Karlsruhe Reports in Informatics 2011,13
Edited by Karlsruhe Institute of Technology,
Faculty of Informatics
ISSN 2190-4782

An Experimental Study on Generating Planar Graphs

Sascha Meinert and Dorothea Wagner

2011

Fakultät für **Informatik**

An Experimental Study on Generating Planar Graphs

Sascha Meinert and Dorothea Wagner

Karlsruhe Institute of Technology (KIT), 76128 Karlsruhe, Germany,
{sascha.meinert,dorothea.wagner}@kit.edu

Abstract. We survey several planar graph generators that were selected according to availability, theoretical interest, easiness of implementation and efficiency. We experimentally study graph properties that allow for a basic classification of the generators. This analysis is extended by means of advanced algorithmical behavior on the generated graphs, in particular kernelization of fixed-parameter tractable problems. We will see the major influence of instance selection on algorithmic behavior. This selection has been disregarded in several publications, which deduce general results from non-representative data sets. Altogether, this study helps experimenters to carefully select sets of planar graphs that allow for a meaningful interpretation of their results.

1 Introduction

Planar graphs are an important class of graphs since many algorithms are custom tailored and thus perform much better on this subclass than in the general case. For an experimental analysis, experimenters have to set up data sets that attest a broad applicability to algorithms. These data sets should be described precisely. However, many publications fail by either sketchy descriptions or inappropriate data set selection. For instance, works exist that describe the use of the LEDA library [17] for random planar graph generation [2, 1, 6]. They speak either of *the* random planar graph generator or *a specific representative* planar graph generating function of LEDA. In fact, LEDA offers at least five functions to randomly generate planar graphs. They are named similar or, even worse, identical and their behavior is completely different, depending on the parameters used. Obviously, experimenters want easy to use planar graph generators and do not want to care about inherent generator details. They use one generator in the hope that a representative data set is created. As we will see later this is a crucial misestimation that may either completely flaw or at least bias experimental results. Note that due to space restrictions some details had to be omitted which can be found in [18].
Related Work. Ideally, an algorithm should be tested on all graphs of a given class to conclude that it performs well. On several classes of planar graphs *Plantri*[1] [15] would be a good choice, as it is capable of generating all isomorphism-

[1] Excluded in our experimental study as it is infeasible to generate all non-isomorphic planar graphs of large size.

free graphs of certain graph size. As the sheer number of planar graphs is way too large it is obvious to ask for a representative sample. Thus, planar graphs that are generated uniformly at random [13, 8, 5] may be an appropriate choice. The implementation of such a generator is a challenging issue. Hence, non uniform but easier to implement generators gain importance. These can often be found in algorithmic libraries, e.g., the LEDA library [17].

Contribution. We empirically study eight selected planar graph generators by means of running time, basic graph properties and algorithmic behavior. Additionally, we report on the *completeness* of each generator, i.e., whether it is able to generate all planar graphs with positive probability. This work gives an overview of several planar graph generators by means of theoretical background and algorithmic behavior, which enables experimenters to compile representative data sets that allow for a meaningful interpretation of algorithmical experiments and their reproducibility.

Outline. First, the functionality, and theoretical background of each planar graph generator is described in Section 2. Additionally, this section provides information about the implementations, which includes abbreviated library function calls, and their modifications where it became necessary. Then, the planar graph generators are experimentally surveyed in Section 3. There, we first describe our generated data sets and our recommended parameter selection. The generated graphs are used to analyze basic graph properties in local as well as global scope to classify the graph generators. Afterwards, we sharpen this classification by means of algorithmic behavior, e.g., *fixed parameter tractable* (FPT) kernelization algorithms. Finally, we conclude this work with a recommendation which generators should be chosen to create a compilation of data sets that allows for meaningful interpretations of experiments.

2 Graph Generators

From here on a graph $G = (V, E)$ is considered to be a simple, labeled, undirected and planar graph, where V is the set of vertices $\{1, \dots, n\}$ and E is the set of edges. A planar graph can be embedded in the plane without edge crossings. A planar graph together with a planar embedding is called a planar map. The selected graph generators presented in this section either are available in libraries, have low running times, are of theoretical interest or are based on ideas that are easy to implement. In our opinion this collection represents the important planar graph generators available. We separate the generators into two groups according to the input parameters. The first group are *(n)-generators* that take as a parameter the number of nodes n. Clearly, the number of edges of graphs generated by such a generator is random. The second group consists of the generators additionally taking the number of edges m as a parameter, denoted by *(n,m)-generator*. They allow for an arbitrary number of edges. As always, a degree of freedom implies the difficult decision on how parameters have to be selected reasonably.

For our evaluation we created benchmark sets with each generator. Each planar graph generator had to generate graphs of size 5, 10, 25, 50, 75, 100, 250, 500, 1 000, 2 500 and 5 000 nodes n in increasing size. Since some graph generators have high running time we set a cut-off time to 14 days. For each graph size n we generated $4n$ graphs. If a graph generator failed to create the requested number of graphs within the time limit we report this behavior in the detailed generator descriptions later on. In the case of the (n,m)-generators, for each node count, graphs were generated having $m = in/4$ edges for $i = 1, \ldots, 11$. The number of graphs generated for each of the n, m combinations is $4n$. All (n,m)-graphs were generated within the time limit. Note that the (n,m)-generators presented here first generate a maximal planar graph and then randomly delete edges until m edges remain.

2.1 (n)-generators

The generators presented here can be subdivided into two groups according to their generation process. The first group consists of combinatorial generators, whereas the geometric generators form the second group. All of the presented (n)-generators are complete. Two of them are expected to draw planar graphs uniformly at random from the set of all planar graphs with the given vertex set $v = \{1, \ldots, n\}$.

Fusy. The planar graph generator developed by Fusy [13] is based on the principles of a Boltzmann Sampler [11]. Labeled graphs of size n are drawn uniformly at random. The running time is in $\mathrm{O}(n^2)$ if the exact number of nodes is sampled or $\mathrm{O}(\frac{n}{\epsilon})$ for an approximated graph size within $[n(1-\epsilon), n(1+\epsilon)]$. The available implementation [12] is the linear-time version of the algorithm. The sampler is based on probabilities described using generating functions, which have to be evaluated laboriously for every number of nodes in a preprocessing step. Note that the Fusy generator cannot be used as an out of the box generator for two reasons. First, the implementation allows the graphs only to consist either of 1 000, 10 000 or 100 000 nodes. Second, only a marginal number of graphs were close to the desired size, i.e., within the interval $[0.95n, 1.05n]$. In particular, the ratio of all generated graphs that have inappropriate size $\hat{n}$ was on average 77% for $3 \leq \hat{n} < 0.95n$ and 21% for $25n \geq \hat{n} > 1.05n$, depending on the targeted graph size. Hence, only 2% of the generated graphs were within the targeted interval. We modified the generator to reject graphs of inappropriate size.

Markov Chain. The planar graph generator by Denise et al. is based on a simple Markov chain [8]. The algorithm chooses a pair of nodes u, v at random. Now the transitions are as follows. If edge $e = (u, v)$ exists, it is deleted. If not, a check is done whether the graph $G = (V, E + e)$ is planar. In the case of planarity the edge is inserted. Otherwise it is discarded and the Markov chain remains in the current state. The stationary distribution of this Markov chain is uniform over all subgraphs of a given graph. Unfortunately, the mixing time of the Markov chain is unknown, but the authors expect that the equilibrium distribution should be reached after $3n^2$ iterations [8]. This behavior is verified by all our tests that we performed on 40 000 graphs of size 50 generated by $3n^2$,

n^3 and n^4 iterations. Generation of larger graph sizes is not feasible because of the long running time. The outcomes of all our tests were similar, independent of the number of iterations.
Kuratowski. The Kuratowski generator is based on the Kuratowski theorem, which states that a graph is planar if and only if no subgraph is present that is a subdivision of K_5 or $K_{3,3}$. The generator starts with a non-planar random graph. Iteratively, the algorithm searchs for a K_5 or $K_{3,3}$ subgraph and removes one of its edges until no more Kuratowski violations exist. Clearly, the running time of the Kuratowski generator strongly depends on the density of the initial graph. To possibly speed up the generation process we analyzed graph properties of graphs that were created from random graphs with edge size m equal to $m = n \log n$ and $m \in \theta(n^2)$. The outcomes of our tests were similar. Thus, graphs with $m = n \log n$ edges were generated as source for applying the Kuratowski algorithm. LEDA subroutine: `KURATOWSKI(graph, V, E, deg)`.
Intersection. This geometric (n)-generator is provided by the LEDA library. It is based on the intersection of line segments. The generator chooses n segments, constructs the corresponding arrangement and keeps the first n nodes. In the last step missing edges are inserted to make the graph connected. LEDA subroutine: `random_planar_graph( graph, xcoords, ycoords, n)`.

2.2 (n,m)-generators

Similar to the (n)-generators, (n,m)-generators rely either on combinatorial or on geometric approaches. Combinatorial generators triangulate the graph while adding nodes to it. Geometric generators first place points at random in a uniform sized rectangle. Afterwards the point set is triangulated. Note that the combinatorial generators create maximal planar graphs whereas the geometric ones generate *planar triangulations*, i.e., each face is bounded by three edges, except possibly the exterior face. No (n,m)-generator is known that creates graphs uniformly at random.
Convex Hull Triangulation. The LEDA library [17] provides a geometric convex hull triangulation algorithm (CHT) that is based on a sweep line. First, the nodes $v \in V$ are sorted lexicographically, i.e., by x- and then by y-coordinate, such that $v_1, \ldots, v_n$ denotes the sorted order. The algorithm processes the vertices in this order. Nodes are inserted and connected to all previous nodes they can see. In this way, after step i, the current graph consists of a triangulation of the first i points. The next vertex is connected to a subset of the previously inserted points lying on the convex hull. LEDA subroutine: `random_planar_map( graph, xcoords, ycoords, int n, int m)`.

By construction, when node v_i is processed, node v_{i-1} is part of the current convex hull and can be seen by node v_i. Thus, an edge (v_i, v_{i-1}) is added and v_i becomes part of the new convex hull. Since edges are never removed, the sequence $v_1, \ldots, v_n$ is a Hamiltonian path in the output triangulation. Thus, CHT cannot generate Non-Hamiltonian triangulations, which are known to exist [16]. Hence, CHT is not complete.

Delaunay. The LEDA library [17] contains a Delaunay triangulation algorithm, which can be used to generate planar graphs. This generator is not complete as not every maximal planar graph is Delaunay realizable [9]. LEDA subroutine: `DELAUNAY_TRIANG( list L, GRAPH DT)`.
Expansion. The node Expansion algorithm was first presented by Bowen et al. [7], who also showed its completeness. Note that the graph generator is combinatorial but generates an embedding as well. In particular, incident edges $e_1, \ldots, e_k$ of each node are cyclically ordered. The algorithm starts with a K_4 graph. Nodes are inserted by expanding a randomly selected node u along two incident edges e_i and e_j, $1 \leq i, j \leq k$, $i \neq j$ where k is the number of edges incident with u. Node u is expanded into two new nodes u_1 and u_2. Additionally the edges e_i and e_j are doubled while updating their target node to u_1 and to u_2, respectively. Thus, a face with four nodes is created, which is then triangulated randomly. The edges e_i and e_j separate the incident edges of u into two subsets. These have to be assigned to u_1 and u_2; which is easy due to the ordering of the stored edges. Thus, a careful implementation yields linear running time.
Insertion. The Node Insertion algorithm is available in the LEDA library [17]. It starts out with a triangle and iteratively picks a face f uniformly at random, inserts a new vertex and connects it to all vertices of f. LEDA subroutine: `random_planar_map( graph, int n, int m)`.

Note that graphs generated by the Insertion generator always contain a node of degree at most 3. Hence, *k-regular graphs* with $k > 3$ cannot be generated. Therefore, Insertion is not complete.

2.3 Summary

Table 1 gives an overview of the origin, complexity, completeness and possible uniform generation. We do not report on uniform generation of subclasses of planar graphs. Hence, an incomplete generator implies non-uniform generation. Due to their complexity Markov and Kuratowski were not capable of generating graphs larger than 500 and 1000 nodes, respectively. Hence, these generators cannot be recommended for large scale planar graph generation. In contrast, the other generators perform quite well. An exception is Delaunay; it generated all requested graph sizes but due to its complexity it may not be capable of generating really large graphs. Note that the running time of LEDA's Intersection implementation is not specified but experiments indicate a running time of $O(n \log n)$. Additionally note that the completeness of Intersection and Kuratowski follows from their definitions and that uniform generation is quite unlikely, which is shown by our experiments.

3 Experiments

We already described the performance of the generators, which together with the theoretical background allows to decide whether a generator is capable of generating graphs of favored size. In this section we further classify the presented

Table 1. Summary of the studied generators. Asterisk marked results can be found in this work.

generator	complexity	complete	uniform
Fusy [13]	$O(n)$	yes	yes
Intersection [17]	$O(n \log n)$	yes	no*
Kuratowski [17]	$O(n^2 \log n)$	yes	no*
Markov [8]	$O(n^3)$	yes	yes

generator	complexity	complete	uniform
CHT [17]	$O(n+m)$	no*	no
Delaunay [17]	$O(n^2)$	no [9]	no
Expansion [7]	$O(n)$	yes	open
Insertion [17]	$O(n)$	no	no

planar graph generators by means of basic graph properties in local as well as globar scope. This classification is strengthened with respect to algorithmical behavior. Altogether, this result will help experimenters to reasonably compile data sets. But first, we report on the generation of our data-sets, which are then used in our experimental studies.

3.1 Dataset Generation

Usually, in experimental studies a broad applicability of newly developed algorithms should be confirmed. In our case it is not feasible to generate all planar graphs of a certain size n, especially when algorithms should be tested on larger instances. Obviously, a representative sample should be chosen. The (n)-generators sample graphs by their underlying behavior, which includes the number of generated edges. In contrast, the (n,m)-generators' number of edges can be set arbitrarily. This raises the difficult question how to set the number of edges in a representative way. A first idea is to set the number of nodes and edges the (n,m)-generators have to create to the number of nodes and edges created by an (n)-generator, which allows to compare them with each other.

As aforementioned, Fusy and Markov are expected to generate graphs uniformly at random but Markov was not capable of generating graphs of larger size. Thus, we generate a data set using the number of nodes and edges predefined by Fusy as input for the (n,m)-generators. This distribution is referred to as *Fusy distribution*. Each generator created 40 000 graphs whose number diverged by $\pm 5\%$ from 1 000 nodes as described in Section 2.1.

Giminez and Noy [14] found that the average degree of a randomly selected planar graph is asymptotically normal. A graph of node count n is expected to contain $\mu \approx 2.21n$ edges on average with a standard deviation of $\sigma \approx 0.65n$. Fusy and Markov exactly fit the theoretical distribution. A consequence of this distribution is that with increasing node count the average degree of a randomly generated planar graph gets closer to its expected value. In particular, the standard deviation of the average degree for 1 000 nodes is $\sigma_{1000} \approx 0.025$, which states with high probability only a small subset within the interval of possible edges $m \in [0, 3n-6]$ is generated. Thus a fourth data set was created by the (n,m)-generators whose average degree correspond to a normal distribution with mean $\mu = 2.21$ and standard deviation $\sigma = 0.65$. This dataset of 40 000 values was chosen to span a larger part of the interval $[0, 3n-6]$, which should give a better overview how (n,m)-generators behave on a spreaded distribution. This distribution is referred to as *Fixed Average Degree distribution*.

Note that all plots showing the (n)-generators base on a graph size of 500 for Intersection, Kuratowski and Markov (lowest common denominator). The smallest size of Fusy graphs is 1 000 so this value was taken. The reason for letting Fusy be the exception was that Kuratowski was able to generate only very few graphs of size 1 000.

3.2 Basic Graph Properties

In the following basic, local and global graph properties of the generated planar graphs will be analyzed. Thereby, a classification of the generators is done. This gives a first sketch which generators should be used for a representative data-set compilation.

Average Degree. One of the very basic questions answered recently by Giminez and Noy [14] was the expected average degree of a random planar graph. According to the theoretical results the distributions of the average degree of Fusy and Markov match the expected distribution, see Figure 1. Intersection and Kuratowski can each clearly be separated from the other generators. For comparison, the Fixed Average Degree distribution used by the (n,m)-generators is shown.

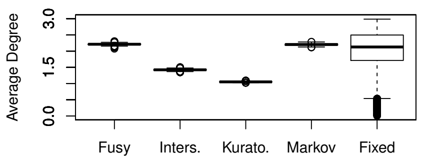


Fig. 1. Inherent distributions of the average degree of the (n)-generators and the Fixed Average Degree distribution.

Degree Sequence. The average degree describes a graph only roughly. A better way to reflect the structure of a graph is its degree sequence. Hence, in Figure 2 the degree sequence distributions of graphs generated by the (n,m)-generators using the Fusy distribution are shown. Note that nodes with a degree larger than 10 have been cut off. All graphs except Fusy consist of more than one connected component. Thus, Fusy contains no nodes with degree zero. The plot shows that CHT, Expansion and Fusy behave very similar and cannot be separated from each other. Insertion has more nodes of degree 2 and the peak at degree 3. This slope then decreases rapidly. The Delaunay generator has a larger mean. Its number of nodes having a degree less than 4 is much smaller compared to the other generators. The peak is at degree 4. More than 50% of the nodes have either degree 4 or 5. The slope then decreases very quickly. Although having similar trends the absolute values differ quite much. The entropy allows for a clear separation of Delaunay and Insertion from the other generators. The entropies of the (n)-generators clearly separate Intersection and Kuratowski from Fusy and Markov, which behave similarly, see Figures 13 and 14 in the appendix for details.

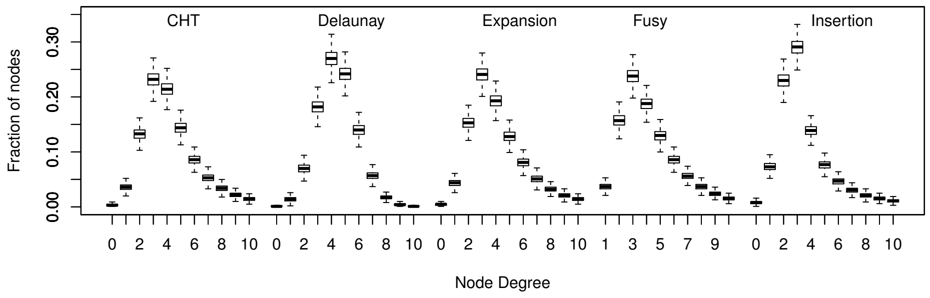


Fig. 2. The distribution of the degree sequence of graphs generated by the (n,m)-generators using the Fusy distribution. For a better overview, outliers have been removed and the degree sequence has been cut off at 10 nodes.

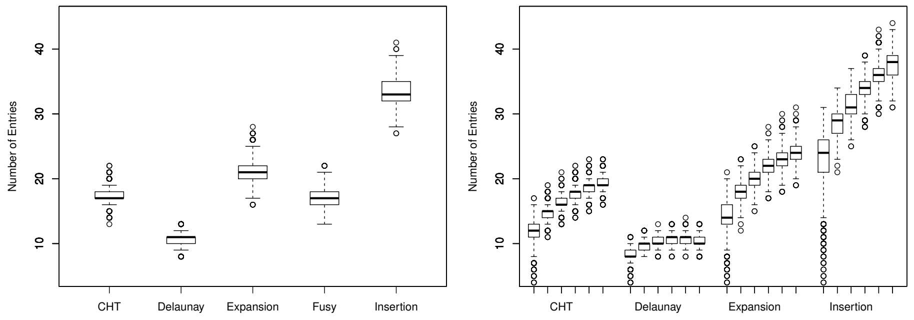


Fig. 3. The number of values of the degree sequence is shown. Both sides show graphs generated with (n,m)-generators. On the left side graphs underlying the Fusy distribution are shown. The graphs on the right side are generated using the Fixed Average Degree distribution, which have been split into six groups of increasing edge count for each generator.

We further study the degree sequence distribution by the number of entries that are necessary to describe a graph. Our assumption is that it correlates with the graphs' structural complexity. Note that we do not expect this to correlate with algorithmic complexity. In Figure 3 the number of degree sequence entries of the (n,m)-generators using the Fusy distribution as well as the Fixed Average Degree distribution are shown. Again, Delaunay and Insertion can be separated clearly from the other generators, which have a similar distribution of the degree seqeunce entries. The Delaunay generator has nearly a constant number of entries for all graphs and thus exhibits a regular structure. In contrast the Insertion generated graphs show the largest variance.

Diameter. So far we analyzed elementary properties that were easy to obtain. By this, we got an idea which generators could complement each other. This classification will now be extended by a global property, namely, the *diameter* of a graph. The diameter of a graph is the length of the longest shortest path between all pairs of nodes in a graph. Thus, their values describes how com-

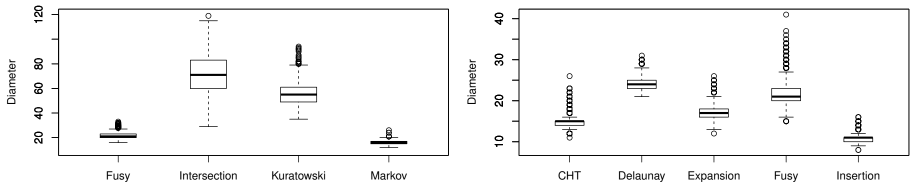


Fig. 4. The diameters of the (n)-generators (left) and of the (n,m)-generators using the Fusy distribution (right).

pact a graph is. Figure 4 shows the diameters of graphs generated by the (n)- and (n,m)-generators using the Fusy distribution. The diameters of Intersection and Kuratowski are very large compared to the other generators. The graphs of both generators are connected and simultaneously have very low average degrees. Thus, the structure is more tree-like, which results in high diameters. The (n,m)-generators show some differences. For example, Insertion has a very small diameter whereas Delaunay has the largest mean diameter. Additionally, Insertion and CHT have small spreads. This difference can be explained by their generation process. Delaunay generates a comb-like regular structure whereas Insertion and CHT insert shortcuts, i.e., long edges, which reduce both the diameter and its spread. Altogether, the studied diameter of graphs confirms the basic classification on a global scale.

Clustering Coefficient. We complete our basic classification of the planar graph generators by the analysis of a local property. Namely, we measure the density of each vertex' neighborhood, in particular we compute the *clustering coefficient* (CC) of a graph [20]. The *neighborhood* of a vertex v is defined as $N(v) = \{u : \{u, v\} \in E\}$. A complete subgraph of three nodes is called a *triangle*. The number of triangles of a node v is defined as $\delta(v) = |\{\{u, w\} \in E : \{v, u\} \in E$ and $\{v, w\} \in E\}|$. A *triple* τ at a node v is a path of length two for which v is the center node. Let $d(v)$ denote the degree of node v. Then the number of triples of a node v is $\tau(v) = \binom{d(v)}{2}$. For nodes v with degree $d(v) \geq 2$ the CC is defined as $c(v) = \delta(v)/\tau(v)$. The CC of the graph is the average over all nodes which is defined as $CC(G) = 1/\hat{n} \sum_{v \in \hat{V}} c(v)$, where $\hat{V}$ is the set of nodes v with $d(v) \geq 2$. Figure 5 shows the clustering coefficients of the (n)-generators. Clearly, Kuratowski and Intersection can be separated from the other generators. This can be explained with the average degree being very small for Kuratowski. The graph is tree like and thus its CC is very small. The Intersection graphs have somewhat larger average degree that allow for a more complex graph than a tree. Hence, it exhibits a slightly larger interconnection. In the case of the (n,m)-generators three groups can be separated. The first consists of the Delaunay generated graphs. Due to the generation process the CC is less than of the other groups. The second consists of CHT, Expansion and Fusy. Minor differences within this group can be seen but they cannot be separated clearly. The Insertion-generated graphs represent the third group. Their generation process leads to a high interconnection of the neighbors, which results in a very large CC.

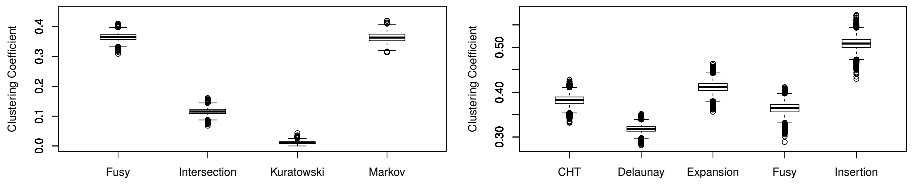


Fig. 5. Clustering coefficients of the (n)-generators (left) and the (n,m)-generators using the Fusy distribution (right).

3.3 Algorithmical Behavior

So far, basic, local and global properties of the generated graphs have been analyzed. This analysis would allow for a classification of the generators with respect to a complementary compilation. We now want to strengthen our proposition by a study of advanced algorithmical behavior, in particular by applying FPT algorithms [19].

K-Core Decomposition. First, we study topological properties of the generated graphs. To this end we use the *k-core decomposition* [21]. This method destructively simplifies the graph. Iteratively, nodes are removed from the graph by increasing degree k. This procedure is applied until all nodes have been pruned off the graph. The k-core of graph is a maximal subgraph in which each vertex has at least degree k. A node has coreness k if it is part of the k-core but not of the $k+1$-core. The k-core decomposition cannot be very large for planar graphs. A k-core implies the graph to have a k-regular subgraph. Because every planar graph contains a vertex with degree at most 5, no 6-regular subgraph can occur in a planar graph. Although 5-regular graphs exist [10] their random generation seems to be unlikely as none of our generated graphs contains a 5-core. In Figure 6 the coreness distribution of graphs generated by the (n)- and (n,m)-generators using the Fusy distribution are shown. Note that outliers have been removed to give a better overview. Evidently, connected graphs do not have nodes with coreness zero. The Intersection and Kuratowski graphs only consist of nodes with coreness one and two. Again, the group consisting of CHT, Expansion, Fusy and Markov can hardly be distinguished. Due to its generation process Insertion does not have a 4-core. Delaunay graphs are dominated by the amount of nodes with coreness three. Thus, Delaunay and Insertion can be clearly separated from the other generators.

K-Vertex Cover. We now take a first glance at an FPT problem. A problem is FPT if it admits an algorithm with running time $O(f(k)n^{O(1)})$, where f is an arbitrary function depending only on k. The problem we examine is *k-vertex cover* (k-VC). A *vertex cover* (VC) of a graph G is a subset $C \in V(G)$ such that $C \cap \{u,v\} \neq \emptyset$, $\forall \{u,v\} \in E(G)$. The *k-VC* problem asks whether a VC of size $|C| \leq k$ exists. To solve k-VC a *kernelization* algorithm is applied in a preprocessing step, which reduces the initial instance to solve in polynomial time to its problem *kernel.* The topological information we gained can be very useful. For instance, it can be used to estimate the size of the kernel when k-VC

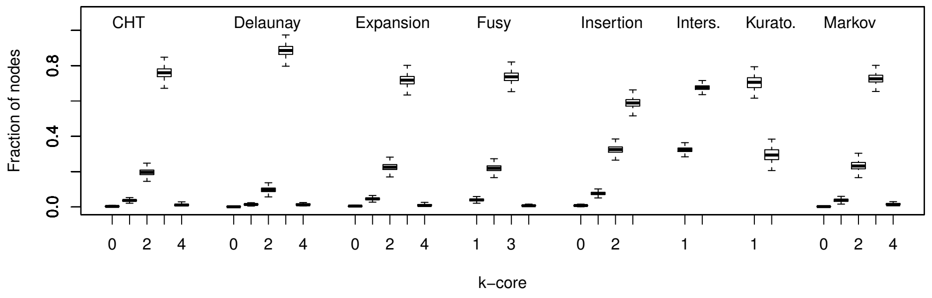


Fig. 6. Coreness distribution of (n)- and (n,m)-generators using the Fusy distribution, outliers have been removed.

kernelization is applied to graphs originating from a certain generator, where others performed extensive experimental studies [2, 6]. Two kernelization rules exist how to handle degree one and two nodes. Thus, the problem reduction roughly equals the sum of the number of the nodes having coreness 0, 1 or 2. Recalling the coreness distribution outcomes in Figure 6 we see that Intersection and Kuratowski can be solved optimally. The best reduction of the remaining generators achieves Insertion. In contrast to that, Delaunay has the smallest reduction of all generators. The remaining generators behave very similarly.

Planar k-Dominating Set. So far we used topological information, in particular the coreness distribution, to estimate algorithmical complexity. Now, we confirm different algorithmical behavior on the generated graphs solving *planar k-dominating set* (PDS), which is also FPT for planar graphs. A *dominating set* (DS) is a set $D \subseteq V(G)$ of vertices such that each vertex in $V(G) \setminus D$ has a neighbor in D. The PDS problem asks whether a DS of size $|D| \leq k$ exists. Similar to k-vertex cover, first a problem kernel is computed. However, the problem reduction of PDS does not rely on the degree of the nodes but the neighborhood of an examined node. The algorithms described in [4] and [3] have been implemented. The reduction is based on two kernelization and seven search tree rules. Alber et al. report impressive kernelization results [1] on several real world instances as well as artificially generated graphs. The authors were mainly interested in showing the algorithmical performance on real world problems but the data set also contained planar graphs that seem to not be selected representatively. Figure 7 presents the size of the PDS kernel. The large reduction reported was only partly achieved. CHT, Delaunay and Expansion show a large variance in the reduction whereas Insertion can be reduced in most times by an amount of about 85%. Thus, the selection of the graph generator has a high impact on the results achieved by this algorithm and a wrong selection would greatly bias the results gained [1].

Treewidth. So far, the topological information and one FPT algorithm confirm our classification. But we aim at establishing a broad base for the classification. We achieve this by studying the *treewidth* of a graph, the number of nodes that are mapped to the tree nodes of a tree decomposition of the graph. It is NP-

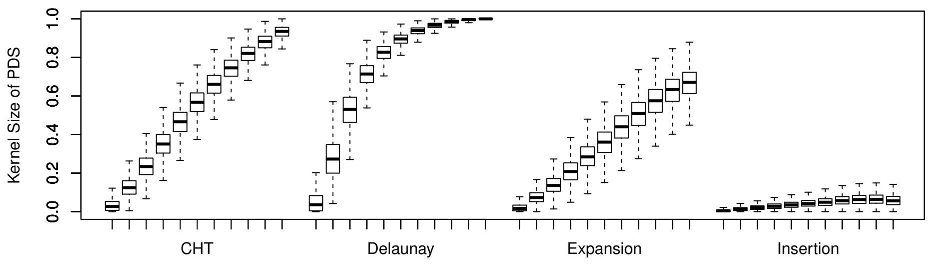


Fig. 7. Size of the PDS kernel of (n,m)-generators using the Fixed Average Degree distribution. The total number of generated graphs has been divided into eleven groups of increasing edge count, outliers have been removed.

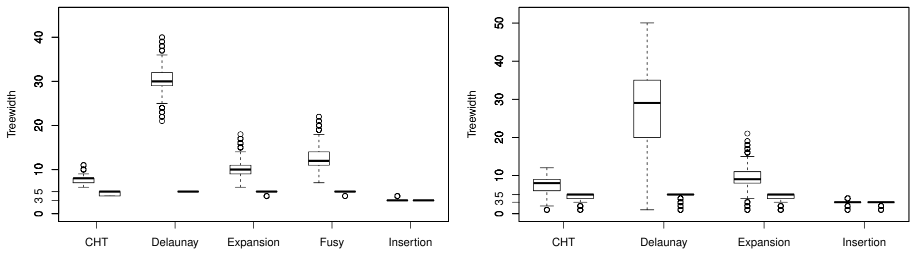


Fig. 8. Upper- and lower bounds of the treedwidth heuristics computed by LibTW [22] of the largest connected component of (n,m)-generated graphs according to the Fusy- (left) and the Fixed Average Degree distribution (right). For each generator the left boxplot shows the upper bound and the right boxplot the lower bound.

hard to compute the treewidth of a graph. Nevertheless, if the treewidth of a graph class is bounded, several NP-hard combinatorial problems can be solved in polynomial time. A k-tree is defined recursively as follows. The complete graph K_k on k vertices is a k-tree. Given a k-tree G on $n \geq k$ vertices, a k-tree on $n + 1$ vertices is obtained by adding a new vertex u and edges connecting u to every vertex of a K_k subgraph in G. A graph is a partial k-tree if it is a subgraph of some k-tree. Partial k-trees are exactly the graphs with tree width up to k. This is exactly the way graphs are generated by the Insertion generator. Hence, Insertion generated graphs have treewidth at most 3 and are often not appropriate instances when dealing with NP-hard problems. Figure 8 shows the outcomes of LibTW [22] upper- and lower-bound heuristics, which were applied to graphs generated by the (n,m)-generators using the Fusy- and Fixed Average Degree distributions. The heuristics confirm the bounded treewidth of Insertion generated graphs. The treewidth of CHT is larger than Insertion but is rather small compared to the other generators. The treewidth of Expansion partially equals the treewidth of Fusy. Delaunay generated graphs seem to be very hard instances since the span of upper- and lower bound is largest.

4 Conclusions

This work is motivated by the ambigous description and inappropriate data-set selection found in many experimental works. To our knowledge no work exists that systematically analyzes planar graph generators, particularly with regard to assembling a complementary compilation of planar graph generators that allow for a meaningful interpretation of experimental work. This study allows for such a classification of a selection of planar graph generators, namely four (n)-generators (Fusy, Markov, Intersection, Kuratowski) and four (n,m)-generators (CHT, Delaunay, Expansion, Insertion). Fusy and Markov are capable of generating graphs uniformly at random. However, with growing size the expected average degree tends to a fixed value of 2.21, which by no means represents the possible interval $[0, \ldots, (3n-6)/n]$. The (n,m)-generators allow for an arbitrary average degree. Hence, we recommend a distribution of the average degree that spans this interval much better.

We empirically studied eight selected planar graph generators by means of running time, basic graph properties and algorithmic behavior. In our implementations we rely on the LEDA library, which provides several planar graph generators. Unfortunately, these are confusably named, which we cleaned up. It turned out that Kuratowski and Markov are not efficient enough to run large scale tests. The efficient generator Fusy is capable of drawing graphs uniformly at random but cannot be used as an out of the box generator. By a detailed analysis of basic graph properties most of the graph generators can be classified into groups. Thus, Delaunay and Insertion can clearly be distinguished from each other and from the group consisting of CHT, Expansion and Fusy. The latter group shows small differences at various tests but none allows for a clear separation. These differences and groupings can also be observed when FPT kernelization algorithms are applied.

As a basic principle, experimental works should precisely describe the origin of the used data sets, which for generated graphs includes their distribution of the average degree. Experimenters should keep in mind which structural properties their algorithms exploit and ensure that the used data sets exhibts well distributed and representative structural properties to allow for significant empirical results. Because of the manifold properties of graphs that may be of interest, it is hard to present a general recommendation which of the studied generators to use. Nevertheless, theoreticians verifying practicability in a small experiment should rely on a uniform generator or use Expansion with a spreaded distribution of the average degree. For detailed experimental works, experimenters should compile data sets that at least consist of Expansion, Delaunay and Insertion generated graphs. Expansion overlaps CHT and Fusy to a certain extent. Nevertheless, both CHT and Fusy complement the data set.

Future work. Often in experimental algorithmics, graphs with a predefined number of nodes and edges are of interest, which Fusy cannot create due to its restrictions. Thus, future work might be a planar (n,m)-generator capable of generating planar graphs uniformly at random. Expansion could be a good starting point as it is complete and compared to Fusy exhibits a similar behavior.

References

1. Jochen Alber, Nadja Betzler, and Rolf Niedermeier. Experiments on data reduction for optimal domination in networks. *Annals of Operations Research*, 146(1):105–117, 2006.
2. Jochen Alber, Frederic Dorn, and Rolf Niedermeier. Experimental evaluation of a tree decomposition-based algorithm for vertex cover on planar graphs. *Discrete Applied Mathematics*, 145(2):219–231, 2005.
3. Jochen Alber, Hongbing Fan, Michael R. Fellows, Henning Fernau, Rolf Niedermeier, Frances Rosamond, and Ulrike Stege. A refined search tree technique for dominating set on planar graphs. *Journal of Computer and System Sciences*, 71(4):385–405, 2005.
4. Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Efficient data reduction for dominating set: a linear problem kernel for the planar case. In *Proceedings of the 8th Scandinavian Workshop on Algorithm Theory (SWAT'2002)*, number 2368 in Lecture Notes in Computer Science, pages 150–159. Springer, July 2002.
5. Manuel Bodirsky, Clemens Gröpl, and Mihyun Kang. Generating labeled planar graphs uniformly at random. *Theor. Comput. Sci.*, 379(3):377–386, 2007.
6. Endre Boros, Peter L. Hammer, and Gabriel Tavares. Preprocessing of unconstrained quadratic binary optimization. Technical report, Rutgers Center for Operations Research (RUTCOR), 2006.
7. Rufus Bowen and Stephen Fisk. Generations of triangulations of the sphere. *Mathematics of Computation*, 21(98):250–252, 1967.
8. Alain Denise, Marcio Vasconcellos, and Dominic J. A. Welsh. The random planar graph. *Congressus Numerantium*, 113:61–79, 1996.
9. Michael B. Dillencourt. Realizability of delaunay triangulations. *Inf. Process. Lett.*, 33:283–287, February 1990.
10. Guoli Ding, Jinko Kanno, and Jianning Su. Generating 5-regular planar graphs. *J. Graph Theory*, 61:219–240, July 2009.
11. Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann samplers for the random generation of combinatorial structures. *Combinatorics, Probability and Computing*, 13(4-5):577–625, 2004.
12. Éric Fusy. Implementation of a boltzman sampler for planar graphs. http://www.lix.polytechnique.fr/∼fusy/ Programs/BoltzmannPlanarGraphs.tar.gz, 2005.
13. Éric Fusy. Uniform random sampling of planar graphs in linear time. *Random Structures & Algorithms*, 35(4):464–522, 2009.
14. Omar Giminéz and Marc Noy. Asymptotic enumeration and limit laws of planar graphs. *Journal of the American Mathematical Society*, 2008.
15. Gunnar Brinkmann and Brendan McKay. Fast generation of planar graphs. http://cs.anu.edu.au/∼bdm/ papers/plantri-full.pdf.
16. Guido Helden. *Hamiltonicity of maximal planar graphs and planar triangulations*. Ph.d. thesis, RWTH Aachen, 2007.
17. Kurt Mehlhorn and Stefan Näher. *The LEDA Platform of Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
18. Sascha Meinert and Dorothea Wagner. An Experimental Study on Generating Planar Graphs. Technical report, ITI Wagner, Faculty of Informatics, Karlsruhe Institute of Technology, 2011. Karlsruhe Reports in Informatics 2011-13.
19. Rolf Niedermeier. *Invitation to Fixed Parameter Algorithms*. Oxford University Press, Oxford, 2006.

20. Thomas Schank and Dorothea Wagner. Approximating Clustering Coefficient and Transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.
21. Stephen B. Seidman. Network Structure and Minimum Degree. *Social Networks*, 5:269–287, 1983.
22. Thomas van Dijk and Jan-Pieter van den Heuvel and Wouter Slob. Computing treewidth with LibTW. http://www.treewidth.com/docs/LibTW.pdf, 2006.

A Appendix

For the sake of completeness we now present some additional results of our investigations. They either confirm our findings but deliver no new insights or are not central to our experimental evaluation, while potentially interesting to the reader.

In some experiments the tested property consists of more than one element, e.g., the degree distribution. To aggregate the data into a single value the *Shannon entropy* known from information theory is used. This entropy is defined as $H = -\sum_{i=1}^{k} \hat{p}_i \log_b(\hat{p}_i)$ where $\hat{p}_i = n_i/N$ with n_i the number of occurrences of outcome i, N the total number of observations, k the number of distinct outcomes and b the base of the logarithm. This estimation of the probabilities of the outcomes is known as *maximum-likelihood* estimation. Depending on the base of the logarithm two different informations are obtained. The *bit entropy* is computed using the base $b = 2$ which determines how many bits per entry on average are necessary to encode the information. The *normalized entropy* to the base $b = k$ informs on the amount of redundancy within the information.

Connected Components. In theory a planar graph created uniformly at random is connected with high probability [14]. The graphs produced by the (n)-generators are all connected, partly by construction. On the contrary, the (n,m)-generators are usually not connected. Here, the number of connected components depends on the number of edges. Nevertheless, the latter very often generate a graph which contains a single, large connected component if the average degree exceeds a certain value. In Figure 9 and Figure 10 the ratio of the largest connected component's size and the total size of the graphs generated by the (n,m)-generators according to the Fixed Average Degree distribution are shown. The data points are sorted increasingly from left to right according to the average degree of the represented graphs. With increasing average degree of a graph, a giant component emerges, which consists of a large fraction of the graph. The giant component has a size of at least 90% at an average degree of 1.6 for CHT, 1.45 for Delaunay, 1.65 for Expansion and 1.8 for Insertion graphs. The size of this large connected component grows further with increasing average degree.

Face-Size Sequence. The *face size* is the number of edges being part of a face. In Figure 11 a histogram of the face-size sequence of Fusy and (n, m)-generated graphs according to the Fusy distribution is shown. Observe that the maximum face size of the generated graphs differ. However, only very few graphs contain a face of size larger than 75% of the maximum size. Thus, the maximum face size cannot be used to distinguish the generators.

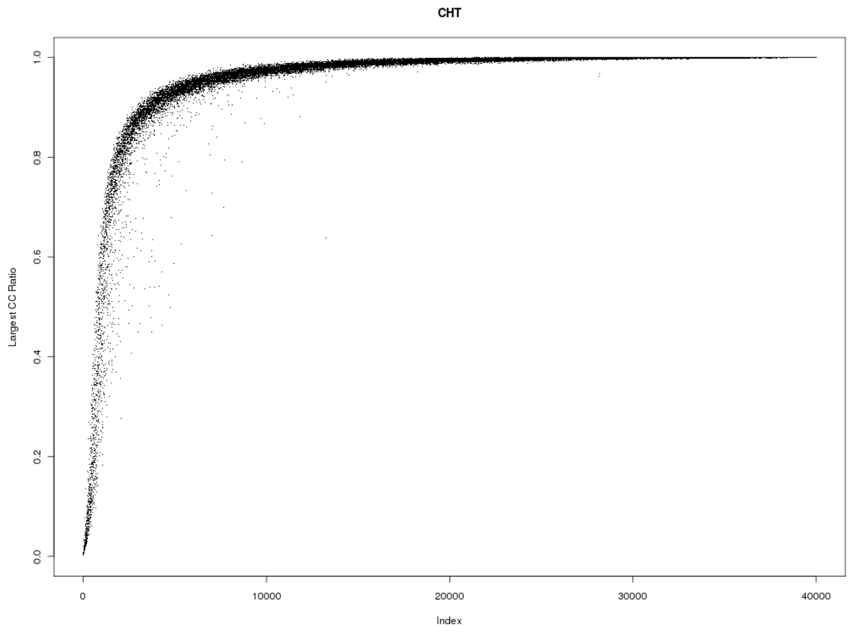
When looking at the slope of the histogram, we can see that more than 99.6% of the faces of the generated graphs have a size within the interval $[3, 10]$. The slope has its peak at size 3 and decreases fast with growing face size. The face-size sequence seems to be approximately equal among all generators as CHT, Delaunay, Expansion and Insertion can hardly be distinguished from each other. Only Fusy can be separated. It has less faces of size 3 and an increased number of faces of size 4 and 5, compared to the (n,m)-generators.

Figure 12 shows boxplots of the face-size sequence distribution of the generated graphs. The variance among all generators is very low, except for Fusy, whose face-size sequence distribution differs slightly from the others.

Degree Sequence. Although the degree sequence distributions in Figure 2 have similar trends, the absolute values differ quite much. This can be seen best when looking at the entropy of the degree sequence. Now Delaunay and Insertion can be separated clearly from the others, see Figure 13. The normalized entropy separates Insertion from the other generators, whereas the bit entropy of Delaunay is much lower than that the others.

For comparison the entropies of the (n)-generators are presented in Figure 14. The bit entropy of Fusy and Markov are very similar, whereas the normalized entropy shows some differences between them. Similar to the average degree distribution, Intersection and Kuratowski can clearly be separated from each other as well as from Fusy and Markov.

Coreness. Figure 6 shows the coreness of graphs that are generated by (n)- and (n,m)-generators using the Fusy distribution. For completion we also report on the coreness entropy in Figure 15. Again, Insertion as well as Delaunay can be clearly separated from each other and from the remaining graphs.



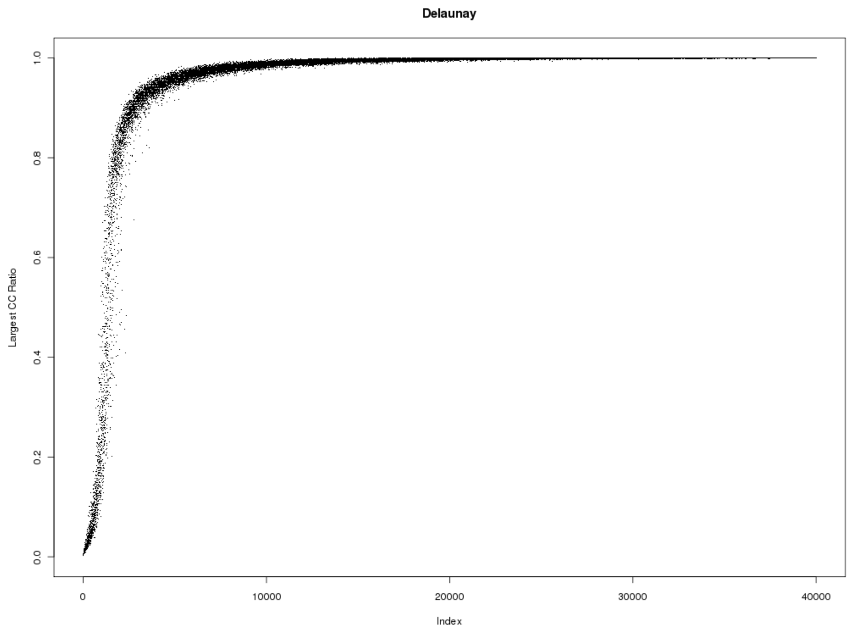
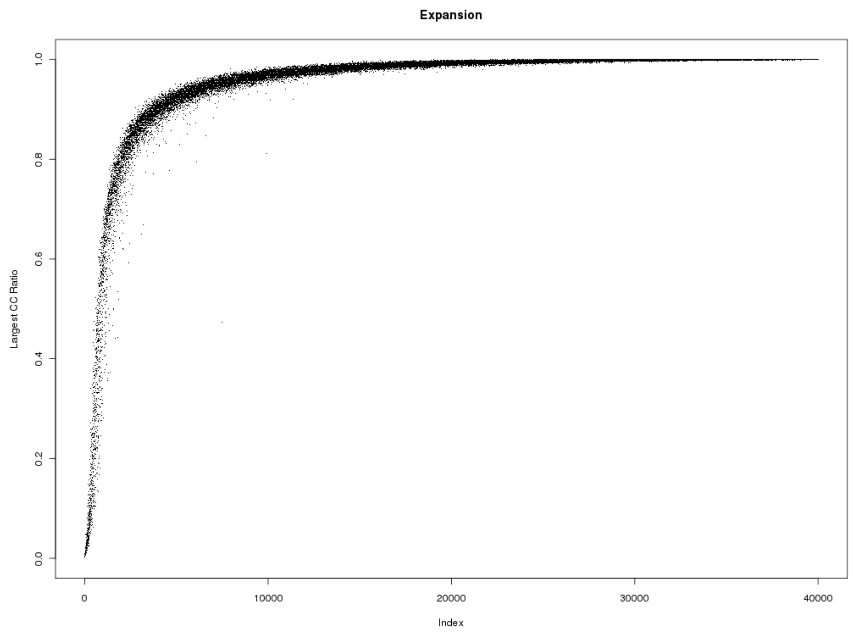


Fig. 9. The size of the largest connected component in relation to the total size of the graph. Shown are CHT and Delaunay generated graphs according to the Fixed Average Degree distribution. The data points are sorted increasingly from left to right according to the average degree of the represented graphs.



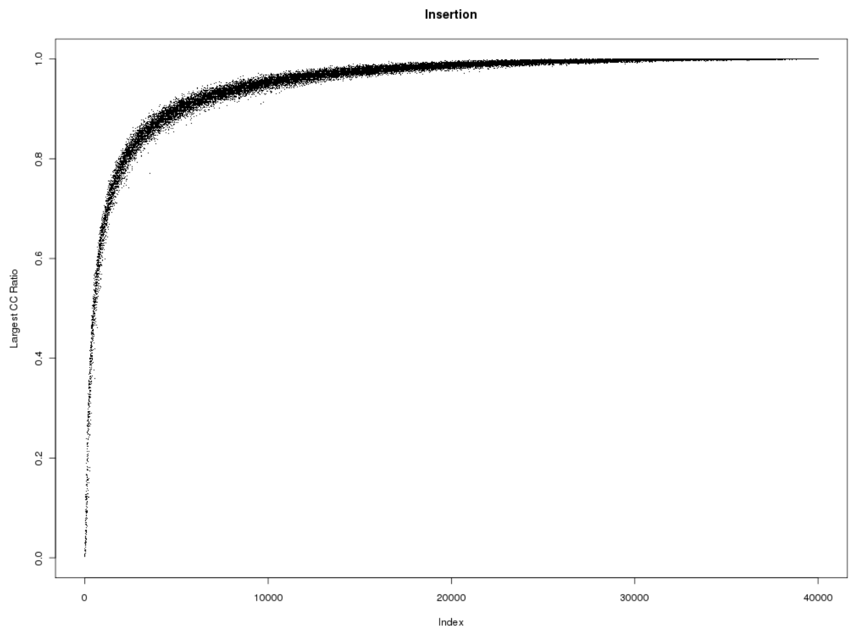


Fig. 10. The size of the largest connected component in relation to the total size of the graph. Shown are Expansion and Insertion generated graphs according to the Fixed Average Degree distribution. The data points are sorted increasingly from left to right according to the average degree of the represented graphs.

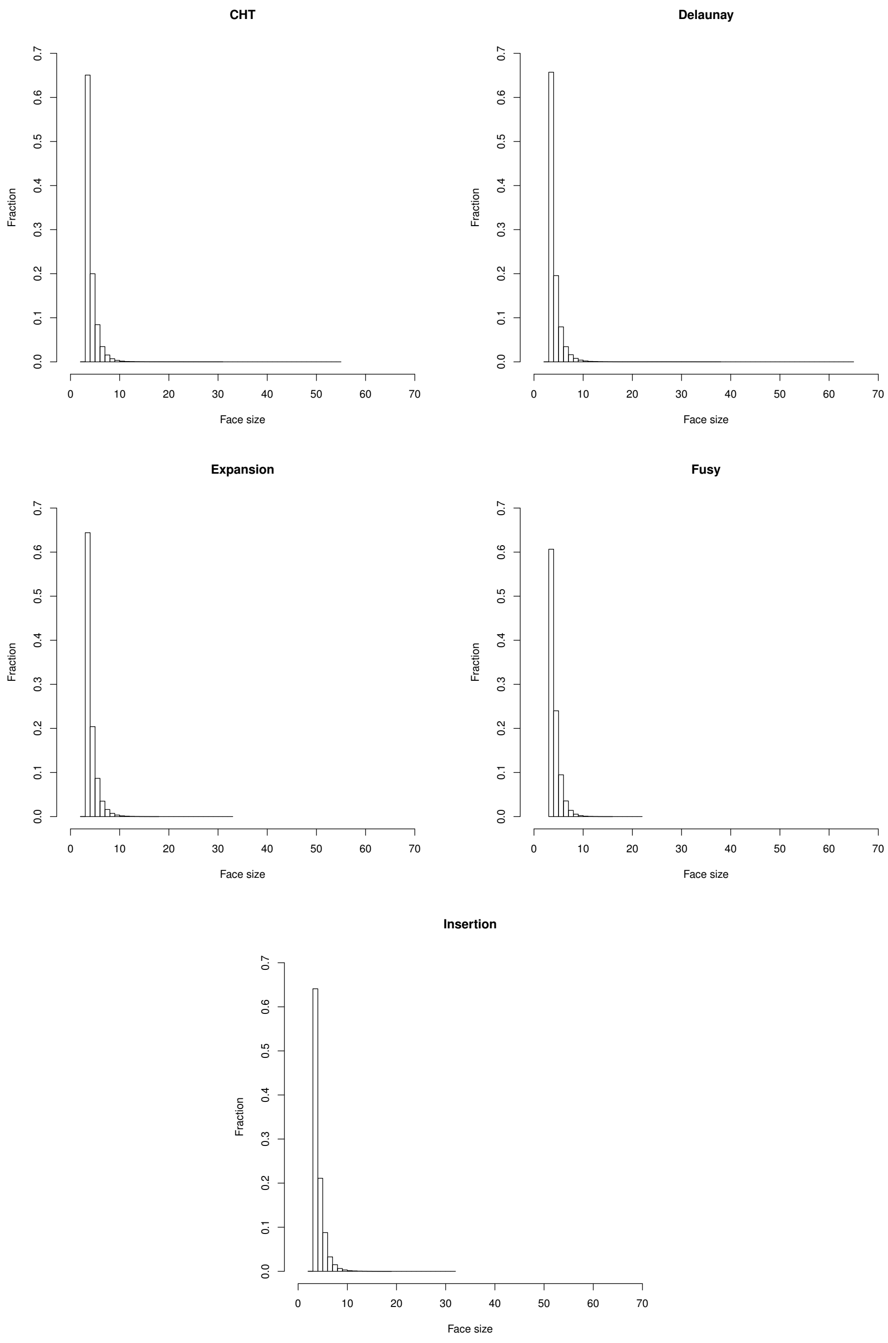


Fig. 11. The face size distribution of Fusy and (n, m)-generated graphs according to the Fusy distribution. The bars sum up to 1.

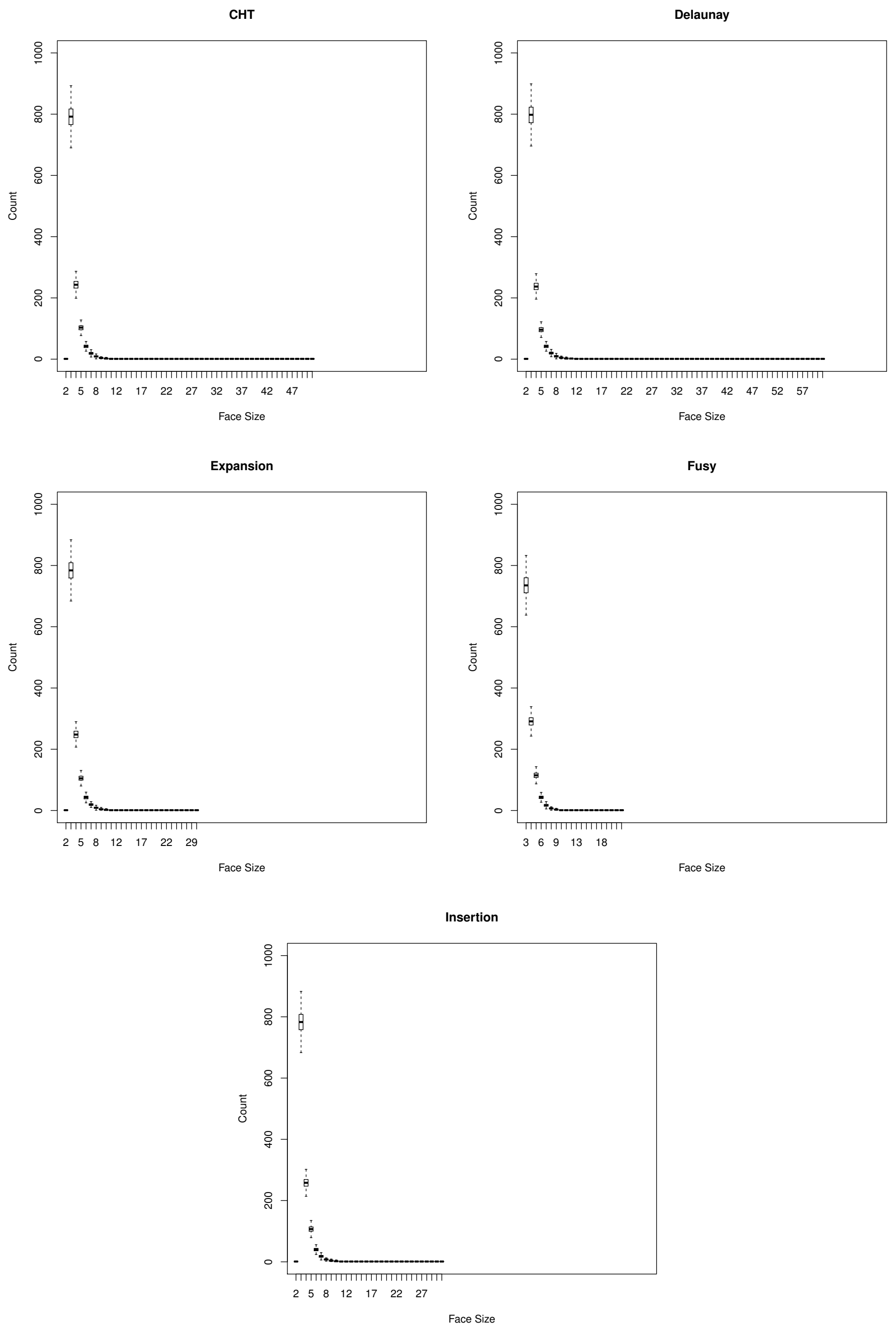


Fig. 12. Boxplots of the face size distribution of Fusy and (n, m)-generated graphs according to the Fusy distribution. For a better overview, outliers have been removed.

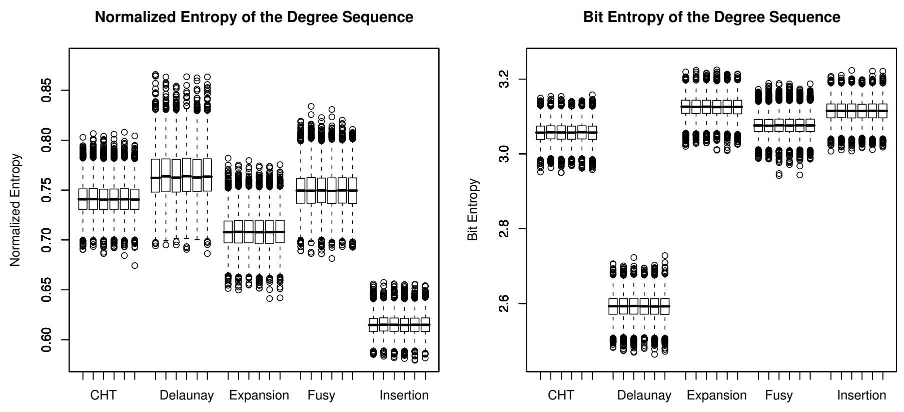


Fig. 13. The entropy of the degree sequence of graphs is shown which are generated by the (n,m)-generators using the Fusy distribution. On the left: the normalized entropy clearly separates Insertion from the other generators. On the right: the bit entropy of Delaunay is less than that of the others.

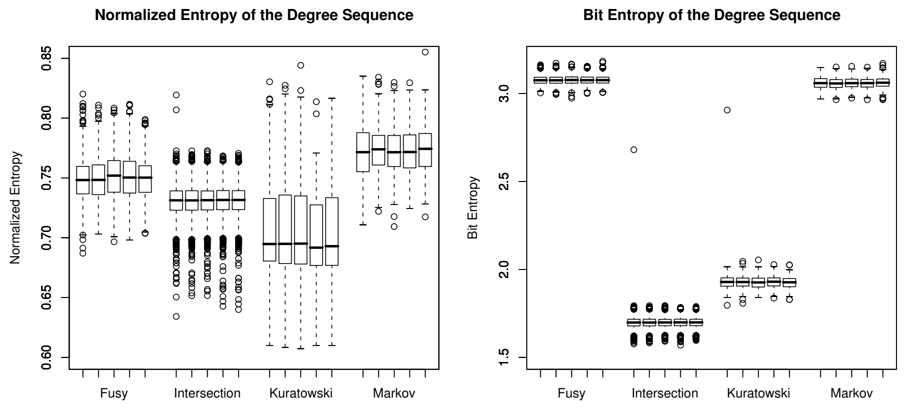
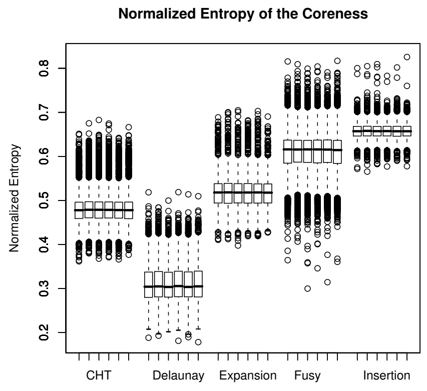


Fig. 14. Shown is the entropy of the degree sequence of graphs computed by the (n)-generators. On the right side, the bit entropy separates Intersection and Kuratowski from each other and from the group consisting of Fusy and Markov. The normalized entropy on the left shows a slightly less value for Fusy than for Markov.



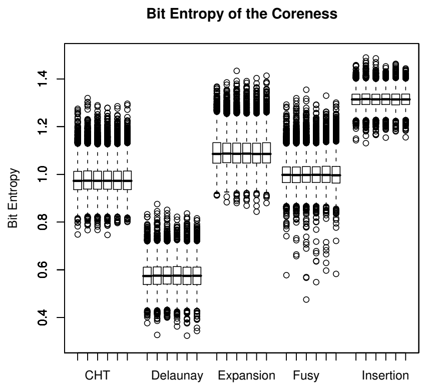


Fig. 15. The coreness entropies of the (n,m)-generators generated using the Fusy distribution are shown.