On the External Validity of Average-Case Analyses of Graph Algorithms

Thomas Bläsius
Karlsruhe Institute of Technology (KIT), Germany

Philipp Fischbeck
Hasso Plattner Institute (HPI), University of Potsdam, Germany

Abstract

The number one criticism of average-case analysis is that we do not actually know the probability distribution of real-world inputs. Thus, analyzing an algorithm on some random model has no implications for practical performance. At its core, this criticism doubts the existence of *external validity*, i.e., it assumes that algorithmic behavior on the somewhat simple and clean models does not translate beyond the models to practical performance real-world input.

With this paper, we provide a first step towards studying the question of external validity systematically. To this end, we evaluate the performance of six graph algorithms on a collection of 2751 sparse real-world networks depending on two properties; the heterogeneity (variance in the degree distribution) and locality (tendency of edges to connect vertices that are already close). We compare this with the performance on generated networks with varying locality and heterogeneity. We find that the performance in the idealized setting of network models translates surprisingly well to real-world networks. Moreover, heterogeneity and locality appear to be the core properties impacting the performance of many graph algorithms.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Average Case, Network Models, Empirical Evaluation

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.21

Related Version *Full Version*: `https://arxiv.org/abs/2205.15066` [4]

Supplementary Material We provide our source code, a docker image for easier reproducibility, the real-world network data set as well as all generated data (networks and statistics):
Software (Source Code): `https://github.com/thobl/external-validity`
Dataset (Source Code Archive, Docker Image, Networks, Statistics): `https://doi.org/10.5281/zenodo.6587324`

1 Introduction

The seminal papers of Cook in 1971 [16] and Karp in 1972 [26] establish that many fundamental combinatorial problems are NP-hard, and thus cannot be solved in polynomial time unless P = NP. Since then, the list of NP-hard problems is growing every year.

Though the non-existence of polynomial time algorithms (unless P = NP) is major bad news, the concept of NP-hardness is limited to the worst case. It thus leaves the possibility of imperfect algorithms that fail sometimes but run correctly and in polynomial time on most inputs. Many algorithms used today are slow in the worst case but perform well on relevant instances. An early attempt to theoretically capture this "good in practice" concept is the *average-case analysis*. There, one assumes the input to be randomly generated and then proves a low failure probability or a good expected performance.[1] On that topic, Karp wrote in 1983 [27] that

[1] We note that within the scope of this paper the term *average case* refers to exactly those situations where the input is drawn from some probability distribution. This includes proving bounds that hold with high probability (instead of in expectation), which would technically be better described as *typical case*. Moreover, it excludes the case of randomized algorithms on deterministic inputs.

> One way to validate or compare imperfect algorithms for NP-hard combinatorial problems is simply to run them on typical instances and see how often they fail. [...] While probabilistic assumptions are always open to question, the approach seems to have considerable explanatory power [...]. (Karp in 1983)

With this promising starting point, one could have guessed that average-case analysis is an important pillar of algorithmic research. However, it currently plays only a minor role in theoretical computer science. The core reason for this was summarized by Karp almost forty years later in 2020 in the Lex Fridman Podcast.[2]

> The field tended to be rather lukewarm about accepting these results as meaningful because we were making such a simplistic assumption about the kinds of graphs that we would be dealing with. [...] After a while I concluded that it didn't have a lot of bite in terms of the practical application. (Karp in 2020)

At its core, this describes the issue that an average-case analysis is lacking *external validity*, i.e., the insights on randomly generated graphs do not transfer to practical instances.

The simplistic probabilistic assumption mentioned in the above quotes is that input graphs are drawn from the Erdős–Rényi model [23], where all edges exist independently at random with the same probability. This assumption has the advantages that it is as unbiased as possible and sufficiently accessible to allow for mathematical analyses. However, in its simplicity, it is unable to capture the rich structural properties present in real-world networks, leading to the lack of external validity.

That being said, since the beginnings of average-case considerations, there have been several decades of research in the field of network science dedicated to understanding and explaining properties observed in real-world networks. This in particular includes the analysis of random network models and the transfer of insights from these models to real networks; indicating external validity. Thus, we believe that it is time to revisit the question of whether average-case analyses of graph algorithms can have external validity. With this paper, we present a first attempt at systematically studying this question.

Before describing our approach and stating our contribution, we want to give two examples from network science, where the existence of external validity is generally accepted.

Examples from Network Science. The Barabási–Albert model [2] uses the mechanism of *preferential attachment* to iteratively build a network. Each newly added vertex chooses a fixed number of neighbors among the existing vertices with probabilities proportional to the current vertex degrees. This simple mechanism yields *heterogeneous* networks, i.e., networks with power-law degree distributions where most vertices have a small degree while few vertices have very high degree.[3] It is well known that networks generated by this model are highly artificial, exhibiting properties that are far from what is observed in real-world networks. Nonetheless, beyond the specific model, it is generally accepted that the mechanism of preferential attachment facilitates power-law distributions. Thus, assuming external validity, the Barabási–Albert model can serve as an explanation of why we regularly observe power-law distributions in real-world data. Moreover, whenever we deal with a process involving preferential attachment, we should not be surprised when seeing a power-law distribution.

---

[2] Transcript of the Lex Fridman Podcast #111. The quote itself starts at 1:39:59. For the full context, start at 1:37:28 (`https://youtu.be/KllCrlfLuzs?t=5848`).

[3] Barabási and Albert were not the first to study a preferential attachment mechanism; see, e.g., Price's model [19]. However, there is no doubt that their highly influential paper [2] popularized the concept.

The Watts–Strogatz model [32] first starts with a ring lattice, i.e., the vertices are distributed uniformly on a circle and vertex pairs are connected if they are sufficiently close. This yields a regular graph with high *locality*, i.e., all connections are short and we observe many triangles. Moreover, ring lattices have high diameter. The second step of the Watts–Strogatz model introduces noise by randomly rewiring edges. This diminishes locality by replacing local connections with potentially long-range edges. Watts and Strogatz demonstrate that only little locality has to be sacrificed before getting a small-world network with low diameter. Again, this model is highly artificial and thus far from being a good representation for real-world networks. However, it seems generally accepted that these observations have implications beyond the specific model, namely that there is a simple mechanism that facilitates the small-world property even in networks with mostly local connections. Thus, in real-world settings where random long-range connections are possible, one should not be surprised to observe the small-world property.

Contribution. We consider algorithms for six different problems that are known to perform better in practice than the worst-case complexity would suggest. We evaluate them on network models that allow for varying amounts of locality and heterogeneity.[4] This shows us the impact of these two properties on the algorithms' performance in the controlled and clean setting of generated networks. We compare this with practical performance by additionally evaluating the algorithms on a collection of 2751 sparse real-world networks. Our overall findings can be summarized as follows. Though the real-world networks yield a less clean and more noisy picture than the generated networks, the overall dependence of the performance on locality and heterogeneity coincides with surprising accuracy for most algorithms. This indicates that there is external validity in the sense that if, e.g., increasing locality in the network models improves performance, we should also expect better performance for real-world networks with high locality. Moreover, it indicates that locality and heterogeneity are the core properties to impact the performance for many networks.

Our insights for the specific algorithms are interesting in their own right, independent of the question of external validity. Moreover, our experiments led to several interesting findings that are beyond the core scope of this paper. These can be found in the full version [4].

In Section 2, we introduce some basic notation and formally define measures for heterogeneity and locality. In Section 3, we describe the set of real-world and generated networks we use in our experiments. Section 4 compares generated and real-world networks for the different algorithms. Related work as well as our insights specific to the algorithms are discussed in this section. In Section 5 we conclude with a discussion of our overall results.

Our source code is available on GitHub[5]. It is additionally archived at Zenodo[6], together with a docker image for easier reproducibility. The latter repository additionally includes the real-world network data set as well as all generated data (networks and statistics). Details omitted from this extended abstract can be found in the full version [4].

[4] For non-local networks, we use the Erdős–Rényi and the Chung–Lu model for homogeneous and heterogeneous degree distributions, respectively. For local networks, we use geometric inhomogeneous random graphs (GIRGs), which let us vary the amount of locality and heterogeneity.

[5] `https://github.com/thobl/external-validity`

[6] `https://doi.org/10.5281/zenodo.6587324`

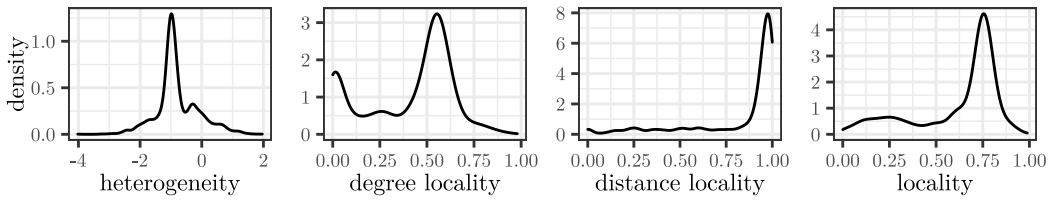

Figure 1 The density (kernel density estimation) of heterogeneity, degree locality, distance locality, and locality of the networks in our data set of real-world networks.

2 Basic Definitions, Heterogeneity, and Locality

Let $G = (V, E)$ be a graph. Throughout the paper, We denote the number of vertices and edges with $n = |V|$ and $m = |E|$. For $v \in V$, let $N(v) = \{u \mid \{u, v\} \in E\}$ be the *neighborhood* of v, and let $\deg(v) = |N(v)|$ be the *degree* of v. Additionally, $N[v] = N(v) \cup \{v\}$ is the *closed neighborhood* of v. An edge $e \in E$ is a *bridge* if $G - e$ is disconnected, where $G - e$ denotes the subgraph induced by $E \setminus \{e\}$.

We define the *heterogeneity* of a graph as the logarithm (base 10) of the coefficient of variation of its degree distribution, i.e., it is $\log_{10}(\sigma/\mu)$ for the average degree $\mu = \frac{1}{n}\sum_{v\in V}\deg(v)$ and the variance $\sigma^2 = \frac{1}{n}\sum_{v\in V}(\deg(v) - \mu)^2$.

We define the locality based on two different measures. For $u, v \in V$, let $\deg(u, v) = |N(u) \cap N(v)|$ be the *common degree* of u and v. For a non-bridge edge $\{u, v\} \in E$ the *degree locality* is defined as

$$L_{\deg}(\{u, v\}) = \frac{\deg(u, v)}{\min(\deg(u), \deg(v)) - 1}.$$

It essentially measures in how many triangles $\{u, v\}$ appears. The *degree locality* $L_{\deg}(G)$ of G is the average degree locality over all non-bridge edges.

For $u, v \in V$, let $\operatorname{dist}(u, v)$ be the distance between u and v in G. For $P \subseteq \binom{V}{2}$, we denote the average distance between vertex pairs in P with $\operatorname{dist}(P)$. For $\{u, v\} \in E$, the *detour distance* $\operatorname{dist}^+(u, v)$ is the distance between u and v in $G - \{u, v\}$. Let $\overline{E} = \binom{V}{2} \setminus E$ and assume that $\overline{E} \neq \emptyset$. For a non-bridge edge $\{u, v\} \in E$, define the *distance locality* as

$$L_{\operatorname{dist}}(\{u, v\}) = 1 - \frac{\operatorname{dist}^+(u, v) - 2}{\operatorname{dist}(\overline{E}) - 2}.$$

It essentially measures how short the edge $\{u, v\}$ is compared to the average distance in the graph. If $\operatorname{dist}(\overline{E}) = 2$, we define $L_{\operatorname{dist}}(\{u, v\}) = 0$. The *distance locality* $L_{\operatorname{dist}}(G)$ of G is the maximum of 0 and the average distance locality over all non-bridge edges.

The *locality* of G is $L(G) = (L_{\deg}(G) + L_{\operatorname{dist}}(G))/2$. To compute the locality, we approximate average distances [12]. See Figure 1 for distribution plots.

3 Networks

Real-World Networks. We use 2751 graphs from Network Repository [29], which we selected as follows. We started with all networks with at most 1 M edges and reduced each network to its largest connected component. We removed multi-edges and self-loops and ignored weights or edge directions. We tested the resulting connected, simple, undirected, and unweighted graphs for isomorphism and kept only one copy for each group of isomorphic networks. This resulted in 3006 networks. From this we removed 255 networks for different reasons.

- We removed 105 networks that were randomly generated and thus do not count as real-world networks.[7]
- We removed 17 trees. They are not very interesting, and locality is not defined for trees.
- We removed 133 graphs with density at least 10 %. Our focus lies on sparse graphs and network models for sparse graphs. Thus, dense graphs are out of scope.

Random Networks. We use three random graph models to generate networks; the Erdős–Rényi model [23] (non-local and homogeneous), the Chung–Lu model [14, 15] (non-local and varying heterogeneity), and the GIRG model [11] (varying locality and heterogeneity). For the latter, we use the efficient implementation in [8].

For each model and parameter configuration, we generate five networks with $n = 50\,\mathrm{k}$ vertices and (expected) average degree 10. As for the real-world networks, we reduce each generated graph to its largest connected component. We average the five generated graphs for each parameter configuration and represent their average as singe dot in the plots.

4 Comparison Between the Models and Real-World Networks

Each of the following subsections compares the performance of a different algorithm between generated and real-world networks. For the cost c of the algorithm, we plot c depending on heterogeneity and locality using color to indicate the cost; see, e.g., Figure 2. The left and middle plot show one data point for each parameter setting of the models and each real-world network, respectively. The right plot aggregates real-world networks with similar locality and heterogeneity. Each point represents a number of networks indicated by its radius (log scale). We regularly assume the cost c to be polynomial in m (or n), i.e., $c = m^x$, and plot $x = \log_m c$. In this case, the color in the left and right plot shows the mean exponent x for the aggregated networks.

4.1 Bidirectional Search

We can compute a shortest path between two vertices s and t using a *breadth first search (BFS)*. The BFS explores the graph layer-wise, where the ith layer contains the vertices of distance i from s. By *exploring* the ith layer, we refer to the process of iterating over all edges with an endpoint in the ith layer, thereby finding layer $i + 1$. The *bidirectional BFS* alternates the exploration of layers between a *forward BFS* from s and a *backward BFS* from t. The alternation strategy we study here greedily applies the cheaper of the two explorations in every step. The cost of exploring a layer is estimated via the sum of degrees of vertices in that layer. The search stops once the current layers of forward and backward search intersect.

The *cost* c for the bidirectional BFS is the average number of edge explorations over 100 random st-pairs. Note that $c \leq 2m$, as each edge can be explored at most twice; once from each side. Figure 2 shows the exponent x for $c = m^x$ depending on heterogeneity and locality.

Impact of Locality and Heterogeneity. For the generated networks, we see that networks with high locality and homogeneous degree distribution (top left corner) have an exponent of around 1 (red). Thus, the cost of the bidirectional search is roughly m, which matches

[7] Finding generated networks was done manually by searching for suspicious naming patterns or graph properties. For each candidate, we checked its source to verify that it is a generated network. Though we checked thoroughly, there are probably a few random networks hiding among the real-world networks.

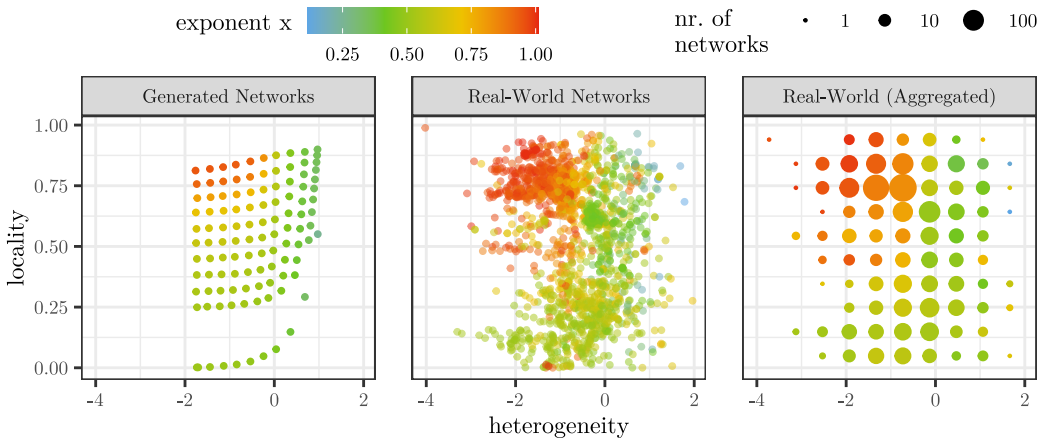

Figure 2 The exponent x of the average cost $c = m^x$ of the bidirectional BFS over 100 st-pairs.

the worst-case bound. If the network is heterogeneous (right) or less local (bottom), we get significantly lower exponents of around 0.5, indicating a cost of roughly $\sqrt{m}$. The cost is particularly low for very heterogeneous networks. Overall, we get a strict separation between hard (high locality, low heterogeneity) and easy (low locality or high heterogeneity) instances.

For real-world networks, we observe the same overall behavior that instances that are local and homogeneous tend to be hard while all others tend to be easy. There are only few exceptions to this, indicating that the heterogeneity and locality are usually the crucial properties impacting the performance of the bidirectional BFS.

Discussion. Borassi and Natale [10] found that the bidirectional BFS is exceptionally efficient on many real-world networks. Though its efficiency is surprising when compared to the worst case, existing average-case considerations suggest that this is actually the expected behavior. It runs in sublinear time on the non-local Erdős–Rényi and Chung–Lu graphs [10]. It is also sublinear on the local and heterogeneous hyperbolic random graphs, but linear on geometric random graphs in Euclidean geometry [7]. The latter two models can be seen as special case of GIRGs, covering the top-right and top-left corner.

4.2 Diameter

The *eccentricity* of $s \in V$ is $\max_{t \in V} \text{dist}(s,t)$, i.e., the distance to the vertex farthest from s. The *diameter* of the graph G is the maximum eccentricity of its vertices. It can be computed using the *iFUB algorithm* [17]. It starts with a root $r \in V$ from which it computes the BFS tree T. It then processes the vertices bottom up in T and computes their eccentricities using a BFS per vertex. This process can be pruned when the distance to r is sufficiently small compared to the largest eccentricity found so far. Pruning works well if r is a central vertex in the sense that it has low distance to many vertices and a shortest path between distant vertices gets close to r. The central vertex r is selected as follows. A *double sweep* [28] starts with a vertex u, chooses a vertex v at maximum distance from u, and returns a vertex w from a middle layer of the BFS tree from v. A *4-sweep* [17] consists of two double sweeps, starting the second sweep with the result w of the first sweep. We consider the *iFUB+4-sweephd* algorithm, which chooses r by doing a 4-sweep from a vertex of maximum degree.

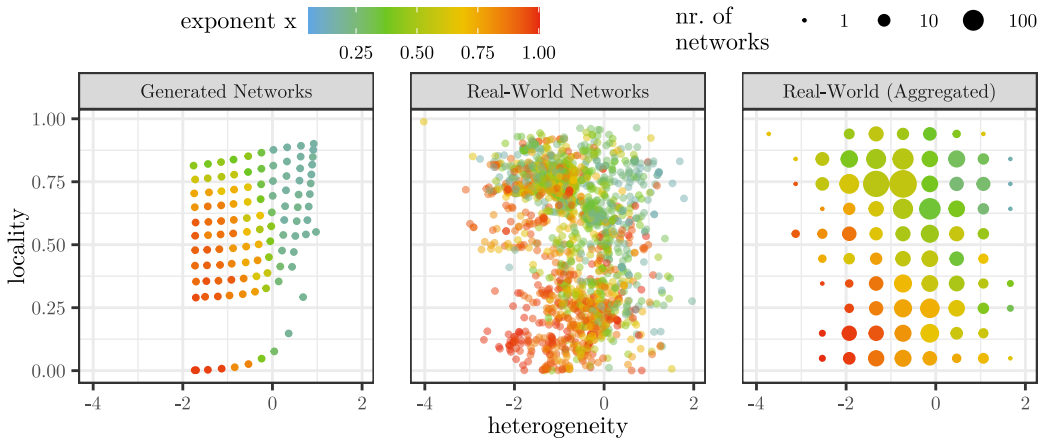

Figure 3 The exponent x of the number of BFSs $c = n^x$ of the iFUB+4-sweephd algorithm, excluding 19 real-world networks with timeout. The GIRG ground space is a square.

The *cost* c of iFUB+4-sweephd is the number of BFSs it performs. Note that $c \leq n$. Figure 3 shows the exponent x for $c = n^x$ depending on heterogeneity and locality. It excludes 19 of the 2751 real-world networks that exceeded the time limit of 30 min. The GIRG model uses a square as ground space instead of the usual torus.

Impact of Locality and Heterogeneity. The general dependence is the same for generated and real-world networks. An almost linear number of BFSs is required for networks lacking locality and heterogeneity. For local or heterogeneous networks, it is substantially sublinear. The picture for real-world networks shows some noise, which indicates that there are properties besides locality and heterogeneity that impact the performance. We note that this observation agrees with the models, where we get highly varying exponents for individual parameter settings, e.g., for GIRGs with $T = 0.82$ and $\beta = 3.4$, we get exponents ranging from 0.21 to 0.92 for the five generated instances.

Discussion. The iFUB algorithm was introduced by Crescenzi, Grossi, Habib, Lanzi, and Marino [17]. They note that it works well on networks with high difference between radius and diameter. This corresponds to the existence of a central vertex with eccentricity close to half the diameter. Additional experiments indicate that locality facilitates the existence of a central vertex (essentially in the center of the geometric ground space). Moreover, in heterogeneous networks, the high degree vertices serve as central vertices. The latter is also supported by the theoretical analysis of Borassi, Crescenzi, and Trevisan [9], who in particular showed that heterogeneous Chung–Lu graphs allow the computation of the diameter in sub-quadratic time, indeed using a vertex of high degree as central vertex.

4.3 Vertex Cover Domination

A vertex set $S \subseteq V$ is a *vertex cover* if every edge has an endpoint in S, i.e., removing S from G leaves a set of isolated vertices. We are interested in finding a vertex cover of minimum size. For two adjacent vertices $u, v \in V$, we say that u *dominates* v if $N[v] \subseteq N[u]$. The *dominance rule* states that there exists a minimum vertex cover that includes u. Thus, one can reduce the instance by including u in the vertex cover and removing it from the graph.

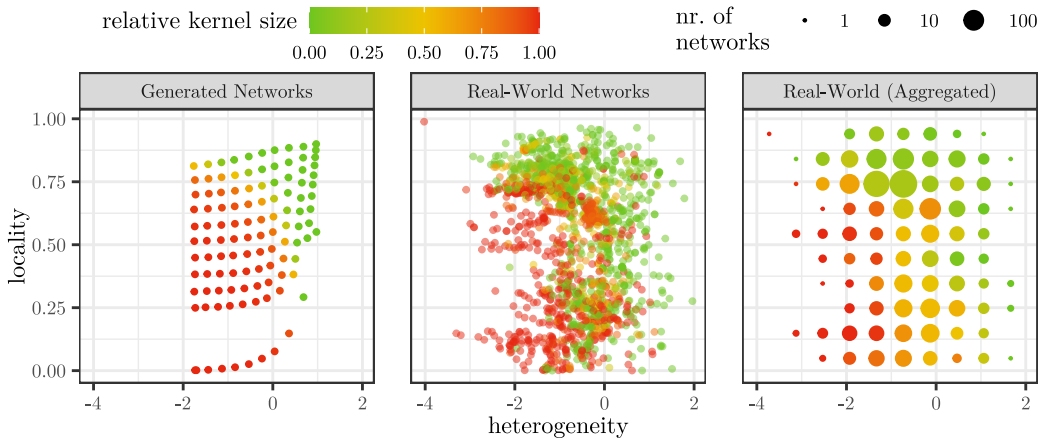

Figure 4 The relative kernel size of the vertex cover domination rule.

To evaluate the effectiveness of the dominance rule, we apply it exhaustively, i.e., until no dominant vertices are left. Moreover, we remove isolated vertices. We refer to the number c of vertices in the largest connected component of the remaining instances as the *kernel size*. Figure 4 shows the *relative kernel size* c/n with respect to locality and heterogeneity.

Impact of Locality and Heterogeneity. We see a sharp separation for the generated networks. For low locality and heterogeneity (bottom left), the reduction rule cannot be applied. For high locality and heterogeneity (top right), the dominance rule completely solves the instance. For real-world networks, the separation is less sharp, i.e., there is a larger range of locality/heterogeneity values in the middle where the dominance rule is effective sometimes. Nonetheless, we see the same trend that the reduction rule is more likely to be effective the higher the locality and heterogeneity. In the extreme regimes (bottom left or top right), we observe the same behavior as for the generated networks with relative kernel sizes close to 1 and 0, respectively, for almost all networks. Moreover, there is dichotomy in the sense that many instances are either (almost) completely solved by the dominance rule or the rule is basically inapplicable.

Discussion. Though vertex cover is NP-hard [26], it is rather approachable: It can be solved in $1.1996^n n^{O(1)}$ time [34] and there is a multitude of FPT-algorithms with respect to the solution size k [18], the fastest one running in $O(1.2738^k + kn)$ [13]. Moreover, there are good practical algorithms [1, 33]. Both algorithms apply a suite of reduction rules, including the dominance rule or a generalization. We note that the dominance rule is closely related to Weihe's reduction rules for hitting set [33]. Previous experiments for Weihe's reduction rules match our results: they work well if the instances are local and heterogeneous [6]. Concerning theoretic analysis on models, we know that on hyperbolic random graphs, the dominance rule is sufficiently effective to yield a polynomial time algorithm [5]. Thus, it is not surprising that the top-right corner in Figure 4 is mostly green.

4.4 The Louvain Algorithm for Graph Clustering

Let $V_1 \cup \cdots \cup V_k = V$ be a *clustering* where each vertex set V_i is a *cluster*. One is usually interested in finding clusterings with dense clusters and few edges between clusters, which is formalized using some quality measure, e.g., the so-called modularity. A common subroutine

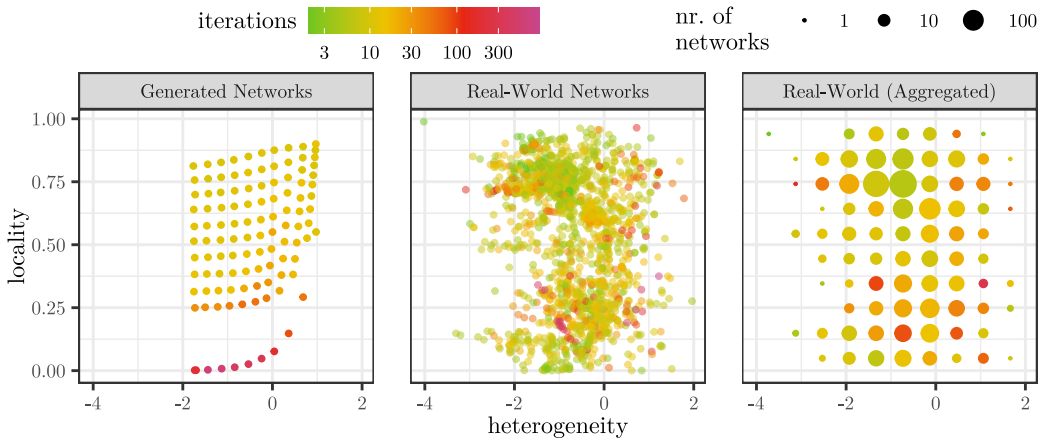

Figure 5 Number of iterations of the first local search of the Louvain algorithm. The color scale is logarithmic. Four outliers (real-world) with more than 1 k iterations are excluded.

in clustering algorithms is to apply the following local search. Start with every vertex in its own cluster. Then, check for each vertex $v \in V$ whether moving v into a neighboring cluster improves the clustering. If so, v is moved into the cluster yielding the biggest improvement. This is iterated until no improvement can be achieved. Doing this with the modularity as quality measure (and subsequently repeating it after contracting clusters) yields the well-known Louvain algorithm [3].

The run time of the Louvain algorithm is dominated by the number of iterations of the initial local search. Figure 5 shows this number of iterations.

Impact of Locality and Heterogeneity. For the generated instances, the number of iterations is generally low but increasing when decreasing the locality. For GIRGs, the average number of iterations ranges from 10.2 to 54.2 for the different parameter configurations. Moreover, the number of iterations starts to rise only for low localities. For 86 % of the configurations the average number of iterations lies below 20. For the Erdős–Rényi graphs we obtain an average of 196.4 iterations and for Chung–Lu graphs it goes up to 591.4 for one configuration. However, these average values over five generated instances have to be taken with a grain of salt as there is a rather high variance, e.g., the number of iterations for the five Chung–Lu graphs with power-law exponent 25 ranges from 82 up to 312.

For the real-world networks there is no clear trend depending on locality or heterogeneity. In general, the number of iterations is rather low except for some outliers. While the strongest outlier requires almost 20 k iterations, 98.6 % of the networks have at most 100 iterations.

Discussion. The worst-case number of iterations of the Louvain algorithm can be upper bounded by $O(m^2)$ due to the fact that the modularity is bounded and each vertex move improves it by at least $\Omega(1/m^2)$. Moreover, there exists a graph family that requires $\Omega(n)$ iterations for the first local search [25, Proposition 3.1].

Our experiments indicate that locality or heterogeneity are not the properties that discriminate between easy and hard instances. For generated instances, there is the trend that low locality increases the number of iterations, which does not transfer to the real-world networks (or is at least less clearly). However, the general picture that most instances require few iterations while there are some outliers coincides for generated and real-world networks.

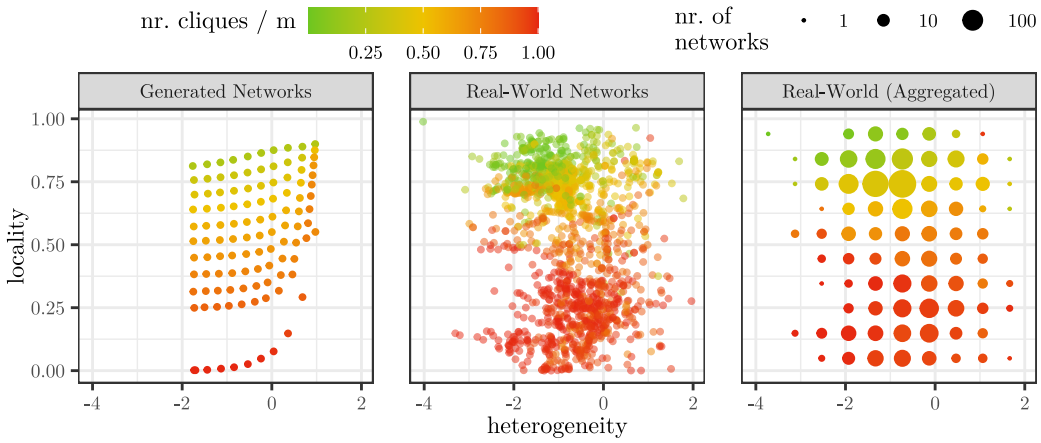

Figure 6 The number of maximal cliques relative to m depending on the heterogeneity and locality, restricted to networks where this value is at most 1 (93 % of the networks).

4.5 Maximal Cliques

A *clique* is a subset of vertices $C \subseteq V$ such that C induces a complete graph, i.e., any pair of vertices in C is connected. A clique is *maximal*, if it is not contained in any other clique. In the following, we are never interested in non-maximal cliques. Thus, whenever we use the term *clique*, it implicitly refers to a maximal clique. To enumerate all cliques, we used the algorithm by Eppstein, Löffler, and Strash [20], using their implementation [21, 22].

As the cliques of a network can be enumerated in polynomial time per clique [30], the number of maximal cliques is a good indicator for the hardness of an instance. For all generated and most real-world networks, the number of cliques does not exceed the number of edges m. Out of the 2751 real-world networks, 2556 (93 %) have at most m and 193 have more than m cliques. For the remaining 2 networks, the timeout of 30 min was exceeded. Figure 6 shows the number of cliques relative to m for all networks with at most m cliques.

Impact of Locality and Heterogeneity. One can see that the networks (generated and real-world) with low locality have roughly m cliques, while the number of cliques decreases for increasing locality. Moreover, among networks with locality, there is the slight trend that networks with higher heterogeneity have more cliques.

It is not surprising that networks with low locality have roughly m cliques, as graphs without triangles have exactly m cliques. For graphs with higher locality, there are two effects counteracting each other. On the one hand, multiple edges combine into larger cliques, which decreases the number of cliques. On the other hand, each edge can be contained in multiple cliques, which increases the number of cliques. Our experiments show that the former effect usually supersedes the latter, i.e., the size of the cliques increases more than the number of cliques each edge is contained in.

Discussion. There are many results on enumerating cliques and on the complementary problem of enumerating independent sets. Here, we focus on discussing two results that are closely related. Eppstein, Löffler and Strash [21] give an algorithm for enumerating all cliques that runs in $O(dn3^{d/3})$ time, where d is the degeneracy (a measure for sparsity). We use this algorithm for our experiments as it performs well in practice.

Fox, Roughgarden, Seshadhri, Wei, and Wein [24] introduced the closure as a measure that captures the tendency that common neighbors facilitate direct connections, i.e., as a measure for locality. Additionally, they introduced the weak closure as a more robust measure. Fox et al. show that weakly c-closed graphs have at most $3^{(c-1)/3}n^2$ cliques. For c-closed graphs they give the additional upper bound of $4^{(c+4)(c-1)/2}n^{2-2^{1-c}}$.

Qualitatively, at a first glance, these theoretical results seem to match our observations on real-world networks: sparse graphs contain few cliques and there are fewer cliques for more local networks (small c). However, a closer look reveals two caveats. First, both upper bounds are exponential in the parameter while 93 % of real-world networks have at most m cliques. Second, the (weak) closure does not actually correlate with the number of cliques or the locality. Weak closure and degeneracy only seem to be a good measure for hardness on the remaining 7 % of the instances. Thus, on the large majority of instances, the generative models yield a much better fit.

4.6 Chromatic Number

We study the effectiveness of a reduction rule for computing the chromatic number based on the size k of the maximum clique [31]. It reduces a graph to its k-core and is thus closely related to the degeneracy of the network.

Whether the reduction rule performs well comes, at its core, down to how the clique size k compares to the degeneracy d. We observe that both, k and d, mainly depend on locality and heterogeneity and behave very similar on real-world and generated networks. However, k and d have the same dependence on locality and heterogeneity, which means that (at least for some regimes) other properties have to tip the scale. We in particular observe a dependence on the average degree for the generated networks; the higher the average degree, the worse the reduction rule performs. More details on this can be found in the full version of the paper [4].

5 Discussion and Conclusion

Networks from different domains have varying structural properties. Thus, trying to find probability distributions that match or closely approximate those of real-world networks seems like a hopeless endeavor. Moreover, even if we knew such a probability distribution, it would most likely be highly domain specific and too complicated for theoretical analysis.

Our View on Average-Case Analysis. A more suitable approach to average case analysis is the use of models that assume few specific structural properties and are as unbiased as possible beyond that. If the chosen properties are the dominant factors for the algorithm's performance, we obtain external validity, i.e., the results translate to real-world instances even though they do not actually follow the assumed probability distributions. There are two levels of external validity.

- The models capture the performance-relevant properties sufficiently well that algorithms perform similar on generated and real-world networks.
- The models are too much of an idealization for this direct translation to practical performance. However, varying a certain structural property in this idealized world has the same qualitative effect on performance as it has on real-world networks.

Though an average case analysis cannot yield strong performance guarantees, with the above notions of external validity, it can give insights into what properties are crucial for performance (first level) and how the performance depends on a property (second level). Moreover, even a

lack of external validity can yield valuable insights in the following sense. Assume we have the hypothesis that property X is the crucial factor for algorithmic performance and thus we study a model with property X as null model. Then, a lack of external validity lets us refute the hypothesis as there clearly has to be a different property impacting the performance.

Impact of Locality and Heterogeneity. Non-surprisingly, the performance on real-world networks depending on locality and heterogeneity is more noisy compared to the generated networks, as real-world networks are diverse and vary in more than just these two properties. That being said, the observations on the models and in practice coincide almost perfectly for the bidirectional search and the enumeration of maximal cliques. For the maximal cliques, the match even includes constant factors, which is particularly surprising, as these numbers are below m while the worst case is exponential.

For the vertex cover domination rule as well as the iFUB algorithm, we obtain a slightly noisier picture. However, the overall trend matches well, which indicates that locality and heterogeneity are crucial factors for the performance, but not the only ones. For the iFUB algorithm, we already identified the existence of central vertices as additional factor (difference between torus and square as underlying geometry).

For the chromatic number, we observe that heterogeneity and locality are important but not the only factors impacting performance. Nonetheless, the performance is similar on the models compared to real-world networks.

For the Louvain clustering algorithm, the models and real-world networks coincide insofar, that the number of iterations is low, with few exceptions. This indicates that locality and heterogeneity are not the core properties for differentiating between hard and easy instances.

Conclusions. Locality and heterogeneity have significant impact on the performance of many algorithms. We believe that it is useful for the design of efficient algorithms to have these two dimensions of instance variability in mind. Moreover, GIRGs [11] with the available efficient generator [8] provide an abundance of benchmark instances on networks with varying locality and heterogeneity. Finally, we believe that average case analyses on the four extreme cases can help to theoretically underpin practical run times. The four extreme cases can, e.g., be modeled using geometric random graphs for local plus homogeneous, hyperbolic random graphs[8] for local plus heterogeneous, Erdős–Rényi graphs for non-local plus homogeneous, and Chung–Lu graphs for non-local plus heterogeneous networks.

References

1 Takuya Akiba and Yoichi Iwata. Branch-and-reduce exponential/FPT algorithms in practice: A case study of vertex cover. *Theoretical Computer Science*, 609:211–225, 2016. doi:10.1016/j.tcs.2015.09.023.

2 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999. doi:10.1126/science.286.5439.509.

3 Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008. doi:10.1088/1742-5468/2008/10/p10008.

4 Thomas Bläsius and Philipp Fischbeck. On the external validity of average-case analyses of graph algorithms. *Computing Research Repository (CoRR)*, abs/2205.15066, 2022. doi:10.48550/arXiv.2205.15066.

[8] GIRGs can be seen as common generalization of geometric and hyperbolic random graphs.

5 Thomas Bläsius, Philipp Fischbeck, Tobias Friedrich, and Maximilian Katzmann. Solving vertex cover in polynomial time on hyperbolic random graphs. *Theory of Computing Systems*, 2021. doi:10.1007/s00224-021-10062-9.

6 Thomas Bläsius, Philipp Fischbeck, Tobias Friedrich, and Martin Schirneck. Understanding the effectiveness of data reduction in public transportation networks. In *Workshop on Algorithms and Models for the Web-Graph (WAW)*, pages 87–101, 2019. doi:10.1007/978-3-030-25070-6_7.

7 Thomas Bläsius, Cedric Freiberger, Tobias Friedrich, Maximilian Katzmann, Felix Montenegro-Retana, and Marianne Thieffry. Efficient shortest paths in scale-free networks with underlying hyperbolic geometry. *ACM Transactions on Algorithms (TALG)*, 18(2):19:1–19:32, 2022. doi:10.1145/3516483.

8 Thomas Bläsius, Tobias Friedrich, Maximilian Katzmann, Ulrich Meyer, Manuel Penschuck, and Christopher Weyand. Efficiently generating geometric inhomogeneous and hyperbolic random graphs. In *European Symposium on Algorithms (ESA)*, pages 21:1–21:14, 2019. doi:10.4230/LIPIcs.ESA.2019.21.

9 Michele Borassi, Pierluigi Crescenzi, and Luca Trevisan. An axiomatic and an average-case analysis of algorithms and heuristics for metric properties of graphs. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 920–939, 2017. doi:10.1137/1.9781611974782.58.

10 Michele Borassi and Emanuele Natale. KADABRA is an ADaptive algorithm for betweenness via random approximation. *ACM Journal of Experimental Algorithmics (JEA)*, 24(1):1.2:1–1.2:35, 2019. doi:10.1145/3284359.

11 Karl Bringmann, Ralph Keusch, and Johannes Lengler. Geometric inhomogeneous random graphs. *Theoretical Computer Science*, 760:35–54, 2019. doi:10.1016/j.tcs.2018.08.014.

12 Shiri Chechik, Edith Cohen, and Haim Kaplan. Average distance queries through weighted samples in graphs and metric spaces: High scalability with tight statistical guarantees. In *International Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 659–679, 2015. doi:10.4230/LIPIcs.APPROX-RANDOM.2015.659.

13 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010. doi:10.1016/j.tcs.2010.06.026.

14 Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002. doi:10.1073/pnas.252631999.

15 Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6(2):125–145, 2002. doi:10.1007/PL00012580.

16 Stephen A. Cook. The complexity of theorem-proving procedures. In *Symposium on the Theory of Computing (STOC)*, pages 151–158, 1971. doi:10.1145/800157.805047.

17 Pilu Crescenzi, Roberto Grossi, Michel Habib, Leonardo Lanzi, and Andrea Marino. On computing the diameter of real-world undirected graphs. *Theoretical Computer Science*, 514:84–95, 2013. doi:10.1016/j.tcs.2012.09.018.

18 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

19 Derek J. de Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the Association for Information Science and Technology (JASIST)*, 27(5):292–306, 1976. doi:10.1002/asi.4630270505.

20 David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in sparse graphs in near-optimal time. In *International Symposium on Algorithms and Computation (ISAAC)*, pages 403–414, 2010. doi:10.1007/978-3-642-17517-6_36.

21 David Eppstein, Maarten Löffler, and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. *ACM Journal of Experimental Algorithmics (JEA)*, 18, 2013. doi:10.1145/2543629.

22 David Eppstein and Darren Strash. Listing all maximal cliques in large sparse real-world graphs. In *Symposium on Experimental and Efficient Algorithms (SEA)*, pages 364–375, 2011. doi:10.1007/978-3-642-20662-7_31.

23 P. Erdős and A. Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959. URL: https://www.renyi.hu/~p_erdos/1959-11.pdf.

24 Jacob Fox, Tim Roughgarden, C. Seshadhri, Fan Wei, and Nicole Wein. Finding cliques in social networks: A new distribution-free model. *SIAM Journal on Computing*, 49(2):448–464, 2020. doi:10.1137/18M1210459.

25 Andrea Kappes. *Engineering Graph Clustering Algorithms*. PhD thesis, Karlsruhe Institute of Technology, 2015. URL: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000049269.

26 Richard M. Karp. Reducibility among combinatorial problems. In *Computational Complexity Conference (CCC)*, pages 85–103, 1972. doi:10.1007/978-1-4684-2001-2_9.

27 Richard M. Karp. The probabilistic analysis of combinatorial optimization algorithms. In *International Congress of Mathematicians (ICM)*, volume 2, pages 1601–1609, 1983. URL: https://www.mathunion.org/fileadmin/ICM/Proceedings/ICM1983.2/ICM1983.2.ocr.pdf.

28 Clémence Magnien, Matthieu Latapy, and Michel Habib. Fast computation of empirically tight bounds for the diameter of massive graphs. *ACM Journal of Experimental Algorithmics (JEA)*, 13, 2008. doi:10.1145/1412228.1455266.

29 Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 4292–4293, 2015. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9553.

30 Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi, and Isao Shirakawa. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing*, 6(3):505–517, 1977. doi:10.1137/0206036.

31 Anurag Verma, Austin Buchanan, and Sergiy Butenko. Solving the maximum clique and vertex coloring problems on very large sparse networks. *INFORMS Journal on Computing*, 27(1):164–177, 2015. doi:10.1287/ijoc.2014.0618.

32 Duncan J. Watts and Steven H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998. doi:10.1038/30918.

33 Karsten Weihe. Covering trains by stations or the power of data reduction. In *Algorithms and Experiments (ALEX)*, pages 1–8, 1998.

34 Mingyu Xiao and Hiroshi Nagamochi. Exact algorithms for maximum independent set. *Information and Computation*, 255:126–146, 2017. doi:10.1016/j.ic.2017.06.001.