Symbolic Regression on Network Properties

Marcus Märtens, Fernando Kuipers, and Piet Van Mieghem

Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology,
PO Box 5031, 2600 GA Delft, The Netherlands

Abstract. Networks are continuously growing in complexity, which creates challenges for determining their most important characteristics. While analytical bounds are often too conservative, the computational effort of algorithmic approaches does not scale well with network size. This work uses Cartesian Genetic Programming for symbolic regression to evolve mathematical equations that relate network properties directly to the eigenvalues of network adjacency and Laplacian matrices. In particular, we show that these eigenvalues are powerful features to evolve approximate equations for the network diameter and the isoperimetric number, which are hard to compute algorithmically. Our experiments indicate a good performance of the evolved equations for several real-world networks and we demonstrate how the generalization power can be influenced by the selection of training networks and feature sets.

Keywords: symbolic regression, complex networks, Cartesian Genetic Programming

1 Introduction

One of the first and most important steps for modelling and analyzing complex real-world relationships is to understand their structure. Networks are an effective way to organize our data so that nodes describe certain actors or entities, while relations are expressed as links connecting two nodes with each other. The resulting topological representation (adjacency matrix) provides an abstract model that is amenable for further analysis. For example, algorithms for finding shortest paths, spanning trees or similar structures usually take the topological representation of the network as input. Community detection algorithms can cluster groups of nodes that are more connected within their group than outside. Computing node centrality metrics allows for the identification of important nodes or critical connections. A well-known example is Google's Pagerank algorithm [1], which uses the eigenvector centrality of a node in order to assess the rank of a webpage with respect to Google's search queries.

Eigenvector centrality [2] is interesting from a different perspective as well. It shows that spectral network properties can improve our understanding of such vast aggregations of data like in the world-wide web. Spectral graph theory explicitly seeks to understand the relations between eigenvalues, eigenvectors and

characteristic polynomials of various network matrices. Many links to fundamental questions of mathematics and complexity theory arise from spectral graph theory, making this area of research both valuable and intricate. It is possible that many topological network properties are reflected in the spectrum, only waiting to be discovered.

This work proposes symbolic regression as a method to automatically derive insights in the spectral domain and their corresponding topological reflections in the network. Only a minimal number of assumptions are needed, in particular in comparison to the frequently used procedure of curve fitting, which assumes already a pre-knowledge of a certain function like a polynomial, exponential, etc. In contrast, symbolic regression is guided by supervised learning for a regression task that explicitly constructs free-form equations out of numeric features and elementary arithmetic operations.

The topological representation may be a cumbersome feature space for machine learning techniques, if only the binary features of the adjacency matrix are considered. Therefore, we examine the usage of features from the spectral domain of the network. By training the symbolic regression system on a set of carefully constructed networks, we are able to estimate target features. Consequently, symbolic regression may assist researchers to unravel the hidden structures in the spectral domain and to propose first-order approximations for difficult-to-compute properties.

Our work is structured as follows: Section 2 introduces the concept of symbolic regression by giving references to previous work where this technique proved useful. Section 3 provides the necessary background in network science by introducing network properties that will be used as features and targets for our experiments. The setup of our experiments is outlined in Section 4 and their results are discussed in Section 5. We conclude with directions for future research in Section 6.

2 Related Work

2.1 Symbolic Regression

One of the most influential works on symbolic regression is due to Michael Schmidt and Hod Lipson [3], who demonstrated that physical laws can be derived from experimental data (observations of a physical system) by algorithms, rather than physicists. The algorithm is guided by evolutionary principles: a set of (initially random) parameters and constants are used as inputs, which are subsequently combined with arithmetic operators like $\{+, -, \times, \div\}$ to construct building blocks of formulas. Genetic operations like crossover and mutation recombine the building blocks to minimize various error metrics. The algorithm terminates after a certain level of accuracy is reached; the formulas that describe the observed phenomenon best are delivered as output for further analysis.

In the work of Schmidt and Lipson [3], symbolic regression was able to find hidden physical conservation laws, which describe invariants over the observed

time of physical systems in motion, like oscillators and pendulums. It is remarkable that symbolic regression was able to evolve the Hamiltonian of the double pendulum, a highly non-linear dynamic system [4] that undergoes complex and chaotic motions. Also, accurate equations of motions were automatically derived for systems of coupled oscillators.

While symbolic regression rarely deduces error-free formulas, the output may deepen our insight in the problem and may help to eventually find exact solutions. One example is the case of solving *iterated functions*, which asks for a function $f(x)$ that fulfills $f(f(x)) = g(x)$ for some given function $g(x)$. Despite the simple description of the problem, there exist difficult cases for which highly non-trivial algebraic techniques seem to be needed to find solutions.

One example is the iterated function $f(f(x)) = x^2 - 2$, for which the best known analytic approach to find $f(x)$ requires the substitution of special function forms and recognizing relations between certain Chebyshev polynomials. Again, Schmidt and Lipson [5] were able to evolve a couple of symbolic expressions that were so close at describing a solution, that a simple proof by basic calculus could be inferred.

Most recently, symbolic regression has been explored in the context of generative network models by Menezes and Roth [6]. They present a stochastic model in which each possible link has a weight computed by an evolved symbolic expression. The weight-computation-rules are executed and the resulting networks are compared by a similarity-metric with some target networks (corresponding to the observations of a physical system), which guides evolution to incrementally improve the underlying generative model.

One particular benefit of symbolic regression and automatic generation of equations is reduction of the bias introduced sometimes unknowingly by human preferences and assumptions. Thus, it is possible for symbolic regression to discover relations that would be deemed counter-intuitive by humans. This makes symbolic regression especially attractive for finding non-linear relationships, for which the human mind often lacks insight and intuition.

With the exception of the deterministic FFX-algorithm by McConaghy [7], most symbolic regression algorithms are based on Genetic Programming [8], where an evolutionary process typically uses grammars [9, 10] to evolve expression trees. Our work can be potentially implemented by many of these Genetic Programming variants, but we selected Cartesian Genetic Programming[1] (CGP) for reasons outlined in the following subsection.

2.2 Cartesian Genetic Programming (CGP)

CGP was originally developed by Julian Miller [11] to represent electronic circuits on 2d-grids (hence the name *Cartesian*), but it soon became a general purpose tool for genetic programming. It has been used in numerous applications, e.g. to develop Robot Controllers [12], Neural Networks [13], Image Classifiers [14] and Digital Filters [15]. A recent result by Vasicek and Sekanina shows how

[1] `http://www.cartesiangp.co.uk/resources.html`

approximate digital circuits can be efficiently evolved by CGP, giving human-competitive results in the field of approximate computing [16]. Vasicek also shows how CGP can be scaled to deal with a large number of parameters in order to optimize combinatorial circuits [17].

The reason why CGP is so popular (especially for circuit design) is due to its internal representation of the Genetic program. CGP uses a flexible encoding that represents the wiring of a computational network. Each node in this network is an arithmetic operation that needs a certain amount of inputs to produce an output. A simple $1 + 4$ evolutionary strategy changes the interconnections between those nodes in order to improve a fitness function (minimizing errors). Input parameters and constants are forward-propagated by applying the computational nodes until output nodes are reached. At these output nodes, the chain of mathematical operations on the inputs can be reconstructed as an equation.

A surprising property of CGP is that only a minor fraction of nodes actually contribute to the final computation. Similar to a human genome, only part of it is actively used, while inactive parts are dormant, but subject to genetic drift. This redundancy is often argued to be beneficial for the evolutionary process in CGP (see Miller and Smith [18]). There is also evidence that CGP does not suffer much from bloat [19], a major issue in other genetic programming techniques that tend to produce very large program sizes even for simple tasks.

3 Networks

In this section, we formally define some network properties and notation that will be used throughout our experiments.

3.1 Network Representations

A network is represented as a graph $G = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of links. The number of nodes is denoted by $N =\mid \mathcal{N} \mid$ and the number of links by $L =\mid \mathcal{L} \mid$. The set $\mathcal{L}$ is typically represented by an $N \times N$ adjacency matrix A with elements $a_{ij} = 1$ if node i and j are connected by a link and $a_{ij} = 0$ otherwise. As we restrict ourselves to simple, undirected networks without self-loops in this work, A is always symmetric. We call A the *topological representation* of G as each element of A directly refers to a structural element (a link) of the network. The number of all neighbors of a node i is called its degree $d_i = \sum_{j=1}^{N} a_{ij}$.

The adjacency matrix A is not the only possible representation of a network. Of equal importance is the Laplacian matrix $Q = \Delta - A$, where Δ is a diagonal matrix consisting of the degrees d_i for each node $i \in \mathcal{N}$.

A different view on the network can be derived by its eigenstructure. Given the adjacency matrix A, there exists an eigenvalue decomposition [20]

$$A = X \Lambda X^T \tag{1}$$

such that the columns of X contain the eigenvectors $x_1, x_2, \ldots, x_N$ belonging to the real eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N$, respectively, contained in the diagonal matrix Λ.

While obtaining the spectral representation of the network requires computational effort by itself (usually, the network is given in its topological representation for which the eigenvalues still need to be determined), it provides a different perspective on the network's properties. For example, the largest eigenvalue λ_1 is linked with the vulnerability of a network to epidemic spreading processes [21].

A similar decomposition is possible for the Laplacian matrix, whose eigenvalues are denoted by $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_N$ and whose real eigenvectors are $y_1, y_2, \ldots, y_N$. The second smallest eigenvalue μ_{N-1} is known as the *algebraic connectivity* [22] and its corresponding eigenvector is known as *Fiedler's vector*. Spectral clustering [20] is a possible application of Fiedler's vector.

Both eigensystems constitute the *spectral representation* of G. Our goal is to describe network properties typically computed by algorithms on the *topological representation* of G by simple functions consisting of elements from the *spectral representation* of G. An example is the number of triangles $\blacktriangle_G$ in a network. A way of computing $\blacktriangle_G$ is to enumerate all possible triples of nodes in a graph and checking whether they are connected in A. However, the number of triangles can also be expressed as

$$\blacktriangle_G = \frac{1}{6} \cdot \sum_{k=1}^{N} \lambda_k^3 \tag{2}$$

and is thus directly computable from the spectral representation without the need of exhaustive enumeration (see Van Mieghem [20], art. 28 for a proof of Equation (2)).

3.2 Network Properties

Network Diameter Many applications of networks are concerned with finding and using shortest-path structures in networks. A *path* between two distinct nodes i and j is a collection of links that can be traversed to reach i from j and vice versa. A *shortest path* is a path with minimal number of links. The *diameter* ρ of a network is defined as the length of the *longest* shortest path in the network, i.e. the maximum over all shortest-path lengths between all node-pairs. Algorithms that solve the all-pairs shortest path problem (like the Floyd-Warshall algorithm) are able to compute the diameter in $\mathcal{O}(N^3)$. While more efficient algorithms exist for sparse networks, an exact computation of the diameter is usually too expensive for very large networks.

There exist multiple upper bounds for the diameter [23, 24], but we find the bound of Chung et al. [25] most tight in almost all cases:

$$\rho \leq \left\lfloor \frac{\cosh^{-1}(N-1)}{\cosh^{-1}\left(\frac{\mu_1+\mu_{N-1}}{\mu_1-\mu_{N-1}}\right)} \right\rfloor + 1. \tag{3}$$

This bound was independently derived by Van Dam and Haemers [26].

Isoperimetric Number For each subset of nodes $X \subset \mathcal{N}$ we can define the set ∂X as the set of links that have exactly one endpoint in X and the other endpoint in $\mathcal{N} \backslash X$. The *isoperimetric number* η of a network is defined as

$$\eta = \min_{\substack{X \subseteq \mathcal{N} \\ |X| \leq \frac{1}{2}N}} \frac{|\partial X|}{|X|}. \tag{4}$$

Essentially, the isoperimetric number is a measure related to *bottlenecks* in networks. Intuitively, a low isoperimetric number indicates that the network can be separated in two reasonably big parts by only cutting a minimum amount of links. While the isoperimetric number is a good descriptor of network robustness, its computation for general networks is intractable, as the computational effort scales with the amount of possible cuts of the network. More information on the isoperimetric number can be found in [20, 27].

4 Experiments

This section describes technical details of the symbolic regression process we deployed to infer equations for the network diameter and the isoperimetric number. As symbolic regression is a supervised learning technique, we describe the sets of networks that were used for training and testing, together with the features we extracted for each case.

4.1 Network Diameter

In order to find a suitable formula for the network diameter, we trained CGP on 3 different sets of networks:

- augmented path graphs,
- barbell graphs and
- the union of both.

The augmented path graphs were generated by iteratively adding random links to a simple path graph of N nodes. With each additional link, there is a chance to lower the diameter of the network. Following this procedure, it is possible to generate a set of relatively sparse graphs of constant node-size with uniformly distributed diameters.

A barbell graph $B(a, b)$ is generated by taking two cliques of size a and connecting them with a path graph of size b. The total number of nodes is $N = 2a + b$. The diameter $\rho(B(a, b))$ is always $b + 3$. Adjusting the length of the path graph allows for generating graphs of different diameters. Changing the size of the cliques allows for creating graphs with the same diameter, but different number of nodes. We sample again such that the network diameter is uniformly distributed within the set of all barbell graphs. Compared with augmented path graphs, barbell graphs are (in general) denser networks.

The set of mixed graphs is the union of the set of augmented path graphs and barbell graphs. See Figure 1 for examples of these networks and Table 1 for a summary of all sets of networks for the experiments.

One reason why we have chosen these sets of networks instead of, for example, Erdős-Rényi (ER) random graphs [28], is to control the distribution of our target feature, the network diameter. Preliminary experiments have shown that too little variance in our target feature will push CGP to converge to a constant function, which does not include any information about the relation between spectral features and the target that we want to extract. For an ER graph of N nodes and with link probability p, Bollobás [28] showed that, for fixed p and N large, ρ can only have one of two possible neighboring values with high probability. Thus, sampling uniform diameters for the random graph model requires careful adjustment of N and p, where we found the usage of augmented paths and barbell graphs more convenient.

For the supervised learning of CGP, each set of networks was separated in a 60% training and a 40% test set. Table 2 gives an overview of the various parameters we set for CGP. In preliminary experiments, we changed each of these parameters independently from another and found the settings of Table 2 to provide the most useful results in terms of fitness and formula complexity. A more thorough parameter tuning approach is needed for maximum performance, but is outside the scope of this work. For the meaning of these parameters, see Miller [11]. Effective tuning of CGP was researched by Goldman and Punch [29].

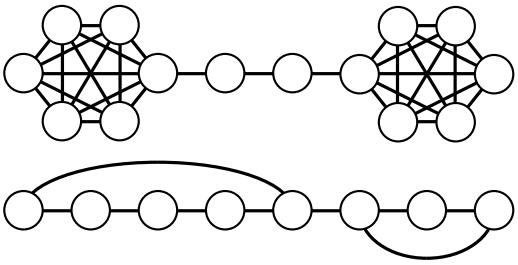

Fig. 1. Example of the barbell graph $B(6, 2)$ with $\rho = 5$ on top and an augmented path graph with $\rho = 4$ at the bottom.

In our experiments, we tried a vast selection of different features to evolve formulas. To keep this section organized, we report only results derived from two of the most generic, but useful, sets of features:

A) $N, L, \lambda_1, \lambda_2, \lambda_3, \lambda_N$
B) $N, L, \mu_1, \mu_{N-1}, \mu_{N-2}, \mu_{N-3}$.

Additionally, the natural numbers $1, \ldots, 9$ were provided as network independent constants for CGP to adjust evolved terms appropriately.

Table 1. Properties of network sets.

	aug. path	**barbell**	**mixed**
networks	1672	1675	3347
nodes	$N = 70$	$7 \leq N \leq 667$	$7 \leq N \leq 667$
diameter	$2 \leq \rho \leq 69$	$4 \leq \rho \leq 70$	$2 \leq \rho \leq 70$
avg. link density	0.04845	0.36985	0.20910

Table 2. Parameterisation of CGP.

parameter	**value**
fitness function	sum of absolute errors
evolutionary strategy	1+4
mutation type and rate	probabilistic (0.1)
node layout	1 row with 200 columns
levels-back	unrestricted
operators	$+, -, \times, \div, \cdot^2, \cdot^3, \sqrt{\cdot}, \log$
number of generations	$2 \cdot 10^5$

The choice of feature sets A and B provides a reasonable trade-off between formula complexity and fitness. While selecting the complete spectrum of eigenvalues as features is possible, we observed that it leads to a high formula complexity without providing considerable improvements in fitness. Additionally, the largest adjacency (smallest Laplacian) eigenvalues are the ones that are suggested to have the strongest influence on network properties [20]. Lastly, since the number of nodes in our network instances is not (in every case) constant, giving the complete spectrum would mean that several features would be missing in networks with low number of nodes. It is unclear, how an appropriate substitution of the missing features should be realized. Thus, some of the discovered formulas could be inapplicable for some networks.

Since the evolutionary procedures of CGP to optimize the fitness of the evolved expressions are stochastic, we deployed multiple runs for each combination of feature and network set. We aggregated those multiple runs into batches, as the test-environment was implemented to run on a computational cluster. Each batch consisted of 20 runs of CGP for a specific set of features. Out of those 20 runs, only the one with the best (lowest) fitness is reported. The fitness is the sum of absolute errors on the test instances of the corresponding set of networks. More formally, if $\hat{\rho_G}$ is the estimate on the diameter ρ_G of network G given by the evolved formula $\hat{\rho}$ and $\mathcal{G}_{test}$ is the set of all networks for testing, the fitness $f(\hat{\rho})$ is defined as:

$$f(\hat{\rho}) = \sum_{G \in \mathcal{G}_{test}} |\rho_G - \hat{\rho_G}|. \tag{5}$$

Furthermore, we define the approximation error $e(\hat{\rho})$ as the average deviation from the diameter over the complete test set:

$$e(\hat{\rho}) = \sum_{G \in \mathcal{G}_{test}} \frac{|\rho_G - \hat{\rho_G}|}{|\mathcal{G}_{test}|}. \tag{6}$$

We present the results over 100 batches for each combination of feature and network test set in Table 3.

Table 3. Experimental results for the network diameter.

networks	feature set	avg. fitness	min. fitness	min. approx. error
aug. path	A	3694.98750	3404.53700	5.08899
	B	842.89691	778.98900	1.16441
barbell	A	1.66654	0.00900	0.00001
	B	50.53473	$< 10^{-5}$	$< 10^{-5}$
mixed	A	5313.91179	4500.68900	3.36123
	B	1462.61943	1134.34100	0.84716

4.2 Isoperimetric Number

The training set of networks for the isoperimetric number η had to be limited to relatively small networks, since the computation of η becomes intractable even for general medium-sized networks. Thus, we decided to exhaustively enumerate all networks of size $N = 7$, for which the computation was still practical. This set consists of 1046 non-isomorphic networks, which we randomly split into a training set of 627 and a test set of 419 networks. We applied the same parameters to CGP as shown in Table 2, with one exception: we created 100 batches for each of the following sets of operators:

1. $+, -, \times, \div, \sqrt{\cdot}, \log$
2. $+, -, \times, \div, \cdot^2, \log$
3. $+, -, \times, \div, \sqrt{\cdot}$
4. $+, -, \times, \div, \cdot^2$.

Since we have only networks of size $N = 7$, we can select the full spectrum as our features, resulting in the following feature sets:

A') $N, L, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6, \lambda_7$
B') $N, L, \mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6$.

Since the smallest Laplacian eigenvalue μ_N always equals 0, $\mu_7 = 0$ and is thus not included as a feature. Additionally, we provided the natural numbers $1, 2$ and 3 as constants. Each batch consisted of 5 independent runs from which only the best one is reported.

Feature set A' delivered on average much better results than feature set B'. The best result was found with feature set A' and operator-set $+, -, \times, \div, \cdot^2, \log$, although not all of these operators appear:

$$\hat{\eta_1} = \frac{L - \lambda_2^2 - 2}{\frac{\lambda_2}{2} + 5} \tag{7}$$

Although Equation (7) is short (low complexity), it had still the best fitness (53.215) of all evolved formulas. The approximation of η on the test set is shown in Figure 2.

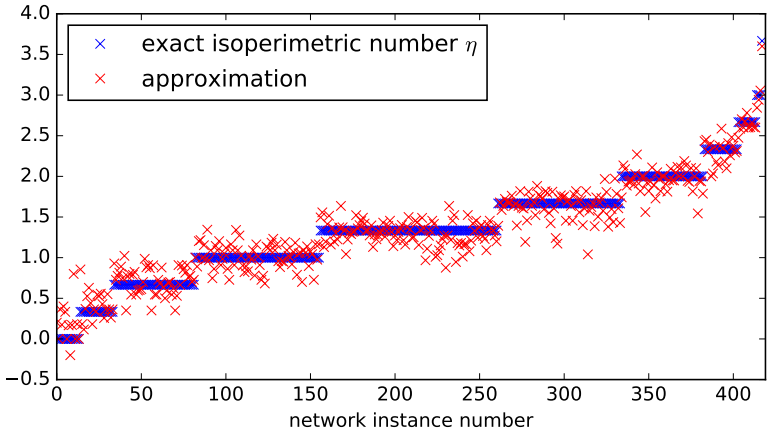

Fig. 2. All 419 networks from the test set ordered by their actual isoperimetric number ascending from left to right. The red crosses show the approximation given by Equation (7).

5 Discussion

In the previous section, we evolved approximate equations for hard to compute network properties. The approximation errors were largely dependent on the used networks for training and testing. For example, the best equations for the diameter found for barbell networks have almost no error, while noticeable errors exist for augmented path and mixed network. This raises two questions:

- how is the quality of the approximate equations influenced by the pre-selected networks and
- how do the approximate equations generalize to other networks?

To give answers, we compute already established analytical equations from the literature as reference points for quality and provide appropriate selections of

networks which were not involved in the generation process of the evolved equations.

5.1 Network Diameter

As a measure for the quality of our evolved equations, we compare their estimates of diameter ρ to the upper bound given by Equation (3). An upper bound and an approximation are different: while the bound is always above the real diameter, approximations may be above or below without any guarantees. Yet, we believe the bound can mark a reference point for a qualitative comparison in addition to the exact diameter itself.

As additional networks, we selected 12 real-world data sets available at networkrepository.com [30], where more information, interactive visualization and analytics can be found. While these networks should only be viewed as examples, they might give an idea about the applicability of the presented technique. To eliminate the selection-bias and gather significant results, one would need to sample the network space in a representative and meaningful way, which is notoriously difficult. For example, simple network generators like Erdős-Rényi random graphs or the Barabási-Albert model are not sufficient, as they will only allow to sample certain degree distributions. Consequently, we restrict ourselves to examples.

We expect that the equations with the lowest fitness give the best results. Because feature set B consistently outperformed feature set A in terms of fitness, we analyzed the approximations given by the best equations of feature set B. The explicit equations are:

$$N - \frac{1 - \frac{1}{(L-N)^{\frac{3}{2}}}}{6 - \frac{6}{\frac{(L-N)^{\frac{3}{2}}}{\sqrt{L-N}} + 4\sqrt{L-N}}} - 2\sqrt{L-N} - \frac{1}{\sqrt{L-N}} \tag{8}$$

$$\frac{\log\left(2L\mu_{N-3} + 6\right) + 6}{\log\left(L\mu_{N-3} + \sqrt{5}\sqrt{\frac{1}{\mu_{N-1}}}\right)} + \sqrt{5}\sqrt{\frac{1}{\mu_{N-1}}} + 3\sqrt{82}\sqrt{\frac{1}{729L\mu_{N-2}\mu_{N-3} - 5}} \tag{9}$$

$$\sqrt{\sqrt{N} + \frac{45\mu_{N-3}}{\left(\mu_{N-1} + \mu_{N-3}\right)^2} + \log\left(\frac{216}{\left(\mu_{N-1} + \mu_{N-3}\right)^2}\right) - \frac{16}{9\mu_{N-3}} + \frac{8\sqrt[4]{\mu_{N-3}}}{L\mu_{N-1}\mu_{N-2}}} \tag{10}$$

The numerical values are all listed together with some basic properties of our validation networks in Table 4. First, we observe that Equation (8) performs extremely poorly by giving huge overestimations of ρ, despite its fitness of almost 0 for the networks of the original test set. The reason is that Equation (8) was evolved on barbell graphs only, which have a fixed and symmetric structure. In particular, the difference $N - L$, which is a frequent subterm of the formula, is

higher in the dense barbell graphs compared to the rather sparse networks of our validation set. Thus, Equation (8) seems to be *overfitted* to the class of barbell graphs.

Equation (9) was evolved only on augmented path graphs and provides a much better approximation of ρ for our validation networks. This might be the case since the validation networks are more similar to the sparse augmented path graphs than to the dense barbell graphs. A visual comparison of the approximation of Equation (9) is given by Figure 3.

Adding the barbell graphs to the training set, like in the evaluation of Equation (10), shows that the accuracy of the approximation of ρ increases by roughly 10%, which must be the effect of the barbell graphs adding a selective force towards accuracy on more denser networks. Moreover, it seems that CGP focused on finding a good approximation for the augmented path graphs in the mix rather than considering to find an equation for both classes of networks. In the majority of the cases, ρ is still overestimated a little, but by far not as much as by the upper bound in Equation (3).

Table 4. Diameter on validation networks.

name	N	L	ρ	Eq. (9)	Eq. (8)	Eq. (10)	Eq. (3)
ca-netscience	379	914	17	21	333	24	160
bio-celegans	453	2025	7	7	374	8	104
rt-twitter-copen	761	1029	14	16	728	18	126
soc-wiki-Vote	889	2914	13	10	799	12	133
ia-email-univ	1133	5451	8	6	1002	8	58
ia-fb-messages	1266	6451	9	7	1122	10	96
web-google	1299	2773	14	29	1222	35	336
bio-yeast	1458	1948	19	19	1414	22	208
tech-routers-rf	2113	6632	12	14	1979	17	237
socfb-nips-ego	2888	2981	9	52	2869	61	2466
web-edu	3031	6474	11	36	2914	40	663
inf-power	4941	6594	46	98	4860	110	749

5.2 Isoperimetric Number

The quality of the equations approximating the isoperimetric number will be related to the Cheeger inequality (see Mohar [27]) that gives us bounds in relation to the algebraic connectivity μ_{N-1} and the maximum degree d_{max} of the network G:

$$\frac{\mu_{N-1}}{2} \leq \eta \leq \sqrt{\mu_{N-1}(2d_{max} - \mu_{N-1})} \tag{11}$$

Since our equations were evolved by an exhaustive enumeration of all non-isomorphic networks of $N = 7$, we are interested how their quality of fit will

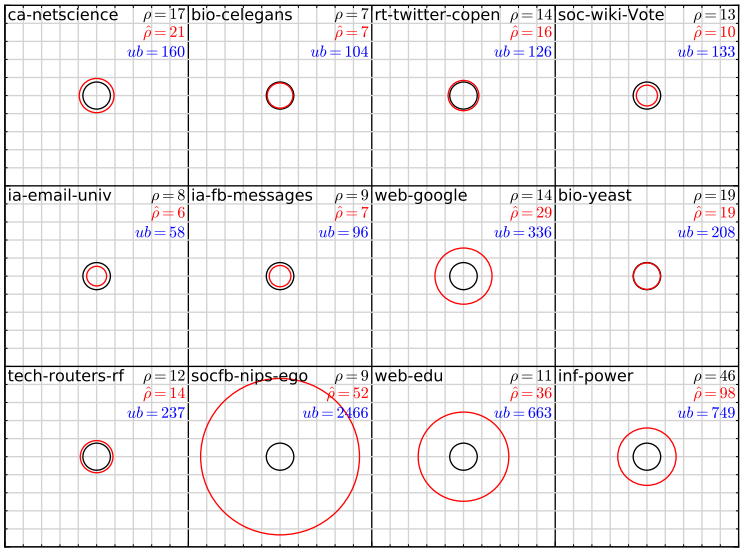

Fig. 3. Red circles: approximate diameter $\hat{\rho}$ by Equation (9) relative to the network diameter ρ as a black circle. All network diameters are scaled in each network to have unit-length in the figure. All values are rounded to the next integer. The upper bound Equation (3) values are given as ub in blue (too large to plot).

differ with N. However, as pointed out before, the computation of the exact value for η is in general only feasible for very small networks. Consequently, we cannot use any of the validation-networks from Table 4. Instead we decided to sample random networks of $N = 20$ nodes and links from $22 \leq L \leq 190$. In total, we generated 4984 non-isomorphic connected networks with roughly uniformly distributed link densities by a variant of the ER random graph model. In these networks, the isoperimetric number η ranges from 0.2 to 10.0, while in our training set η was between 0 and 4.

Surprisingly, Equation (7) deduced from all networks with $N = 7$ nodes is performing poorly on the set of random networks, as shown by the green dots in Figure 4. The estimates are most of the time not even below the bound of the Cheeger inequality, shown in grey. By analyzing the sum of absolute errors on this new set of networks, we found that from all evolved formulas, the following equation for the isoperimetric number gives the best performance:

$$\hat{\eta}_2 = \frac{1}{N^2}\left(L\left(\frac{\mu_1}{\mu_2} + \mu_2\right) - 1\right). \tag{12}$$

We observe that for over 98% of the random networks, the estimate of Equation (12) was within $(1 \pm 0.2) \cdot \eta$. Since this equation incorporates not only spectral features, but also N and L, we believe it generalizes better to networks of different size other than those used in the training set. Additionally, our ex-

periments show that a low fitness value does not necessarily correspond to good generalization. Out of the 800 batches used to find a formula for η, only 259 returned expressions that did not create artifacts (like square roots of negative numbers or divisions by zero, which CGP evaluates to 0 by definition). While Equation (7) ranked first with a fitness of 53.215, Equation (12) was one of the unranked expressions, since on some of the unconnected networks of the training set, μ_2 was 0, while $\mu_2 = 0$ did never appear for the connected random networks.

It is also noteworthy that Equation (12) seems to slightly overestimate η as soon as networks with $\eta > 4$ are encountered. This does not seem to be a coincidence, as 4 was the maximum value for η in the training set.

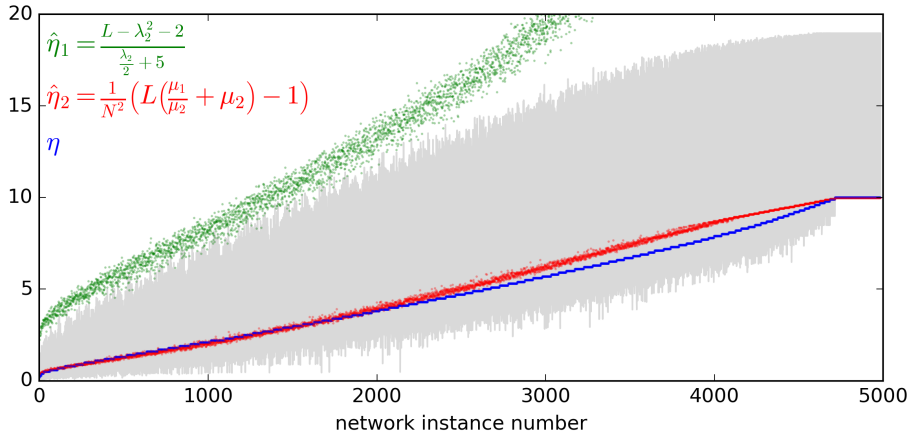

Fig. 4. A set of 4984 random networks with $N = 20$ ordered by their isoperimetric number (blue). The grey area corresponds to the Cheeger inequalities given by Equation (11). The lower bound coincides with η and the red approximation formula for the last 100 networks, as they are all fully connected.

6 Conclusion

Our experiments provide a first demonstration that symbolic regression can be applied to analyze networks. For the first time, to the best of our knowledge, an automated system has inferred approximate equations for network properties, which otherwise would have required a high algorithmic effort to be determined. Although these equations are not rigorously proven and might be cumbersome for humans to comprehend, they are able to exploit the hidden relationships between the topological and the spectral representation of networks, which has been elusive to analytical treatment so far.

We do not expect that symbolic regression at the current level will substitute researchers deriving meaningful equations, but we do believe that symbolic

regression can be a meaningful tool for these researchers. As the proposed techniques are not biased by human preconceptions, unexpected results might provide inspiration and stimulating starting points for the development of formal proofs or more accurate formulas. While this lack of bias can be an advantage, the proposed system nevertheless allows for the incorporation of *a priori* expert knowledge. If certain features and operators are suspected to be correlated to an unknown target quantity, their usage can be enforced easily.

Understanding which conditions give rise to equations with a high generalization power for networks will be the main challenge for the future. While a good fitness of an equation does not necessarily imply a high generalization power, our experiments indicate that symbolic regression is clearly able to produce equations that are reasonably accurate for unknown networks. In order to prevent overfitting to the training set and to increase this generalization power, selecting a good set of networks for training seems to be the key. This makes symbolic regression especially appealing when dealing with networks that can be characterized by their structural and degree-related properties, like scale-free or small-world networks. As networks with such properties are ubiquitous, discovering explicit relations between their features will pave the way for a deeper insight into our increasingly connected environments.

References

1. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web. Technical report, Stanford University, Stanford, CA (1998)
2. Van Mieghem, P.: Graph eigenvectors, fundamental weights and centrality metrics for nodes in networks. arXiv preprint arXiv:1401.4580 (2014)
3. Schmidt, M., Lipson, H.: Distilling free-form natural laws from experimental data. Science **324**(5923) (2009) 81–85
4. Strogatz, S.H.: Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering. Studies in Nonlinearity. Westview Press (2014)
5. Schmidt, M., Lipson, H.: Solving iterated functions using genetic programming. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, ACM (2009) 2149–2154
6. Menezes, T., Roth, C.: Symbolic regression of generative network models. Scientific reports **4** (2014)
7. McConaghy, T.: FFX: Fast, scalable, deterministic symbolic regression technology. In: Genetic Programming Theory and Practice IX. Springer (2011) 235–260
8. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection. Volume 1. MIT press (1992)
9. McConaghy, T., Eeckelaert, T., Gielen, G.: CAFFEINE: Template-free symbolic model generation of analog circuits via canonical form functions and genetic programming. In: Proceedings of the conference on Design, Automation and Test in Europe. Volume 2., IEEE Computer Society (2005) 1082–1087
10. Augusto, D.A., Barbosa, H.J.: Symbolic regression via genetic programming. In: 6th Brazilian Symposium on Neural Networks, IEEE (2000) 173–178

11. Miller, J.F.: Cartesian genetic programming. Springer (2011)
12. Harding, S., Miller, J.F.: Evolution of robot controller using cartesian genetic programming. In: Genetic programming. Springer (2005) 62–73
13. Khan, M.M., Khan, G.M.: A novel neuroevolutionary algorithm: Cartesian genetic programming evolved artificial neural network (cgpann). In: Proceedings of the 8th International Conference on Frontiers of Information Technology, ACM (2010) 48
14. Harding, S., Graziano, V., Leitner, J., Schmidhuber, J.: MT-CGP: Mixed type cartesian genetic programming. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation, ACM (2012) 751–758
15. Miller, J.F.: Evolution of digital filters using a gate array model. In: Evolutionary Image Analysis, Signal Processing and Telecommunications. Springer (1999) 17–30
16. Vasicek, Z., Sekanina, L.: Evolutionary approach to approximate digital circuits design. IEEE Transactions on Evolutionary Computation **19**(3) (2015) 432–444
17. Vasicek, Z.: Cartesian GP in optimization of combinational circuits with hundreds of inputs and thousands of gates. Genetic Programming (January 2015)
18. Miller, J.F., Smith, S.L.: Redundancy and computational efficiency in cartesian genetic programming. IEEE Transactions on Evolutionary Computation **10**(2) (2006) 167–174
19. Turner, A.J., Miller, J.F.: Cartesian genetic programming: Why no bloat? In: Genetic Programming. Springer (2014) 222–233
20. Van Mieghem, P.: Graph spectra for complex networks. Cambridge University Press (2011)
21. Van Mieghem, P., Omic, J., Kooij, R.: Virus spread in networks. IEEE/ACM Transactions on Networking **17**(1) (2009) 1–14
22. Fiedler, M.: Algebraic connectivity of graphs. Czechoslovak mathematical journal **23**(2) (1973) 298–305
23. Alon, N., Milman, V.D.: $\lambda 1$, isoperimetric inequalities for graphs, and superconcentrators. Journal of Combinatorial Theory, Series B **38**(1) (1985) 73–88
24. Mohar, B.: Eigenvalues, diameter, and mean distance in graphs. Graphs and combinatorics **7**(1) (1991) 53–64
25. Chung, F.R., Faber, V., Manteuffel, T.A.: An upper bound on the diameter of a graph from eigenvalues associated with its laplacian. SIAM Journal on Discrete Mathematics **7**(3) (1994) 443–457
26. Van Dam, E.R., Haemers, W.H.: Eigenvalues and the diameter of graphs. Linear and Multilinear Algebra **39**(1-2) (1995) 33–44
27. Mohar, B.: Isoperimetric numbers of graphs. Journal of Combinatorial Theory, Series B **47**(3) (1989) 274–291
28. Bollobás, B.: Random graphs. Springer (1998)
29. Goldman, B.W., Punch, W.F.: Analysis of cartesian genetic programming's evolutionary mechanisms. IEEE Transactions on Evolutionary Computation **19**(3) (2015) 359–373
30. Rossi, R.A., Ahmed, N.K.: An interactive data repository with visual analytics. SIGKDD Explor. **17**(2) (2016) 37–41