entropy

MDPI

Article

Extreme Value Statistics of Community Detection in Complex Networks with Reduced Network Extremal Ensemble Learning (RenEEL)

Tania Ghosh [1,2], Royce K. P. Zia [1,3] and Kevin E. Bassler [1,2,4,*]

[1] Department of Physics, University of Houston, Houston, TX 77204, USA; tghosh3@uh.edu (T.G.); rkpzia@vt.edu (R.K.P.Z.)
[2] Texas Center for Superconductivity, University of Houston, Houston, TX 77204, USA
[3] Center for Soft Matter and Biological Physics, Department of Physics, Virginia Tech, Blacksburg, VA 24061, USA
[4] Department of Mathematics, University of Houston, Houston, TX 77204, USA
* Correspondence: bassler@uh.edu; Tel.: +1-713-743-3568

Abstract: Arguably, the most fundamental problem in Network Science is finding structure within a complex network. Often, this is achieved by partitioning the network's nodes into communities in a way that maximizes an objective function. However, finding the maximizing partition is generally a computationally difficult NP-complete problem. Recently, a machine learning algorithmic scheme was introduced that uses information within a set of partitions to find a new partition that better maximizes an objective function. The scheme, known as RenEEL, uses Extremal Ensemble Learning. Starting with an ensemble of K partitions, it updates the ensemble by considering replacing its worst member with the best of L partitions found by analyzing a reduced network formed by collapsing nodes, which all the ensemble partitions agree should be grouped together, into super-nodes. The updating continues until consensus is achieved within the ensemble about what the best partition is. The original K ensemble partitions and each of the L partitions used for an update are found using a simple "base" partitioning algorithm. We perform an empirical study of how the effectiveness of RenEEL depends on the values of K and L and relate the results to the extreme value statistics of record-breaking. We find that increasing K is generally more effective than increasing L for finding the best partition.

Keywords: community detection; complex networks; modularity; extreme value statistics

Received: 6 November 2024
Revised: 28 May 2025
Accepted: 11 June 2025
Published: 13 June 2025

Citation: Ghosh, T.; Zia, R.K.P.; Bassler, K.E. Extreme Value Statistics of Community Detection in Complex Networks with Reduced Network Extremal Ensemble Learning (RenEEL). *Entropy* **2025**, *27*, 628. https://doi.org/10.3390/e27060628

1. Introduction

Finding the community structure within a complex network that relates to its function or behavior is a fundamental challenge in Network Science [1–5]. It is a highly non-trivial problem, as even defining what one means by "structure" must be specified [6]. A variety of approaches can be used to partition the nodes of a network into communities [7,8]. Each approach will divide the nodes according to a different definition of the structure and, in general, will find a different partition [9,10]. Often, the goal is to find a partition that maximizes an objective function. However, finding such a partition can be a computationally difficult NP-complete problem [11,12]. Finding a guaranteed exact solution for a large network is therefore generally infeasible. Thus, for practical applications, it is important to have an approximate algorithm that has polynomial-time complexity, i.e., is fast, and that finds near-exact best solutions for large networks, i.e., is accurate.

Recently, an algorithmic scheme has been introduced that uses information in an ensemble of partitions to produce a better, more accurate partition when seeking to maximize an objective function. The scheme, Reduced Network Extremal Ensemble Learning (RenEEL) [13], uses a machine learning paradigm for graph partitioning, Extremal Ensemble Learning (EEL). EEL begins with an ensemble $\mathcal{P}_K$ of $k \leq K$ unique partitions. It then iteratively updates the ensemble using extremal criteria until consensus is reached within the ensemble about what the "best" partition is, i.e., the one with the largest value of the objective function. Each update considers adding a new partition P to $\mathcal{P}_K$ as follows: If $P \in \mathcal{P}_K$, then the "worst" partition in $\mathcal{P}_K$, P_{worst}, is removed from $\mathcal{P}_K$, reducing the size of $\mathcal{P}_K$ by one $k \to k-1$, and the update is complete. If $P \notin \mathcal{P}_K$ and P is worse than P_{worst}, then again P_{worst} is removed from $\mathcal{P}_K$, reducing the size of $\mathcal{P}_K$ by one $k \to k-1$, and the update is complete. If $P \notin \mathcal{P}_K$ and P is better than P_{worst}, then if $k = K$, P_{worst} is replaced by P in $\mathcal{P}_K$ and the update is complete, or if $k < K$, P is added to $\mathcal{P}_K$, increasing the size of $\mathcal{P}_K$ by one $k \to k+1$, and the update is complete. Iterative updates continue until $k = 1$. The remaining partition is the consensus choice for the best partition.

To ensure fast convergence to a consensus choice in EEL updates, RenEEL conserves the consensus within $\mathcal{P}_K$ that exists up to that point each time it finds a new partition to be used in an update. It achieves this by partitioning a *reduced network* rather than the original network. The reduced network is constructed by collapsing the nodes that every partition in $\mathcal{P}_K$ agrees should be in the same community into "super" nodes. Reduced networks are smaller than the original network and can be analyzed faster, focusing effort only on improving partitioning where there is disagreement within $\mathcal{P}_K$. The consensus within $\mathcal{P}_K$ increases monotonically, and the size of the reduced networks decreases monotonically as the EEL updates are made. RenEEL creates an ensemble $\mathcal{P}_L$ consisting of L partitions found by analyzing the reduced network to decide the partition to use in each EEL update. The best partition in $\mathcal{P}_L$ is then used in the update.

There is wide flexibility within the RenEEL scheme. A *base algorithm* is used to find the partitions that initially form the $\mathcal{P}_K$ ensemble and those that form each $\mathcal{P}_L$ ensemble. The base algorithm can be any algorithm that finds a partition that maximizes an objective function. Multiple base algorithms can even be used. There is also freedom in choosing the values of K and L, which represent the maximum size of $\mathcal{P}_K$ and the size of each $\mathcal{P}_L$, respectively. The best base algorithm choice and values of K and L depend on the network being analyzed, desired accuracy, and available computational resources.

This paper investigates the effect of varying K and L on the performance of RenEEL. Larger values of K and L will typically lead to a final, consensus best partition with larger values of the objective function [13]. But how does the value of the objective function of the consensus partition reached typically depend on K and L? How quickly is the value found expected to approach its true maximum value? Given only limited computational resources, is it better to increase K or L? We empirically study these questions when seeking the partition of three well-known real-world networks that maximizes the objective function Modularity.

2. Results

A commonly used approach to find structure in a complex network is to partition the nodes into communities that are more densely connected than expected in a random network. In this approach, the community structure corresponds to the partition that maximizes an objective function called *Modularity* [2,14]. For a given nodal partition $C \equiv \{c_1, c_2, \ldots\}$, Modularity q is defined as

$$q(C) = \frac{1}{2m} \sum_{\langle ij \rangle} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta_{c_i c_j}, \tag{1}$$

where the sum is over all pairs of nodes $\langle ij \rangle$, c_i is the community of the ith node, and m is the total number of links present in the network. k_i and A_{ij} are, respectively, the degree of the ith node and the ijth element of the adjacency matrix. Thus, Modularity is the difference between the fraction of links inside the partition's communities, the first term in Equation (1), and what the expected fraction would be if all links of the network were randomly placed, the second term in Equation (1). The task is to find the partition C that maximizes q. We denote the maximum value of q as Q, the value of which is called "the Modularity" of the network.

A number of algorithms with polynomial-time complexity have been developed to find a partition that maximizes Modularity. They range from very fast but not-so-accurate algorithms, such as the Louvain method [15] or randomized greedy agglomerative hierarchical clustering [16], to more accurate but slower algorithms [17], such as one that combines both agglomeration and division steps [18,19]. The accuracy of all of the algorithms tends to decrease as the size of the network increases.

All of the fast Modularity-maximizing algorithms are stochastic because at intermediate steps of their execution, there are seemingly equally good choices to make that are randomly made. In the end, those choices can be consequential because different runs of an algorithm with different sets of random intermediate choices can result in different solutions. Because of this, multiple runs of an algorithm are often made, say 100, to analyze a network, producing an ensemble of approximate partitions. The partition in the ensemble with the largest Modularity is then taken as the network's community structure, while all other partitions in the ensemble are discarded. RenEEL instead uses the information within the ensemble to find a more accurate partition.

Here we use a RenEEL algorithm that has a randomized greedy base algorithm [16] to find the community structure by maximizing Modularity in real-world networks A, B, and C. Network A is the As-22july06 Network [20]. It is a snapshot in time of the structure of the Internet at the level of autonomous systems. It has 22,963 nodes, which represent autonomous systems, and 48,436 links of data connection. Network B is the PGP Network [20]. It is a snapshot in time of the giant component of the Pretty-Good-Privacy (PGP) algorithm user network. It has 10,680 nodes, which are the users of the PGP algorithm, and 24,316 links indicating the interactions among them. Lastly, Network C is the Astro-ph network. It is a coauthorship network of scientists in Astrophysics consisting of 16,706 nodes representing scientists and 12,1251 links representing coauthorship in preprints on the Astrophysics arXiv database [21].

For each of the three networks, 300 different runs of RenEEL were made for independent values of K and L of 10, 20, 40, 80, 160, and 320, respectively. The compute time required to find the consensus partition was measured for each run. The mean and standard errors of the compute times for the runs at a given value of K and L were then calculated. The full results are listed in Tables A1, A4 and A7 in Appendix A. For a fixed value of L or K, we find that the mean compute times $\langle t \rangle$ increase asymptotically as a power of the other ensemble size,

$$\langle t \rangle \sim K^{\alpha_K}|_{L \text{ fixed}} \quad \text{and} \quad \langle t \rangle \sim L^{\alpha_L}|_{K \text{ fixed}}. \tag{2}$$

For example, Figure 1 shows this power-law behavior for Network A when L and K have fixed values of 80.

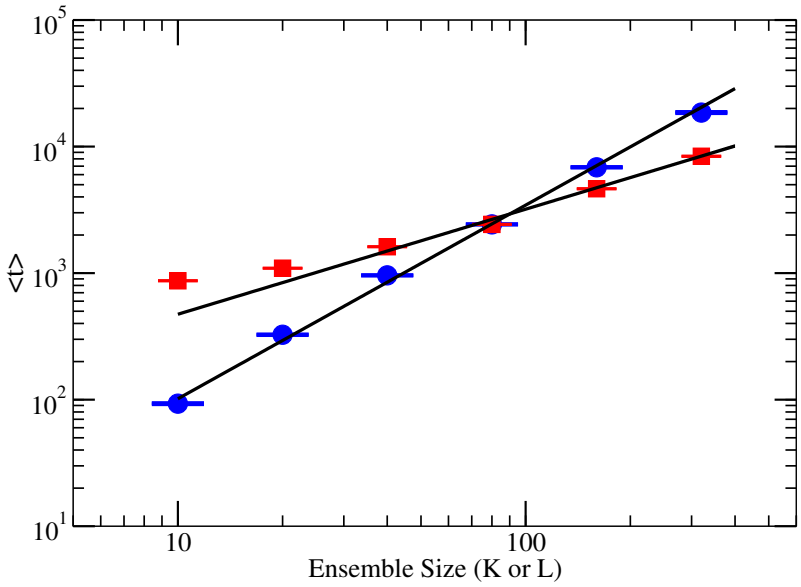

Figure 1. The compute time for RenEEL to complete when analyzing Network A. Data are measured in seconds. Blue circles are results as a function of K for fixed $L = 80$, and red squares are results as a function of L for fixed $K = 80$. The straight lines are power-law fits to the data of ensemble sizes greater than 10. The slope of the fit to the blue circles is $\alpha_K = 1.489(4)$, and to the red squares is $\alpha_L = 0.810(3)$.

Two-parameter, nonlinear least-squares fits to data for ensemble sizes greater than 10 were then used to determine the proportionality constant and α. Table 1a,b show the values for α_K and α_L, respectively, that result from fits at different fixed values of L and K for each of the three networks. All statistical errors reported in this paper are $\pm 2\sigma$.

Table 1. (a) Values of scaling exponent α_K of the compute time divergence at fixed values of L. (b) Values of scaling exponent α_L of the compute time divergence at fixed values of K.

(**a**)			
L	**Network A**	**Network B**	**Network C**
20	1.382(2)	1.262(2)	1.331(4)
40	1.431(2)	1.212(2)	1.300(4)
80	1.489(4)	1.240(2)	1.300(2)
160	1.483(2)	1.164(2)	1.320(4)
320	1.469(2)	1.194(4)	1.311(6)
(**b**)			
K	**Network A**	**Network B**	**Network C**
20	0.732(3)	0.800(6)	0.821(4)
40	0.752(2)	0.820(4)	0.872(4)
80	0.810(3)	0.851(2)	0.900(8)
160	0.810(4)	0.800(2)	0.924(4)
320	0.792(3)	0.879(4)	0.898(6)

The values of the exponents $\alpha_K(L)$ and $\alpha_L(K)$ weakly vary with the value of L and K, respectively. The standard errors of $\alpha_K(L)$ and $\alpha_L(K)$ tend to remain consistent between smaller and larger ensemble sizes. The distribution of computed time, as depicted in Figure A1 for Network A, does not follow a normal distribution. Consequently, increasing the ensemble size does not lead to a decrease in the standard error. For each network, however, the value of α_K is significantly larger than α_L. Thus, the expected compute time increases faster with K than L. Given that larger values of K and L typically lead to a better

result, i.e., a consensus partition with a larger Q, one might naively conclude that it is better to increase L rather than K. But, to determine if that conclusion is, in fact, correct, the way that Q increases with K and L must be taken into account.

To this end, we begin by noting that for any finite-size network, there is only a finite number of possible partitions. Many modularity-maximizing algorithms will consistently find the actual best partition for very small networks. As the network size and number of possible partitions grow, the task becomes harder; algorithms start to fail to find the exact solution and only provide estimates of the actual, or exact, best partition. RenEEL appears to perform very well at finding the actual best partition of networks of sizes of up to a few thousand nodes [13]. Still, even RenEEL can only find estimates of the exact best partition of larger networks, such as the three we analyze in this paper. As values of K and L increase, the estimates improve, and the value of Q of the consensus partition approaches $Q_{\max}$, the Modularity of the exact best partition. To explore how the values of Q of RenEEL's consensus partitions approach $Q_{\max}$ as a function of K and L, the mean and standard errors of Q found in the runs that were made on each network were calculated as a function of K and L. The results are listed in Tables A2, A3, A5, A6, A8 and A9 in Appendix A. For a fixed value of L or K, we find that Q approaches a maximum value, $Q_{\max}$, as a power-law of the other ensemble size,

$$\begin{aligned} Q &\approx Q_{max} - A_K K^{-\gamma_K}\big|_{L \text{ fixed}} \\ \text{and} & \\ Q &\approx Q_{max} - A_L L^{-\gamma_L}\big|_{K \text{ fixed}}, \end{aligned} \tag{3}$$

where the As are constants. Figure 2 shows this behavior for Network A when L and K have fixed values of 80. The exact value of $Q_{\max}$ is unknown for Networks A, B, and C. Three-parameter, nonlinear least-squares fits were used to determine the values of $Q_{\max}$, A, and γ. Table 2a,b list the values of $Q_{\max}$ and γ_K, respectively, that result from fits at fixed values of L for each of the three networks. Similarly, Table 3a,b list the values of $Q_{\max}$ and γ_L that result from fits at fixed values of K for each of the three networks.

Table 2. (a) Values of asymptotic maximum Modularity Q_{max} found at fixed values of L. (b) Values of scaling exponent γ_K of the Modularity convergence to $Q_{\max}$ at fixed values of L.

(**a**)			
L	**Network A**	**Network B**	**Network C**
10	0.679265(4)	0.886761(2)	0.742303(12)
20	0.679309(6)	0.886784(6)	0.742589(18)
40	0.679320(6)	0.886814(2)	0.742829(16)
80	0.679368(6)	0.886825(4)	0.742956(18)
160	0.679370(8)	0.886832(2)	0.743013(16)
320	0.679377(4)	0.886834(2)	0.743168(20)
(**b**)			
L	**Network A**	**Network B**	**Network C**
10	1.21(2)	2.16(4)	1.18(18)
20	1.29(2)	2.15(6)	1.31(20)
40	1.38(2)	2.01(8)	1.28(20)
80	1.34(2)	2.04(4)	1.71(16)
160	1.35(4)	2.29(14)	1.22(18)
320	1.25(2)	2.05(6)	1.29(20)

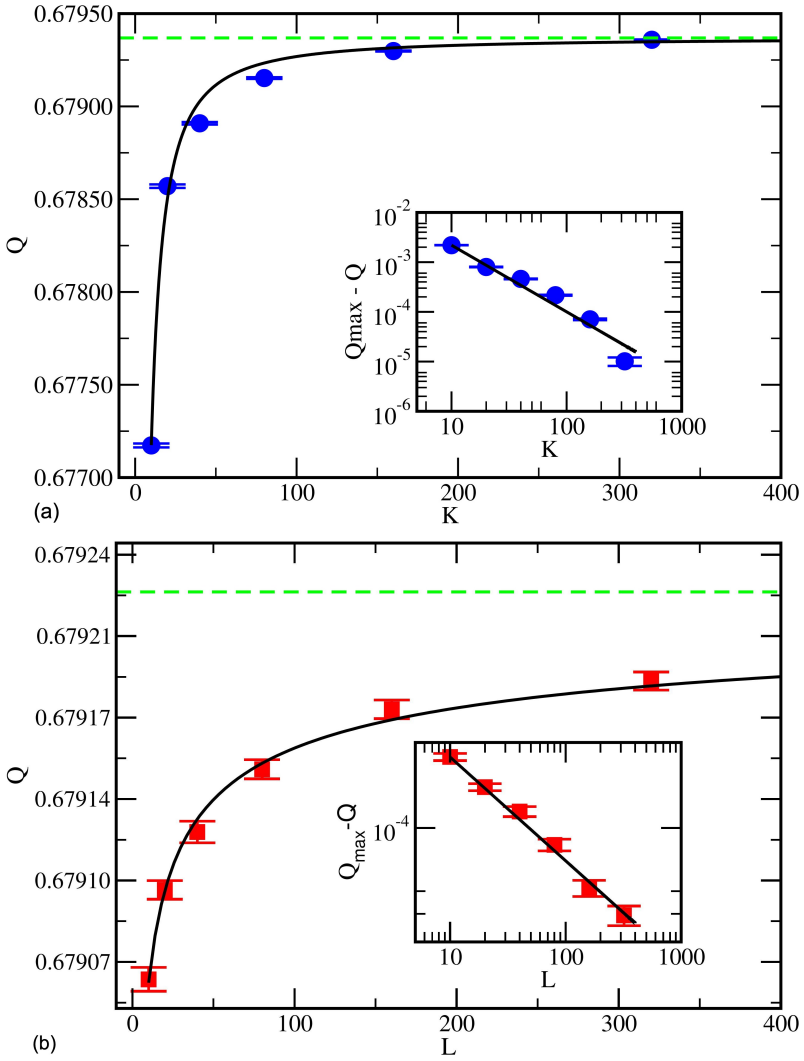

Figure 2. Modularity Q of RenEEL consensus partitions for Network A (**a**) as a function of K for fixed $L = 80$ (blue circles), and (**b**) as a function of L for fixed $K = 80$ (red squares). The black solid lines are fits to the data, representing a power-law approach to the estimated maximum value $Q_{\max}$ indicated by the green dashed lines. The value of $Q_{\max}$ is 0.679368(6) in (**a**) and 0.679229(40) in (**b**). The insets show the data on log-log axes. The slope of the fit to the blue circles in the inset of (**a**) is $\gamma_K = 1.34(2)$ and to the red squares in the inset of (**b**) is $\gamma_L = 0.36(18)$.

The fitted values of $Q_{\max}$ increase systematically with increasing L and K and converge to statistically equivalent values at the largest ensemble sizes studied (320), regardless of whether L or K is increased. However, the values of $Q_{\max}$ are generally larger when fixing L rather than K when comparing results from when they are fixed at the same size. This fact implies that the maximum value of Q is approached faster by fixing L and increasing K rather than the opposite. The rate of convergence to $Q_{\max}$ is quantified by the γ exponents. The values of the exponents $\alpha_K(L)$ and $\alpha_L(K)$ depend on the network but only weakly vary with the values of L and K, respectively. For each network, however, the value of α_K is significantly larger than that of α_L. Thus, the expected compute time increases faster with K than it does with L.

Table 3. (a) Values of asymptotic maximum Modularity Q_{max} found at fixed values of K. (b) Values of scaling exponent γ_L of the Modularity convergence to $Q_{\max}$ at fixed values of K.

(**a**)			
K	**Network A**	**Network B**	**Network C**
10	0.677480(34)	0.886521(12)	0.740259(170)
20	0.678725(24)	0.886763(10)	0.741572(174)
40	0.678963(36)	0.886830(12)	0.743070(172)
80	0.679229(40)	0.886832(10)	0.743072(170)
160	0.679246(26)	0.886832(6)	0.743077(172)
320	0.679370(20)	0.886833(8)	0.743088(168)
(**b**)			
K	**Network A**	**Network B**	**Network C**
10	0.37(6)	0.86(10)	0.29(26)
20	0.49(8)	0.88(10)	0.29(22)
40	0.50(12)	0.85(16)	0.22(24)
80	0.36(18)	0.85(12)	0.20(22)
160	0.47(20)	0.69(12)	0.25(20)
320	0.67(10)	0.86(16)	0.25(20)

To understand these results, recognize that finding the best partition is an extremal process that, when repeated, is akin to the process of record-breaking. Let us recall some of the theory of the extreme value statistics of record-breaking [22]. Consider a sequence of independent and identically distributed random numbers

$$\{x_1, x_2, x_3, \ldots, x_t, \ldots\} \tag{4}$$

chosen from a probability distribution of the form

$$p(x) = \mu B^{-\mu}(B - x)^{\mu - 1}, \qquad 0 \le x \le B, \tag{5}$$

where B is the maximum possible value of x, and define the *record* $R(t)$ as the maximum value of x in the first t numbers in the sequence:

$$R(t) \equiv \max\{x_1, x_2, x_3, \ldots, x_t\}. \tag{6}$$

Then, in the limit of large t, the mean record will approach B as

$$\langle R(t) \rangle = B\left(1 - \Gamma(1/\mu)\, t^{-1/\mu}\right), \tag{7}$$

i.e., as a power-law function with an exponent of $1/\mu$. From this, we see that $\mu = 1$ is a borderline case; from Equation (5), it is the case of a uniform distribution of x. If $\mu < 1$, then $p(x)$ is maximal at $x = B$, and if $\mu > 1$, then $p(x)$ vanishes as $x \to B$.

While the analogy with this simple, analytically tractable model of record-breaking is not perfect, Equation (3) can be compared with Equation (7) by identifying $Q_{\max}$ with B and γ with $1/\mu$. Then, the fact that empirically $\gamma_K > 1$ and $\gamma_L < 1$ suggests that the distributions of the Q of the consensus partitions found by increasing K and L correspond to different cases. Namely, as K is increased, the consensus partition Q is likely to be near $Q_{\max}$ as it is for $\mu < 1$, while as L is increased, it is more likely to have a smaller value as it is for $\mu > 1$. To confirm this, we made 800 runs of RenEEL analyzing Network A with $L = 80$ and $K = 320$ and with $L = 320$ and $K = 80$. Figure 3 shows the consensus values of Q found in those runs. As expected, the values found with $L = 80$ and $K = 320$ (red bars)

are much more likely to be near the maximum value than those found with $L = 320$ and $K = 80$ (blue bars).

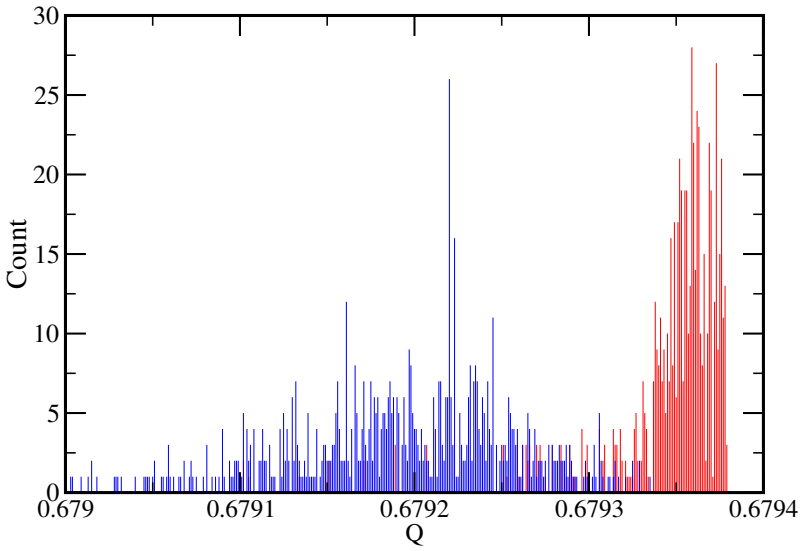

Figure 3. Distribution of Q obtained for Network A. Blue bars correspond to the data for $L = 320$ and $K = 80$, and red bars to $L = 80$ and $K = 320$.

We can now answer the central question of this paper: Given only limited computational resources, is it better to increase K or L? We have found that the average compute time grows faster with K than with L, but also that the consensus Q approaches $Q_{\max}$ faster with K than with L. Does the consensus Q approach $Q_{\max}$ as a function of compute time faster by increasing K or L? To answer this, we invert Equation (2) and combine it with Equation (3) to obtain

$$Q_{max} - Q \sim \langle t \rangle^{-\gamma/\alpha} \tag{8}$$

So, the larger γ/α is, the faster Q approaches Q_{max} as a function of average compute time. Table 4a shows the values of γ_K/α_K at different fixed L for the three networks. Similarly, Table 4b shows the values of γ_L/α_L at different fixed L for the three networks.

Table 4. (a) Values of scaling exponent $\frac{\gamma}{\alpha}$ at fixed values of L. (b) Values of scaling exponent $\frac{\gamma}{\alpha}$ at fixed values of K.

(**a**)			
L	**Network A**	**Network B**	**Network C**
20	0.93(1)	1.70(5)	0.98(15)
40	0.96(1)	1.65(6)	0.98(15)
80	0.90(1)	1.64(3)	1.31(12)
160	0.91(2)	1.96(12)	0.93(13)
320	0.89(1)	1.71(5)	0.98(12)
(**b**)			
K	**Network A**	**Network B**	**Network C**
20	0.66(11)	1.09(13)	0.35(26)
40	0.66(16)	1.03(20)	0.25(26)
80	0.44(22)	0.99(14)	0.22(24)
160	0.57(22)	0.88(15)	0.27(21)
320	0.68(12)	0.97(18)	0.28(22)

From these results, it can be clearly concluded that increasing K rather than L will cause Q to approach Q_{max} faster. With limited computational resources, it is therefore better to increase K rather than L. Although we have shown that this is true only for three example networks and only when maximizing Modularity, we speculate that these networks are not special and maximizing modularity, rather than a different objective function, is also not special. Therefore, the conclusion that it is more computationally efficient to increase K rather than L in RenEEL should generally be true. However, it would be interesting to explore this question when maximizing other objective functions with RenEEL.

Author Contributions: T.G., R.K.P.Z. and K.E.B. conceived of the presented idea. T.G. performed the numerical simulations and data analysis. K.E.B. verified the analytical methods and supervised the project. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The three networks used in this study are openly available at https://sites.cc.gatech.edu/dimacs10/archive/clustering.shtml (accessed on 19 April 2023). The raw data, which are used for analysis, are provided in Appendix A.

Acknowledgments: The authors would like to thank the Sabine cluster at the University of Houston, which was utilized for performing the simulations presented in this work.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

RenEEL	Reduced Network Extremal Ensemble Learning
EEL	Extremal Ensemble Learning

Appendix A

Table A1. Average computed time for Network A.

	K					
L	**10**	**20**	**40**	**80**	**160**	**320**
10	50.6(0.3)	123(1)	298(4)	755(5)	2065(57)	5811(104)
20	59.7(0.3)	161.2(0.9)	391(2)	1094(6)	2732(24)	7241(42)
40	72.0(0.3)	202(1)	564(3)	1617(9)	3972(21)	10,839(58)
80	93(1)	326(2)	960(5)	2427(13)	6857(41)	18,568(212)
160	145.1(0.9)	501(3)	1559(10)	4641(29)	12,123(109)	34,007(245)
320	215(1)	919(5)	2908(16)	8385(56)	22,964(220)	58,568(388)

Table A2. Modularity values for Network A.

	K		
L	**10**	**20**	**40**
10	0.67691112(300)	0.67829673(134)	0.67877639(779)
20	0.677076101(1973)	0.67844664(446)	0.67881534(751)
40	0.67715786(897)	0.67853550(1947)	0.67887617(823)
80	0.67717013(1549)	0.67851858(941)	0.67890690(564)
160	0.67730312(899)	0.67863411(271)	0.67890899(716)
320	0.67732834(2819)	0.67865861(980)	0.67893038(621)

Table A3. Modularity values for Network A.

	K		
L	**80**	**160**	**320**
10	0.67906248(512)	0.67921816(584)	0.67928542(149)
20	0.67910093(401)	0.67925767(629)	0.67932669(188)
40	0.67912582(465)	0.67928322(294)	0.67933913(177)
80	0.67915278(412)	0.67929836(261)	0.67935888(196)
160	0.67917852(401)	0.67931103(397)	0.67936292(202)
320	0.67919068(391)	0.67932548(421)	0.67936962(97)

Table A4. Average computed time for Network B.

	K					
L	**10**	**20**	**40**	**80**	**160**	**320**
10	13.84(0.07)	30.7(0.2)	63.1(0.1)	143.1(0.6)	293(1)	625(3)
20	16.37(0.09)	39.7(0.2)	88.5(0.3)	196(1)	426(2)	833(4)
40	20.2(0.1)	60.5(0.3)	130(1)	293(1)	649(3)	1340(7)
80	31.1(.2)	102(0.9)	256(2)	514(3)	1199(8)	2604(22)
160	46.8(0.3)	170(1)	431(3)	969(6)	2036(13)	4363(26)
320	73.5(0.6)	282(2)	877(6)	1955(14)	4016(29)	8214(51)

Table A5. Modularity values for Network B.

	K		
L	**10**	**20**	**40**
10	0.88642816(536)	0.88674884(440)	0.88671202(790)
20	0.88647127(1895)	0.88674307(594)	0.88675001(918)
40	0.88641007(295)	0.88677498(371)	0.8867800(1021)
80	0.88649938(1104)	0.88674605(430)	0.88678501(865)
160	0.88655162(1003)	0.88676799(246)	0.88680202(672)
320	0.88655147(675)	0.88676975(315)	0.88681905(495)

Table A6. Modularity values for Network B.

	K		
L	**80**	**160**	**320**
10	0.88673690(1014)	0.88674209(745)	0.88675501(832)
20	0.88677493(799)	0.88678016(634)	0.88678290(405)
40	0.88680491(805)	0.88680996(678)	0.886801492(418)
80	0.88680099(697)	0.88681788(395)	0.88681484(302)
160	0.88682706(707)	0.88683401(471)	0.88683493 (247)
320	0.88683201(381)	0.88683687(382)	0.88683980(201)

Table A7. Average computed time for Network C.

	K					
L	**10**	**20**	**40**	**80**	**160**	**320**
10	139.5(0.7)	295(2)	640(5)	1466(10)	3483(23)	8485(70))
20	181(1)	415(2)	1010(9)	2389(16)	5806(35)	14274(108)
40	295(2)	716(6)	1754(10)	4350(31)	10,846(82)	25,722(161)
80	484(4)	1251(13)	3203(30)	8033(68)	20,680(162)	45,012(402)
160	909(9)	2340(28)	6096(52)	15,600(123)	41,308(344)	87,912(801)
320	1790(19)	4600(53)	11,936(106)	29,605(249)	77,968(620)	170,610(1001)

Table A8. Modularity values for Network C.

	K		
L	**10**	**20**	**40**
10	0.73982575(12141)	0.74042051(10865)	0.74119763(9309)
20	0.73946114(16171)	0.74082625(12067)	0.74199647(6703)
40	0.73847614(7023)	0.74163987(6899)	0.74230803(7021)
80	0.73959559(15273)	0.74192572(9061)	0.74221369(6149)
160	0.73966281(10312)	0.7411455(13235)	0.7424959(6301)
320	0.74011840(13382)	0.74154016(12306)	0.74209565(8293)

Table A9. Modularity values for Network C.

	K		
L	**80**	**160**	**320**
10	0.74155587(8022)	0.74188600(8036)	0.74189623(8136)
20	0.74220959(7080)	0.74202369(8062)	0.74203165(8518)
40	0.74230761(6930)	0.74229803(8301)	0.74216325(8772)
80	0.74239582(7500)	0.74251785(6805)	0.74251525(6181)
160	0.7425311(7104)	0.7426539(7409)	0.7426924(6417)
320	0.74273495(6921)	0.74270134(4921)	0.74271412(2012)

Appendix B

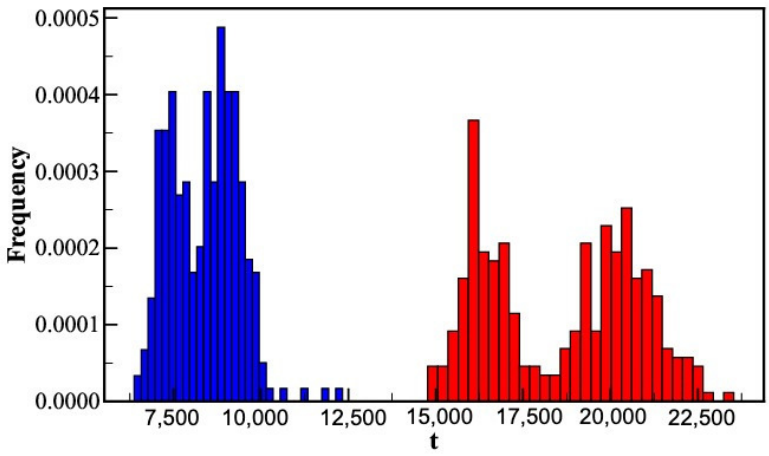

Figure A1. Distribution of required computed time to find the consensus partition for Network A. Blue bars correspond to the data for $L = 320$ and $K = 80$, and red bars to $L = 80$ and $K = 320$.

References

1. Fortunato, S. Community detection in graphs. *Phys. Rep.* **2010**, *486*, 75–174. [CrossRef]
2. Newman, M.E.J.; Girvan, M. Finding and evaluating community structure in networks. *Phys. Rev. E* **2004**, *69*, 026113. [CrossRef] [PubMed]
3. Newman, M.E.J. *Networks: An Introduction*; Oxford University Press: Oxford, UK, 2010.
4. Peixoto, T.P. *Descriptive vs. Inferential Community Detection in Networks: Pitfalls, Myths and Half-Truths*; Elements in the Structure and Dynamics of Complex Networks; Cambridge University Press: Cambridge, MA, USA, 2023.
5. Guimerà, R.; Sales-Pardo, M.; Amaral, L.A.N. Modularity from fluctuations in random graphs and complex networks. *Phys. Rev. E* **2004**, *70*, 025101. [CrossRef] [PubMed]
6. Schaub, M.T.; Delvenne, J.C.; Rosvall, M.; Lambiotte, R. The many facets of community detection in complex networks. *Appl. Netw. Sci.* **2017**, *2*, 4. [CrossRef] [PubMed]
7. Newman, M.E.J. The Structure and Function of Complex Networks. *SIAM Rev.* **2003**, *45*, 167–256. [CrossRef]

8. Ng, A.; Jordan, M.; Weiss, Y. On Spectral Clustering: Analysis and an algorithm. In Proceedings of the Advances in Neural Information Processing Systems, NIPS 2001, Vancouver, BC, Canada, 3–8 December 2001; Dietterich, T., Becker, S., Ghahramani, Z., Eds.; MIT Press: Cambridge, MA, USA, 2001; Volume 14.
9. Peel, L.; Larremore, D.B.; Clauset, A. The ground truth about metadata and community detection in networks. *Sci. Adv.* **2017**, *3*, e1602548. [CrossRef] [PubMed]
10. Wolpert, D.; Macready, W. Coevolutionary free lunches. *IEEE Trans. Evol. Comput.* **2005**, *9*, 721–735. [CrossRef]
11. Brandes, U.; Delling, D.; Gaertler, M.; Gorke, R.; Hoefer, M.; Nikoloski, Z.; Wagner, D. On Modularity Clustering. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 172–188. [CrossRef]
12. Brandes, U.; Delling, D.; Gaertler, M.; Görke, R.; Hoefer, M.; Nikolski, Z.; Wagner, D. *On Modularity—NP-Completeness and Beyond*; Technical Report 19; Universität Karlsruhe (TH): Karlsruhe, Germany, 2006. [CrossRef]
13. Guo, J.; Singh, P.; Bassler, K.E. Reduced network extremal ensemble learning scheme for community detection in complex networks. *Sci. Rep.* **2019**, *9*, 14234. [CrossRef] [PubMed]
14. Newman, M.E.J. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* **2006**, *103*, 8577–8582. [CrossRef] [PubMed]
15. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008. [CrossRef]
16. Ovelgönne, M.; Geyer-Schulz, A. Cluster Cores and Modularity Maximization. In Proceedings of the 2010 IEEE International Conference on Data Mining Workshops, Sydney, Australia, 13 December 2010; pp. 1204–1213.
17. Sun, Y.; Danila, B.; Josić, K.; Bassler, K.E. Improved community structure detection using a modified fine-tuning strategy. *Europhys. Lett.* **2009**, *86*, 28004. [CrossRef]
18. Treviño, S.; Nyberg, A.; Genio, C.I.D.; Bassler, K.E. Fast and accurate determination of modularity and its effect size. *J. Stat. Mech. Theory Exp.* **2015**, *2015*, P02003. [CrossRef]
19. Liu, Z.; Ma, Y. A divide and agglomerate algorithm for community detection in social networks. *Inf. Sci.* **2019**, *482*, 321–333. [CrossRef]
20. Newman, M. Network Data. Available online: http://www-personal.umich.edu/~mejn/netdata/ (accessed on 19 April 2025).
21. Newman, M.E.J. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. USA* **2001**, *98*, 404–409. [CrossRef] [PubMed]
22. Schmittmann, B.; Zia, R.K.P. "Weather" records: Musings on cold days after a long hot Indian summer. *Am. J. Phys.* **1999**, *67*, 1269–1276. [CrossRef]