HAL
archives-ouvertes.fr

On Computing the Hyperbolicity of Real-World Graphs

Michele Borassi, David Coudert, Pierluigi Crescenzi, Andrea Marino

▶ **To cite this version:**

Michele Borassi, David Coudert, Pierluigi Crescenzi, Andrea Marino. On Computing the Hyperbolicity of Real-World Graphs. 23rd Annual European Symposium on Algorithms (ESA), Sep 2015, Patras, Greece. Springer, 9294, pp.215-226, 2015, Lecture Notes in Computer Science. <10.1007/978-3-662-48350-3_19>. <hal-01199860>

HAL Id: hal-01199860
https://hal.inria.fr/hal-01199860

Submitted on 16 Sep 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On Computing the Hyperbolicity of Real-World Graphs*

Michele Borassi[1], David Coudert[2], Pierluigi Crescenzi[3] and Andrea Marino[4]

[1]IMT Institute for Advanced Studies Lucca, Italy
[2]Inria, France
[3]Dipartimento di Ingegneria dell'Informazione, Università di Firenze
[4]Dipartimento di Informatica, Università di Pisa

Abstract

The (Gromov) hyperbolicity is a topological property of a graph, which has been recently applied in several different contexts, such as the design of routing schemes, network security, computational biology, the analysis of graph algorithms, and the classification of complex networks. Computing the hyperbolicity of a graph can be very time consuming: indeed, the best available algorithm has running-time $\mathcal{O}(n^{3.69})$, which is clearly prohibitive for big graphs. In this paper, we provide a new and more efficient algorithm: although its worst-case complexity is $\mathcal{O}(n^4)$, in practice it is much faster, allowing, for the first time, the computation of the hyperbolicity of graphs with up to 200,000 nodes. We experimentally show that our new algorithm drastically outperforms the best previously available algorithms, by analyzing a big dataset of real-world networks. Finally, we apply the new algorithm to compute the hyperbolicity of random graphs generated with the Erdös-Renyi model, the Chung-Lu model, and the Configuration Model.

1 Introduction

In recent years, the analysis of complex networks has provided several significant results, with a huge amount of applications in sociology, biology, economics, statistical physics, electrical engineering, and so on. These results are based on the analysis of very big real-world networks, now made available by improvements in computer technology and by Internet. One of the major challenges in this field is to understand which properties distinguish these networks from other kinds of graphs, like random graphs [25], and which properties distinguish networks of different kinds [16], in order to classify general and particular behavior.

In this context, a significant role is played by the hyperbolic structure underlying a complex network, that is usually not present in random graphs [24, 6]. For instance, if we draw points from a hyperbolic space and we connect nearby points, we obtain a graph that shares many properties with real-world networks [21]. Furthermore, the Internet graph can be embedded in the hyperbolic space, preserving some metric properties [26, 4].

Consequently, researchers have tried to measure this hyperbolic structure of complex networks, using Gromov's definitions of hyperbolicity [15], which works in any metric space, and does not rely on complicated structures not available in graphs (geodesics, connections, and so

*This work has been supported in part by the Italian Ministry of Education, University, and Research under PRIN 2012C4E3KT national research project AMANDA (Algorithmics for MAssive and Networked Data), and by ANR project Stint under reference ANR-13-BS02-0007, ANR program "Investments for the Future" under reference ANR-11-LABX-0031-01.

on). Intuitively, this parameter reflects how the metric space (distances) of a graph is close to the metric space of a tree. In particular, given an undirected graph $G = (V, E)$ (in this paper, all graphs will be undirected), the Gromov hyperbolicity of a quadruple of nodes $\delta(x, y, v, w)$ is defined as half the difference between the biggest two of the three sums $d(x, y) + d(v, w)$, $d(x, v) + d(y, w)$, and $d(x, w) + d(y, v)$, where $d(\cdot, \cdot)$ denotes the distance function between two nodes, that is, the lenght of the shortest path connecting the two nodes. The hyperbolicity of G is $\delta(G) = \max_{x,y,v,w \in V} \delta(x, y, v, w)$ (the smaller this value, the more hyperbolic the space is).

Several network properties are connected to the value of the hyperbolicity: here we will just recall some of them. In [8], it is shown that a small hyperbolicity implies the existence of efficient distance and routing labeling schemes. In [23], the authors observe that a small hyperbolicity, that is, a negative curvature of an interconnection network, implies a faster congestion within the core of the network itself, and in [18] it is suggested that this property is significant in the context of network security and can, for example, mitigate the effect of distributed denial of service attacks. In [12], instead, the hyperbolicity is used to implement a distance between trees, successively applied to the estimation of phylogenetic trees. From a more algorithmic point of view, it has been shown that several approximation algorithms for problems related to distances in graphs (such as diameter and radius computation [7], and minimum ball covering [9]) have an approximation ratio which depends on the hyperbolicity of the input graph. Moreover, some approximation algorithms with constant approximation factor rely on a data-structure whose size is proportional to the hyperbolicity of the input graph [20]. More in general, the hyperbolicity is connected to other important graph quantities, like treelength [7] and chordality [30]. In the field of the analysis of complex networks, the hyperbolicity and its connection with the size and the diameter of a network has been used in [2] in order to classify networks into three different classes, that is, strongly hyperbolic, hyperbolic, and non-hyperbolic, and to apply this classification to a small dataset of small biological networks (a more extensive analysis of the hyperbolicity of real-world networks has been also recently done in [5]). In general, it is still not clear whether the hyperbolicity value is small in all real-world networks (as it seems from [19, 2]), or it is a characteristic of specific networks (as it seems from [1]). Finally, the hyperbolicity of random graphs has been analyzed in the case of several random graph models, such as the Erdös-Renyi model [24] and the Kleinberg model [6]. Moreover, in this latter paper, it is stated that the design of more efficient exact algorithms for the computation of the hyperbolicity would be of interest.

Indeed, it is clear that the hyperbolicity computation problem is polynomial-time solvable by the trivial algorithm that computes $\delta(x, y, v, w)$ for each quadruple of nodes. However, the running-time is $\mathcal{O}(n^4)$, where n is the number of nodes, which is prohibitive for real-world networks. The best known algorithm uses fast (max,min)-matrix multiplication algorithm to obtain a running time $\mathcal{O}(n^{3.69})$ [14], and it has been shown that hyperbolicity cannot be computed in $\mathcal{O}(n^{3.05})$ time, unless there exists a faster algorithm for (max,min)-matrix multiplication than currently known. Such running times are prohibitive for analysing large-scale graphs with more than 10,000 nodes.

Recently, new algorithms have been developed [11, 10]. Although these algorithms have worst-case running time $\mathcal{O}(n^4)$, they perform well in practice, making it possible to compute the hyperbolicity of graphs with up to 50,000 nodes.

In this paper, we propose a new algorithm to compute the hyperbolicity of a graph, taking some ideas from the algorithm in [11]. The new algorithm heavily improves the performances through significant speed-ups in the most time-consuming part. The speed-ups will be so efficient that the running-time of the new algorithm will be dominated by the pre-processing part, which needs time $\mathcal{O}(mn)$, where m is the number of edges (we assume the input graph to be connected, and consequently $m+n = \mathcal{O}(m)$). This way, the $\mathcal{O}(n^4)$ bottleneck is almost removed, at least in

practical instances. For this reason, we will be able for the first time to compute the hyperbolicity of graphs with up to 200,000 nodes. We will experimentally show these claims by analyzing a big dataset of real-world networks of different kinds. Finally, we apply our algorithm to the computation of the hyperbolicity of random graphs. In particular, in the Chung-Lu model, we compute the hyperbolicity of graphs with up to 200,000 nodes, improving previous experiments that stop at 1,100 nodes [13].

In Section 2, we will sketch the main features of the algorithm in [11], in order to make the paper self-contained. Section 3 will explain how to modify that algorithm, in order to obtain significant improvements, and Section 4 contains our experimental results. Finally, in Section 5, we apply our algorithm to the analysis of the hyperbolicity of random graphs, as suggested in [6], and Section 6 concludes the paper.

2 CCL: The Currently Best Available Algorithm

In this section, we will sketch the algorithm proposed in [11], whose main ideas and lemmas will also be used in the next section. This algorithm improves the trivial algorithm by analyzing quadruples in a specific order, and by cutting the exploration of the quadruples as soon as some conditions are satisfied. We will name this algorithm CCL, from the initials of the surnames of the authors. In particular, for each quadruple (p, q, r, s) of nodes, CCL computes $\tau(p, q; r, s)$ as defined below, instead of computing $\delta(p, q, r, s)$.

$$\tau(p, q; r, s) = \frac{d(p, q) + d(r, s) - \max\{d(p, r) + d(q, s), d(p, s) + d(q, r)\}}{2}.$$

Note that $\delta(G) = \max_{p,q,r,s \in V} \tau(p, q; r, s)$, because if $d(p, q) + d(r, s)$ is the maximum sum, then $\tau(p, q; r, s) = \delta(p, q, r, s)$, otherwise $\tau(p, q; r, s) < 0$.

Lemma 1 (Lemma 3.2 of [11]). *For any quadruple (p, q, r, s) of nodes, $2\tau(p, q; r, s) \leq \min(d(p, q), d(r, s))$.*

In order to exploit this lemma, CCL stores all the $N = \frac{n(n-1)}{2}$ pairs of nodes inside a sequence $\mathtt{P} = (\{x_1, y_1\}, \ldots, \{x_N, y_N\})$, in decreasing order of distance (that is, if $d(x_i, y_i) > d(x_j, y_j)$, then $i < j$). For each i, CCL iterates over all pairs $\{x_j, y_j\}$ with $j < i$, and computes $\tau(x_i, y_i; x_j, y_j)$, storing the maximum value found in a variable δ_L (clearly, δ_L is a lower bound for $\delta(G)$). Even if iterating over the whole sequence P would lead us to the trivial algorithm, by applying Lemma 1 we may cut the exploration as soon as $d(x_i, y_i) \leq 2\delta_L$, because the τ value of all remaining quadruples is at most $d(x_i, y_i)$.

A further improvement is provided by the following lemma.

Lemma 2 ([29]). *Let x, y, v, w be four nodes, and let us assume that there exists an edge (x, x') such that $d(x', y) = d(x, y) + 1$. Then, $\tau(x, y; v, w) \leq \tau(x', y; v, w)$.*

Definition 3. *A pair $\{x, y\}$ is far apart if there is no edge (x, x') such that $d(x', y) = d(x, y) + 1$ and no edge (y, y') such that $d(x, y') = d(x, y) + 1$.*

By Lemma 2, CCL only needs to analyze far apart pairs, and, hence, in the following we will denote by P (respectively, N) the list (number) of far apart pairs. The pseudo-code of CCL is provided in Algorithm 1.

Other improvements of this algorithm involve pre-processing the graph: first of all, we may analyze each biconnected component separately [11, Section 2], then, we may decompose the graph by modular decomposition, split decomposition [29], and clique decomposition [10].

Algorithm 1: Hyperbolicity algorithm proposed in [11], CCL.

Let $\mathtt{P} = (\{x_1, y_1\}, \dots, \{x_N, y_N\})$ be the list of far apart pairs, in decreasing order of distance.
$\delta_L \leftarrow 0$;
for $i \in [1, N]$ **do**
  if $d(x_i, y_i) \leq 2\delta_L$ **then**
    return δ_L;
  for $j \in [1, i-1]$ **do**
    $\delta_L \leftarrow \max(\delta_L, \tau(x_i, y_i; x_j, y_j))$;
return δ_L;

3 HYP: The New Algorithm

In this section, we propose a new algorithm, that we will call HYP, that improves upon Algorithm 1 by further reducing the number of quadruples to consider.

3.1 Overview

The new algorithm HYP speeds-up the inner `for` loop in Algorithm 1, by decreasing the number of pairs to be analyzed. In particular, let us fix a pair (x_i, y_i) in the outer `for` loop and a lower bound δ_L: a node v is (i, δ_L)-*skippable* or simply *skippable* if, for any w, $\tau(x_i, y_i; v, w) \leq \delta_L$. It is clear that if a node v is skippable, the algorithm could skip the analysis of all quadruples containing x_i, y_i, and v. Even if it is not easy to compute the set of skippable nodes, we will define easy-to-verify conditions that imply that a node v is skippable (Section 3.2): a node not satisfying any of these conditions will be named (i, δ_L)-*acceptable* or *acceptable*. Our algorithm will then discard all quadruples (x_i, y_i, v, w) where either v or w is not acceptable.

Furthermore, we will define another condition such that if $\tau(x_i, y_i; v, w) > \delta_L$, then either v or w must satisfy this condition (an acceptable node also satisfying this condition will be defined (i, δ_L)-*valuable* or *valuable*). Hence, our algorithm will not only discard all quadruples (x_i, y_i, v, w) where either v or w is not acceptable, but also all quadruples where both v and w are not valuable.

In order to apply these conditions, when analyzing a pair (x_i, y_i), HYP computes the set of acceptable and valuable nodes in time $\mathcal{O}(n)$ (actually, several nodes are skipped, thanks to implementation tricks, so that the time might be much smaller). Then, for each valuable node v, it analyzes pairs (v, w) preceding (x_i, y_i) such that w is acceptable. For this latter loop, we record for each node v the list $\mathtt{mate}[v]$ of previously seen pairs (v, w), and then test each time if w is acceptable. The pseudo-code for HYP is provided by Algorithm 2.

Lemma 4. *The algorithm is correct.*

Proof. First of all, $\delta_L \leq \delta(G)$ during the whole algorithm, so we only have to rule out the possibility that the output is strictly smaller than $\delta(G)$. Let x, y, v, w be a quadruple such that $\tau(x, y; v, w) = \delta(G)$. We may assume without loss of generality that $\{x, y\}$ and $\{v, w\}$ are far-apart (otherwise, we change the pairs using Lemma 2), and that $\{v, w\}$ is before $\{x, y\}$ in the ordering of pairs (otherwise, we swap the pairs). By Lemma 1, $d(x, y) \geq 2\delta(G) \geq 2\delta_L$ at any step of the algorithm: if $2\delta_L = d(x, y) \geq 2\delta(G)$ at some step, the algorithm is correct because δ_L never decreases. Otherwise, the pair $\{x, y\}$ is analyzed at some step i, v and w will be (i, δ_L)-acceptable, and either v or w will be (i, δ_L)-valuable (by definition of acceptable and valuable). Hence, in the inner loop, $\tau(x, y; v, w)$ is computed, and afterwards $\delta_L = \tau(x, y; v, w) = \delta(G)$. □ □

Algorithm 2: The new algorithm, HYP

```
Let P = ({x_1, y_1}, ..., {x_N, y_N}) be the ordered list of far apart pairs.
δ_L ← 0;
mate[v] ← ∅ for each v;
for i ∈ [1, N] do
    if d(x_i, y_i) ≤ 2δ_L then
        return δ_L;
    (acceptable, valuable) ← computeAccVal ();
    for v ∈ valuable do
        for w ∈ mate[v] do
            if w ∈ acceptable then
                δ_L ← max(δ_L, τ(x_i, y_i; v, w));
    add y_i to mate[x_i];
    add x_i to mate[y_i];
return δ_L
```

It remains to define which nodes are acceptable and which nodes are valuable, which is the topic of the following section.

3.2 Acceptable and Valuable Nodes

First of all, let us fix i and δ_L, since in this section they play the role of parameters. Moreover, for the sake of clarity, we will denote x_i and y_i simply by x and y. The following lemmas will provide conditions implying that v is skippable, that is, there is no pair $\{v, w\}$ appearing in P before $\{x, y\}$ such that $\tau(x, y; v, w) > \delta_L$. An acceptable node must not satisfy these conditions. The first lemma holds by definition of skippable.

Lemma 5. *If v does not belong to any (far-apart) pair $\{v, w\}$ before $\{x, y\}$ in P, then v is skippable.*

A second possibility to prove that a node is skippable is given by a simple corollary of the following lemma.

Lemma 6 ([11])**.** *For each quadruple of nodes (x, y, v, w), $\tau(x, y; v, w) \leq \min_{a,b \in \{x,y,v,w\}} d(a, b)$.*

Corollary 7. *If $d(x, v) \leq \delta_L$ or $d(y, v) \leq \delta_L$, then v is skippable.*

Proof. If the assumptions are satisfied, for each w, $\tau(x, y; v, w) \leq d(x, v) \leq \delta_L$, or $\tau(x, y; v, w) \leq d(y, v) \leq \delta_L$. □

The next lemmas make use of the notion of the eccentricity $e(v)$ of a node, defined as $\max_{w \in V} d(v, w)$.

Lemma 8. *If $2e(v) - d(x, v) - d(y, v) < 4\delta_L + 2 - d(x, y)$, then v is skippable.*

Proof. By contradiction, let us suppose that there exists a node w such that $\delta_L < \tau(x, y; v, w)$. Then, $2\delta_L + 1 \leq 2\tau(x, y; v, w) = d(x, y) + d(v, w) - \max(d(x, v) + d(y, w), d(x, w) + d(y, v)) \leq d(x, y) + d(v, w) - \frac{1}{2}(d(x, v) + d(y, w) + d(x, w) + d(y, v)) \leq d(x, y) + e(v) - \frac{1}{2}(d(x, v) + d(y, v)) - \frac{1}{2}d(x, y)$. By rearranging this inequality, we would contradict the hypothesis. □ □

Lemma 9. *If $e(v) + d(x, y) - 3\delta_L - \frac{3}{2} < \max\{d(x, v), d(y, v)\}$, then v is skippable.*

Proof. By contradiction, let us suppose that there exists a node w such that $\delta_L < \tau(x, y; v, w)$. By Corollary 7, $d(y, w) > \delta_L$, that is, $d(y, w) \geq \delta_L + \frac{1}{2}$. Consequently, $2\delta_L + 1 \leq 2\tau(x, y; v, w) = d(x, y) + d(v, w) - \max(d(x, v) + d(y, w), d(x, w) + d(y, v)) \leq d(x, y) + d(v, w) - d(x, v) - d(y, w) \leq d(x, y) + e(v) - d(x, v) - \delta_L - 1/2$. By exchanging the roles of x and y, we obtain $2\delta_L + 1 \leq d(x, y) + e(v) - d(y, v) - \delta_L - \frac{1}{2}$. These two inequalities contradict the hypothesis. □ □

Definition 10. *A node is acceptable if it does not satisfy the assumptions of Lemmas 5, 8 and 9 and Corollary 7.*

Remark 11. *Lemma 5 can be verified "on the fly", by keeping track of already-seen nodes. The other items are clearly verifiable in time $\mathcal{O}(1)$ for each node, and consequently the running-time of this operation is $\mathcal{O}\left(|\{v \in V : \exists\{v, w\} < \{x, y\}\}|\right)$, which is less than or equal to $\mathcal{O}(n)$.*

Remark 12. *A variation of* HYP *verifies on the fly Lemma 9 and not Lemma 5. At the beginning of the algorithm, for each node x, we pre-compute a list of all nodes v in decreasing order of $e(v) - d(x, v)$ (in time $\mathcal{O}(n^2 \log n)$). Then, when computing acceptable nodes, we scan the list corresponding to x, and we stop as soon as we find a node v such that $e(v) + d(x, y) - 3\delta_L - \frac{3}{2} < d(x, v)$. In this case, the running-time of this operation is $\mathcal{O}\left(|\{v \in V : e(v) + d(x, y) - 3\delta_L - \frac{3}{2} \geq d(x, v)\}|\right)$. Since we may swap the roles of x and y, at each step, our algorithm chooses between x and y the less central node, according to* closeness centrality *measure [3].*

The two remarks above correspond to two versions of our algorithm HYP, that we will call HYP1 and HYP2, respectively. Now we need to define valuable nodes, using the following lemma, which involves a given node c (formally, we would need to write c-valuable instead of valuable). All choices of c are feasible, but if c is "central", the running-time improves. We decided to set c as the most central node according to *closeness centrality* measure [3].

Lemma 13. *Let c be any fixed node, and, for any node z, let $f_c(z) := \frac{1}{2}(d(x, y) - d(x, z) - d(z, y)) + d(z, c)$. Then, for any two nodes v and w, we have $2\tau(x, y; v, w) \leq f_c(v) + f_c(w)$.*

Proof. We have that, $2\tau(x, y; v, w) = d(x, y) + d(v, w) - \max(d(x, v) + d(y, w), d(x, w) + d(y, v)) \leq d(x, y) + d(v, c) + d(c, w) - (d(x, v) + d(y, w) + d(x, w) + d(y, v))/2 = f_c(v) + f_c(w)$. The lemma is thus proved. □ □

As a consequence, if $2\tau(x, y; v, w) > 2\delta_L$, either $f_c(v) > \delta_L$ or $f_c(w) > \delta_L$. This justifies the following definition.

Definition 14. *An acceptable node v is c-valuable or valuable if $f_c(v) > \delta_L$.*

Hence, if $\tau(x, y; v, w) > \delta_L$, then at least one of v and w must be valuable.

Remark 15. *It is possible to compute if an acceptable node is valuable in time $\mathcal{O}(1)$, so there is no time overhead for the computation of valuable nodes.*

4 Experimental Results

In this section, we compare the best algorithm available until now [11] (CCL, whose pseudo-code is Algorithm 1), with the two versions of our new algorithm, denoted as HYP1 and HYP2 (using Remark 11 and Remark 12, respectively). Other available algorithms are the trivial algorithm, which is significantly outperformed by CCL in [11], and the algorithm in [14]. This latter is not practical, because it is based on fast matrix multiplication: indeed using $\mathcal{O}(n^3)$ matrix multiplication implementation we get the same time of the trivial algorithm. As far as we know, no other competitors are available.

Both CCL and our algorithm, in both versions HYP1 and HYP2, share the following preprocessing (see [11]):

- compute biconnected components to treat them separately;
- computing the distances between all pairs of nodes;
- computing and sorting the list `P` of all far-apart pairs.

All the operations above need time $\mathcal{O}(m \cdot n)$ and they will be not part of the comparison since they are common to all three algorithms. Our tests were performed on an AMD Opteron(TM) Processor 6276, running Fedora release 21. Our source code has been written in C and compiled with gcc 4.9.2 with optimization level 3. The code is available at `http://piluc.dsi.unifi.it/lasagne` and has also been included in the graph module of Sagemath [27].

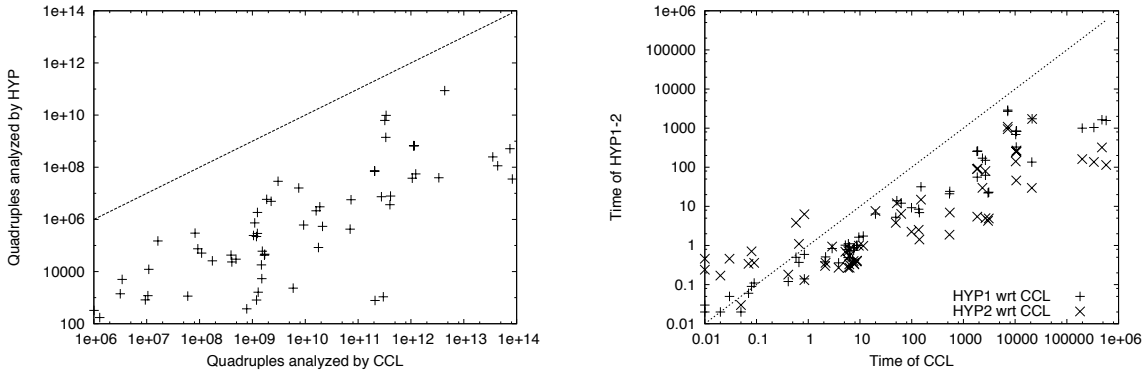
We have collected a dataset composed by 62 graphs (available with the code) of different kinds: social, peer-to-peer, autonomous systems, citation networks, and so on. The networks were selected from the well-known SNAP dataset (`http://snap.stanford.edu/`), and from CAIDA (`http://www.caida.org`). The number of nodes varies between 4,039 and 265,009 (1,396 and 50,219 after the preprocessing).

Number of quadruples. The first comparison analyzes how many quadruples are processed before the hyperbolicity is computed - note that HYP1 and HYP2 analyze the same number of quadruples, since the only difference between them is how acceptable and valuable nodes are computed. The results are summarized in Figure 1a, which plots the number of quadruples processed by the new algorithms with respect to CCL. More precisely, for each graph G, we draw a point in position (x, y) if CCL analyzed x quadruples and both HYP1 and HYP2 analyzed y quadruples to compute the hyperbolicity of G. We observe that the new algorithm analyzes a much smaller number of quadruples, ranging from one hundred to few millions, drastically outperforming CCL, which often analyzes millions of millions of quadruples, and even billions of millions. Of course, the new algorithm is never outperformed, because the quadruples analyzed by HYP1 and HYP2 are always a subset of the quadruples analyzed by CCL.

Running time. Since the computation of acceptable and valuable nodes has a non-negligible impact on the total running time, for a more fair comparison, we have also considered the running time of the algorithms. In Figure 1b we report the time used by HYP1 and HYP2 with respect to the time used by CCL. Also in this case, both HYP1 and HYP2 drastically outperform CCL: the running-time is lower in most of the graphs, and the only cases where CCL is faster need a very small amount of time (a few seconds at most). On the other hand, the new algorithms are much faster when the total time is big: for instance, on input `as-20120601.caida`, CCL needs at least one week (this lower bound was computed from the actual hyperbolicity and all the distances between the nodes), while HYP1 is 367 times faster, needing less than half an hour, and HYP2 is more than 5,000 times faster, needing less than two minutes. Similar results hold in all graphs where the total running-time is big. This does not only mean that we have improved upon CCL, but also that the improvement is concentrated on inputs where the running-time is high. Furthermore, we observe that on all graphs the total running time of algorithm HYP2 is less than half an hour: this means that, even if the worst-case complexity of this algorithm is $\mathcal{O}(n^4)$, in practice, the time used by the second part is comparable to the preprocessing time, which is $\mathcal{O}(m \cdot n)$. Hence, from a practical point of view, since real-world graphs are usually sparse, the algorithm may be considered quadratic.

5 Synthetic Graphs

Recently, a different line of research has tried to compute asymptotic values for the hyperbolicity of random graphs, when the number of nodes n tends to infinity. The simplest model considered



(a) Quadruples analyzed by HYP1 and HYP2 with respect to CCL.

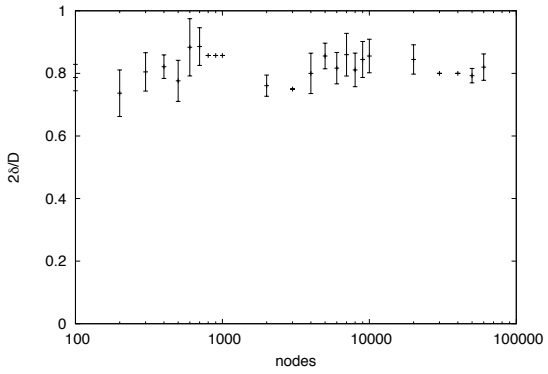
(b) Time used by HYP1 and HYP2 with respect to CCL.

Figure 1: Comparisons of quadruples analyzed and running-time of HYP1, HYP2, and CCL. The line $y = x$ separates the region where CCL is better (above) from the region where HYP1 and HYP2 are better (below).

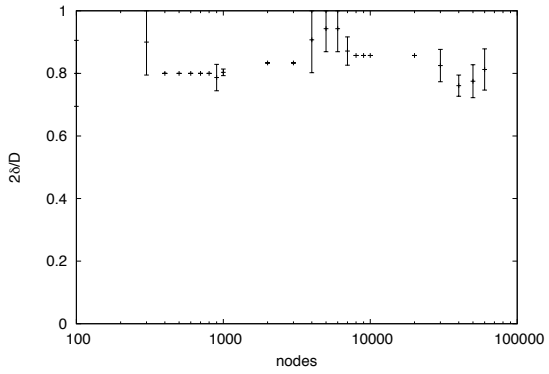
is the Erdös-Renyi random graph $G_{n,m}$, that is, we choose a graph with n nodes and m edges uniformly at random. In this model, it has been proved that the hyperbolicity tends to infinity [24], and, if m is "much bigger than n", exact asymptotics for δ have been computed [22]. Instead, the hyperbolicity of sparse Erdös-Renyi graphs is not known, and it is mentioned as an open problem in [22]. Among the other possible models, the Chung-Lu model and the Configuration Model stand out for their simplicity (for more background, we refer to [17]). On these models, as far as we know, it was only proved [28] that the hyperbolicity of a graph generated through the Chung-Lu model tends to infinity if the maximum and minimum degree are "close to each other" (meaning that their ratio is smaller than $2^{\frac{1}{3}}$). Other models were analyzed in [6]: also in that paper, the estimation of the hyperbolicity of random graphs of different kind is mentioned as an open problem.

Following [6], we use our algorithm to shed some light on the behavior of these random graphs, at least experimentally, in order to help formulating sensible conjectures on possible asymptotics. In particular, we have restricted our attention to four examples, chosen among the models where exact asymptotics have not been proved: Erdös-Renyi random graphs with $m = 3n$ and $m = 5n$, and graphs generated through the Chung-Lu and the Configuration Model, with power-law degree distribution with exponent 2.5 (similar to the degree distribution of several real-world networks [25]). For each number of nodes $n = k \cdot 10^i$ where $k < 10$ and $i \geq 2$, we have generated 10 graphs and we have computed their hyperbolicity. More precisely, we have computed the value $\frac{2\delta}{D}$, where D is the diameter, which is always between 0 and 1 because of Lemma 1: this value might be more interesting than the plain hyperbolicity value, since, for most models, asymptotics for the diameter are known. We believe that this ratio can then be used to formulate sensible conjectures. Figure 2 shows the average value of $\frac{2\delta}{D}$ and the corresponding standard error over the 10 measures performed.

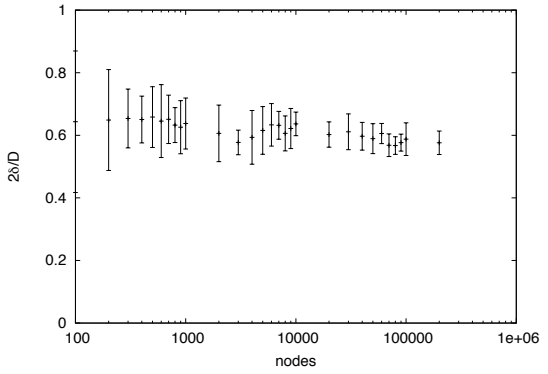
We have been able to compute the hyperbolicity of Erdös-Renyi graphs with up to $60,000$ nodes, and graphs generated with the Configuration Model or the Chung-Lu model with up to $200,000$ nodes. In all models considered, it is quite evident that the ratio $\frac{2\delta}{D}$ does not tend to 0, and consequently $\delta = \Theta(D)$. Furthermore, the ratio in Erdös-Renyi graphs is not very far from 1, even if the results are not precise enough to discriminate between $\delta = \frac{D}{2}$ or $\delta = cD$ for some $c < \frac{1}{2}$. Instead, in graphs generated through the Configuration Model or the Chung-Lu model, this ratio seems to tend to a value between 0.5 and 0.7.



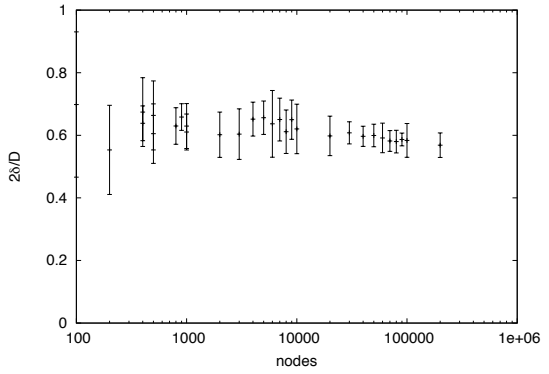
(a) Erdös-Renyi with average degree 3.



(b) Erdös-Renyi with average degree 5.



(c) Chung-Lu.



(d) Configuration Model

Figure 2: Mean and Standard Deviation of the ratio $\frac{2\delta}{D}$ at growing values of n.

6 Conclusion and Open Problems

In this paper, we have provided a new and more efficient algorithm to compute the hyperbolicity of a graph: even if the running time is $\mathcal{O}(n^4)$ in the worst case, it turns out to be $\mathcal{O}(m \cdot n)$ in practice. As an example of application, we have studied the hyperbolicity of random graphs. The space requirement of the algorithm, as well as of its predecessors, is $\mathcal{O}(n^2)$: in our case this is needed to store all distances and the list of far-apart pairs. It would be nice to better deal with memory usage (for instance, working on the disk) or avoiding the computation and storage of all pairwise distances by using lower and upper bounds instead. Furthermore, this algorithm may be parallelized, by analyzing at the same time different nodes v, or different pairs (x, y). An open issue is determining how parallelization can improve performances. The algorithm can be adapted to deal with weighted graphs. On the other hand, a widely accepted definition of hyperbolicity for directed graphs is still missing. Finally, it would be nice to prove more precise asymptotics for the hyperbolicity of random graphs.

References

[1] R. Albert, B. DasGupta, and N. Mobasheri. Topological implications of negative curvature for biological and social networks. *Physical Review E*, 89(3):032811, 2014.

[2] H. Alrasheed and F. F. Dragan. Core-Periphery Models for Graphs based on their delta-Hyperbolicity: An Example Using Biological Networks. *Studies in Computational Intelligence*, 597:65–77, 2015.

[3] A. Bavelas. Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America*, 22:725–730, 1950.

[4] M. Boguna, F. Papadopoulos, and D. Krioukov. Sustaining the Internet with Hyperbolic Mapping. *Nature Communications*, 1(62), Oct. 2010.

[5] M. Borassi, A. Chessa, and G. Caldarelli. Hyperbolicity Measures "Democracy" in Real-World Networks. *Preprint on arXiv*, pages 1–10, Mar. 2015.

[6] W. Chen, W. Fang, G. Hu, and M. Mahoney. On the hyperbolicity of small-world and treelike random graphs. *Internet Mathematics*, pages 1–40, 2013.

[7] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, and Y. Vaxès. Notes on diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs. *Electronic Notes in Discrete Mathematics*, 31:231–234, 2008.

[8] V. Chepoi, F. F. Dragan, B. Estellon, M. Habib, Y. Vaxès, and Y. Xiang. Additive Spanners and Distance and Routing Labeling Schemes for Hyperbolic Graphs. *Algorithmica*, 62(3-4):713–732, 2012.

[9] V. Chepoi and B. Estellon. Packing and covering δ-hyperbolic spaces by balls. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, volume 4627 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2007.

[10] N. Cohen, D. Coudert, G. Ducoffe, and A. Lancin. Applying clique-decomposition for computing Gromov hyperbolicity. Research Report RR-8535, HAL, 2014.

[11] N. Cohen, D. Coudert, and A. Lancin. On computing the Gromov hyperbolicity. *ACM J. Exp. Algor.*, 2015.

[12] A. Dress, K. Huber, J. Koolen, V. Moulton, and A. Spillner. *Basic Phylogenetic Combinatorics*. Cambridge University Press, Cambridge, UK, Dec. 2011.

[13] W. Fang. On Hyperbolic Geometry Structure of Complex Networks. Master's thesis, MPRI at ENS and Microsoft Research Asia, 2011.

[14] H. Fournier, A. Ismail, and A. Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *Information Processing Letters*, 115(6-8):576–579, 2015.

[15] M. Gromov. Hyperbolic groups. In *Essays in Group Theory*. Springer, 1987.

[16] S. Havlin and R. Cohen. *Complex networks: structure, robustness and function*. Cambridge University Press, Cambridge, 2010.

[17] R. V. D. Hofstad. Random Graphs and Complex Networks, 2014.

[18] E. A. Jonckheere and P. Lohsoonthorn. Geometry of network security. In *American Control Conference*, volume 2, pages 976–981, Boston, MA, USA, 2004. IEEE.

[19] W. S. Kennedy, O. Narayan, and I. Saniee. On the Hyperbolicity of Large-Scale Networks. *CoRR*, abs/1307.0031:1–22, 2013.

[20] R. Krauthgamer and J. R. Lee. Algorithms on negatively curved spaces. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 119–132, 2006.

[21] D. Krioukov, F. Papadopoulos, M. Kitsak, and A. Vahdat. Hyperbolic Geometry of Complex Networks. *Physical Review E*, 82(3):36106, 2010.

[22] D. Mitche and P. Pralat. On the hyperbolicity of random graphs. *The Electronic Journal of Combinatorics*, 21(2):1–24, 2014.

[23] O. Narayan and I. Saniee. The Large Scale Curvature of Networks. *Physical Review E*, 84:66108, Dec. 2011.

[24] O. Narayan, I. Saniee, and G. H. Tucci. Lack of Hyperbolicity in Asymptotic Erdös–Renyi Sparse Random Graphs. *Internet Mathematics*, pages 1–10, 2015.

[25] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, Jan. 2003.

[26] F. Papadopoulos, D. Krioukov, M. Boguna, and A. Vahdat. Greedy forwarding in scale-free networks embedded in hyperbolic metric spaces. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):15–17, 2009.

[27] The Sage Developers. *Sagemath open-source Mathematics Software (Version 6.9)*, 2015. `http://www.sagemath.org`.

[28] Y. Shang. Non-Hyperbolicity of Random Graphs with Given Expected Degrees. *Stochastic Models*, 29(4):451–462, Oct. 2013.

[29] M. A. Soto Gómez. *Quelques propriétés topologiques des graphes et applications à internet et aux réseaux*. PhD thesis, Univ. Paris Diderot (Paris 7), 2011.

[30] Y. Wu and C. Zhang. Hyperbolicity and chordality of a graph. *The Electronic Journal of Combinatorics*, 18(1):1–22, 2011.