Generating Graphs with Predefined k-Core Structure*

Michael Baur, Marco Gaertler, Robert Görke, Marcus Krug, and Dorothea Wagner

Faculty of Informatics, Universität Karlsruhe (TH),
{baur,gaertler,goerke,krug,wagner}@informatik.uni-karlsruhe.de

Summary. The modeling of realistic networks is of great importance for modern complex systems research. Previous procedures typically model the natural growth of networks by means of iteratively adding nodes, geometric positioning information, a definition of link connectivity based on the preference for nearest neighbors or already highly connected nodes, or combine several of these approaches.

Our novel model is based on the well-know concept of k-cores, originally introduced in social network analysis. Recent studies exposed the significant k-core structure of several real world systems, e.g. the AS network of the Internet. We present a simple and efficient method for generating networks which strictly adhere to the characteristics of a given k-core structure, called core fingerprint. We showcase our algorithm in a comparative evaluation with two well-known AS network generators.

1 Introduction

The interest in modeling classes of graphs has significantly increased by recent studies of complex systems such as the Internet, biological networks, river basins, or social networks. While random graphs have been studied for a long time, the standard models appear to be inappropriate because they do not share certain abstract characteristics observed for those systems. One of these characteristics is the k-core structure which can be interpreted as a nested decomposition separating parts of the network based on their density. This decomposition is commonly applied in order to identify central parts of the networks since it peels the network layer by layer, filtering out less important parts that are sparsely connected with the remaining graph. Example applications are network fingerprinting with LaNet-vi [1], protein network analyses [18] or the exploration of modern social networks [6].

* This work was partially supported by the DFG under grant WA 654/14-3 and EU under grant DELIS (contract no. 001907).

A crucial field of application of graph generators is the simulated evolution of a given network, granting insights in both its past development and its anticipated future behavior. One prominent example is the Internet at the Autonomous System level where various models have emerged over the last few years, including BRITE [12], Inet [9], nem [10], and various models presented by Pastor-Satorras and Vespignani [14]. While this network has been observed to possess a very distinct k-core structure, kept track of over a long period of time, all generating tools so far ignore this structure, and thus largely fail to do justice to this significant property. Overall, up to our knowledge an approach to create networks with a given k-core structure is missing so far.

To address this issue we refine the abstract measurement of core sizes to a *core fingerprint* that additionally includes information on the interconnectivity of each pair of shells. This allows us to design a simple and efficient method to incrementally generate randomized networks with a predefined k-core structure, starting with the maximum core. By utilizing two results on edge rewiring we thus achieve a structure that precisely matches the core fingerprint.

This paper is organized as follows: first we clarify the preliminaries and state some basic properties for k-core structures in Section 2, then we give the description of the network generator in Section 3. In Section 4 we evaluate our model in comparison to two well-known generators with respect to commonly used network properties. Finally we give some concluding remarks.

2 Preliminaries

Let $G = (V, E)$ be a simple, undirected graph. A subset $V' \subseteq V$ of the node set induces a subgraph $G[V'] := (V', E')$ with $E' := \{\{u, v\} \mid u, v \in V', \{u, v\} \in E\}$. The degree $\deg(v)$ of the node v is the number of incident edges. A nested decomposition of G is a finite sequence $(V_0, \ldots, V_k)$ of subsets of nodes such that $V_0 = V$, $V_{i+1} \subseteq V_i$ for $i < k$, and $V_k \neq \emptyset$.

Cores are a widely used realization of nested decompositions. The concept was originally introduced by Seidman [17] and generalized by Batagelj and Zaversnik [3]. Constructively speaking, the *i-core* of an undirected graph is defined as the unique subgraph obtained by iteratively removing all nodes of degree less than i. This procedural definition immediately gives rise to a construction algorithm that can easily be implemented. Moreover, it is equivalent to the closed definition of the i-core as the set of all nodes with at least i adjacencies to other nodes in the i-core. The *core number* of a graph is the smallest i such that the $(i + 1)$-core is empty, and the corresponding i-core is called the *core* of a graph. Figure 1 depicts the core decomposition of an example graph with a core number of 4. The core decomposition can be computed in linear time with respect to the graph size [2].

A node has *coreness* i, if it belongs to the i-core but not to the $(i+1)$-core. We call the collection of all nodes having coreness i the *i-shell*. An edge $\{u, v\}$

is an *intra-shell edge* if both u and v have the same coreness, otherwise it is an *inter-shell edge*.

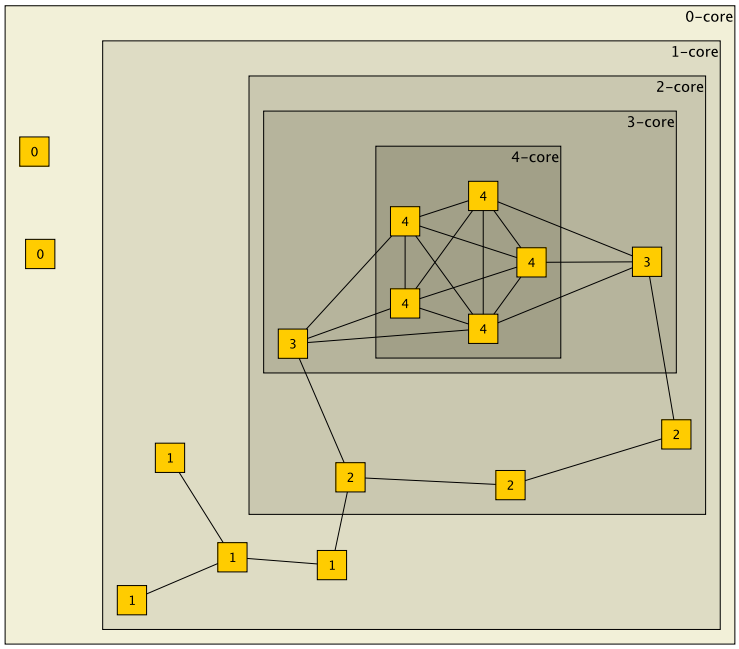


Fig. 1. A k-core decomposition with 5 core shells.

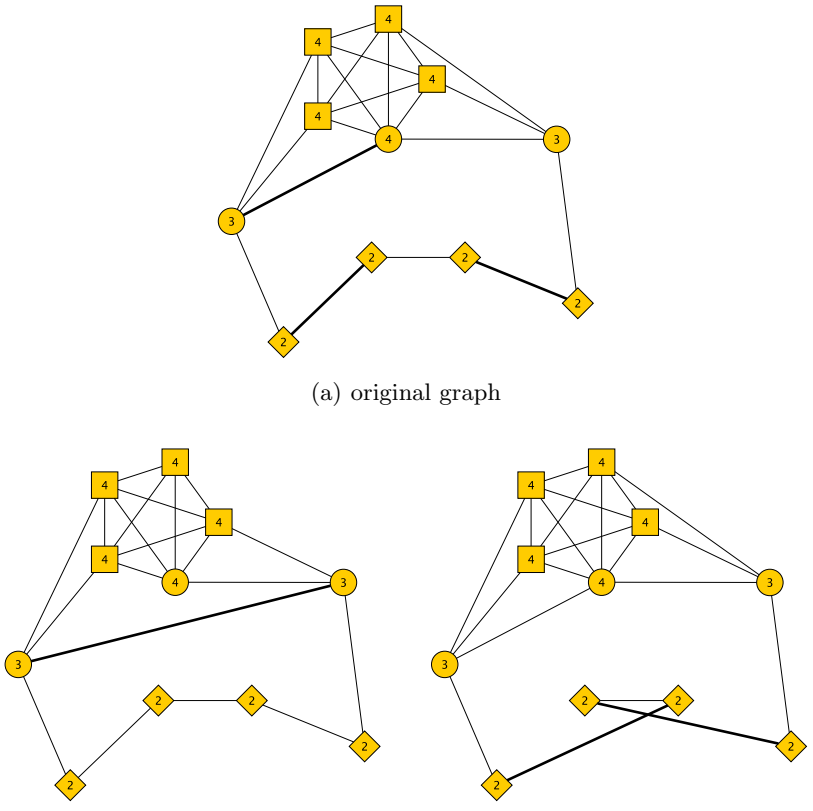
Informally speaking, the coreness of a node can be viewed as a robust version of the degree, i. e., a node of coreness i retains its coreness even after the removal of an arbitrary number of nodes of smaller coreness.

Next, we summarize two simple facts about the relation of intra- and inter-shell edges.

Lemma 1 (Rewiring). *Let $G = (V, E)$ be a graph. Let $u, v \in V$ be two non-adjacent nodes with the same coreness and $\{u, w\}, \{v, w'\} \in E$ two edges such that* $\text{coreness}(u) \leq \min\{\text{coreness}(w), \text{coreness}(w')\}$. *Then $G' := (V, E')$ with $E' := E \setminus \{\{u, w\}, \{v, w'\}\} \cup \{u, v\}$ has the same core decomposition as G. Conversely, let $u, v \in V$ be two adjacent nodes with the same coreness and $w, w' \in V$ such that* $\text{coreness}(u) \leq \min\{\text{coreness}(w), \text{coreness}(w')\}$ *and $\{u, w\}, \{v, w'\} \notin E$. Then $G'' := (V, E'')$ with $E'' := E \setminus \{u, v\} \cup \{\{u, w\}, \{v, w'\}\}$ has the same core decomposition as G.*

Lemma 2 (Swapping). *Let $G = (V, E)$ be a graph, $u, v, w, w' \in V$ be four nodes all having the same coreness, $\{u, v\}, \{w, w'\} \in E$ be two intra-shell edges, and $\{u, w\}, \{v, w'\} \notin E$. Then the graph $G' := (V, E')$ with $E' := E \setminus \{\{u, v\}, \{w, w'\}\} \cup \{\{u, w\}, \{v, w'\}\}$ has the same core decomposition as G.*

The correctness of both lemmas follows directly from the definition. Informally speaking, Lemma 1 allows for two disconnected nodes of the same coreness to each remove one edge to some nodes of higher coreness and instead become connected, and vice versa, without changing the decomposition. Furthermore, according to Lemma 2 we can swap the endnodes of intra-shell edges if this does not interfere with existing connections. Figure 2 illustrates these two lemmas for an example graph. Using these statements, we can now establish



(a) original graph

(b) after rewiring

(c) after swapping

Fig. 2. Rewiring and swapping in an example graph.

(tight) bounds of the sizes of cores and shells.

Lemma 3. *Let $G = (V, E)$ be a graph, $(V_0, \ldots, V_k)$ its core decomposition and $G_i := (V_i, E_i) := G[V_i]$ the i-core. Then the size of every i-core is bounded as follows:*

$$i + 1 \leq |V_i| \quad \text{and} \quad \frac{(i+1)i}{2} \leq |E_i| \ . \tag{1}$$

Let $n_i := |V_i \setminus V_{i+1}|$ *be the number of nodes with coreness* i *and* $m_i := |E_i \setminus E_{i+1}|$ *the number of all edges whose endnodes with minimum coreness has coreness* i *for* $0 \leq i \leq k$ *(for convenience we define* $V_{k+1} := \emptyset$ *and* $E_{k+1} := \emptyset$*). Then the size of the* i*-shell is bounded as follows:*

$$0 \leq n_i \leq |V| \tag{2}$$

$$\left.\begin{matrix} \left\lceil \frac{i \cdot |n_i|}{2} \right\rceil & \text{, if } n_i > i \\ \binom{n_i}{2} + n_i \cdot (i - n_i + 1) & \text{, if } n_i \leq i \end{matrix}\right\} \leq m_i \leq \begin{cases} i \cdot n_i & \text{, if } i < k \\ i \cdot n_i - \frac{i^2+i}{2} & \text{, if } i = k \end{cases} \tag{3}$$

Note that the bounds for the i-core (Eq. 1) are trivially obtained from the definition. The bounds for the i-shell (Eq. 2 and 3), however, use the above two lemmas, i. e., the shell has the minimum number of edges, if it has the maximum possible number of intra-shell edges, since each such edge contributes twice, and a minimum number of inter-shell edges. An analogous reasoning yields the upper bounds.

3 Core Generator

In this section, we first introduce a set of relevant parameters for the construction of core structures and discuss which combinations of these lead to feasible instances, i. e., are capable of realizing a graph with a predefined core structure. Then we describe our basic algorithm that generates such graphs, and point out several variations.

As the 0-shell only contains isolated nodes and in order to reduce technical peculiarities, we restrict ourselves to generating graphs with an empty 0-shell.

3.1 Input Parameters

There are several possibilities to specify core structures. Of the quantitative approaches, the most obvious is to give the number of nodes per shell, the number of intra-shell edges, and the number of inter-shell edges (for each pair of shells). This can be coded as a vector $N \in \mathbb{N}_0^k$ where N_i is the number of nodes in the i-shell and a symmetric matrix $M \in \mathbb{N}_0^{k \times k}$, where $M_{i,j}$ contains the number of edges connecting the i-shell with the j-shell. For example, the graph (omitting isolated nodes) given in Figure 1 has the following representation:

$$N := (4, 3, 2, 5) \quad \text{and} \quad M := \begin{pmatrix} 3 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 \\ 0 & 2 & 0 & 6 \\ 0 & 0 & 6 & 10 \end{pmatrix}$$

Clearly, the implied sizes of the shells have to respect the bounds established in Lemma 3. On the one hand, this kind of specification of core structures provides the maximum degree of freedom, i. e., the user can configure the size

distribution of each shell and is only limited by constraints ensuring consistency. On the other hand, the user has to specify every detail which can be fairly complex and time consuming for larger networks and non-trivial core structures. For example, the current network of the Autonomous Systems has a (maximum) core index of about 30. In order to generate a graph with such a core decomposition almost 500 parameters need to be specified in this model.

We propose a slightly modified set of input parameters where the matrix M is replaced by a vector $D \in [0,1]^k$ specifying the density of the shells with respect to the bounds given in Lemma 3. This greatly reduces the number of parameters and only $2 \cdot k$ parameters has to be provided. Moreover, a consistency check of the vector D is not required anymore as D is defined as a relative density (with respect to tight bounds) and not absolute values. On the downside, the user cannot precisely specify where edges will be realized in order to provide the given density of the shell, i. e., most of the edges could be inter-shell edges instead of intra-shell edges. This can partially be fixed by a post-processing step applying the rewiring lemma. Another minor disadvantage is the numerical issue that the given density may only be approximated as the defined density value implies a fractional number of edges.

3.2 General Method

Next, we describe our technique to generate graphs with a predefined core fingerprint. As we will shortly see, the method can handle both kinds of inputs, i. e., the shell connectivity matrix M as well as the density vector D.

The basic idea of our generator is to grow a graph according to the core structure from the inside to the outside. We start from the empty graph which realizes the empty $(k+1)$-core and iteratively extend this graph in order to add the next lower shell. The correctness follows by the invariant that newly added nodes of a lesser coreness cannot interfere with previously built shells by definition. The pseudo code for our generator using the shell-connectivity matrix M is given in Algorithm 1. As the absolute numbers of edges both between and inside core shells are given, it is sufficient to choose just a non-adjacent node pair. In order to guarantee that the coreness of nodes in the i-shell will not exceed i, we predefine an order σ that can be used a proof. More precisely, we define newly created edges as directed such that inter-shell edges point for the lower shell to the upper shell and intra-shell edges are directed in accordance to our predefined order σ. The crucial point is that we ensure, during the generation, that each node in V_i has a maximum out-degree of k. This allows us to use σ as an ordering to remove nodes with degree less or equal than k, thus proving that the coreness of nodes in V_i does not exceed i. We are therefore left to guarantee that the coreness is not strictly less than i but exactly i. An example where this property is violated is given in Figure 3(a). However, we can remedy such a situation by a sophisticated movement of edges. Two cases exists, an inter-shell edge starting from a node in V_i with degree greater than i can be moved to another node in V_i with

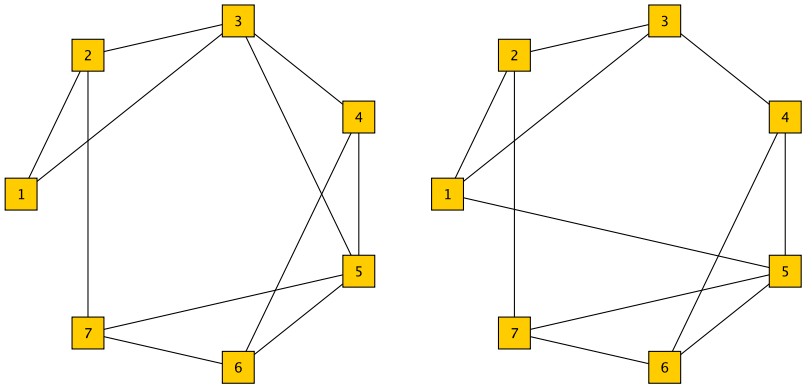
Algorithm 1: Core Generator

Input: Integer k, vector $N \in \mathbb{N}_0^k$, symmetric matrix $M \in \mathbb{N}_0^{k \times k}$
Output: Graph $G = (V, E)$
Data: Lists V_i for $1 \leq i \leq k$

$V \leftarrow \emptyset$; $E \leftarrow \emptyset$
for $i \leftarrow k$ **to** 1 **do**
  create N_i new nodes and store them in V_i
  define an ordering σ on V_i
  for $j \leftarrow i$ **to** k **do**
    for $m \leftarrow 1$ **to** $M_{i,j}$ **do**
      if $\min_{v \in V_i} \text{outdeg}(v) \geq k$ **then**
1   *Error, too many edges are requested!*
      if i=j **then**
        select $v_i \in V_i, v_j \in V_i$ with $\{v_i, v_j\} \notin E$, $\sigma(v_i) < \sigma(v_j)$, and $\text{outdeg}(v_i) < k$
        $E \leftarrow E \cup (v_i, v_j)$
      else
        select $v_i \in V_i, v_j \in V_j$ with $\{v_i, v_j\} \notin E$ and $\text{outdeg}(v_i) < k$
        $E \leftarrow E \cup (v_i, v_j)$
  remove direction of edges
2   **while** $\min_{v \in V_i} \deg(v) < i$ **do**
    $v_i \leftarrow \text{argmin}_{v \in V_i} \deg(v)$
    if $\exists \{v_j, w\} \in E$ *with* $v_j \in V_i \setminus N(v_i)$ *and* $w \in V$ **then**
3     $E \leftarrow (E \setminus \{\{v_j, w\}\}) \cup \{\{v_i, v_j\}\}$
    else
      randomly select $v_j \in V_i$ with $\deg(v_j) > i$
      randomly select $v_j' \in N(v_j) \setminus N(v_i)$
4     $E \leftarrow (E \setminus \{\{v_j, v_j'\}\}) \cup \{\{v_i, v_j'\}\}$
  $V \leftarrow V \cup V_i$
return *graph* $G = (V, E)$

an erroneously low degree. If such an edge does not exist, then an intra-shell edge starting from a node in V_i with degree greater than i can be moved in the same way. It is easy to see that if the latter case does not hold either, then the lower bound for the number of edges (Lemma 3) is violated and the core fingerprint invalid.

In principle, the same generation scheme can be used if the density vector is defined instead of the shell-connectivity matrix. The major difference now is, that the number of edges to be inserted has to be estimated and that each time we introduce a new edge, we have to decide whether to insert it as an intra-shell or as an inter-shell edge. A suitable approach is to insert inter-shell edges with probability D_i and to insert intra-shell edges with probability $(1 - D_i)$, correspondingly. In some cases, however, the density score implies strict numbers of intra-shell and inter-shell edges, thus this randomization can lead



(a) before rewiring (b) after rewiring

Fig. 3. Example which requires a rewiring. The input parameters are $k = 3$, $N = (0, 0, 7)$, and $M_{i,j} = 0$ for $(i, j) \neq (3, 3)$ and $M_{3,3} = 11$. Figure 3(a) shows the resulting graph according to the selection of the following pairs: $(1, 2), (3, 4), (5, 6), (7, 2), (3, 2), (5, 3), (7, 6), (1, 3), (4, 5), (7, 5), (6, 4)$. Although eleven edges are generated, node 1 only has degree 2. After the rewiring of edge $\{3, 5\}$ where $v_j = 3$ was selected, every nodes has coreness three.

to situations where each node already has an out-degree of i although further edges have to be inserted. Applying our Lemma 1 can amend this dead end and further edges can be inserted.

3.3 Refinements

As the core fingerprint is only one of the interesting characteristics, we briefly discuss which other relevant features can be integrated in the generator.

The first characteristic is connectivity, i. e., how many connected components exist per shell. Although the whole graph or even the i-core is connected, the i-shell can have several disconnected components. In order to prevent this, the user can specify the number and size of connected components. The generator will then first create a spanning forest and mark those edges as not rewirable. This approach implies a trade-off since not every combination of a valid shell-connectivity matrix/density vector and number and size of connected components can be realized. This conflict can be resolve by increasing or decreasing the predefined number of edges or the number and size of connected components depending on the users' interests.

A second highly popular feature is the degree distribution. In a post-processing step, we can apply a sequence of rewirings (Lemma 1) and swappings (Lemma 2) in order to approach a given degree distribution.

4 Modeling the AS Network

An important application of a core-aware network generator is the simulation of the Internet at the AS level. In this section we compare networks generated by our method and established topology generators with an exemplary snapshot of the real AS network at the router level provided by the Oregon Routeviews project [15] at midnight on January 1, 2006 (oix-full-snapshot-2006-01-01-0000). This graph consists of 21419 nodes and 45638 edges.

4.1 Topology Generators

The first methods to generate networks with Internet-like structure date back to the 1990s and a multitude of techniques has been proposed since then. Among the most popular and widely used tools we have chosen Inet-3.0 [9] and BRITE [12] for our comparison since these are commonly included in other studies which cover a broader range of existing models [11, 9]. Although nem [10] also seems promising we do not take it into account because of its limitation to networks not greater than 4000 nodes.

The *Internet topology generator Inet* [9] generates an AS-level representation of the Internet. Its developers claim that *"it generates random networks with characteristics similar to those of the Internet from November 1997 to Feb 2002, and beyond"*. Basically, Inet generates networks with a degree distribution which fits to one of the power laws originally found by Faloutsos et al. [7], namely that the frequency of nodes with degree d is proportional to d raised to a power of a constant α: $f(d) \propto d^{\alpha}$. Since this law does not cover all nodes and in order to match other relevant properties as well, optimizations for various specific conditions were added to the original procedure over time. The complete generation method is explained in [9]. Since the procedures of Inet are already customized to AS networks, only a small number of input parameters can be specified: the total number of nodes, the fraction of degree-one nodes, and the size of the square used for node placement.

The *Boston university Representative Internet Topology gEnerator BRITE* [12] can generate networks for different levels of the Internet topology. Beside this, it offers various other options to customize the generation procedure.

Drawing area. The nodes of the generated topology are distributed in a square of a certain size.

Node distribution. In the drawing area, nodes are either distributed uniformly at random or Pareto.

Outgoing links. New nodes are connected with a specific number of outgoing links to other, already existing nodes.

Connectivity. The neighborhood of a node is selected based on certain guidelines such as geometric locality, preferential attachment, or a combination of both.

Procedure. Nodes can either be placed before the addition of edges or in an incremental fashion. In the latter case each new node introduces a number of new edges that can only connect to already existing nodes.

4.2 Characteristics

In [9], an extensive collection of characteristics is evaluated that judge the fitness of a generated graph with respect to its real world counterpart. We repeated this evaluation for a representative selection of these properties with a focus on the assessment of the core generator. In the following, we summarize the properties we employed in our analyses.

General statistics. To see how well the generated networks fit to the most obvious characteristics we computed some basic properties: the number of edges, the minimum and the maximum degree. Note that all models strictly meet the given number of nodes, so the number of edges corresponds to density and average degree.

Cores. The core decomposition is a significant structural property of an AS network. We compare not only the core number but the extensive core fingerprint.

Clustering coefficient. The clustering coefficient is a measure for the local density around a node. It counts how many of a node's pairs of neighbors are themselves adjacent. These values are averaged to get a single measure for the network. Closely related characteristics are the numbers of triangles and triples and the transitivity [16].

Path length. We compare two properties based on path length: *characteristic path length*, which is the average of the distances of all node pairs and average *eccentricity*. The eccentricity of a node is its maximum distance to all other nodes. Average eccentricity then is the average of all nodes eccentricities.

Frequency versus degree. One of the classic power laws found by Faloutsos et al. [7] is $f(d) \propto d^{\alpha}$, that is, the frequency of nodes with degree d, is proportional to d raised to a power of a constant α. Since this power law does not hold for nearly 2% of the highest degree nodes, we use a modified version [4, 5]:

$$F(d) = \sum_{i>d} f(i) \propto d^{\alpha} \ .$$

Size of k-neighborhood. Another power law indentified in [7] is $\mathcal{N}(k) \propto k^{\beta}$, where $\mathcal{N}(k)$ is the sum over all nodes of their neighborhood sizes within distance k, i. e., $\mathcal{N}(k) = \sum_{u \in V} \sum_{v \in V} \mathrm{dist}_k(u, v)$, where

$$\mathrm{dist}_k(u, v) = \begin{cases} 1 & \text{, if } \mathrm{dist}(u, v) \leq k \\ 0 & \text{, otherwise.} \end{cases}$$

Note that this characteristic can also be measured as an average over all nodes, and it is also known as the *number of pairs within k hops*.

4.3 Evaluation

In the following, we detail the findings of our systematic evaluation. We gathered results on the three generators as described in Sections 3 and 4.1 and on the real AS network for all the properties listed in Section 4.2.

Based on the previous studies we set appropriate parameters for the generators Inet and BRITE. For Inet we have chosen the default input parameters except for the number of nodes and the random seed. As the results in [13] suggest, we have used preferential attachment and incremental growth for BRITE. Furthermore, we add two edges for each new node to fit the average degree of AS networks.

	Real AS	Core	BRITE	Inet
Number of Nodes	21,419	21,419	21,419	21,419
Number of Edges	45,638	45,638	42,835	58,069
Minimum Degree	1	1	2	1
Maximum Degree	2,408	470	411	3,572
Core Number	26	26	2	19
Number of Triples	12,161,105	5,197,409	637,716	30,643,657
Number of Triangles	46,256	34,465	177	75,547
Transitivity	0.011	0.020	0.001	0.007
Clustering Coeff.	0.375	0.143	0.002	0.535
Avg. Path Length	3.811	3.812	5.305	3.072
Avg. Eccentricity	8.519	7.911	8.626	6.431

Table 1. Characteristics of the AS network and the three generators.

By construction, the numbers of nodes match the reference AS network, however, the numbers of edges already differ heavily. While the number of edges is only slightly lower for graphs generated by BRITE, and exaclty fits the reference for core generator (called *Core* in the following), the edge set created by Inet is larger by one third.

The well-known phenomenon of higly connected hubs in the AS network accompanied by the power-law degree distribution is regarded as one of the most significant properties of the Internet. Inet reproduces these quite well, but overstates the maximum degree. In contrast, the degree distribution of Core oscillates around the reference but fails to produce high-degree nodes due to its lack of preferential attachment and the degree distribution of BRITE suggests that the preference of new nodes to connect to existing hubs is not strong enough either. These facts can be observed in Figure 4.

At a first glance, BRITE clearly fails to build up any kind of deep core structure (the core number is 2). The reason for this becomes evident from the incremental generation process of BRITE: the iterative addition of nodes incident to two new edges can simply be reversed, resulting in a valid removal

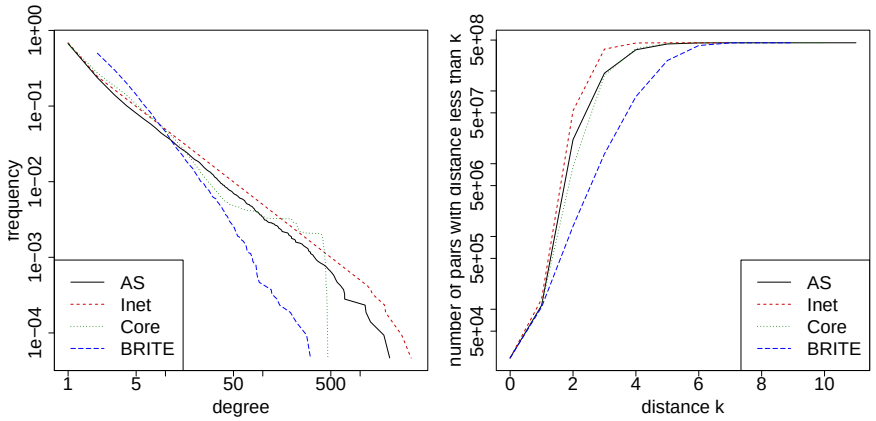


Fig. 4. The fraction of nodes with a degree lesser or equal to d (left). The k-neighborhood for distances $k \in [0, 10]$ (right).

sequence for the 2-core that ultimately yields an empty 3-core. Figure 5 plots both the number of nodes and the number of edges per k-core. Inet builds up a decent core hierarchy but fails to attain a sufficient depth, obviously resulting in larger mid-level shells, in terms of both nodes and edges. By construction, Core perfectly matches the reference. The plots in Figure 6 show the numbers of nodes and edges per k-shell. They confirm the above observations and additionally grant an insight into the absolute numbers of elements per shell.

The shallow core structure created by BRITE is accompanied by a very low transitivity alongside a negligible number of triangles and a tiny clustering coefficient, suggesting that the BRITE graph is primarily composed of a set of paths of length two. The high average path length further corroborates this conjecture, since by virtue of preferential attachment hubs of high degree evolve, which, however, are interconnected via paths of length two by construction.

The absolute numbers of triples and triangles as well as the transitivity and the clustering coefficient are acceptable for both Core and Inet. The discrepancy of the latter generator from the reference can quite generally be explained by the increased number of edges. The behavior of Core with respect to these values is largely due to the absence of high-degree nodes, since, intuitively speaking, star-shaped structures yield a high number of triples. The relatively high number of triangles thus yields an increased transitivity. The low clustering coefficient, however, suggests, that there is large number of nodes with a sparse direct neighborhood. Since, at the same time, Core exhibits a high number of triangles, the majority of these triangles is incident to nodes with higher degree.

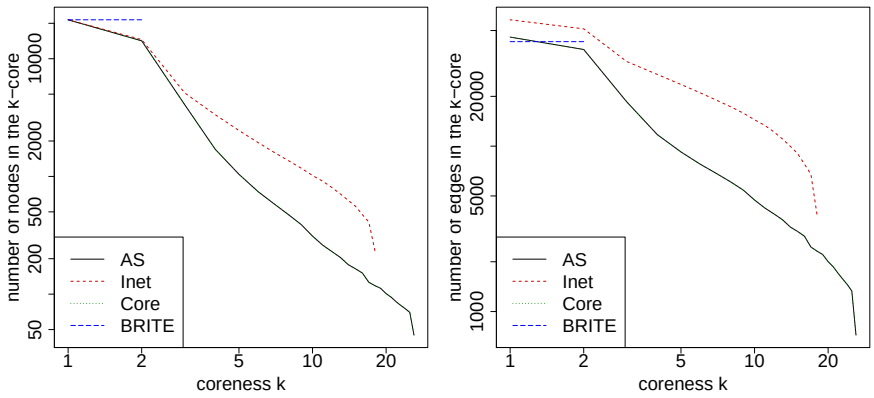


Fig. 5. The numbers of nodes (left figure) and of edges (right figure) per k-core. Note that BRITE generates only nodes in the 2-core and that the lines of the AS and Core perfectly match by construction.

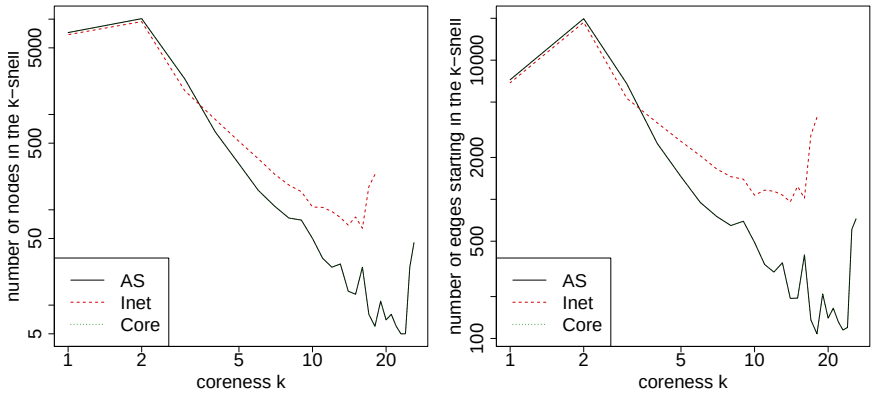


Fig. 6. The numbers of nodes (left figure) and of edges (right figure) per k-shell (BRITE omitted). An edge is considered to belong to the ℓ-shell if its endnode with smallest coreness has coreness ℓ. Note that the lines of the AS and Core perfectly match by construction.

Figure 4 depicts the size of the neighborhood within k hops (sum over all nodes). Note that the high average path length of BRITE mentioned earlier comes along with the slow growth of the neighborhood size. The low average path length and the low average eccentricity exhibited by Inet are, again, due to the large edge set. With respect to these values, Core excels. Both the average path length and the k-neighborhood practically match the reference.

5 Conclusion

In the recent past, the core decomposition has been found to be a crucial characteristic of real world complex systems. To our knowledge this paper is the first to scrutinize and clarify how to specify the core fingerprint of a network by examining the interconnectivity of each pair of shells. We employ this core fingerprint to introduce a simple and efficent algorithm for the generation of random graphs based on the core decomposition.

We exemplify the feasibility of our technique in a case study using the AS network of the Internet, comparing our generator to the established topology generators BRITE [12] and Inet [9]. Our results yield that the our generator is highly suitable for the simulation of AS topologies, confirming the importance of the core decomposition. Moreover we show that BRITE largely fails to capture significant characteristics of the AS network, including its core structure, and that Inet roughly matches the reference except for its general tendency to be too densely connected. While our core generator and BRITE create a topology within seconds, a major drawback of Inet is its generation time of several minutes.

The high customizability of our rather generic core generator suggests several adaptations that can further increase the fitness to the specific peculiarities of the AS network. Such adaptations to special networks can be realized by employing a number of structural modifications such as swapping, rewiring and the use of preferential attachment without interfering with the core decomposition.

References

1. José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large Scale Networks Fingerprinting and Visualization Using the k-Core Decomposition. In *Advances in Neural Information Processing Systems 18*, pages 41–50. MIT Press, 2006.
2. Vladimir Batagelj and Matjaž Zaveršnik. An $\mathcal{O}(m)$ Algorithm for Cores Decomposition of Networks. Technical Report 798, IMFM Ljublana, Ljubljana, 2002.
3. Vladimir Batagelj and Matjaž Zaveršnik. Generalized Cores. Preprint 799, IMFM Ljublana, Ljubljana, 2002.

4. Tian Bu and Don Towsley. On Distinguishing between Internet Power Law Topology Generators. In INFOCOM'02 [8].
5. Qian Chen, Hyunseok Chang, Ramesh Govindan, and Sugih Jamin. The Origin of Power Laws in Internet Topologies Revisited. In INFOCOM'02 [8], pages 608–617.
6. Nicolas Ducheneaut, Nicholas Yee, Eric Nickell, and Robert J. Moore. Alone Together?: Exploring the Social Dynamics of Massively Multiplayer Online Games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'06)*, pages 407–416. ACM Press, 2006.
7. Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262. ACM Press, 1999.
8. *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom)*, volume 1. IEEE Computer Society Press, 2002.
9. Cheng Jin, Qian Chen, and Sugih Jamin. Inet Topology Generator. Technical Report CSE-TR-433, EECS Department, University of Michigan, 2000.
10. Damien Magoni. nem: A Software for Network Topology Analysis and Modeling. In *Proceedings of the 10th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. IEEE Computer Society, 2002.
11. Damien Magoni and Jean Jacques Pansiot. Analysis and Comparison of Internet Topology Generators. In *Proceedings of the 2nd International IFIP-TC6 Networking Conference*, pages 364–375. Springer, 2002.
12. Alberto Medina, Anukool Lakhina, Ibrahim Matta, and John Byers. BRITE: An Approach to Universal Topology Generation. In *Proceedings of the 9th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 2001.
13. Alberto Medina, Ibrahim Matta, and John Byers. On the Origin of Power Laws in Internet Topologies. *Computer Communication Review*, 30(2), April 2000.
14. Romualdo Pastor-Satorras and Alessandro Vespignani. *Evolution and Structure of the Internet: A Statistical Physics Approach*. Cambridge University Press, 2004.
15. University of Oregon Routeviews Project. `http://www.routeviews.org/`.
16. Thomas Schank and Dorothea Wagner. Approximating Clustering Coefficient and Transitivity. *Journal of Graph Algorithms and Applications*, 9(2):265–275, 2005.
17. Stephen B. Seidman. Network Structure and Minimum Degree. *Social Networks*, 5:269–287, 1983.
18. Stefan Wuchty and Eivind Almaas. Peeling the Yeast Protein Network. *Proteomics*, 5(2):444–449, 2005.