties that hold for almost all graphs. 2) **Scale-Free Networks:** Scale-free networks, whose degree distribution follows a power law [5]. They are extremely common in systems that exhibit preferential attachment [2] such as citation and signaling biological networks. 3) **Small-World Networks:** Small-world networks where most nodes can be reached through a small number of neighbors resulting in the famous 'six degrees of separation' [24]. 4) **Fully-Connected Networks:** Networks where every node is connected to every other node.

Each of these network families can be parametrized by the number of nodes N, and their degree distribution. Erdos-Renyi networks, for example, are parametrized by their average density p ranging from 0 to 1, where 0 would lead to a completely disconnected graph (no nodes are connected), and 1.0 would lead back to a fully-connected graph. The lower p is, the sparser a network is. Similarly, the degree distribution of scale-free networks is defined by the exponent of the power distribution. Because each graph is generated randomly, two graphs with the same parameters will be different if they have different random seeds, even though, on average, they will have the same average degree (and therefore the same number of links).

4 Experimental Procedure and Reproducibility

We evaluate our algorithm on a series of popular benchmarks for deep reinforcement learning tasks, selected from two frameworks—the open source Roboschool [18] benchmark, and the MuJoCo framework [23]. The five benchmark tasks we evaluate on are: Humanoid-v1 (Roboschool and Mujoco), HalfCheetah-v1 (MuJoCo), Hopper-v1 (MuJoCo) and Ant-v1 (MuJoCo). Our choice of benchmarks is motivated by the inherent difficulty of these walker-based problems. The code we used to compute the benchmark performances is based off the freely-available code from [21].

To maximize reproducibility of our empirical results, we use the standard evaluation metric of collecting the total reward agents obtain during a test-only episode, which we compute periodically during training [13, 4, 21]. Specifically, with a probability of 0.08, we pause training, take the parameters of the best agent and run this parameter (without added noise pertubation) for 1000 episodes, and take the average total reward over all episodes - the exact same evaluation procedure of [21]. After evaluation, training is resumed with the same pre-evaluation parameters (i.e. evaluation does not change training parameters). When training eventually stabilizes to a maximum 'flat' line, we record the maximum of evaluation performance values (averaged over all episodes) during this 'flat' period as our recorded performance for this particular experimental run. As such, the training performance (as shown in Fig. 4) will be slightly lower that the corresponding maximum evaluation performance (as shown in Table 1). We observe this procedure to be quite robust to noise.

Because we are trying to evaluate the performance of communication topologies, we then repeat the former evaluation procedure for different instances of the same network topology (by varying

the random seed of network generation, we can create, at the start of each experiment, a different network topology with the same global properties) with the same average density p (i.e. the same average number of links) and the same number of nodes N. Since each node runs the same number of episode time steps per iteration, different networks with the same p can be fairly compared. We then report the average performance over 6 runs with 5-95% confidence intervals. We will share the JSON files that fully describe our experiments, including the analysis script that calculates our evaluation metric, in our code release [1].

In addition to using the evaluation procedure of [21], we also use their exact same neural network architecture: multilayer perceptrons with two 64-unit hidden layers separated by $tanh$ nonlinearities. We also keep all the modifications to the update rule introduced by [21] to improve performance: (1) training for one complete episode for each iteration; (2) employing antithetic or mirrored sampling, also known as mirrored sampling [7], where we explore $\epsilon_i^{(t)}, -\epsilon_i^{(t)}$ for every sample $\epsilon_i^{(t)} \sim \mathcal{N}(0, I)$; (3) employing fitness shaping [25] by applying a rank transformation to the returns before computing each parameter update, and (4) weight decay in the parameters for regularization. We also use the exact same hyperparameters as the original OpenAI (fully-connected and centralized) implementation [21], varying only the network topology for our experiments.

We also verify that our implementation using alternative network topologies takes approximately the same wall-clock time as when using a fully-connected network (baseline ES): although each iteration using our network takes longer because of increased communication (about 60 seconds for NetES vs 50 seconds for ES with 1000 agents each), NetES is still superior because our 1000 agents can learn at the same (higher) performance level as 3000 ES agents (which takes more than 2 minutes).
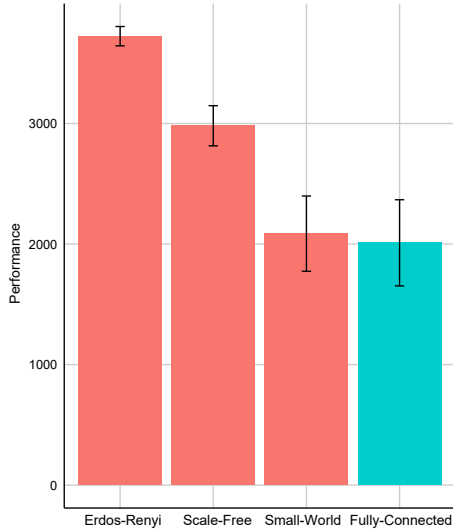

Figure 1: Learning performance on all network families: Erdos-Renyi graphs do best, fully-connected graphs do worst (Results from Mu-JoCo Ant-v1 with small networks of 100 nodes).
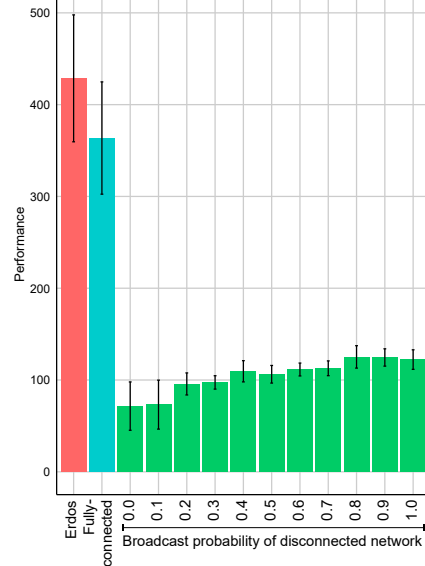
Figure 2: Agents with only broadcast do not learn (Results from 1000 agents on RoboSchool Humanoid-v1).

5 Results

In this section, we present our empirical results as per the experimental procedure described in the previous section. We first present the results showing how NetES running alternative topologies outperforms ES running fully-connected networks - for these results we use networks with 1000 agents (or more). We then run controls (generally using smaller networks of 100 nodes) on all modifications to the ES algorithm to make sure that any improvements we see come exclusively from using alternative topologies. Whenever the performance of alternative topologies are presented, we only

---

[1] our code and JSON experiment files can be found at `https://github.com/d-val/NetES`

Type	Task	Fully-connected	Erdos	Improvement %
MuJoCo	Ant-v1	4496	4938	**9.8**
MuJoCo	HalfCheetah-v1	1571	7014	**346.3**
MuJoCo	Hopper-v1	1506	3811	**153.1**
MuJoCo	Humanoid-v1	762	6847	**798.6**
Roboschool	Humanoid-v1	364	429	**17.9**

Table 1: Summary of improvements for Erdos-Renyi networks with 1000 nodes compared to fully-connected networks.

compare networks with the same number of nodes and average number of links (defined by the average density) within the same plot so that networks can be fairly commpared. Throughout this paper, we use an average network density of 0.2 for all network families and sizes of networks because it is sparse enough to provide good learning performance, and consistent (not noisy) empirical results.

5.1 Empirical performance of different network families

Using the MuJoCo Ant-v1 benchmark task (because it runs the fastest), we run a series of experiments on the different network families we previously introduced: Erdos-Renyi, scale-free, small-world and the conventional fully-connected network. For network families to be fairly compared, it is important to note that for a given average density, all networks from all topological families have the same approximate number of links (and nodes) and only the distribution of links (degree distribution) changes. Because these are exploratory experiments, we choose to run on smaller networks (number of agents, N=100). As can be seen in Fig. 1, Erdos-Renyi outperforms all other network families, and fully-connected networks (the de facto traditional network) perform worst. We present some theoretical insights as to why Erdos-Renyi networks do best in a later section.

5.2 Empirical performance on all benchmarks

Using Erdos-Renyi networks (as they previously performed best compare to other network families), we run larger networks of 1000 agents on all 5 benchmark results. As can be seen in Table 1, our Erdos-Renyi networks outperform fully-connected networks on all benchmark tasks, resulting in improvements ranging from MuJoCo 9.8% on Ant-v1 to 798% on MuJoCo Humanoid-v1. All results are statistically significant (based on 5-95% confidence intervals).

We note that the difference in performance between Erdos-Renyi and fully-connected networks is higher for smaller networks (as in Fig. 1 and Fig. 5) compared to larger networks (as in in Table 1) for the same benchmark - and we observe this behavior across different benchmarks. We believe that this is because NetES is able to achieve higher performance with fewer agents due to its efficiency of exploration, as supported in our theoretical result in a later section.

5.3 Multiplicative Learning Performance

So far, we have compared alternative network topologies with fully-connected networks containing the same number of agents. In this section, we investigate whether organizing the communication topology using Erdos-Renyi networks can outperform larger fully-connected networks. For maximum reproducibility of our results, we choose one of the benchmarks that has the lowest improvement for 1000 agents, Roboschool Humanoid-v1. As can be seen in Fig. 3 and the training curves (which display the training performance, not the evaluation metric results) in Fig. 4, an Erdos-Renyi network with 1000 agents provides comparable performance to 3000 agents arranged in a fully-connected network. This shows that networks with alternative topologies not only provide improvements over fully-connected networks, but also have a multiplicative effect on performance.
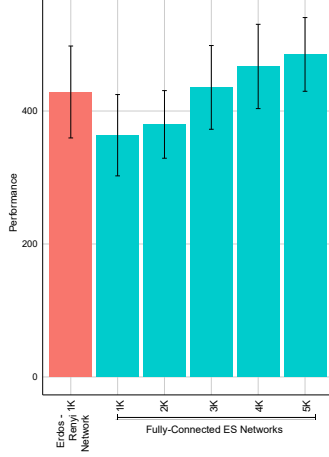
Figure 3: Evaluation results for Erdos-Renyi graph with 1000 agents compared to fully-connected networks (Results from RoboSchool Humanoid).
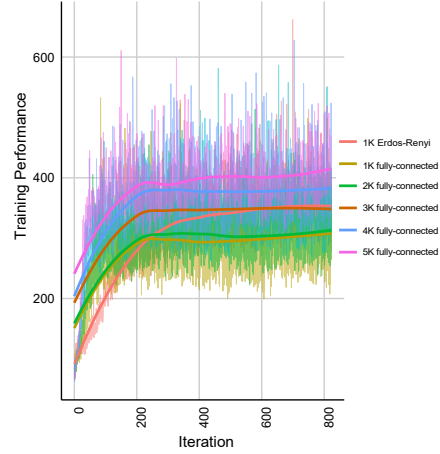
Figure 4: Training performance (not evaluation metric) for Roboschool Humanoid-v1

5.4 Control Experiments

To make sure that none of the modifications we implemented in the ES algorithm (to generalize it to be able to use alternative topologies) is causing improvements in performance instead of just the use of alternative network topologies, we run control experiments on each modification, namely: 1) the use of broadcast, 2) the fact that each agent/node has a different parameter set.

5.4.1 Broadcast effect

We implement parameter broadcast as such: at every iteration, with a probability p_b, we choose to replace all agents' current parameters with the best agent's performing weights, and then continue training (as per Equation 2) after that. Even though broadcast is common in machine learning (e.g. 'exploit' in Population-based Training [9] replaces current weights with weights with highest rewards), we want to make sure that broadcast (over different probabilities ranging from 0.0 to 1.0) does not contribute significantly to learning. Therefore, we compare 'disconnected' networks where agents can only learn from their own parameter update (they do not see the rewards and parameters of any other agents) and from broadcast with our Erdos-Renyi network and fully-connected networks of 1000 agents on the Roboschool Humanoid-v1 task. As can be seen in Fig. 2 practically no learning happens with **just** broadcast. We therefore experimentally verify that broadcast does not drive the learning performance improvement we observe when using alternative topologies. Broadcast is treated as a hyperparameter that we choose to fix in this paper as 0.8.

5.4.2 Each agent their own

The other change we introduce in NetES is to have each agent hold their own parameter set $\theta_i^{(t)}$ instead of a global (noised) parameter $\theta^{(t)}$. We therefore investigate the performance of the following 4 control baselines: fully-connected ES (as per [21]) with 100 agent running: (1) same global parameter, no broadcast; (2) same global parameter, with broadcast; (3) different parameters, with broadcast; (4) different parameters, no broadcast; compared to NetES running an Erdos-Renyi network. As can be seen in Fig 5, NetES does better than all 4 other control baselines on MuJoCo Ant-v1.
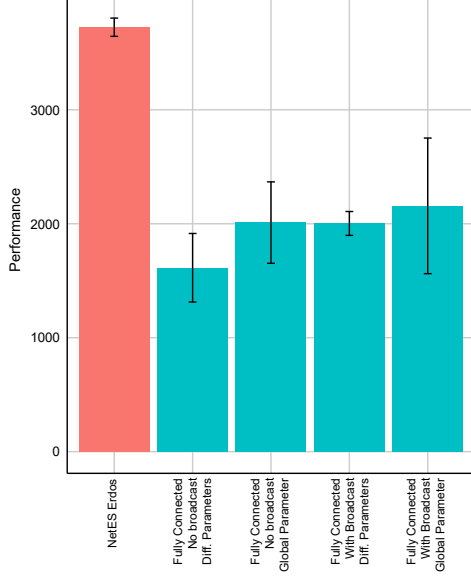
7

Figure 5: NetES using an Erdos-Renyi graph does significantly better than all 4 other control baselines (Results from MuJoCo Ant-v1 with 100 agents).
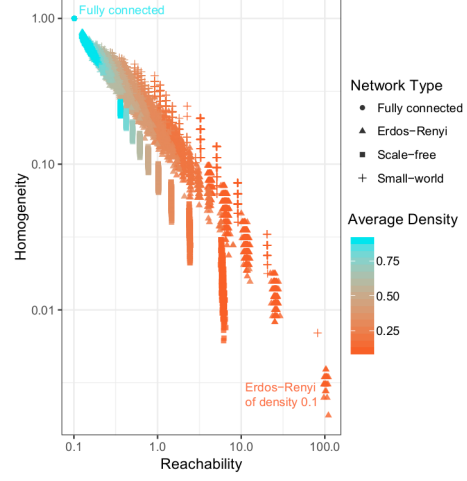

Figure 6: We generate large instances of networks (using N=100) from the three main families of networks, and observe that Erdos-Renyi graphs maximize the diversity of parameter updates.

6 Theoretical Insights

In this section, we present some intuitive theoretical insights into why alternative topologies can do better than fully-connected topologies, and why Erdos-Renyi networks outperform all other network families we have tested.

A motivating factor for introducing sparse connectivity and having each agent hold their own parameters (as per Equation 2) is to search the parameter space more effectively, a common motivation in DRL and optimization in general. One possible heuristic for measuring the capacity to explore the parameter space is the diversity of parameter updates during each iteration, which can be measured by the variance of parameter updates:

Theorem 1. *In a NetES update iteration t for a system with N agents with parameters $\Theta = \{\theta_1^{(t)}, ..., \theta_N^{(t)}\}$, agent communication matrix $A = \{a_{ij}\}$, agent-wise perturbations $\mathcal{E} = \{\epsilon_1^{(t)}, ..., \epsilon_N^{(t)}\}$, and parameter update $u_i^{(t)} = \frac{\alpha}{N\sigma^2} \sum_{j=1}^{N} a_{ij} \cdot \left(R(\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) \cdot ((\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) - (\theta_i^{(t)}))\right)$ as per Equation 2, the following relation holds:*

$$\mathsf{Var}_i[u_i^{(t)}] \leq \frac{\max^2 R(\cdot)}{N\sigma^4}\left\{\left(\frac{\|A^2\|_F}{(\min_l |A_l|)^2}\right) \cdot f(\Theta, \mathcal{E}) - \left(\frac{\min_l |A_l|}{\max_l |A_l|}\right)^2 \cdot \frac{\sigma^2}{N}(\sum_{i,j} \epsilon_i^{(t)}\epsilon_j^{(t)})\right\} \quad (3)$$

Here, $|A_l| = \sum_j a_{jl}$, $\qquad f(\Theta, \mathcal{E}) = \sqrt{(\sum_{j,k,m}\left((\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_m^{(t)}) \cdot (\theta_k^{(t)} + \sigma\epsilon_k^{(t)} - \theta_m^{(t)})\right)^2)}.$

The proof for Theorem 2 is provided in the supplementary material. This theoretical upper-bound is merely expository; it is not indicative of the *worst-case* performance, which requires the optimization of a lower-bound. We use this theoretical insight to understand the *capacity* for parameter exploration supplied by any network topology, and not to **choose** the best network topology (which would require a lower bound).

It is also important to note that the quantity in Theorem 2 is not the **variance of the value function gradient**, which is typically minimized in reinforcement learning. It is instead the **variance in the positions in parameter space** of the agents after a step of our algorithm. This quantity is more productively conceptualized as akin to a radius of exploration for a distributed search procedure

8

rather than in its relationship to the variance of the gradient. The challenge is then to maximize the search radius of positions in parameter space to find high-performing parameters. As far as the side effects this might have, given the common wisdom that increasing the variance of the value gradient in single-agent reinforcement learning can slow convergence, it is worth noting that noise (i.e. variance) is often critical for escaping local minima in other algorithms, e.g. via stochasticity in SGD.

By Theorem 2, we see that the diversity of exploration in the parameter updates across agents is affected by two quantities that involve the connectivity matrix A: the first being the term $(\|A^2\|_F/(\min_l |A_l|))^2$ (henceforth referred to as the reachability of the network), which we want to maximize, and the second being $(\min_l |A_l|/\max_l |A_l|)^2$ (henceforth referred to as the homogeneity of the network), which we want to be as small as possible in order to maximize the diversity of parameter updates across agents. Reachability and homogeneity are not independent and are statistics of the degree distribution of a graph.

Reachability is the squared ratio of the total number of paths of length 2 in A to the minimum number of links of all nodes of A. The sparser a network, the larger the reachability. For Erdos-Renyi graphs, $(\|A^2\|_F/(\min_l |A_l|))^2 \approx (pN)^{-1/2}$, where p is the average density of the network (the inverse of sparsity), the probability that any two nodes being connected.

Homogeneity is the squared ratio of the minimum to maximum connectivity of all nodes of A: the higher this value, the more homogeneously connected the graph is. The sparser a network is, the lower is the homogeneity of a network. In the case of Erdos-Renyi networks, $(\min_l |A_l|/\max_l |A_l|)^2 \approx 1 - 8\sqrt{(1-p)/(Np)}$ (the proofs and plots for Erdos-Renyi are provided in the supplementary material).

Using the above definitions for reachability and homogeneity, we generate random graphs of each network family, and plot them in Fig. 6. Two main observations can be made from this result: (1) Erdos-Renyi networks maximize reachability and minimize homogeneity, which means that they maximize the diversity of parameter exploration. (2) Fully-connected networks (which are the de facto communication network used for distributed learning) are the single worst network in terms of exploration diversity (they minimize reachability and maximize homogeneity, the opposite of what would be required for maximizing parameter exploration).

We find that this theoretical result is in accordance with our empirical results: Erdos-Renyi networks perform best, followed by scale-free networks, while fully-connected networks do worse.

7 Conclusion

In this work, we extend ES, a DRL algorithm, to run alternative topologies and empirically show that the conventional fully-connected de facto topology used in almost all machine learning algorithms is sub-optimal. We also run control experiments on all modification to the ES algorithm and show that improvements come exclusively from the use of alternative topologies. We then provide some theoretical insights as to why that might be. Overall, our work suggests that distributed machine learning algorithms would learn more efficiently if the communication topology between learning agents was optimized. Future work could explore how to learn the network structure itself, how to learn with evolving networks, and the investigation of the performance of naturally occurring (non-synthetic) network topologies such as networks of autonomous vehicles.

References

[1] Anne Auger and Nikolaus Hansen. A restart cma evolution strategy with increasing population size. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 2, pages 1769–1776. IEEE, 2005.

[2] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[3] Daniel Barkoczi and Mirta Galesic. Social learning strategies modify the effect of network structure on group performance. *Nature communications*, 7, 2016.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[5] Krzysztof Choromański, Michał Matuszak, and Jacek Miekisz. Scale-free graph with preferential attachment and evolving internal vertex structure. *Journal of Statistical Physics*, 151(6):1175–1183, 2013.

[6] P ERDdS and A R&WI. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[7] John Geweke. Antithetic acceleration of monte carlo integration in bayesian inference. *Journal of Econometrics*, 38(1-2):73–89, 1988.

[8] Yannis M Ioannides et al. Random graphs and social networks: An economics perspective. In *IUI Conference on Business and Social Networks, Vaxholm, Sweden, June*, 2004.

[9] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017.

[10] David Lazer and Allan Friedman. The network structure of exploration and exploitation. *Administrative Science Quarterly*, 52(4):667–694, 2007.

[11] Sergio Valcarcel Macua, Aleksi Tukiainen, Daniel García-Ocaña Hernández, David Baldazo, Enrique Munoz de Cote, and Santiago Zazo. Diff-dac: Distributed actor-critic for multitask deep reinforcement learning. *arXiv preprint arXiv:1710.10363*, 2017.

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. pages 1928–1937, 2016.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[14] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[15] Angelia Nedic. Asynchronous broadcast-based convex optimization over a network. *IEEE Transactions on Automatic Control*, 56(6):1337–1351, 2011.

[16] Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *arXiv preprint arXiv:1709.08765*, 2017.

[17] Angelia Nedic and Asuman Ozdaglar. 10 cooperative distributed multi-agent. *Convex Optimization in Signal Processing and Communications*, 340, 2010.

[18] OpenAI. Roboschool. `https://github.com/openai/roboschool`, 2017. Accessed: 2017-09-30.

[19] Ingo Rechenberg. Evolution strategy: Optimization of technical systems by means of biological evolution. *Fromman-Holzboog, Stuttgart*, 104:15–16, 1973.

[20] Wei Ren. Averaging algorithms and consensus. *Encyclopedia of Systems and Control*, pages 1–10, 2013.

[21] Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[22] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie: mit einer vergleichenden Einführung in die Hill-Climbing-und Zufallsstrategie*. Birkhäuser, 1977.

[23] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[24] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. In *Social Networks*, pages 179–197. Elsevier, 1977.

[25] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(1):949–980, 2014.

[26] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[27] David H Wolpert and Kagan Tumer. An introduction to collective intelligence. *arXiv preprint cs/9908014*, 1999.

8 Appendix 1 : Algorithm

Algorithm 1 Networked Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters $\theta_i^{(0)}$ where $i = 1, 2,$ ..., N (for N workers), adjacency matrix A, global broadcast probability p_b

Initialize: n workers with known random seeds, initial parameters $\theta_i^{(0)}$

for $t = 0, 1, 2,...$ **do**

    for each worker $i = 1, 2, ..., $ N **do**

        Sample $\epsilon_j^{(t)} \sim \mathcal{N}(0, I)$

        Compute returns $R_i = R(\theta_j^{(t)} + \sigma \epsilon_j^{(t)})$

    Sample $\beta^{(t)} \sim \mathcal{U}(0, 1)$

    if $\beta^{(t)} < p_b$ **then**

        Set $\theta_i^{(t+1)} \leftarrow \arg\max_{\theta_i^{(t)}} R(\theta_j^{(t)} + \sigma \epsilon_j^{(t)})$

    else

        for each worker $i = 1, 2, ..., n$ **do**

            Set $\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} + \frac{\alpha}{N\sigma^2} \sum_{j=1}^{N} a_{ij} \cdot \left(R(\theta_j^{(t)} + \sigma \epsilon_j^{(t)}) \cdot (\theta_j^{(t)} + \sigma \epsilon_j^{(t)} - \theta_i^{(t)}) \right)$

---

9 Appendix 2 : Diversity of Parameter Updates

Here we provide proofs Theorem 1 from the main paper concerning the diversity of the parameter updates.

Theorem 2. *In a multi-agent evolution strategies update iteration t for a system with N agents with parameters $\Theta = \{\theta_1^{(t)}, ..., \theta_N^{(t)}\}$, agent communication matrix $A = \{a_{ij}\}$, agent-wise perturbations $\mathcal{E} = \{\epsilon_1^{(t)}, ..., \epsilon_N^{(t)}\}$, and parameter update $u_i^{(t)}$ given by the sparsely-connected update rule:*

$$u_i^{(t)} = \frac{\alpha}{N\sigma^2} \sum_{j=1}^{N} a_{ij} \cdot \left(R(\theta_j^{(t)} + \sigma \epsilon_j^{(t)}) \cdot ((\theta_j^{(t)} + \sigma \epsilon_j^{(t)}) - (\theta_i^{(t)})) \right)$$

The following relation holds:

$$\mathsf{Var}_i[u_i^{(t)}] \quad \leq \quad \frac{\max^2 R(\cdot)}{N\sigma^4} \left\{ \left(\frac{\|A^2\|_F}{(\min_l |A_l|)^2} \right) \cdot f(\Theta, \mathcal{E}) - \left(\frac{\min_l |A_l|}{\max_l |A_l|} \right)^2 \cdot g(\mathcal{E}) \right\} \quad (4)$$

Here, $|A_l| = \sum_j a_{jl}$, $f(\Theta, \mathcal{E}) = \left(\sum_{j,k,m}^{N,N,N} ((\theta_j^{(t)} + \sigma \epsilon_j^{(t)} - \theta_m^{(t)}) \cdot (\theta_k^{(t)} + \sigma \epsilon_k^{(t)} - \theta_m^{(t)}))^2 \right)^{\frac{1}{2}}$, *and* $g(\mathcal{E}) = \frac{\sigma^2}{N} \left(\sum_{i,j}^{N,N} \epsilon_i^{(t)} \epsilon_j^{(t)} \right)$.

Proof. From Equation 2, the update rule is given by:

$$u_i^{(t)} = \frac{\alpha}{N\sigma^2} \sum_{j=1}^{N} a_{ij} \cdot \left(R(\theta_j^{(t)} + \sigma \epsilon_j^{(t)}) \cdot ((\theta_j^{(t)} + \sigma \epsilon_j^{(t)}) - (\theta_i^{(t)})) \right) \tag{5}$$

The variance of $u_i^{(t)}$ can be written as:

$$\mathsf{Var}_i[u_i^{(t)}] = \mathbb{E}_{i \in \mathcal{A}}[(u_i^{(t)})^2] - (\mathbb{E}_{i \in \mathcal{A}}[(u_i^{(t)})])^2 \tag{6}$$

Expanding $\mathbb{E}_{i \in \mathcal{A}}[(u_i^{(t)})^2]$:

$$= \frac{1}{N} \sum_{i \in \mathcal{A}} \left\{ \frac{\gamma}{N\sigma^2} \sum_{j=1}^{N} a_{ij} \cdot R(\theta_j^{(t)} + \sigma \epsilon_j^{(t)}) \cdot (\theta_j^{(t)} + \sigma \epsilon_j^{(t)} - \theta_i^{(t)}) \right\}^2 \tag{7}$$

Simplifying:

$$= \frac{1}{N\sigma^4} \sum_{i,j,k} \left(\frac{a_{ij}a_{ik}}{|A_i|^2} R(\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) R(\theta_k^{(t)} + \sigma\epsilon_k^{(t)}) \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \cdot (\theta_k^{(t)} + \sigma\epsilon_k^{(t)} - \theta_i^{(t)}) \right) \tag{8}$$

Since $R(\cdot) \leq \max R(\cdot)$, therefore:

$$\leq \frac{\max^2 R(\cdot)}{N\sigma^4} \sum_{i,j,k} \frac{a_{ij}a_{ik}}{|A_i|^2} \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \cdot (\theta_k^{(t)} + \sigma\epsilon_k^{(t)} - \theta_i^{(t)}) \tag{9}$$

$$\leq \frac{\max^2 R(\cdot)}{N\sigma^4} \sum_{i,j,k} \frac{a_{ij}a_{ik}}{\min_l |A_l|^2} \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \cdot (\theta_k^{(t)} + \sigma\epsilon_k^{(t)} - \theta_i^{(t)}) \tag{10}$$

By the Cauchy-Schwarz Inequality:

$$\mathbb{E}_{i \in \mathcal{A}}[(u_i^{(t)})^2] \leq \frac{\max^2 R(\cdot)}{N\sigma^4} \left(\sum_{i,j,k} \frac{(a_{ij}a_{ik})^2}{\min_l |A_l|^4} \right)^{\frac{1}{2}} \cdot \left(\sum_{i,j,k} \left((\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \cdot (\theta_k^{(t)} + \sigma\epsilon_k^{(t)} - \theta_i^{(t)}) \right)^2 \right)^{\frac{1}{2}} \tag{11}$$

Since $a_{ij} \in \{0,1\} \forall (i,j)$, $(a_{ij}a_{ik})^2 = a_{ij}a_{ik} \forall (i,j,k)$. Additionally, we know that $a_{ij} = a_{ji}$, since A is symmetric. Therefore, $\sum_i a_{ij}a_{ik} = \sum_i a_{ji}a_{ik} = A_{jk}^2$. Using this:

$$\mathbb{E}_{i \in \mathcal{A}}[(u_i^{(t)})^2] \leq \frac{\max^2 R(\cdot)}{N\sigma^4} \cdot \left(\frac{|A^2|^{\frac{1}{2}}}{\min_l |A_l|^2} \right) \cdot \left(\sum_{i,j,k} \left((\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \cdot (\theta_k^{(t)} + \sigma\epsilon_k^{(t)} - \theta_i^{(t)}) \right)^2 \right)^{\frac{1}{2}} \tag{12}$$

Replacing $\left(\sum_{i,j,k} \left((\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \cdot (\theta_k^{(t)} + \sigma\epsilon_k^{(t)} - \theta_i^{(t)}) \right)^2 \right)^{\frac{1}{2}} = f(\Theta, \mathcal{E})$, where $\Theta = \{\theta_i^{(t)}\}_{i=1}^N, \mathcal{E} = \{\epsilon_i\}_{i=1}^N$ for compactness, we obtain:

$$\mathbb{E}_{i \in \mathcal{A}}[(u_i^{(t)})^2] \leq \frac{\max^2 R(\cdot)}{N\sigma^4} \cdot \left(\frac{|A^2|^{\frac{1}{2}}}{\min_l |A_l|^2} \right) \cdot f(\Theta, \mathcal{E}) \tag{13}$$

Similarly, the squared expectation of $(u_i^{(t)})$ over all agents can be given by:

$$(\mathbb{E}_{i \in \mathcal{A}}[u_i^{(t)}])^2 = \left(\frac{1}{N} \sum_{i \in \mathcal{A}} \left\{ \frac{\gamma}{N\sigma^2} \sum_{j=1}^N a_{ij} \cdot R(\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \right\} \right)^2 \tag{14}$$

$$= \frac{1}{N^2\sigma^4} \left(\sum_{i \in \mathcal{A}} \left\{ \frac{1}{|A_i|} \sum_{j=1}^N a_{ij} \cdot R(\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \right\} \right)^2 \tag{15}$$

$$= \frac{1}{N^2\sigma^4} \left(\sum_{i,j} \left\{ \frac{a_{ij}}{|A_i|} \cdot R(\theta_j^{(t)} + \sigma\epsilon_j^{(t)}) \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \right\} \right)^2 \tag{16}$$

Since $R(\cdot) \geq \min R(\cdot)$, therefore:

$$\geq \frac{\min^2 R(\cdot)}{N^2\sigma^4} \left(\sum_{i,j} \left\{ \frac{a_{ij}}{|A_i|} \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \right\} \right)^2 \tag{17}$$

$$\geq \frac{\min^2 R(\cdot)}{N^2\sigma^4 \max_l |A_l|^2} \left(\sum_{i,j} \left\{ a_{ij} \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) \right\} \right)^2 \tag{18}$$

Since A is symmetric, $\sum_{i,j}^{N,N} a_{ij} \cdot (\theta_j^{(t)} + \sigma\epsilon_j - \theta_i^{(t)}) = \sum_{i,j}^{N,N} a_{ij} \cdot (\theta_i^{(t)} + \sigma\epsilon_i - \theta_j^{(t)})$. Therefore:

$$= \frac{\min^2 R(\cdot)}{N^2\sigma^4 \max_l |A_l|^2} \Big(\sum_{i,j} \frac{1}{2} \{ a_{ij} \cdot (\theta_j^{(t)} + \sigma\epsilon_j^{(t)} - \theta_i^{(t)}) + a_{ij} \cdot (\theta_i^{(t)} + \sigma\epsilon_i^{(t)} - \theta_j^{(t)}) \} \Big)^2 \quad (19)$$

Therefore,

$$(\mathbb{E}_{i \in \mathcal{A}}[u_i^{(t)}])^2 \quad = \quad \frac{\min^2 R(\cdot)}{N^2\sigma^2 \max_l |A_l|^2} \Big(\sum_{i,j} \frac{1}{2} \{ a_{ij} \quad \cdot \quad (\epsilon_j^{(t)} \quad + \quad \epsilon_i^{(t)}) \} \Big)^2 \quad (20)$$

Using the symmetry of A, we have that $\sum_{i,j}^{N,N} a_{ij}\epsilon_i = \sum_{i,j}^{N,N} a_{ij}\epsilon_j$. Therefore:

$$= \frac{\min^2 R(\cdot)}{N^2\sigma^2 \max_l |A_l|^2} \Big(\sum_{i,j} a_{ij} \cdot \epsilon_j^{(t)} \Big)^2 \quad (21)$$

$$= \frac{\min^2 R(\cdot)}{N^2\sigma^2 \max_l |A_l|^2} \Big(\sum_{j} |A_j| \cdot \epsilon_j^{(t)} \Big)^2 \quad (22)$$

$$\geq \frac{\min^2 R(\cdot) \min_l |A_l|^2}{N^2\sigma^2 \max_l |A_l|^2} \Big(\sum_{i,j} \epsilon_i^{(t)}\epsilon_j^{(t)} \Big) \quad (23)$$

Combining both terms of the variance expression, and using the normalization of the iteration rewards that ensures $\min R(\cdot) = -\max R(\cdot)$, we can obtain (using $g(\mathcal{E}) = \frac{\sigma^2}{N}\Big(\sum_{i,j} \epsilon_i^{(t)}\epsilon_j^{(t)} \Big)$):

$$\mathsf{Var}_{i \in \mathcal{A}}[u_i^{(t)}] \quad \leq \quad \frac{\max^2 R(\cdot)}{N\sigma^4} \Big\{ \Big(\frac{|A^2|^{\frac{1}{2}}}{\min_l |A_l|^2} \Big) \quad \cdot \quad f(\Theta, \mathcal{E}) \quad - \quad \Big(\frac{\min_l |A_l|^2}{\max_l |A_l|^2} \Big) \quad \cdot \quad g(\mathcal{E}) \Big\} \quad (24)$$

$\square$

10 Appendix 3 : Approximating Reachability and Homogeneity for Large Erdos-Renyi Graphs

Recall that a Erdos-Renyi graph is constructed in the following way

1. Take n nodes
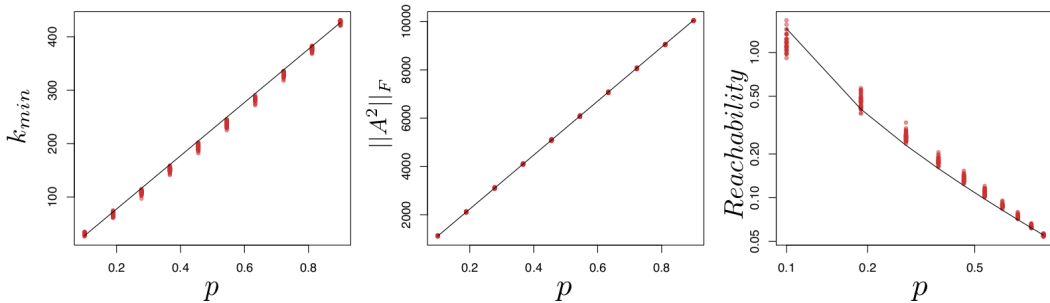2. For each pair of nodes, link them with probability p

Figure 7: Comparison between the values of $k_{min}, ||A^2||_F$, and Reachability as a function of p for different realizations of the Erdos-Renyi model (points) and their approximations given in Equations (26), (25) and (27) respectively (lines).

The model is simple, and we can infer the following:

- The average degree of a node is $p(n-1)$

- The distribution of degree for the nodes is the Binomial distribution of $n-1$ events with probability p, $B(n-1,p)$.

- The (average) number of paths of length 2 from one node i to a node $j \neq i$ ($n_{ij}^{(2)}$) can be calculated this way: a path of length two between i and j involves a third node k. Since there are $n-2$ of them, the maximun number of paths between i and j is $n-2$. However, for that path to exists there has to be a link between i and k and k and j, an event with probability p^2. Thus, the average number of paths between i and j is $p^2(n-2)$

Estimating Reachability

We can then estimate Reachability:

$$Reachability = \frac{||A^2||_F}{(min_l|A_l|)^2} = \frac{\sqrt{\sum_{i,j} n_{ij}^{(2)}}}{k_{min}^2}$$

where $k_{min} = (min_l|A_l|)$ is the minimum degree in the network. Given the above calculations we can approximate

$$\sum_{i,j} n_{ij}^{(2)} = \sum_i n_{ii}^{(2)} + \sum_{i \neq j} n_{ij}^{(2)} \approx n \times [p(n-1)] + n(n-1) \times [p^2(n-2)]$$

where the first term is the number of paths of length 2 from i to i summed over all nodes, i.e. the sum of the degrees in the network. The second term is the sum of $p^2(n-2)$ for the terms in which $i \neq j$. For large n we have that

$$\sum_{i,j} n_{ij}^{(2)} \approx p^2 n^3$$

and thus,

$$||A^2||_F \approx \sqrt{p^2 n^3}. \tag{25}$$

For the denominator k_{min} we could use the distribution of the minimum of the binomial distribution $B(n-1,p)$. However, since it is a complicated calculation we can approximate this way: since the binomial distribution $B(n-1,p)$ looks like a Gaussian, we can say that the minimum of the distribution is closed to the mean minus two times the standard deviation:

$$k_{min} \approx p(n-1) - 2\sqrt{p(n-1)(1-p)} \tag{26}$$

Once again in the case of large n we have

$$k_{min} \approx pn$$

Thus

$$Reachability \approx \frac{\sqrt{p^2 n^3}}{[p(n-1) - 2\sqrt{p(n-1)(1-p)}]^2} \tag{27}$$

As we can see in the figure those approximations work very well for realizations of the Erdos-Renyi networks.

Assuming that n is large, we can approximate

$$Reachability \approx \frac{pn^{3/2}}{p^2 n^2} = \frac{1}{pn^{1/2}}$$

Thus the bound decreases with increasing n and p. Note that the density of the Erdos-Renyi graph (the number of links over the number of possible links) is p. And thus for a fixed n more sparse networks $p \simeq 0$ have larger Reachability than more connected networks $p \simeq 1$.

Estimating Homogeneity

The Homogeneity is defined as

$$Homogeneity = \left(\frac{k_{min}}{k_{max}} \right)^2$$

As before we can approximate

$$k_{max} \approx p(n-1) + 2\sqrt{p(n-1)(1-p)}$$

And thus

$$Homogeneity \approx \left(\frac{p(n-1) - 2\sqrt{p(n-1)(1-p)}}{p(n-1) + 2\sqrt{p(n-1)(1-p)}} \right)^2$$

For large p we can approximate it to be

$$Homogeneity \approx 1 - 8\frac{\sqrt{1-p}}{\sqrt{np}} \tag{28}$$

which shows that for $p \simeq 1$ we have that Homogeneity grows as a function of p. Thus for fixed number of nodes n, increasing p we get larger values of the Homogeneity. See figure 2
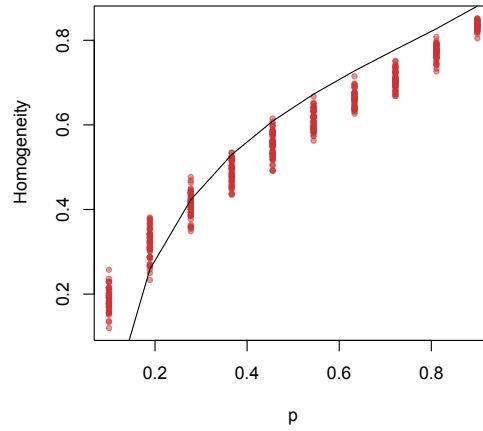

Figure 8: Comparison for the Homogeneity in the Erdos-Renyi case for different values of p and $n = 500$. Points correspond to the real data, while the lines are the approximations given by Equation (28).