Better Approximation of Betweenness Centrality*

Robert Geisberger† Peter Sanders† Dominik Schultes†

Abstract

Estimating the importance or centrality of the nodes in large networks has recently attracted increased interest. *Betweenness* is one of the most important centrality indices, which basically counts the number of shortest paths going through a node. Betweenness has been used in diverse applications, e.g., social network analysis or route planning. Since exact computation is prohibitive for large networks, approximation algorithms are important. In this paper, we propose a framework for unbiased approximation of betweenness that generalizes a previous approach by Brandes. Our best new schemes yield significantly better approximation than before for many real world inputs. In particular, we also get good approximations for the betweenness of unimportant nodes.

1 Introduction

One of the most important aspects of automatic analysis of networks is the computation of *centrality indices* that measure the importance of a node in some well defined way. Recently, the focus of attention in network analysis has shifted to the analysis of ever larger networks that are rapidly becoming available in such diverse areas as transportation networks (e.g., public transportation or road networks), social networks (e.g., friendship circles, recommendation networks, or citation networks), computer networks (e.g., the internet or peer-to-peer networks), or networks in bioinformatics (e.g., protein interaction networks).

In this paper we consider *betweenness centrality* [8, 1], which is one of the most frequently considered centrality indices. Our results might also be applicable to related concepts such as *stress centrality* [15] that are also based on counting shortest paths. Consider a weighted directed (multi)-graph $G = (V, E)$ with $n = |V|$, $m = |E|$. Let SP_{st} denote the set of shortest paths between source s and target t and $SP_{st}(v)$ the subset of SP_{st} consisting of paths that have v in their interior. Then, the betweenness centrality for node v is

$$c_B(v) := \sum_{s,t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{1.1}$$

where $\sigma_{st} := |SP_{st}|$ and $\sigma_{st}(v) := |SP_{st}(v)|$.

This definition counts the number of shortest paths through v, counting paths with alternatives only fractionally.

Our original motivation for considering betweenness was to identify sets of important nodes that can define a highway-node hierarchy [14], which is used for (dynamic) routing in road networks. For this application, the requirement is to process huge networks with many million nodes in a few minutes. In this context, we also need reasonable approximations for the betweenness of *all* nodes since we have to decide which of several neighboring unimportant nodes will make it to the first level of the hierarchy.

1.1 Related Work.

Brandes [4] gives an exact algorithm for computing betweenness of all nodes that is based on solving a single source shortest path problem (SSSP) from each node. An SSSP computation from s produces a directed acyclic graph (DAG) encoding all shortest paths starting at s. By backward aggregation of counter values, the contributions of these paths to the betweenness counters can be computed in linear time (Section 3 gives more details). Depending on the graph model, the exact algorithm takes time between $\Theta(nm)$ (e.g., for unit edge weights) and $\Theta\big(nm + n^2 \log(n)\big)$ (comparison based general edge weights). Although this is polynomial time, it is prohibitive for networks with many millions of nodes and edges. Bader and Madduri [2] present a massively parallel implementation of the exact algorithm that can handle a few million nodes.

Brandes and Pich [5] investigate how the exact algorithm can be turned into an approximation algorithm by extrapolating from a subset of k starting nodes (*pivots*), otherwise using the same aggregation strategy as the exact algorithm. Subsequently, we refer to this approximation algorithm as "Brandes' algorithm". A random sample of starting nodes turns out to work well. In particular, the randomized method yields an unbi-

*Partially supported by DFG grant SA 933/1-3.

†Universität Karlsruhe (TH), 76128 Karlsruhe, Germany, {robert.geisberger, sanders, schultes}@ira.uka.de

ased estimator[1] for betweenness. Unfortunately, this method produces large overestimates for unimportant nodes that happen to be near a pivot. For example, consider a degree-two node v connecting a degree-one node u to the rest of the network. If u is selected as a pivot, the betweenness of v is overestimated by a factor of n/k.

1.2 Our Contributions. Our main idea is to solve the problem described above by changing the scheme for aggregating betweenness contributions so that nodes do not unduly 'profit' from being near a pivot. Section 2 describes a general framework for this idea that yields unbiased estimators for betweenness.

Our framework also applies to a simpler variant of betweenness, *canonical-path betweenness centrality* $c_C(v)$, which we usually just call *canonical centrality*. We introduce this variant due to our original motivation of dealing with road networks, where shortest routes are 'almost unique'[2]. Note that some route planning methods even enforce unique shortest paths by perturbing the edge weights (e.g., [9]). Consequently, in case of canonical centrality, we consider only a single *canonical* shortest path between any source-target pair. Our approach does not require edge perturbation. It is sufficient that some deterministic tie breaking ensures that only a single shortest path is found.

Section 3 describes how to efficiently implement two instantiations of our framework—*linear scaling*, where the contribution of a sample depends linearly on the distance to the sample, and *bisection scaling*, where a sample only contributes 'on the second half' of a path. Linear scaling can be implemented using a slight variation of Brandes' original aggregation scheme. Bisection scaling requires a quite different approach and another level of random sampling. Section 4 reports on extensive experiments with a wide range of large graphs. The linear scaling is always better than [5]. (Sampled) bisection scaling is in many ways even better. In particular, it yields good approximation of the betweenness also for less important nodes already with a small number of pivots—an area in which the original method fails. Section 5 summarizes the results and outlines possible further improvements. For example, we have evidence that betweenness approximations can help to construct better highway-node hierarchies for road networks.

[1] That means the expectation of the estimated betweenness is the actual betweenness.

[2] Indeed, multiple shortest routes *do* appear in practice. However they usually share most edges so that in most cases, the term $\sigma_{st}(v)/\sigma_{st}$ is one or zero.

2 A Generalized Framework for Betweenness Approximation

Our estimator is parametrized by a *length function* $\ell : E \to \mathbb{R}$ on the edges[3] and a *scaling function* $f : [0,1] \to [0,1]$. For a path $P = \langle e_1, \ldots, e_k \rangle$ let $\ell(P) := \sum_{1 \le i \le k} \ell(e_i)$.

In each iteration, our algorithm performs one of $2n$ possible shortest path searches with uniform probability $1/2n$. Namely, forward search in $G = (V, E)$ from a pivot $s \in V$ ($|V| = n$ possibilities) or backward search from a pivot $t \in V$, i.e., a search from t in $(V, \{(v,u) : (u,v) \in E\})$ (another n possibilities). For each shortest path of the form

$$P = \langle \overbrace{s, \ldots, v}^{Q}, \ldots, t \rangle$$

found in this way, we define a *scaled contribution*

$$\delta_P(v) := \begin{cases} \frac{f(\ell(Q)/\ell(P))}{\sigma_{st}} & \text{for a forward search} \\ \frac{1-f(\ell(Q)/\ell(P))}{\sigma_{st}} & \text{for a backward search} \end{cases} .$$

Overall, v gets a contribution

$$\delta(v) := \delta_s(v) := \sum_{t \in V} \sum \{\delta_P(v) : P \in SP_{st}(v)\}$$

for a forward search, and

$$\delta(v) := \delta_t(v) := \sum_{s \in V} \sum \{\delta_P(v) : P \in SP_{st}(v)\}$$

for a backward search.

THEOREM 2.1. *$X := 2n\delta(v)$ is an unbiased estimator for $c_B(v)$, i.e., $E(X) = c_B(v)$.*

Proof. Summing over all $2n$ possible events with probability $1/2n$ each, we get

$$\begin{aligned} E(X) &= 2n \frac{\sum_{s \in V} \delta_s(v) + \sum_{t \in V} \delta_t(v)}{2n} \\ &= \sum_{s,t \in V} \sum \left\{ \frac{\overbrace{f\left(\frac{\ell(Q)}{\ell(P)}\right) + 1 - f\left(\frac{\ell(Q)}{\ell(P)}\right)}^{=1}}{\sigma_{st}} : P \in SP_{st}(v) \right\} \\ &= \sum_{s,t \in V} \frac{\overbrace{|SP_{st}(v)|}^{=\sigma_{st}(v)}}{\sigma_{st}} \overset{(1.1)}{=} c_B(v) \ . \end{aligned}$$

□

[3] ℓ may or may not be identical to the edge-weight function used for shortest-path calculations.

Hence, by averaging k independent runs of the above unbiased estimator, we obtain an approximation $\frac{X_1+\ldots+X_k}{k}$ of the betweenness value of node v.

In the following, we will instantiate the length function ℓ either with the original edge weight or with unit edge weight (hop counting). We will consider three variants for the choice of f. For constant $f(x) = 1/2$ we essentially get Brandes' algorithm.[4] Our new schemes use $f(x) = x$ for *linear scaling* and

$$f(x) = \begin{cases} 0 & \text{for } x \in [0, 1/2) \\ 1 & \text{for } x \in [1/2, 1] \end{cases}$$

for *bisection scaling*. The intuition behind these methods is to reduce the contributions for nodes close to the pivot. In a sense, bisection scaling is best for this purpose. However, it will turn out that it is easier to implement linear scaling.

3 Linear Time Computation of Contributions

A naive implementation of the definitions from Section 2 could take quadratic time even for a single pivot and canonical centrality. We will give algorithms that evaluate the shortest path DAG in time linear in its size. We only explain the computations for forward search from source s. Backward search works analogously.

3.1 Brandes' Algorithm. Brandes [4] already explains how to compute σ_{st} on the fly during the shortest path calculations: $\sigma_{ss} = 1$ and, for $s \neq t$, $\sigma_{st} = \sum_{v \in pred(t)} \sigma_{sv}$ where $pred(t)$ is a multiset containing the immediate predecessors of t in the shortest path DAG.[5] In a subsequent aggregation phase, the nodes are processed in reverse topological order, i.e., by non-increasing distance from s. We get,

$$\delta_s(v) = \sum_{w \in succ(v)} \frac{\sigma_{sv}}{\sigma_{sw}}(1 + \delta_s(w))$$

where $succ(v)$ denotes the immediate successors of v in the shortest path DAG.[6] The factor $\frac{\sigma_{sv}}{\sigma_{sw}}$ takes care of distributing the contributions from w to possible multiple parents. The '1' is the contribution due to paths of the form $\langle s, \ldots, v, w\rangle$ and the $\delta_s(w)$ takes care of nodes reached indirectly via w.

3.2 Linear Scaling is easiest to implement by using the original edge weights as the length function ℓ. Then it is possible to implement it with only small deviations from Brandes' scheme. We have

$$\delta_s(v) = \sum_{w \in succ(v)} \frac{\mu(s,v)}{\mu(s,w)} \cdot \frac{\sigma_{sv}}{\sigma_{sw}}(1 + \delta_s(w))$$

where $\mu(s, w)$ is the shortest path distance from s to w. Less formally, the modification compared to Brandes' scheme is to put values $1/\mu(s, w)$ into the aggregation rather than 1. At the end, multiplying with $\mu(s, v)$ yields the correct scaling of $\mu(s, v)/\mu(s, w)$ for all values put into the aggregation.

3.3 Bisection Scaling For Canonical Centrality. Here, we mostly use unit distances for the length function ℓ. (Recall that ℓ is not necessarily the same as the edge weight.) We do aggregation by a depth-first traversal of the shortest-path tree. This allows us to keep an array storing the path from s to the currently explored node. Aggregation works similarly to Brandes' algorithm with one major modification. When a node v at depth d is explored, let v' denote the node on position $\max(0, \lfloor d/2 \rfloor - 1)$. We decrement the current value for $\delta_s(v')$. This has the effect of dropping the contribution of v from the aggregation just where it is prescribed by the scaling function f. We have also implemented a variant for general length functions. Here, we have to search the stack for $\ell(\langle s, \ldots, v\rangle)/2$ starting at the position used for the predecessor of v. Although this has linear worst case complexity, it works reasonably well for road networks using travel time.

3.4 Bisection Scaling For General Betweenness Centrality. We have a direct implementation of *enumerative bisection scaling* that enumerates all paths by backtracking. This works well for graphs with few redundant paths but can be very slow in the worst case. What is more interesting is that the general case can be reduced to the canonical case: we randomly sample a parent pointer for each node t in the shortest path DAG. We call this variant *bisection sampling*. If a parent p of w is selected with probability $\frac{\sigma_{sp}}{\sigma_{sw}}$, we get an unbiased estimator. We can extract more information out of the shortest path DAG by performing several sampling steps for the same DAG, which does not invalidate the fact that the estimator is unbiased.

[4] One can also do just forward searches in this case.

[5] Since there may be an exponential number of paths, this recurrence might lead to arithmetic overflows if one would use integer arithmetics on machine words. However, we are only interested in approximations, so that it suffices to compute with floating point numbers with mantissas of logarithmically many bits. Our implementation uses double precision arithmetics that flags overflows as ∞-values. This never happened yet for the inputs we considered.

[6] In our framework with backward searches we have to divide by two at the end.

graph	nodes	edges	source	average time per pivot [ms]
Belgian road network	463 514	596 119	PTV AG	1 337
Belgian road network with unit distance	463 514	596 119	PTV AG	914
Actor co-starring network	392 400	16 557 451	[12]	6 242
US patent network	3 774 769	16 518 947	[10]	5
World-Wide-Web graph	325 729	1 497 135	[12]	144
CNR 2000 Webgraph	325 557	3 216 152	[11]	362
CiteSeer undir. citation network	268 495	2 313 294	[6]	1 711
CiteSeer co-authorship network	227 320	1 628 268	[6]	835
CiteSeer co-paper network	434 102	32 073 440	[6]	4 110
DBLP co-authorship network	299 067	1 955 352	[7]	1 323
DBLP co-paper network	540 486	30 491 458	[7]	5 024

Table 1: Overview of used graphs. A co-paper network has papers as nodes and edges between papers that share at least one author. The values in the last column refer to general betweenness centrality and Brandes' algorithm.

4 Experiments

The experiments were done on one core of a single AMD Opteron Processor 270 clocked at 2.0 GHz with 8 GB main memory and 2 × 1 MB L2 cache, running SuSE Linux 10.0 (kernel 2.6.13). The program was compiled by the GNU C++ compiler 4.0.2 using optimization level 3.

We have approximated canonical centrality for several real-world road networks with up to 42 million edges using travel times as edge weights. We do most quality evaluations here with a subnetwork for Belgium with 463 514 nodes because for this we can still compute exact values.[7]

For general betweenness we used networks with unit edge weights stemming from a variety of applications including citation networks, coauthorship networks, web graphs and communication networks (AS graph, router level graphs). Table 1 gives additional information on these networks. Our focus was on large, real-world graphs where we can still compute exact betweenness in reasonable time. This includes the largest graphs from [2]. We do not use the graphs considered in [5] since they are mostly randomly generated and comparatively small. The evaluation here uses the most difficult of these instances[8]—a movie actor multigraph whose edges indicate costarring in some movie. This graph has 392 400 nodes, 16 557 451 edges, and many shortest paths between most pairs of nodes.

For assessing the quality of the estimation, we adopt the measures proposed by Brandes and Pich [5]—Euclidean distance of the normalized n-dimensional vectors for exact and estimated centrality, respectively, and the number of inversions[9] when comparing the rankings produced by exact and estimated centrality, respectively. The number of inversions is computed using a mergesort-based $\mathcal{O}(n \log n)$ algorithm. The Euclidean distance is mostly governed by the nodes with large betweenness, whereas the number of inversions treats all nodes equally. We have also looked at other measures. For example, the *average absolute betweenness error* turned out to give similar results as the Euclidean distance[10], whereas the *geometric mean of relative rank errors*[11] has similar characteristics as the number of inversions.

Since our new algorithms need slightly more time than Brandes' algorithm for evaluating the shortest path DAG, we ensure a fair comparison by giving our algorithms no more *time* than Brandes' algorithm; thus, they will perform less iterations. Effectively, the time needed by Brandes' algorithm for one pivot is our unit of time.

Figure 1 evaluates the approximation of canonical centrality for Belgium. With respect to the Euclidean distance, all algorithms show benign and predictable behavior. Still, our new methods fare uniformly better with bisection slightly ahead. In the same running time we get an about two times smaller distance or alternatively, we achieve about the same quality in eight times less running time. With respect to the number of inversions, the difference is much more dramatic. Even after

[7] Computing exact values for our complete Western European road network would take about 12 years.

[8] The US patent network is much bigger but relatively easy to solve exactly since most searches reach only a small number of nodes.

[9] i.e., the total number of pairs that are in the wrong order.

[10] Note that the normalization is likely to have little effect since all the compared methods are unbiased and hence the computed vectors are likely to have similar length anyway.

[11] Let r denote the ratio between the estimated rank and the true rank. Then the relative rank error is $\max\{r, 1/r\}$.

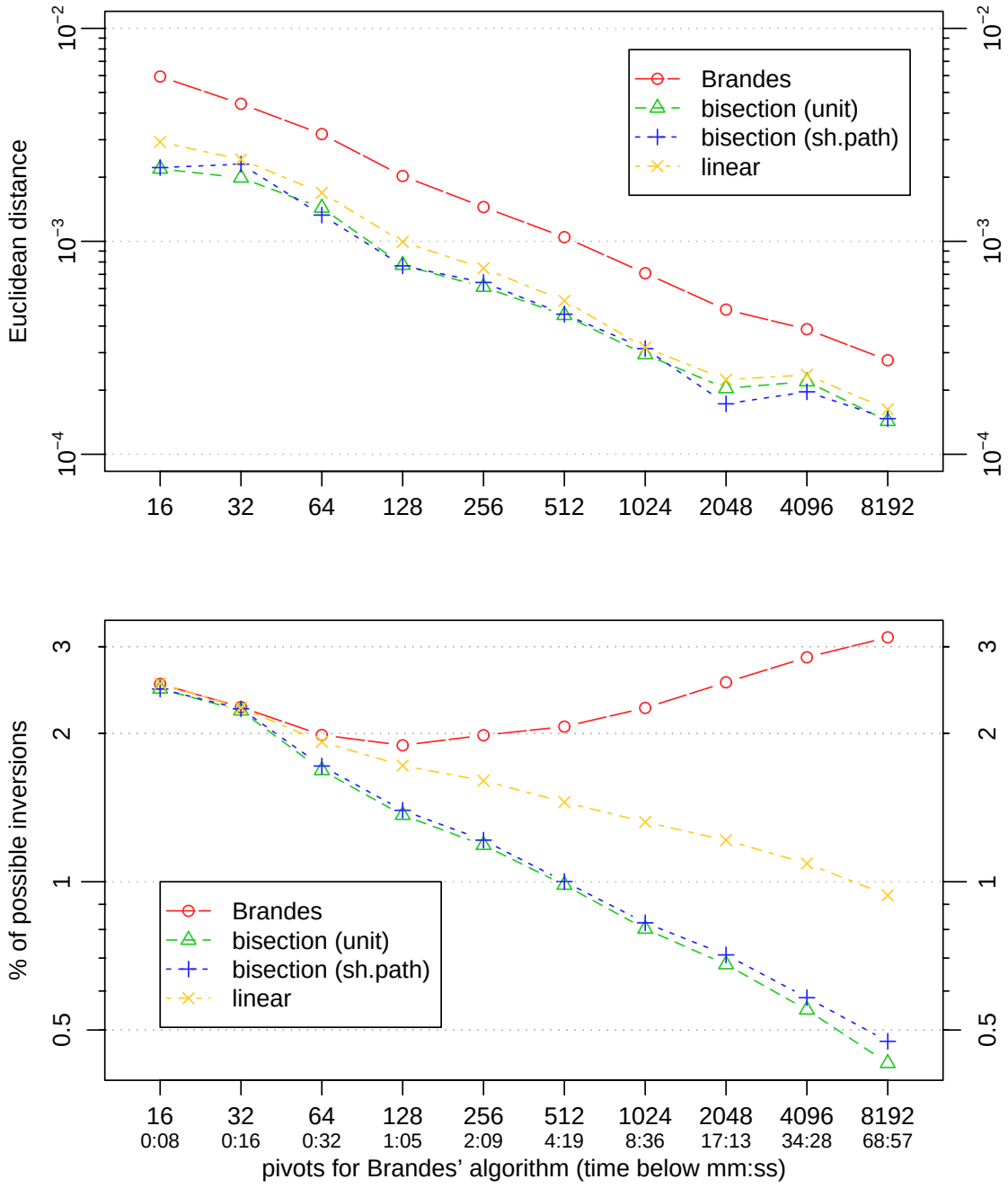

Figure 1: Euclidean distance and inversions for canonical centrality of Belgium.

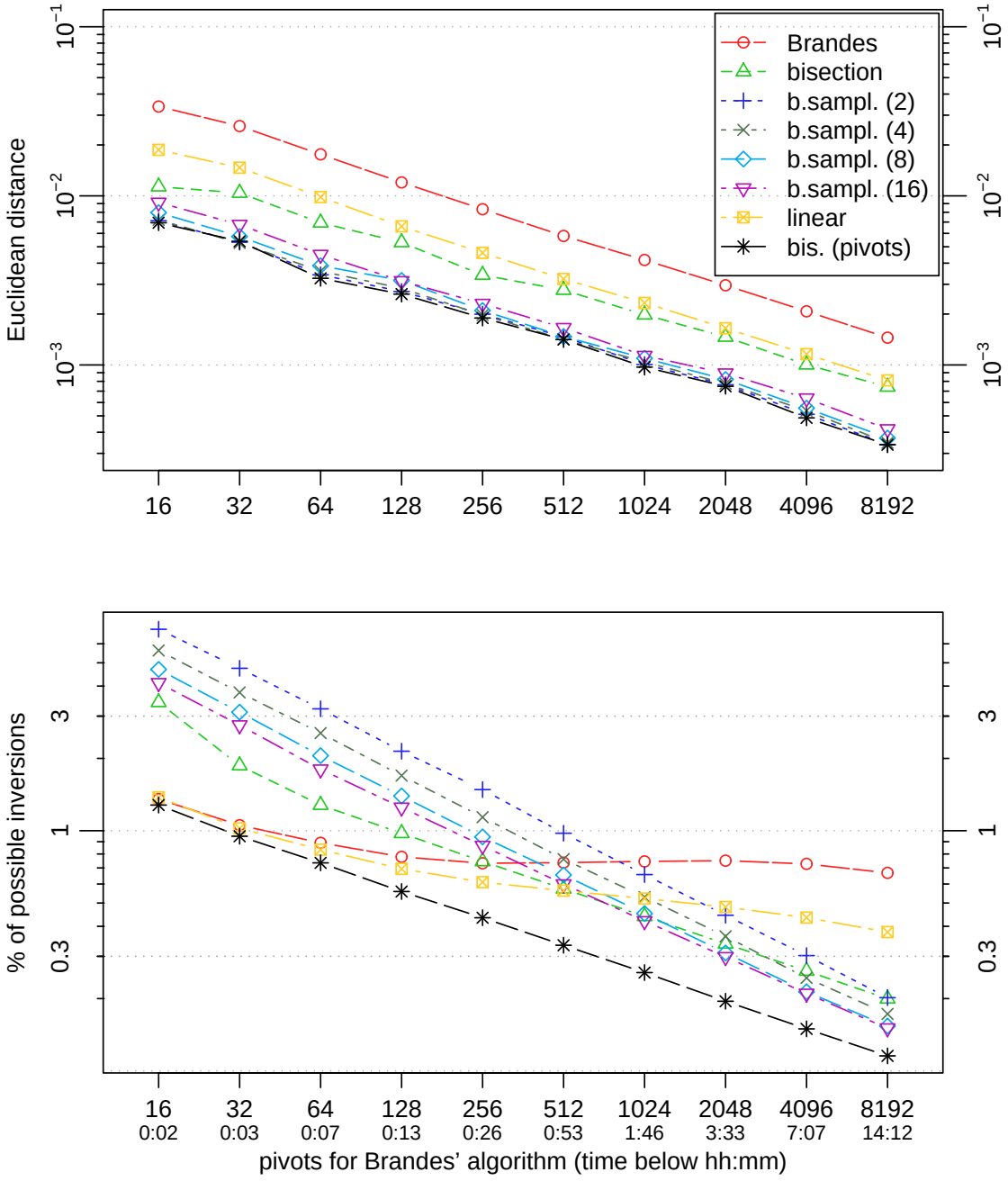

Figure 2: Euclidean distance and inversions for betweenness in the actor network.

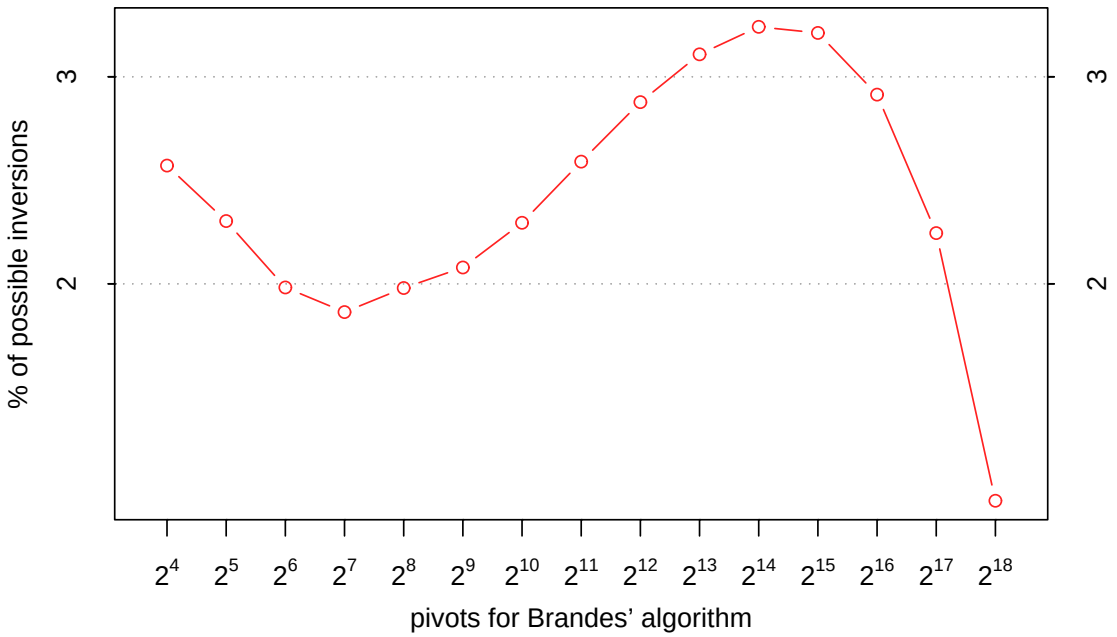

Figure 3: Inversions, canonical centrality of Belgium, Brandes' algorithm.

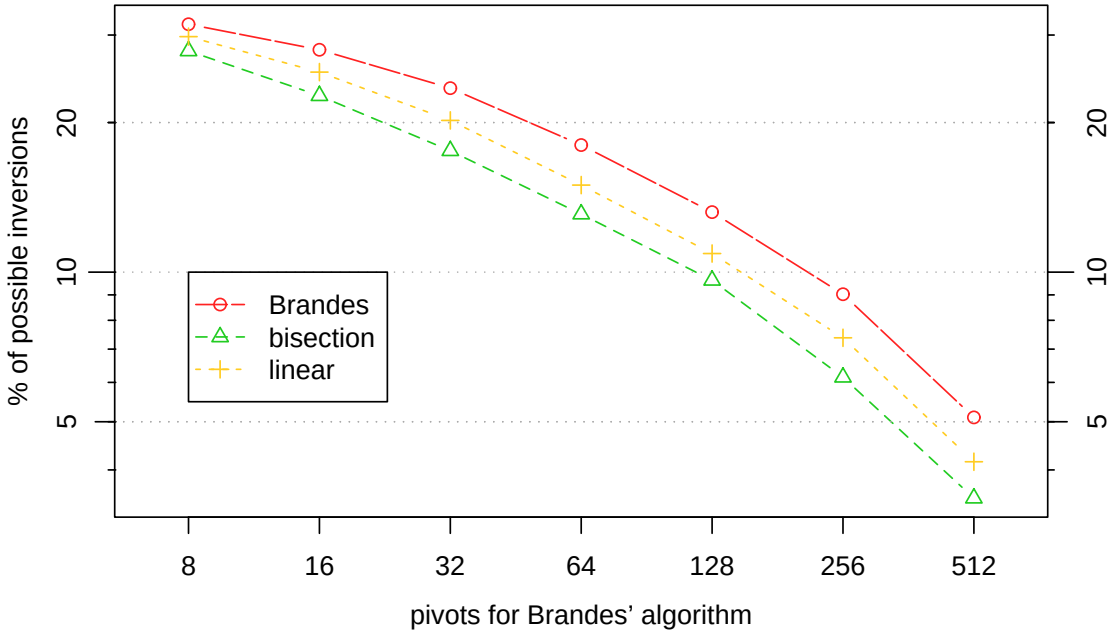

Figure 4: Inversions, canonical centrality of a random graph with 1 000 nodes and 10 074 edges.

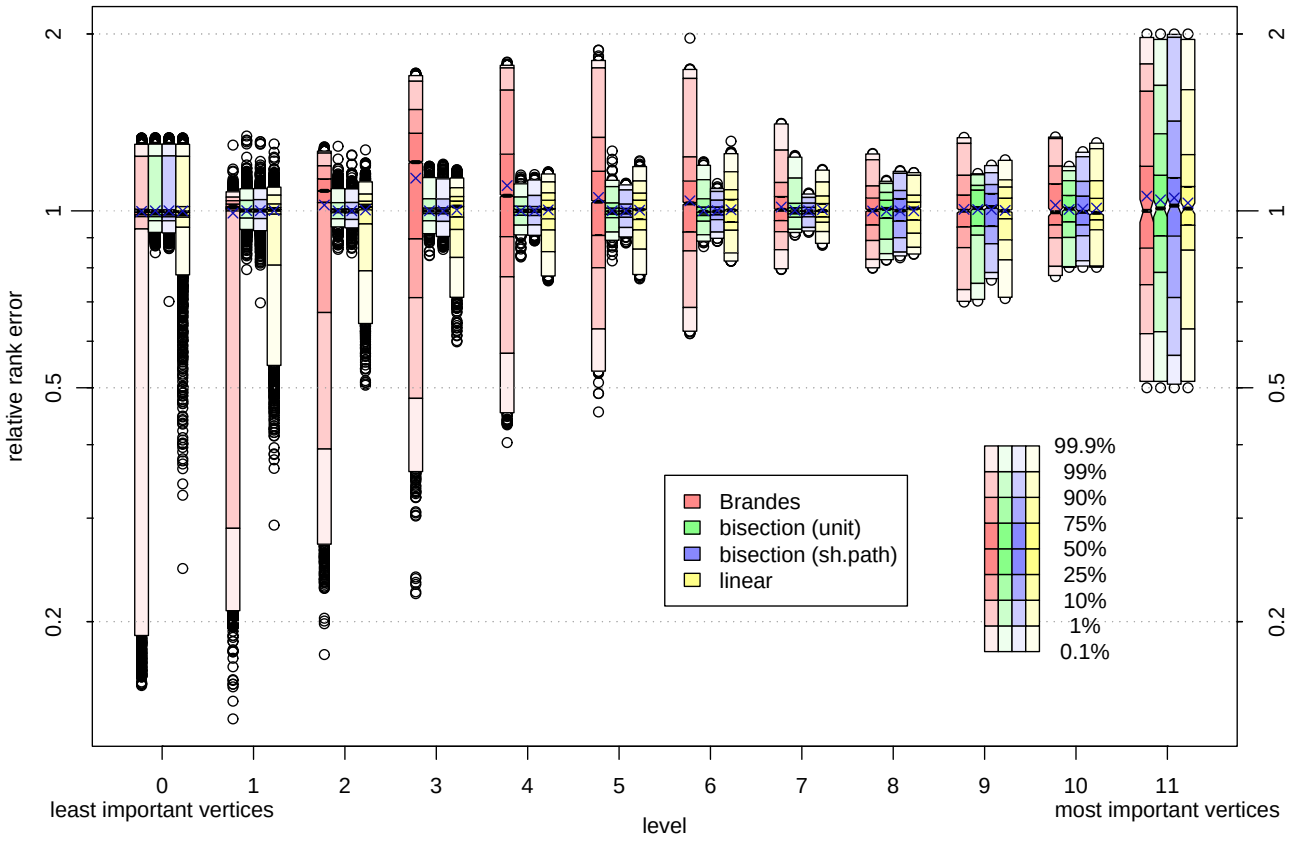

Figure 5: Distribution of the relative rank errors for 12 different levels (categories) of importance in the Belgian road network after 1024 iterations of Brandes' algorithm. Level 11 contains the 128 nodes with largest exact canonical centrality, Level 10 the next 256 most important nodes, and so on. The cross gives the average value. The circles denote outliers.

8192 iterations, Brandes' algorithm does not even begin to converge.[12] This behavior is unexpected since the random graphs used in [5] are much more 'well behaved' (see also Figure 4). A possible explanation is that the fate of most unimportant nodes is that their centrality is initially *slightly* underestimated. However, the more pivots are used, the more unimportant nodes near the pivots become *grossly* overestimated. The net effect on the number of inversions is positive. Our new algorithms show a nice, near linear behavior on a double-logarithmic plot. Bisection scaling clearly outperforms linear scaling. To understand what is going on, let us analyze in more detail how approximation errors are distributed over the nodes. Figure 5 illustrates the distribution of the relative rank errors for 12 different levels (categories) of importance. We can see that bisection scaling gives uniformly good approximation quality over all levels, while Brandes' algorithm has orders of magnitude larger errors for the less important levels, which constitute the majority of the nodes. All methods have comparatively large errors for the highest level of importance. The reason is that the betweenness values for these nodes are very similar so that already small errors in the absolute value can lead to large deviations in relative ranks.

Figure 2 evaluates betweenness approximation for the actor network. For the Euclidean distance, we get similar results as before. Interestingly, enumerative bisection fares quite well, although it is about four

[12] Figure 3 indicates that the numbers, in the very end, *do* go down.

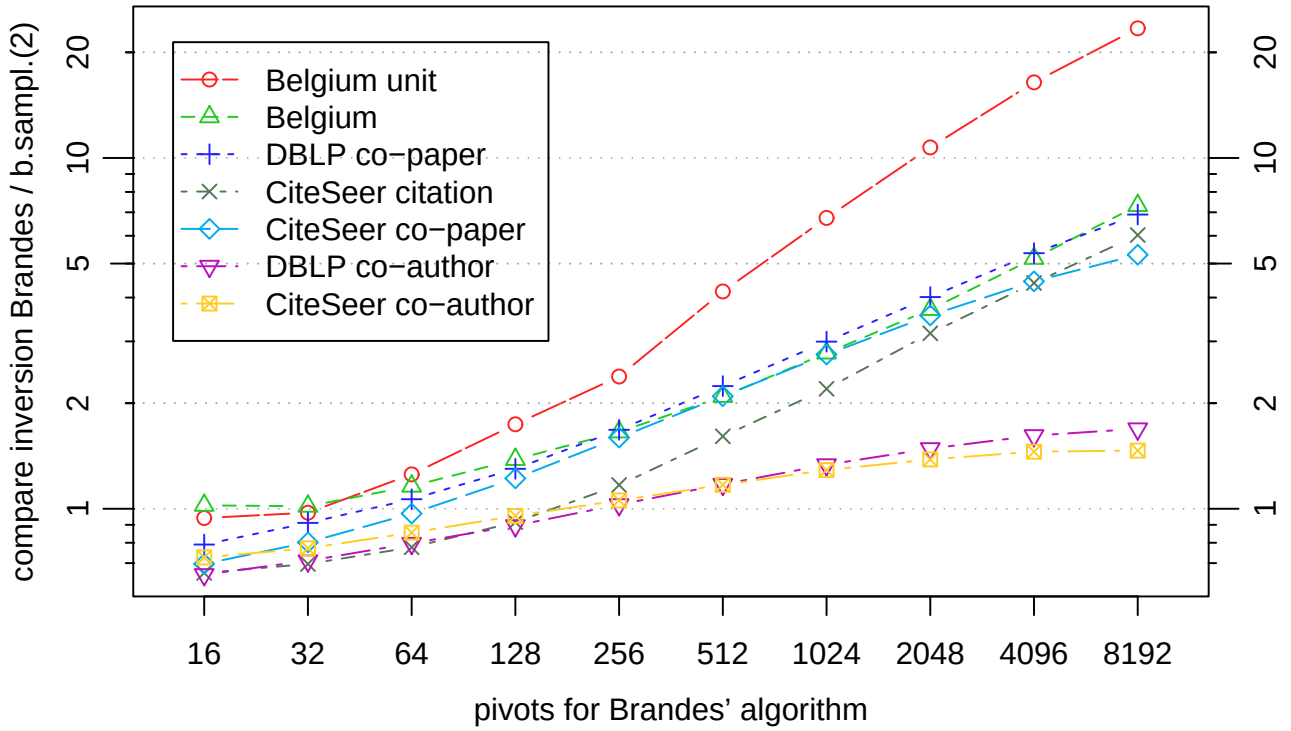

Figure 6: Number of inversions for Brandes' algorithm divided by number of inversions for bisection scaling with two samples for different networks.

times slower per iteration than the other methods. In particular, it outperforms linear scaling. Bisection sampling fares best. It has about four times smaller errors and needs about 16 times less time to achieve the same error bound as Brandes' algorithm with 8192 iterations. The number of sampled trees per pivot does not have a big influence on performance.

For the inversion number, the ranking of the algorithms is less clear. The linear scaling algorithm always outperforms Brandes' algorithm, which stagnates for many iterations before finally heading down after thousands of iterations. However, both Brandes and linear scaling have a better start than the other variants. Only after 1024 iterations, the bisection based algorithms start to take over. The main difference to the more clear-cut ranking for road networks seems to be the presence of highly redundant shortest paths. In this situation, bisection sampling throws away information that is essential if only few pivots are used. An analysis of a level plot similar to Figure 5 reveals that bisection sampling outperforms Brandes' algorithm on most levels but has deficits for the least important levels. Still, bisection sampling has a nice near-linear decrease of the inversions on a double-logarithmic plot that eventually, after a sufficient number of iterations, leads to superior performance. It is also interesting to see what would happen if we had a fast algorithm for exploiting all the information available. The lowest curve in Figure 2 plots the performance of our enumerative bisection algorithm based on just counting iterations of the outer loop. We see that this hypothetical algorithm outperforms all other algorithms from the beginning.

We see a similar pattern for many of the other networks. Figure 6 shows the improvement yielded by bisection sampling over Brandes' algorithm. For few pivots, Brandes' algorithm is up to a factor of two better. For many pivots, bisection sampling is up to a factor 20 better. With respect to the Euclidean distance, bisection sampling is always better for all of these networks. The only deviation from this pattern we found were the directed networks from Table 1—the US patent network and the web graphs. Here,

all approximation algorithms had problems to achieve good approximation. Linear scaling was always slightly better than Brandes' algorithm. Bisection sampling was sometimes better, sometimes worse. Shortest path searches turned out to be very local. This means that all approximation algorithms have trouble eliminating false zeroes for nodes of small betweenness. This is a disadvantage for bisection sampling since it is slower and since it produces zero contributions where Brandes and linear scaling produce positive contributions. On the other hand, the same effect makes these networks relatively easy for the *exact* algorithm. For example, our implementation solves the US patent graph in 127 minutes—only 2.5 times more time than Bader and Madduri [2] need using 16 IBM-P5 processor cores.

5 Conclusion and Future Work

Our new bisection scaling algorithm is a versatile method for approximating canonical centrality that works well even for huge networks. The algorithm has considerably better performance than previous methods for all graphs tried and for all global quality measures. This good evaluation largely extends to the original definition of betweenness and the sampled bisection algorithm. However, for a small number of pivots, the linear scaling variant of our framework is still better. Enumerative bisection might be the overall winner here if we found a near linear time implementation.

All algorithms discussed are easy to parallelize with up to k processors. We have not yet attempted any pragmatic tuning of the algorithms. For example, by using robust statistics (e.g., omitting the smallest and largest measurements) it might be possible to reduce the number of nodes with grossly overestimated betweenness. At least for road networks it should also help to do some preprocessing like getting rid of degree-one nodes. In multigraphs like the actor network, we might get improved performance by modeling parallel edges more explicitly.

For some directed networks, none of the approximation algorithms gives very convincing results since a good approximation takes almost as much time as an exact calculation. Although we did not find undirected graphs with similarly bad performance, better theoretical performance bounds[13] or a more detailed experimental investigation on the influence of graph structure on performance remain an interesting open problem.

[13] [5] gives a probabilistic tail bound which contains a bug however. Repairing this bug yields a bound only useful for nodes with near quadratic betweenness.

5.1 Path Sampling. Bias due to the selection of sampled source nodes can be avoided altogether by sampling both source and target node rather than attempting to gain as much information as possible from a single-source (all-target) shortest-path computation. The obvious drawback is that we get $n-1$ times fewer paths per sample. We can mitigate this problem by using speedup techniques for answering the queries. In [13] this is done using inexact heuristics so that it is unclear what is actually computed. At least for road networks we can do much better using recently developed exact speedup techniques that are very efficient. In particular, transit-node routing [3] achieves nearly constant query time and thus might lead to very good results. More precisely, transit-node routing computes *transit nodes* u and v such that there is a shortest path from source s via u and v to target t. We can process this information by incrementing counters for the access connections $\langle s, u\rangle$ and $\langle v, t\rangle$ and the transit connection $\langle u, v\rangle$. Only at the end, we have to propagate these counters down to the actual nodes (or edges) of the underlying network. This propagation cannot take longer than building the data structures during preprocessing. We have not implemented path sampling because it is complicated and unlikely to work better than bisection sampling for road networks. In particular, we need a huge number of samples before unimportant nodes get a significant contribution. On the other hand, if we were interested in a small selection of nodes with highest betweenness and the transit-node preprocessing would be needed anyway, then path sampling would be very efficient and we could even derive good performance guarantees using Chernoff bounds.

5.2 Including Local Searches. We have developed a generalization of our framework that allows a mix of local and global search and remains unbiased. This might lead to better estimates for not-so-important nodes. In particular, false zero values might be reduced by local searches around nodes with zero betweenness estimate. Preliminary experiments indicate that the general idea works but does not yield dramatic improvements for the instances tried.

5.3 Application to Highway-Node Routing. Highway-node routing [14] requires a nested hierarchy of more and more important nodes. We have preliminary results on how to use betweenness estimations to define improved highway-node hierarchies. At the same time, the resulting method is simpler than the approach used in [14]. However, a really good scheme seems to require additional heuristics beyond the scope of this paper.

Acknowledgments. Robert Görke has provided us with citation networks crawled from DBLP and Citeseer. Kamesh Madduri has provided the patent network and the actor network. We would like to thank Reinhard Bauer, Ulrik Brandes, Daniel Delling and Dorothea Wagner for interesting discussions. Several anonymous reviewers provided valuable suggestions.

References

[1] J. M. Anthonisse. The rush in a directed graph. Technical Report BN 9/71, Stichting Mathematisch Centrum, Amsterdam, 1971.

[2] David A. Bader and Kamesh Madduri. Parallel algorithms for evaluating centrality indices in real-world networks. In *ICPP*, pages 539–550. IEEE Computer Society, 2006.

[3] H. Bast, S. Funke, P. Sanders, and D. Schultes. Fast routing in road networks with transit nodes. *Science*, 316(5824):566, 2007.

[4] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[5] Ulrik Brandes and Christian Pich. Centrality estimation in large networks. *International Journal of Bifurcation and Chaos*, to appear. special issue on Complex Networks' Structure and Dynamics.

[6] CiteSeer. Scientific Literature Digital Library. `http://citeseer.ist.psu.edu/`, 2007.

[7] DBLP. DataBase systems and Logic Programming. `http://dblp.uni-trier.de/`, 2007.

[8] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40:35–41, 1977.

[9] A. Goldberg, H. Kaplan, and R. Werneck. Reach for A^*: Efficient point-to-point shortest path algorithms. In *Workshop on Algorithm Engineering & Experiments*, pages 129–143, Miami, 2006.

[10] A. B. Jaffe Hall, B. H. and M. Tratjenberg. The NBER Patent Citation Data File: Lessons, Insights and Methodological Tools. *NBER Working Paper*, 8498, 2001.

[11] Laboratory for Web Algorithmics. `http://law.dsi.unimi.it/index.php?option=com_include&Itemid=65`.

[12] Notre Dame CNet resources. `http://www.nd.edu/~networks/`.

[13] M. J. Rattigan, M. Maier, and D. Jensen. Using structure indices for efficient approximation of network properties. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, pages 357–366. ACM, 2006.

[14] D. Schultes and P. Sanders. Dynamic highway-node routing. In *6th Workshop on Experimental Algorithms*, volume 4525 of *LNCS*, pages 66–79. Springer, 2007.

[15] Alfonso Shimbel. Structural parameters of communication networks. *Bulletin of Mathematical Biophysics*, 15:501–507, 1953.