Approximating Clustering-Coefficient and Transitivity*

Thomas Schank† and Dorothea Wagner‡
University of Karlsruhe, Germany
Department of Computer-Science
ILKD Wagner §
(Dated: June 11, 2004)

Since its introduction in the year 1998 by Watts and Strogatz, the clustering-coefficient has become a frequently used tool for analyzing graphs. In 2002 the transitivity was proposed by Newman, Strogatz and Watts as an alternative to the clustering-coefficient. However, as we illustrate by several examples both parameters may differ vastly. On the other hand, an extension of the definitions to weighted versions provides the formal relation between them. As many networks considered in complex systems are huge, the efficient computation of such network parameters is crucial. Several algorithms with polynomial running time can be derived from results known in graph theory. The main contribution of this work is a new fast approximation algorithm for the weighted clustering-coefficient which also gives very efficient approximation algorithms for the clustering-coefficient and the transitivity. We namely present an algorithm with running time in $O(1)$ for the clustering-coefficient, respectively with running time in $O(n)$ for the transitivity. By an experimental study we demonstrate the performance of the proposed algorithms on real-world data as well as on generated graphs. These results also support the assumption that normally the values of clustering-coefficient and transitivity differ considerably.

Keywords: network analysis, clustering-coefficient, transitivity, counting triangles, approximation algorithms, Monte-Carlo algorithms, sublinear algorithms, graph generator, preferential attachment

I. INTRODUCTION

Recently, there is growing interest in understanding the structure, dynamics and evolution of large networks like the Internet, the World Wide Web, technological and biological networks or social networks. One way of analyzing specific properties of networks consists in computing and comparing certain local or global network indices like degree distribution and connectedness. Algorithmic aspects in network analysis concern the correct and fast computation of such indices. Vertex indices are often easily computable in polynomial time. However, if networks are large even polynomial running times that are e.g. cubic in the number of nodes are not acceptable.

A frequently used tool for analyzing graphs is the clustering-coefficient introduced in [WS98] respectively transitivity proposed in [NSW02]. This paper concentrates on the algorithmic aspects of computing those indices. First, we illustrate that clustering-coefficient and transitivity may differ considerably. However, an extension of the definitions to weighted versions provides the formal relation between them. The main contribution of this work is a new fast approximation algorithm for the weighted clustering-coefficient which also gives very efficient approximation algorithms for the clustering-coefficient and for transitivity. An experimental study demonstrates the performance of the proposed algorithms on real-world data as well as on generated graphs. These results also support the assumption that the values of clustering-coefficient and transitivity differ in general.

A. Basic Definitions

Let $G = (V, E)$ be a undirected, simple (no self-loops, no multiple edges) graph (network) with a set of nodes (vertices) V and a set of edges E. We use the symbol n for the number of nodes and the symbol m for the number of edges. The *degree* $d(v) := |\{u \in V : \text{there is} \{v, u\} \in E\}|$ of node v is defined to be the number of nodes in V that are adjacent to v.

We use Landau's O-Notation to compare asymptotic behavior of functions, e.g. the running time of algorithms. Let g, f be functions from $\mathbb{N}$ to $\mathbb{R}$ then $O(g(n)) := \{f(n) : \text{there are positive constants } c, N \text{ such that } 0 \leq f(n) \leq c\ g(n) \text{ for all } n \geq N\}$, $\Omega(g(n)) := \{f(n) : \text{there are positive constants } c, N \text{ such that } 0 \leq c\ g(n) \leq f(n) \text{ for all } n \geq N\}$ and $\Theta(g(n)) := O(g(n)) \cap \Omega(g(n))$.

B. The Clustering-Coefficient

The clustering-coefficient was introduced by Watts and Strogatz [WS98] in the context of social network analysis. Given three actors u, v and w with mutual relations between v and u as well as between v and w, it is supposed to represent the likeliness that u and w are also related. We formalize this notion by defining for $v \in V$

*The authors gratefully acknowledge financial support from DFG under grant WA 654/13-2 and from the European Commission within FET Open Project COSIN (IST-2001-33555).
†Electronic address: schank@ira.uka.de
‡Electronic address: dwagner@ira.uka.de
§URL: http://i11www.ira.uka.de/algo/

$m(v) := |\{\{u,w\} \in E : \{v,u\} \in E \text{ and } \{v,w\} \in E\}|$ and $t(v) := d(v)(d(v)-1)/2$. We call $m(v)$ the *number of opposite edges of* v, and $t(v)$ the *number of potential opposite edges of* v. For a node v with $d(v) \geq 2$, the *clustering-coefficient* is defined as

$$c(v) := \frac{m(v)}{t(v)}.$$

Then the clustering-coefficient of graph $G = (V, E)$ is defined as

$$C(G) := \frac{1}{|V'|} \sum_{v \in V'} c(v), \tag{1}$$

where V' is the set of nodes v with $d(v) \geq 2$. It should be mentioned that there is some variation in the literature with respect to nodes of degree less than two. Sometimes, $c(v)$ is defined to be either zero or one for such nodes v. Alternatively, nodes of degree less than two are not taken into account for the computation of $C(G)$. However, the choice of the definition is important as can be seen from the results of our experiments for the AS-graph in Sec. IV, Tab. IV.

C. The Transitivity

The transitivity was introduced by Newman, Watts and Strogatz in [NSW02] where it was claimed to be equal to the clustering-coefficient.

A *triangle* $\Delta = (V_\Delta, E_\Delta)$ of a graph $G = (V, E)$ is a three node subgraph with $V_\Delta = \{v_1, v_2, v_3\} \subset V$ and $E_\Delta = \{\{v_1, v_2\}, \{v_2, v_3\}, \{v_3, v_1\}\} \subset E$. We use the symbol $\delta(G)$ for the *number of triangles* in graph G. A *triple* $\Upsilon = (V_\Upsilon, E_\Upsilon)$ of a graph $G = (V, E)$ is a three node subgraph with $V_\Upsilon = \{v_1, v_2, v_3\} \subset V$ and $E_\Upsilon = \{\{v_1, v_2\}, \{v_2, v_3\}\} \subset E$, where v_2 is the *center* of the triple. We will use $\tau(G)$ to denote the *number of triples* in graph G. Note that each triangle contains exactly three triples. Motivated by this property, Newman, Watts and Strogatz defined the *transitivity* for a graph as

$$T(G) := \frac{3\,\delta(G)}{\tau(G)}. \tag{2}$$

If the graph is clear from context, we simply use C, T, δ and t.

II. EXTENDING THE DEFINITIONS AND THE RELATION BETWEEN T AND C

As already mentioned in [BR02], T and C are different. Actually, we show that the values of T and C differ significantly for some networks of various density. By extending the definition of the clustering-coefficient to a weighted version, we can derive a formal relation between C and T. Considering the analogous generalization of the transitivity will be useful for the approximation algorithm introduced in Sec. III C.

TABLE I: Behavior of density, clustering-coefficient and transitivity for different classes of graphs. The last row points to the section in the appendix where the construction of the graph families is described.

D	C	T	Sec.
sparse	$C \to 0$	$T \to 0$	C 1
sparse	$C \to 1$	$T \to 0$	C 3
sparse	$C \to 0$	$T \to 1$	C 4
sparse	$C \to 1$	$T \to 1$	C 2
dense	$C \to 0$	$T \to 0$	C 5
dense	$C \to 1$	$T \to 0$	C 6
dense	$C \to 0$	$T \to 1$	C 7
dense	$C \to 1$	$T \to 1$	C 8

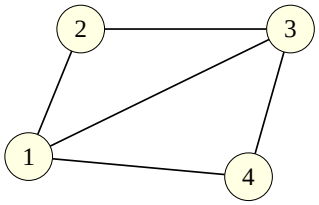

FIG. 1: For this graph $T = 0.75 \neq 0.83 = C$.

A. The Difference between Values of C and T

The smallest graph with differing values for C and T is depicted in Fig. 1. Tab. I shows the values for C, T and the density $D := m/\binom{n}{2}$ for some graph families whose construction is described in App. C. That is, one can construct graphs for which the values of T and C differ as much as possible, independent from the density of the graph.

B. Generalization of the Clustering-Coefficient with Weights

The definition of the clustering-coefficient does not consider the fact that, depending on the network, some nodes might be more important than others. This might be specified by a weight function that is either induced by the graph structure or given explicitly. For a weight function $w : V \to \mathbb{R}^+$ we define the *weighted clustering-coefficient* to be

$$C_w(G) := \frac{1}{\sum_{v \in V'} w(v)} \sum_{v \in V'} w(v)c(v). \tag{3}$$

This definition maintains the property that C_w is within the range between zero and one. Two implicit weight functions w are immediate, the degree-based weight $w(v) = d(v)$, and the number of potential opposite edges $w(v) = t(v)$. In the second case, $t(v)$ simply cancels out in each additive term of the numerator, and we get

$$C_t(G) = \frac{\sum m(v)}{\sum t(v)}. \tag{4}$$

We return to the definition of the transitivity. Each potential opposite edge has a one-to-one correspondence to one triple. Therefore, the sum of numbers of potential opposite edges over all nodes is equal to the number of triples in a graph, i.e. $\sum t(v) = \tau$. Furthermore $\sum m(v) = 3\,\delta$, because each triangle implies exactly three opposite edges. We now recognize that the transitivity is just the weighted clustering-coefficient with respect to the number of potential opposite edges

$$T(G) = C_t(G). \tag{5}$$

An equivalent formulation of Eq. 5 was already presented in [BR02]. The following properties are immediate.

Corollary 1 *For graphs where*

- *all nodes have the same degree, or*
- *all nodes have the same clustering-coefficient*

C and T are equal.

The first property is quite interesting with respect to the small-world networks of Watts and Strogatz [WS98] where node degrees do not differ much.

C. Generalization of the Transitivity with Weights

Let Π be the set of all triples in a graph G, then $\tau(G) = |\Pi|$. Further consider the mapping $X : \Pi \to \{0, 1\}$, where $X(\Upsilon)$ equals one if there is an edge between the outer nodes of the triple Υ, and zero otherwise. Then we can rewrite the transitivity as

$$T(G) = \frac{1}{|\Pi|} \sum_{\Upsilon \in \Pi} X(\Upsilon).$$

This equation is similar to the definition of the clustering-coefficient in Eq. 1. Again we can consider a weight function $\varpi : \Pi \to \mathbb{R}^+$ and define

$$T_\varpi(G) := \frac{1}{\sum\limits_{\Upsilon \in \Pi} \varpi(\Upsilon)} \sum_{\Upsilon \in \Pi} \varpi(\Upsilon) X(\Upsilon) \tag{6}$$

similar as in Eq. 3.

Lemma 1 *The weighted clustering-coefficient is a special case of the weighted transitivity.*

Proof of Lemma 1 For a node weight function w, let $\varpi(\Upsilon) := \frac{w(v)}{t(v)}$ where v is the center of triple Υ. Further, let Π_v be the set of triples with center v. Then accordingly $t(v) = |\Pi_v|$ and $\sum_{\Pi_v} X(\Upsilon) = m(v)$. By appropriate transformation we get $C_w = T_\varpi$. ⊡

Note that for $w = t$, i.e. $\varpi \equiv 1$, $T_\varpi = T$ and for $w \equiv 1$, i.e. $\varpi(\Upsilon) = 1/t(v)$, $T_\varpi = C$, where again v is assumed to be the center of triple Υ.

We observe that nodes of degree less than two cannot be center of any triple and thus do not contribute to T_ϖ. Accordingly, the convention to ignore nodes of degree less than two for the definition of the clustering-coefficient is more convenient. Altogether, the notion of the weighted transitivity can be viewed as the most general definition from which the unweighted transitivity as well as the weighted clustering-coefficient can be derived as special cases.

III. ALGORITHMS

A. Relation to Other Problems in Graph Theory

We now consider the efficient computation of the weighted clustering-coefficient C_w. Let us assume that for each node v the weight $w(v)$ can be computed in constant time. First note that $t(v)$ can be computed in constant time for each node, provided that the degree $d(v)$ is known. Otherwise, the degree of all nodes can be computed in a preprocessing step in linear time if the graph is represented appropriately, e.g. by adjacency lists. It remains to compute the number of opposite edges $m(v)$ for each node v. As already mentioned in Sec. II B, this is equivalent to computing the number of triangles containing node v.

Note that the transitivity requires only to compute the total number of triangles in a graph. For the (weighted) clustering-coefficient, however, the number of triangles containing node v have to be computed locally for each node v. Hence, it might be possible to compute the transitivity more efficiently than the clustering-coefficient. It is an open question if such an algorithm exists. All algorithms known so far for counting all triangles in a graph can be modified to count also locally for all nodes the triangles containing that node without additional running time.

B. Exact Algorithms

1. The Matrix Multiplication Method

The diagonal elements of the third power of a graph's adjacency matrix contain the number of triangles for each node. This gives an algorithm with running time in

$O(n^\gamma)$, with γ being the matrix multiplication exponent ($2 \leq \gamma \leq 2.37$). It should be mentioned, that due to its complexity (respectively implementation efforts) and numerical instabilities, fast matrix multiplication is hardly used. So, in practice the matrix multiplication method has running time in $O(n^3)$.

2. An Alternative $O(n^3)$-Algorithm

For each node simply check if edges connecting two adjacent nodes exist. Since there are $d(v)(d(v)-1)/2$ potential opposite edges for a node v, the running time is in $O\left(n \cdot \max\{d(v)\}^2\right)$ which is in $O\left(n^3\right)$.

3. The AYZ-Algorithm

The most efficient algorithm for counting triangles we are aware of is proposed by Alon, Yuster and Zwick in [AYZ97]. The running time is in $O\left(m^{2\gamma/(\gamma+1)}\right)$, which is in $O\left(m^{1.41}\right)$ with fast matrix multiplication and in $O(m^{1.5})$ with standard matrix multiplication.

4. Discussion of the Algorithms

The matrix multiplication method is mentioned due to its apparent popularity. However, besides the fact that the running time is not optimal it has a great disadvantage with respect to its space requirement. Unless special techniques for sparse matrices are used, the space consumption is in $\Theta(n^2)$. Hence this algorithm is not recommendable.

The alternative $O(n^3)$ algorithm is space efficient. It is also time efficient if there are no high degree vertices in the network. Otherwise the AYZ-algorithm is the method of choice for obtaining exact results.

C. Approximation Algorithms

In very large networks, the exact computation of the clustering-coefficient might not be practicable as it is too time consuming. Relaxing exactness of the computation in order to enable more efficient algorithms is an alternative. Therefore, we present a Monte-Carlo algorithm based on sampling to approximate the weighted clustering-coefficient.

1. An Approximation Algorithm for the Weighted Clustering-Coefficient

Roughly speaking our approximation algorithm samples triples with appropriate probability. It then checks whether an edge between the non-center nodes of the triple is present. Finally, it returns the ratio between the number of existing edges and the number of samples. The pseudo-code is presented in Alg. 1. As it is the case for a lot of randomized algorithms, Alg. 1 is quite simple. However, its time complexity and correctness is less obvious.

Algorithm 1: C_w-Approximation

Input: array A of nodes $v \in V$ with $d(v) \geq 2$
adjacency array for each node
weight function $w : V \to \mathbb{R}^+$ with $w(v) = 0$ for $d(v) \leq 1$
number k of samples
Output: approximation of C_w
Data : real variables: $rand, weightsum$
node variables: u, v, w
integer variable: l
real weight array $W[]$ of size n

```
weightsum ← 0
1 for v_i ∈ V do
      weightsum ← weightsum + w(v_i)
      W[v_i] ← weightsum
l ← 0
2 for i ∈ (1, ..., k) do
      rand ← UniformRandomNumber([0, weighsum])
3     v ← FindNode(node v in A with W[v] ≤ rand and
      W[v] ≥ W[x] for all x ∈ V with W[x] ≤ rand )
4     u ← RandomAdjacentNode(v)
      repeat
          w ← RandomAdjacentNode(v)
      until u ≠ w
5     if EdgeExists(u, w) then
          l ← l + 1
return l/k
```

Theorem 1 *For a graph G with node weights w, a value $C_w^{approx}(G)$ that is in $[C_w(G) - \epsilon, C_w(G) + \epsilon]$ with probability at least $\frac{\pi-1}{\pi}$ can be computed in time $O\left(n\, g(n) + m + \frac{\ln \pi}{\epsilon^2} \ln n\right)$, where the worst-case running time required to compute $w(v)$ is in $O(g(n))$.*

Proof of Theorem 1: We prove that Alg. 1 has the requested properties. Let us first consider the time complexity. The running time of the first for-loop (starting at line-number 1) is obviously in $O(ng(n))$. For the second for-loop (line-number 2), the *FindNode* function (line-number 3) can be executed in $\ln n$ steps by performing binary search. Choosing two adjacent nodes (line-number 4 to line-number 5) is expected to be in constant time. The *EdgeExists* function (line-number 5) is expected to be in constant time as well if a hash data structure is used for the edges. Finally, defining $k := \lceil \ln \pi / 2\epsilon^2 \rceil$ gives total expected running time of $O\left(\frac{\ln \pi}{\epsilon^2} \ln n\right)$ for the second for-loop.

In order to prove the correctness for our choice of k, we make use of Hoeffding's bound; see Sec. B of the appendix for further details. We have to prove that the expectation

$E(l/k)$ is equal to C_w and that the bounds on ϵ and $\frac{\pi-1}{\pi}$ are fulfilled for our choice of k. Lemma 1 states that C_w can be computed by testing for each triple whether it is contained in a triangle or not. With the same notation as in Sec. II C and particularly as in the proof of Lemma 1, we get

$$C_w(G) = \frac{1}{\sum_{v \in V'} w(v)} \sum_{v \in V'} \sum_{\Upsilon \in \Pi_v} \frac{w(v)}{t(v)} X(\Upsilon)$$

where $w(v)/t(v)$ is the weight of the corresponding triple. However, $w(v)/t(v)$ is also the probability that a triple is being chosen in Alg. 1. Hence, by linearity of the expectation $E(l/k) = C_w$.

The random variable $X(\Upsilon)$ is a mapping from Π to $\{0, 1\}$, and hence $M = 1$ in Hoeffding's bound; see Sec. B of the appendix. We can now immediately see that the bounds on ϵ and the probability $\frac{\pi-1}{\pi}$ are fulfilled for our choice of k. ⊡

One may regard the error bound ϵ and the probability π as fixed parameters. Under this assumption, we get immediately the following corollary.

Corollary 2 *For a fixed error bound ϵ and a fixed probability parameter π, the clustering-coefficient C and the transitivity T can be approximated in linear time within the interval $C \pm \epsilon$ respectively $T \pm \epsilon$ with probability $\frac{\pi-1}{\pi}$.*

D. Sublinear Approximation Algorithms

With the evolution of massive networks the study of sublinear algorithms gained interest. Such algorithms process only parts of the input graph. However, there is usually a requirement to preprocess the input in order to obtain a suitable representation of the graph resp. compute an appropriate data structure. This takes at least linear time. However, once the preprocessing is performed, several sublinear algorithms might be applied using the same data structures.

In the following, we assume that a random sampling of a node, random sampling of adjacent nodes and the test of the presence of an edge can be done in constant time. The first two conditions can be easily satisfied by using arrays for nodes and adjacency lists. The third one requires more advanced data structures, e.g. a matrix for dense graph or hash-maps for sparse graphs. We further assume the error bound ϵ and the probability parameter π to be fixed and that $w(v)$ is computable in $O(1)$ time.

Corollary 3 *Under the above assumptions the weighted clustering-coefficient C_w can be approximated in $O(n)$ time.*

This statement follows directly from the proof of Theorem 1. If we further assume that the nodes of degree less than two are kept in a separate array, the following holds:

Corollary 4 *The clustering-coefficient C can be approximated in constant time.*

Proof of Corollary 4 For the unweighted case, Alg. 1 can be modified in the following way:

1. remove the *weightsum* variable and related functions,
2. remove the for loop beginning at line-number 1,
3. remove the *FindNode* function in the second for-loop (beginning at line number 2) and instead sample each node by simply choosing a node at random in $O(1)$ time.

With these modifications the algorithm has running time $O(k)$ with k depending only on ϵ and π which we assumed to be constant. ⊡

IV. IMPLEMENTATION AND EXPERIMENTS

A. Implementation of the Algorithms

We use abbreviations AYZ for our implementation of the AYZ-Algorithm, N3 for the implementation of the alternative $O(n^3)$-algorithm, and APPROX for our implementation of Alg. 1.

Our implementations are publicly available from `http://i11www.ira.uka.de/algo/people/schank/cct`. The programs should compile and run on any recent Linux-based or similar operating system.

B. Experiments

We computed the clustering-coefficient C and transitivity T. In order to enable comparison to previous results we also give the values of C^0 where $c(v) = 0$ for $d(v) \leq 1$, and C^1 where $c(v) = 1$ for $d(v) \leq 1$.

The parameters of APPROX were set to $\epsilon = 0.001$ and $\pi = 10^{10}$. The running time is always measured without reading and building the graph in memory, which corresponds to the sublinearity of the algorithms mentioned in Sec. III D. The APPROX algorithm needs some amount of random bits, which were taken from the device */dev/urandom*. The running time was measured with the *getrusage*() function. The binaries were compiled with gnu cplusplus compiler version 3.3 and optimization level -O3. The experiments were finally carried out on a 2.4 GHz Intel-Xeon (we used only one of the two processors) based machine running a Linux-based operating system. All graphs considered did fit in main memory.

TABLE II: Transitivity and clustering-coefficient for the movie-actor network.

T	C	C^0	C^1
0.166	0.785	0.780	0.786

TABLE III: Size and running time in seconds for the movie-actor network.

n	m	N3	AYZ	APPROX
$382 \cdot 10^3$	$15 \cdot 10^6$	3369	3265	125

1. *The Movie Actor-Network*

The computation of the clustering coefficient was already done in [NSW02]. Our value for T deviates by 0.033 from the result given in [NSW02] see Tab. II.

It can be clearly seen in Tab. III that APPROX, running in about two minutes, beats N3 and AYZ which both run for about one hour. There is no substantial difference in running time between N3 and AYZ. The reason is the almost complete absence of high degree nodes. AYZ and N3 will have the same running time if all nodes have degree less than $\sqrt{m}$, see [AYZ97] for details.

2. *The AS-graph*

We computed the parameters for the AS-graph of 2002/04/06 collected by S. Argawahl and made available to public under http://www.cs.berkeley.edu/~sagarwal/research/BGP-hierarchy/data.

The AS graph is a prominent example of the necessity to specify the way the clustering-coefficient for nodes of degree less than two is defined. As can be seen in table IV the difference between C^0 and C^1 is 0.321.

Tab. V contains the running times. It takes more than one minute to calculate the approximation whereas AYZ runs less than a second. This shows the negative effects of the number of samples being independent of the input size of the graph.

3. *Graph-Generator*

In the following we present results of the running times on graphs generated by an implementation of Alg. 2. For each step of n a new graph was created and than the algorithms were applied on this graph.

TABLE IV: Transitivity and clustering-coefficient for the AS graph from 2002/04/06.

T	C	C^0	C^1
0.012	0.458	0.311	0.632

TABLE V: Size and running time in seconds for the AS-graph from 2002/04/06.

n	m	N3	AYZ	APPROX
13164	28510	2	0	70

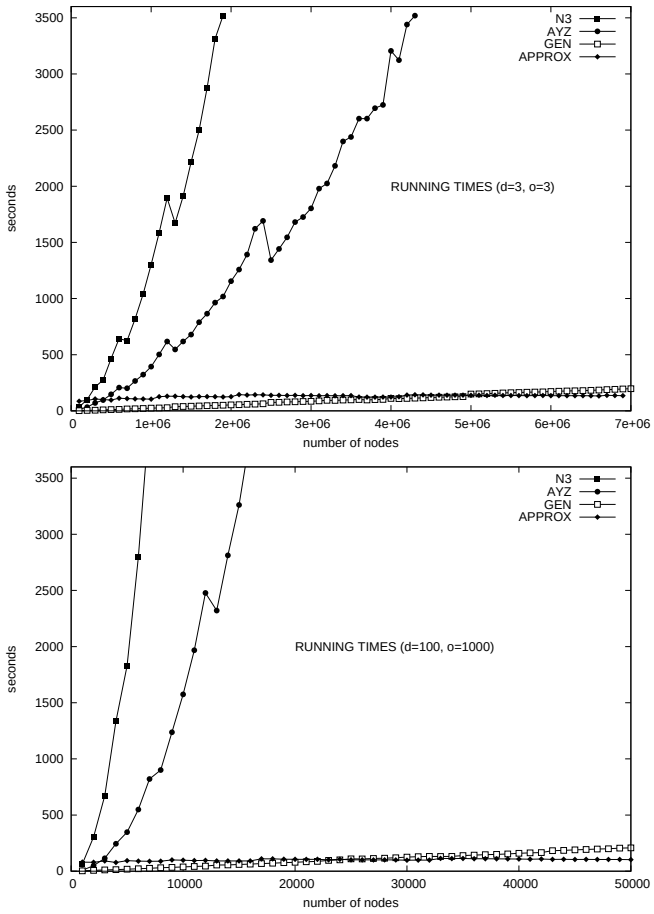

FIG. 2: Running-Times for the Generator

The plots in Fig. 2 show that APPROX is much faster compared to AYZ or N3 for the generated graphs for growing n. The difference is more pronounced for denser graphs (high d and o parameters). The abbreviation GEN is used for our implementation of the generator algorithm 2. It can be seen that APPROX becomes even faster then GEN at some point. This fits again to the mentioned sublinearity of APPROX.

There are some spikes in Fig. 2 which seem to repeat within a certain range of number of nodes. They actually happen within a fixed range of number of edges and are caused by rebuilding the internal data structure of the edge-hash-map.

Acknowledgments

We wish to thank Ulrik Brandes, Marco Gärtler and Christoph Gulden for fruitful discussions and for hinting to relevant work. We further wish to thank student-workers Matthias Broghammer and Lars Volkhard which were involved in the implementation of the algorithms.

APPENDIX A: A FAST GRAPH-GENERATOR WITH ADJUSTABLE CLUSTERING-COEFFICIENT

Algorithm 2: Graph-Generator

Input: initial graph G: two connected nodes
N
d
o
Output: graph G
for $3, \ldots, N$ **do**
  $v \leftarrow$ `NewNode( )`
  for $1, \ldots, Min(n, d)$ **do**
    repeat
      $u \leftarrow$ `RandomNode(`*with prob* $d_u / \sum_V d(v)$ `)`
    until *not* `EdgeExists(`v, u`)`
    `AddEdge(`v, u`)`
  if $d \geq 2$ **then**
    for $1, \ldots, o$ **do**
      $u \leftarrow$ `RandomAdjacentNode(`v`)`
      repeat
        $w \leftarrow$ `RandomAdjacentNode(`v`)`
      until $w \neq u$
      if *not* `EdgeExists(`u, w`)` **then**
        `AddEdge(`u, w`)`
return G

Our goal was to produce simple graphs according to the preferential attachment rule with non vanishing clustering-coefficient by a fast and simple algorithm. The running time of Alg. 2 is in $O(N(d+m))$. The final number of edges is in $\Omega(Nd)$ and in $O(Nd^2)$.

Fig. 3 shows the clustering-coefficient and transitivity for the generator with $d = 10$ and several values for o. The clustering-coefficient is adjustable where T seems to approach zero for $n \to \infty$.

APPENDIX B: THE HOEFFDING BOUND

Let X_i be independent real random variables bounded by $0 \leq X_i \leq M$ for all i. With P denoting the probability, E denoting the expectation, k the number of samples

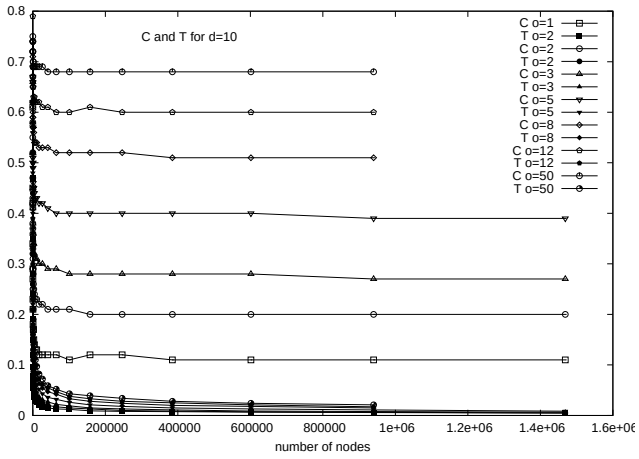

FIG. 3: C and T for the generator with $d = 10$ and several o

and ϵ some error bound Hoeffding's bound [Hoe63] states:

$$\mathrm{P}\left(\left|\frac{1}{k}\sum_{i=1}^{k} X_i - \mathrm{E}\left(\frac{1}{k}\sum_{i=1}^{k} X_i\right)\right| \geq \epsilon\right) \leq e^{\frac{-2k\epsilon^2}{M^2}} \qquad \text{(B1)}$$

This bound is refered to in the literature also as the Chernoff-Hoeffding or simply as one of the Chernoff bounds.

APPENDIX C: EXAMPLE GRAPH FAMILIES

The following graph families give examples for the divergence of D, C and T. For each example we consider the family of graphs for $n \to \infty$ and evaluate the parameters.

1. sparse, $C \to 0$ and $T \to 0$

The rings of n nodes and n edges.

2. sparse, $C \to 1$ and $T \to 1$

The graph family consisting of $\frac{n}{3}$ disconnected triangles.

3. sparse, $C \to 1$ and $T \to 0$

The graph family of $n + 2$ nodes as in Fig. 4. Here: $\delta = n$; $t = 2(n^2 - n)/2 + n = n^2$; $C = n/(n + 2)$, hence $D \to 0, C \to 1$ and $T \to 0$ for $n \to \infty$.

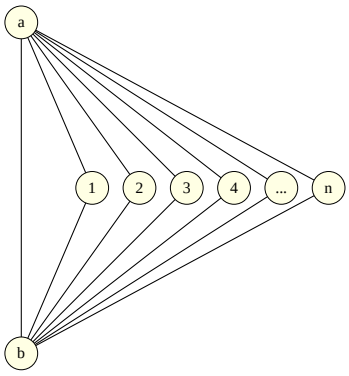

FIG. 4: sparse, $C \to 1$,$T \to 0$

4. sparse, $C \to 0$ and $T \to 1$

The graph family of $n = q + k$ nodes with q nodes as a clique and k nodes as a ring. Here: $\delta = \binom{q}{3}$ and $t = 3\binom{q}{3} + k$, hence $T = 3\binom{q}{3}/(3\binom{q}{3} + k)$; $C = q/(q + k)$; for $k \in \Theta(q^2)$ and $n \to \infty : m \in O(n)$, $C \to 0$, and $T \to 1$.

5. dense, $C \to 0$ and $T \to 0$

The complete bipartite graph with equal sized partitions.

6. dense, $C \to 1$ and $T \to 0$

The family of a equal-sized bipartition of b nodes and $a/3$ disconnected triangles. Here $T = a \Big/ \left(a + b\binom{b/2}{2}\right)$ and $C = a/(b + a)$; with $a \in \Theta(b \log b)$ gives $m \in \Omega(n^2/\log^2 n)$, $C \to 1$ and $T \to 0$.

7. dense, $C \to 0$ and $T \to 1$

Same graph-family as in App. C 4 but now with $k \in \Theta(q \log q)$ gives $m \in \Omega(n^2/\log^2 n)$, $C \to 0$ and $T \to 1$.

8. dense, $C \to 1$ and $T \to 1$

The complete graph.

[AYZ97] Noga Alon, Raphael Yuster, and Uri Zwick. Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997.

[BR02] Béla Bollobás and Oliver Riordan. Mathematical results on scale-free random graphs. In Stefan Bornholdt and Heinz Georg Schuster, editors, *Handbook of Graphs and Networks: From the Genome to the Internet*. Wiley-VCH, 2002.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Journal*, 58:13–30, 1963.

[NSW02] Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Random graph models of social networks. *Proceedings of the National Academy of Science of the United States of America*, 99:2566–2572, 2002.

[WS98] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.