On Compressing Social Networks

Flavio Chierichetti[*†]
Dipartimento di Informatica
Sapienza University of Rome
chierichetti@di.uniroma.it

Ravi Kumar
Yahoo! Research
Sunnyvale, CA
ravikumar@yahoo-inc.com

Silvio Lattanzi[†]
Dipartimento di Informatica
Sapienza University of Rome
lattanzi@di.uniroma.it

Michael Mitzenmacher[‡†]
School of Engg. & Appl. Sci.
Harvard University
michaelm@eecs.harvard.edu

Alessandro Panconesi[†]
Dipartimento di Informatica
Sapienza University of Rome
ale@di.uniroma.it

Prabhakar Raghavan
Yahoo! Research
Sunnyvale, CA
pragh@yahoo-inc.com

ABSTRACT

Motivated by structural properties of the Web graph that support efficient data structures for in memory adjacency queries, we study the extent to which a large network can be compressed. Boldi and Vigna (WWW 2004), showed that Web graphs can be compressed down to three bits of storage per edge; we study the compressibility of social networks where again adjacency queries are a fundamental primitive. To this end, we propose simple combinatorial formulations that encapsulate efficient compressibility of graphs. We show that some of the problems are NP-hard yet admit effective heuristics, some of which can exploit properties of social networks such as link reciprocity. Our extensive experiments show that social networks and the Web graph exhibit vastly different compressibility characteristics.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Database Applications—*Data Mining*; G.2.2 [**Mathematics of Computing**]: Graph Theory—*Graph algorithms*

General Terms

Algorithms, Experimentation, Measurement, Theory

Keywords

Compression, Social networks, Linear arrangement, Reciprocity

1. INTRODUCTION

We study the extent to which social networks can be compressed. There are two distinct motivations for such studies. First, Web properties require high-speed indexes for serving adjacencies in the social network: thus, a typical query seeks the neighbors of a node (member) of a social network. Maintaining these indexes in memory demands that the underlying graph be stored in a compressed form that facilitates efficient adjacency queries. Secondly, there is a wealth of evidence (e.g., [17]) that social networks are not random graphs in the usual sense: they exhibit certain distinctive local characteristics (such as degree sequences). Studying the compressibility of a social network is akin to studying the degree of "randomness" in the social network. The Web graph (Web pages are nodes, hyperlinks are directed edges) is a special variant of a social network, in that we have a network of pages rather than of people. It is known that the Web graph is highly compressible [6, 11]. Particularly impressive results have been obtained by Boldi and Vigna [6], who exploit *lexicographic locality* in the Web graph: when pages are ordered lexicographically by URL, proximal pages have similar neighborhoods. More precisely, two properties of the ordering by URL are experimentally observed to hold:

- *Similarity:* pages that are proximal in the lexicographic ordering tend to have similar sets of neighbors.
- *Locality:* many links are intra-domain, and therefore likely to point to pages nearby in the lexicographic ordering.

These two empirical observations are exploited in the BV-algorithm to compress the Web graph down to an amortized storage of a few bits per link, leading to efficient in-memory data structures for Web page adjacency queries (a basic primitive in link analysis). Do these properties of locality and similarity extend to social networks in general? Whereas the Web graph has a natural lexicographic order (by URL) under which this locality holds, there is no such obvious ordering for social networks. Can we find such an ordering for social networks, leading to compression through lexicographic locality?

Our main contributions in the paper are the following. We propose a new compression method that exploits link reciprocity in social networks (Section 3). Motivated by this and BV, we formulate a genre of graph node ordering problems that distill the essence of locality in BV-style algorithms (Section 4.1). We develop a simple and practical heuristic based on shingles for obtaining an effective node ordering; this ordering can be used in BV-style compression algorithms (Section 4.3). We then perform an extensive set of experiments on large real-world graphs, including two social networks (Section 5). Our main findings are: social networks appear

[*]Work done in part while visiting Yahoo! Research.
[†]Supported in part by a grant from Yahoo! Research.
[‡]Supported in part by NSF grant NSF CNS-0721491.

KDD'09, June 28–July 1, 2009, Paris, France.
Copyright 2009 ACM 978-1-60558-495-9/09/06 ...$5.00.

far less compressible than Web graphs yet closer to host graphs and exploiting link reciprocity in social networks can vastly help its compression.

2. RELATED WORK

Prior related work falls into three major categories, namely, compressing Web graphs, compressing indexes, and graph ordering problems.

Adler and Mitzenmacher introduced the idea of finding pages with similar sets of neighbors in the context of compressing Web graphs, and obtained some hardness results for compression in this context [1]. Randall et al. [22] suggested lexicographic ordering as a way to obtain good Web graph compression, utilizing both similarity and locality. Raghavan and Garcia-Molina [21] considered a hierarchical view of the Web graph to achieve compression; see also Suel and Yuan [27] for a structural approach to compressing Web graphs. A major step was taken by Boldi and Vigna [6], who both developed a generic Web graph compression framework that takes into account the locality and similarity of Web pages and obtained very strong compression performance; our work is based on this framework. Boldi and Vigna [7] also developed ζ-codes, to exploit power law distributed integer gaps. Buehrer and Chellapilla [11] used the frequent pattern mining approach to compress Web graphs; using this, they were able to achieve a compression of under two bits per link. Recently, Boldi, Santini, and Vigna [5] studied the effectiveness of various orderings, including Gray ordering, in compressing the transpose of the Web graph.

The problem of assigning or reassigning document identifiers in order to compress text indexes has a long history. Blandford and Blelloch [4] considered the problem of compressing text indexes by permuting the document identifiers to create locality in an inverted index. Silvestri, Perego, and Orlando [26] proposed a clustering approach for reassigning document identifiers. Shieh et al. [24] proposed a document identifier reassignment method based on a heuristic for the traveling salesman problem. Recently, Silvestri [25] showed that assigning document identifiers to Web documents based on URL lexicographic ordering improves compression.

There are many classical node ordering problems on graphs. The minimum bandwidth problem, where the goal is to order the nodes to minimize the maximum stretch of edges, and the minimum linear arrangement problem, where the goal is to order the nodes to minimize the sum of stretch of edges, have a rich history. We refer to [13] and the online compendium at `www.nada.kth.se/~viggo/wwwcompendium/node52.html`.

3. COMPRESSION SCHEMES

In this section we outline the compression framework used in the rest of the paper. The framework is based on the algorithm of Boldi and Vigna for compressing Web graphs [6]; their algorithm achieved a compression down to about three bits per link on a snapshot of the Web graph. We henceforth refer to this as the *BV compression* scheme, which we first describe. Next, we describe what we call the *backlinks compression (BL)* scheme, which targets directed graphs that are highly reciprocal.

Notation. Let $G = (V, E)$ be a directed graph and let $|V| = n$. The nodes in V are bijectively identified with the set $[n] = \{1, \ldots, n\}$ of integers. For a node $u \in V$, let $\mathrm{out}(u) \subseteq V$ denote the set of out-neighbors of u, i.e., $\mathrm{out}(u) = \{v \mid (u, v) \in E\}$. Likewise, let $\mathrm{in}(u)$ denote the set of in-neighbors of u. Let $\mathrm{outdeg}(u) = |\mathrm{out}(u)|$ and $\mathrm{indeg}(u) = |\mathrm{in}(u)|$. If both $(u, v) \in E$ and $(v, u) \in E$ and $u < v$, then we call the edge (v, u) to be *reciprocal*. For a node $u \in V$, let $\mathrm{rec}(u)$ be $\{v \mid (v, u)$ is reciprocal $\}$. Let lg denote $\log_2$.

We will encode all integers using one of three different encoding schemes, namely, Elias's γ-code, δ-code, and Boldi–Vigna ζ-code with parameter 4 (which we found to be the best in our experiments) [7]. These integer encoding schemes encode an integer $x \in Z^+$ using close to the informatic-theoretic minimum of $1 + \lfloor \lg(x) \rfloor$ bits. For example, the number of bits used by the γ-code to represent x is $1 + 2\lfloor \lg x \rfloor$. We refer to [28] for more background on these codes.

3.1 BV compression scheme

BV incorporates three main ideas. First, if the graph has many nodes whose neighborhoods are similar, then the neighborhood of a node can be expressed in terms of other nodes with similar neighborhoods. Second, if the destinations of edges exhibit locality, then small integers can be used to encode them (relative to their sources). Third, rather than store the destination of each edge separately, one can use *gap encodings* to store a sequence of edge destinations. Given a sorted list of positive integers (say, the destinations of edges from a node), we write down the sequence of *gaps* between subsequent integers on the list, rather than the integers themselves. The idea is that even if the integers are big (requiring many bits to record), the gaps between integers on the list could be recorded with fewer bits.

We now detail the BV scheme for compressing Web graphs. The nodes are Web pages and the directed edges are the hyperlinks. First, order Web pages lexicographically by URL. This assigns to each Web page a unique integer identifier (ID), which is its position in this ordering. Let w be a window parameter; for the Web, BV recommend $w = 8$.

Let v be a Web page. Its encoding will be as follows.

1. *Copying.* Check if the list $\mathrm{out}(v)$ of v's out-neighbors is a small variation on the list of one of the $w - 1$ preceding Web pages in the lexicographic ordering. Let u be such a prototype page, if it exists.

2. *Encoding.* Encode v's out-neighbors as follows. If the copying step found a prototype u, then use $\lg w$ bits to encode the (backward) offset from v to u, followed by the changes from u's list to v's. If none of the $\lg w$ preceding pages in the lexicographic ordering offers a good prototype, set the first $\lg w$ bits to all 0's, then explicitly write down v's out-neighbors. (BV also optimize further by storing a list $i, i + 1, \ldots, j - 1, j$ of consecutive out-neighbors by storing the interval $[i, j]$ instead.)

Note that locality and similarity are captured by the copying step. By using clever gap encoding schemes (using the integer codes mentioned earlier) on top of the basic method above, BV obtain their best results. Note that the availability of a natural ordering on the Web pages facilitates exploitation of locality. For more details, we refer to the original paper [6] and [19, Chapter 20].

This general method of compression has two nice properties. First, it is dependent only on locality in some canonical ordering. Second, adjacency queries (fetch all the out-neighbors of a given node) can be served fairly efficiently. Given a Web page whose out-neighbors are sought, we enumerate these out-neighbors by decoding backwards through the chain of prototypes, until we arrive at a list whose encoding begins with at least $\lg w$ 0's. While in principle this chain could be arbitrarily long, in practice most chains are short. For instance, few chains would go backwards across URL domains.

3.2 Backlinks (BL) compression scheme

We now describe a slightly different compression scheme that is motivated by the observed properties of social networks. This scheme, called *BL*, incorporates an additional idea on top of BV, namely, link *reciprocity*. In the BL scheme, reciprocal links are encoded in a special way. Since social networks are known to be mostly reciprocal (if Alice is Bob's friend, then Bob is very likely to be Alice's friend), this will turn out to be advantageous.

Suppose we obtain an ordering of the nodes in the graph through some process to be discussed later. We will identify each node in the graph with its position in this ordering. Let v be a node. Its encoding will consist of the following.

1. *Base information.* The outdegree $|\mathrm{out}(v)|$, minus 1 if v has a self-loop, and minus the number of reciprocal edges from v. Also, include a bit specifying if v has a self-loop.

2. *Copying.* The node u that v uses as a prototype to copy from: as $u \leq v$ in the ordering, u is encoded as the difference between u and v. If $u = v$, then no copying is performed. Otherwise, a bit is added for each out-neighbor of u, representing whether or not that out-neighbor of u is also an out-neighbor of v.

3. *Residual edges.* Let $v_1, \ldots, v_k$ be the out-neighbors of v that are yet to be encoded after the above step. Let $v_1 \leq \cdots \leq v_k$. We write one bit stating if $v > v_1$ or $v < v_1$. Then we encode the gaps $|v_1 - v|, |v_2 - v_1|, \ldots, |v_k - v_{k-1}|$.

4. *Reciprocal edges.* Finally, we encode the reciprocal out-neighbors of v. For each $v' \in \mathrm{out}(v)$ such that $v' > v$, we encode whether $v' \in \mathrm{rec}(v)$ or not using one bit per link and discard (v', v).

Note that reciprocal edges are succinctly encoded by the last step. Thus, this method potentially outperform BV in terms of compression. However, it has a drawback: unlike in BV, adjacency queries may be slower. This is because BV limits the "length" of prototype chains but we do not impose such a limit in BL. If the compressed representation of a network bottlenecks adjacency query serving, then a limit on the length of copying chain can be introduced in BL as well.

4. COMPRESSION-FRIENDLY ORDERINGS

In both the BV and BL schemes, the ordering of nodes plays a crucial role in the performance of the compression scheme. The performance of suggests that the lexicographic ordering of URL's for the Web graph is both natural and crucial, begging the question: *can we find such orderings for other graphs, in particular, social networks*? If we could, we would be able to apply either the BV or the BL scheme. In this section we study ordering problems that are directly motivated by the BV and BL compression schemes.

4.1 Formulation

We first formalize the problem of finding the best ordering of nodes in a graph for the BV and BL schemes. As we saw earlier, both algorithms benefit if locality and similarity are captured by this ordering. This leads to the following natural combinatorial optimization problem, which we call *minimum logarithmic arrangement*.

PROBLEM 1 (MLOGA). *Find a permutation* $\pi : V \to [n]$ *such that* $\sum_{(u,v)\in E} \lg|\pi(u) - \pi(v)|$ *is minimized.*

The motivation behind this definition is to minimize the sum of the logarithms of the edge lengths according to the ordering (where the length of the edge $u \to v$ is $|\pi(u) - \pi(v)|$). Notice this cost represents the compression size of the length of the edge in an encoding that is information-theoretically optimal (or nearly so).

Also note that if the term inside the summation were just $|\pi(u) - \pi(v)|$, then this is the well-known *minimum linear arrangement* (MLINA) problem. MLINA is NP-hard [14]; little, however, is known about its approximability. The best algorithm [23] approximates MLINA to $O(\sqrt{\log n}\log\log n)$ and this algorithm is not practical for large graphs. From the standpoint of the hardness of approximation, only the existence of a PTAS has been ruled out [3]. One cannot hope to use an approximate solution to MLINA to solve MLOGA since we can show (see Appendix A) that these problems are very different in their structure.

In actually compressing the graph, it is more efficient to compress the gaps induced by the neighbors of a node. Suppose $u < v_1 < v_2$ and $(u, v_1), (u, v_2) \in E$. Then, compressing the gaps $v_1 - u$ and $v_2 - v_1$ is always and could be far less expensive than compressing the lengths of the edges, namely, $v_1 - u$ and $v_2 - u$. For this reason, we introduce a slightly modified problem, called *minimum logarithmic gap arrangement*. Let $f_\pi(u, \mathrm{out}(u))$ be the cost of compressing $\mathrm{out}(u)$ under the ordering π, i.e., if $u_0 = u, \mathrm{out}(u) = \{u_1, \ldots, u_k\}$ with $\pi(u_1) \leq \cdots \leq \pi(u_k)$, then

$$f_\pi(u, \mathrm{out}(u)) = \sum_{i=1}^{k} \lg|\pi(u_i) - \pi(u_{i-1})|.$$

PROBLEM 2 (MLOGGAPA). *Find a permutation* $\pi : V \to [n]$ *such that* $\sum_{u\in V} f_\pi(u, \mathrm{out}(u))$ *is minimized.*

Once again, as a problem, MLOGGAPA turns out to be very different from MLINA and MLOGA (see Appendix A).

Both formulations MLOGA and MLOGGAPA capture the essence of obtaining an ordering that will benefit BV and BL compressions. We believe a good approximation algorithm for either of these problems will be of practical interest.

4.2 Hardness results

We show that MLOGA is hard in general. The proof is in Appendix B.

THEOREM 3. MLOGA *is NP-hard on multi-graphs.*

While we are currently unable to show that MLOGGAPA is NP-hard, we can show that its "linear" version (i.e., without the logarithms), MLINGAPA, is indeed hard. The proof is in Appendix C.

THEOREM 4. MLINGAPA *is NP-hard.*

We can also show a lower bound on the solution to MLOGA for expander-like graphs, suggesting that they are not compressible with constant number of bits per edge via BV/BL schemes. The proof is in Appendix D.

LEMMA 5. *If G has constant conductance, then the cost of* MLOGA *on $G = (V, E)$ is $\Omega(|E| \log n)$. If, instead, G has constant node or edge expansion, then the cost of* MLOGA *on G is $\Omega(n \log n)$.*

4.3 The shingle ordering heuristic

In this section we propose a simple and practical heuristic for both MLOGA and MLOGGAPA problems. Our heuristic is based on obtaining a fingerprint of the out-neighbors of a node and ordering the nodes according to this fingerprint. If the fingerprint can succinctly capture the locality and similarity of nodes, then it can be effective in BV/BL compression schemes.

To motivate our heuristic, we recall the *Jaccard coefficient* $J(A, B) = |A \cap B|/|A \cup B|$, a natural notion of similarity of two sets. Let σ be a random permutation of the elements in $A \cup B$. For a set A, let $M_\sigma(A) = \sigma^{-1}(\min_{a\in A}\{\sigma(a)\})$, the smallest element in A according to σ; we call it the *shingle*. It can be shown [10] that the probability that the shingles of A and B are identical is precisely the Jaccard coefficient $J(A, B)$, i.e.,

$$\Pr[M_\sigma(A) = M_\sigma(B)] = \frac{|A \cap B|}{|A \cup B|} = J(A, B).$$

Instead of using random permutations, it was shown that the so-called *min-wise independent* family suffices [10]; in practice, even pairwise independent hash functions work well. It is also easy to boost the accuracy of this probabilistic estimator by combining multiple shingles obtained from independent hash functions.

The intuition behind our heuristic is to treat the out-neighbors $\mathrm{out}(u)$ of a node u as a set and compute the shingle $M_\sigma(\mathrm{out}(u))$ of this set for a suitably chosen permutation (or hash function) σ. The nodes in V can then be ordered by the shingles. By the property stated above, if two nodes have significantly overlapping out-neighbors, i.e., share a lot of common neighbors, then with high probability they will have the same shingle and hence be close to each other in a shingle-based ordering. Thus, the properties of locality and similarity are captured by the *shingle ordering* heuristic. (Gibson, Kumar, and Tomkins [15] used a similar heuristic, but for identifying dense subgraphs of large graphs.)

4.4 Properties of shingle ordering

In this section we show some theoretical justification for the shingle ordering heuristic: using shingle ordering, it is possible to copy a constant fraction of the edges in a large class of random graphs with certain properties. The well-known preferential attachment (PA) model [2, 8], for instance, generates graphs in this class. Our analysis thus shows that it is indeed possible to obtain provable performance guarantees on shingle ordering with respect to copying (hence compression) in stylized models.

We first prove the following general statement about the sufficient conditions under which using shingle ordering can copy a constant fraction of edges. The proof is given in Appendix E.

THEOREM 6. *Let $G = (V, E)$ be such that $|E| = \Theta(n)$ and $\exists S \subseteq V$ such that*
(i) $|S| = \Theta(n)$,
(ii) $\forall v \in S, \exists v' \in S, v \neq v'$, s.t. $|\mathrm{out}(v) \cap \mathrm{out}(v')| \geq 1$,
(iii) there exists a constant k, s.t. $\forall v \in S, \mathrm{outdeg}(v) \leq k$,
(iv) $\forall v \in S, \forall w \in \mathrm{out}(v), \mathrm{indeg}(w) \leq n^{\frac{1}{2}-\epsilon}$.
Then, with probability $1 - o(1)$ (over the space of permutations), at least a constant fraction of the edges will be "copied" (even with a window of size 1) when using the shingle ordering.

It is trivial to note that this holds even for undirected graphs; indeed, each undirected edge $\{u, v\}$ can be substituted by two directed edges $(u, v), (v, u)$. Then, for each node, its original set of neighbors will be the same as its new sets of in- and out-neighbors.

We now show the main result of the section: using shingle ordering it is possible to copy a constant fraction of the edges of graphs generated by the PA model.

THEOREM 7. *With high probability, the graphs generated by the PA model satisfies the properties of Theorem 6.*

PROOF. We start by removing the nodes incident to multi-edges or self-loops — there are $o(n)$ such nodes and their incident edges[1]. Also, we remove all nodes of degree $> k$, for some constant k — by [9] only $\epsilon_k n$ edges and nodes will be removed this way.

The resulting graph will thus have at most n nodes and at least $(1 - 2\epsilon_k)mn \geq (1 - 2\epsilon_k)n$ edges. Also its maximum degree will be k. By averaging, a graph having these three properties will contain at least $(1 - 2\epsilon_k)\frac{n}{2k}$ nodes of degree at least 2.

Now take all the nodes v in this graph incident to a neighbor of degree ≥ 2. There are $\geq (1 - 2\epsilon_k)\frac{n}{2k}$ such neighbors and each of them will be connected to at most k such v's. Thus, the number of these v's is at least $\Omega(n/(2k^2)) = \Omega(n)$. The set of these v's is the set S of Theorem 6. □

As our experiments show, shingle ordering allows both BL and BV schemes to take significant advantage of copying.

5. EXPERIMENTAL RESULTS

In this section we describe the experimental results. The goal of our experiments is two-fold: (1) study the performance of BV/BL schemes using the shingle ordering on social networks; (2) obtain insights into the differences between the Web and social networks in terms of their compressibility. First we begin with the description of the data sets we use for our experiments. Next we discuss the baselines we use (to compare against shingle ordering). Finally we present and discuss our experimental results.

5.1 Data

For our experiments, we chose four large directed graphs: (i) a 2008 snapshot of LiveJournal (a social network site, `livejournal.com`) and an induced subgraph of users, called LiveJournal (zip), for whom we know their zip codes; (ii) monthly snapshots of Flickr (a photosharing site, `flickr.com`) from March 2004 until April 2008; (iii) the host graph[2] of a 2005 snapshot of the `.uk` Web graph; and (iv) the host graph of a 2004 snapshot of the IndoChina (`.in`, `.cn`) Web graph.

Graph	n	$\|E\|$	% reciprocal edges
LiveJournal	5,363,260	79,023,142	72.0
LiveJournal (zip)	1,314,288	8,040,562	79.0
Flickr	25,158,667	69,702,479	64.4
UK-host	587,205	12,825,465	18.6
IndoChina-host	19,123	233,380	10.6
IndoChina	7,414,866	194,109,311	20.9

Table 1: Basic properties of our graphs.

In Table 1, we summarize the properties of the graphs we have considered. Notice the magnitude of the reciprocity of the social networks (LiveJournal and Flickr); the BL scheme will critically leverage this property.

5.2 Baselines

We use the following orderings as our baselines to compare against the shingle ordering.

(1) *Random* order. We use a random permutation of all the nodes in the graph.

(2) *Natural* order. This is the most basic order that can be defined for a graph. For Web and host graphs, a natural order is the URL lexicographic ordering (used by BV). For a snapshot of LiveJournal, a natural order is the order in which the user profiles were

[1] This can be easily shown by noting that the expected number of multiple edges and self-loops added by the nth inserted node is $O(m^3/n^{1/2-\epsilon})$, conditioned on the fact that the highest degree at that point is $O(n^{1/2+\epsilon})$ whp [12]. Then, by Markov's inequality the claim follows.

[2] Host graph refers to a directed graph whose nodes are the hosts and an edge exists from host a to host b if there is a Web page on host a that points to a Web page on host b.

crawled. For Flickr, since we know the exact time at which each node and edge was created, a natural order is the order in which users joined the network.

(3) *Gray* order. Consider the binary adjacency matrix induced by, say, the natural order. Now, a node v precedes node w in the Gray ordering if the row of v precedes the row of w (both interpreted as binary strings) in the canonical Gray code [5, 22].

(4) *Geographic* order. In a social network, if geographic information is available in the form of a zip code, then this defines a geography-based order. Liben-Nowell et al. [18] showed that about the 70% of social network links arise from geographical proximity, suggesting that friends can be grouped together using geographical information. Notice that this only defines a partial order (i.e., with ties).

(5) *DFS* and *BFS* order. Here, the orderings are given by these common graph traversal algorithms. We also try the undirected versions of these traversals, where the edge direction is discarded.

To test the robustness of shingle ordering, we also use an ordering obtained by two shingles instead of just one, where the second shingle is used to break ties produced by the first. We call this the *double shingle* ordering. When only one shingle was used, ties were broken using the natural order or if specified, the Gray order.

Our performance numbers are always measured in bits/link.

5.3 Compression performance

In Table 2, we present the results of the different compression/orderings on our graphs. This table shows that double shingle ordering produces the best or the near-best compression, for both BV and BL. In some cases, it cuts almost half the number of bits used by the natural order. Also we note that the improvement of BL over BV is significant for networks that are highly reciprocal, i.e., social networks. Finally, the numbers show interesting similarities between social networks and host graphs. In both cases, their compressibility using the best compression (BL with double shingle order) is on par with each another.

It is interesting to note that the best compression rates for the host graphs are similar to that of the social networks, even though the former are much smaller in size than the latter. For comparison, we note how the snapshot of the UK domain (IndoChina domains) that we used to obtain the host graph, was found to be compressible to 1.701 (1.472) bits/link (see [6] and `http://law.dsi.unimi.it/`). This seems to indicate that the host graphs are very hard to compress.

Graph	BV		BL	
	Shingle	Double Shingle	Shingle	Double Shingle
LiveJournal	15.977	15.838	10.421 (ζ_4)	10.415 (ζ_4)
Flickr	13.602	13.503	10.938 (δ)	10.891 (δ)
UK-host	8.318	8.097	8.197 (ζ_4)	8.094 (ζ_4)
IndoChina-host	7.328	7.260	7.082 (δ)	7.080 (δ)

Table 3: Performances of the compression techniques using Gray orderings to break ties.

Next we present the effect of breaking ties in shingle and double shingle orderings using the Gray ordering (Table 3). Modest improvements are obtained by this method for some graphs. Once again, LiveJournal does not appear to be amenable to the shingle approach.

We also note how (Table 4) the BFS/DFS orderings are always suboptimal, almost as bad as a random order. In Table 5, we show the performance of geographical ordering on the induced subgraph of LiveJournal, restricted to users in US with a known zip code. We see how ordering by zip code (i.e., in such a way that people at small geographic distance are close to each other in the ordering) is much worse than ordering by shingle, suggesting that geographic ordering is perhaps not useful for compression.

5.4 Temporal analysis

In Figure 1, we see how the different ordering and compression techniques achieve different results on the monthly snapshots of the Flickr social network. The upper half of the figure shows how the Flickr network grew over time. Here, we see that BL with shingle ordering beats the competition uniformly over all the snapshots. We also see an interesting pattern: BL obtains a better compression rate, with each of the orderings. It is remarkable to note that even though the number of edges in Flickr grew by an enormous number between March 2005 and April 2008, the compressibility of the network (under a variety of schemes and orderings) has remained robust.

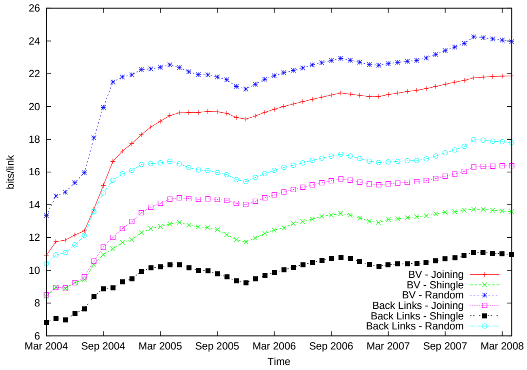

Figure 1: Performance on the temporal Flickr graph.

5.5 Why does shingle ordering work best?

Figures 2 and 3 show one reason why the shingle ordering helps compression: in the LiveJournal, IndoChina-host and UK-host graphs the number of small gaps is higher with shingle ordering than with any other ordering. The notable exception is the LiveJournal graph, where the natural ordering is marginally better.

In Figure 2, the upper panel represents the number of gaps (y-axis) of a certain length (x-axis) for the LiveJournal graph. The lower panel represents a sub-sampled version of the same data: for each length i we deleted the length $= i$ point with probability $\Theta(1/i)$. This way, in expectation, the number of points in each interval $10^k, \ldots, 10^{k+1}$ is the same. The bottom panel is more readable. Recall that in LiveJournal, the natural (crawl) ordering beats the shingle ordering by a small amount.

In Figure 3, the upper (lower) panel represents the number of gaps (y-axis) of a certain length (x-axis) for (top to bottom) for the UK-host and the IndoChina-host. These are the sub-sampled versions of the actual data. Note that in both cases, shingle ordering is the best, i.e, the shingle ordering creates many more gaps of small length than the other orderings — the smaller the length of a gap, the fewer bits it takes for encoding.

From these, we see that shingle ordering reduces the lengths of gaps. As we argued earlier, shingle ordering also helps the BV and BL schemes exploit copying. These two benefits together appear to be the main reasons why shingle ordering almost always outperforms many other orderings.

Graph	BV					BL				
	Natural	Random	Gray	Shingle	Double Shingle	Natural	Random	Gray	Shingle	Double Shingle
LiveJournal	14.435	23.566	14.308	15.956	15.828	9.564 (ζ_4)	15.169 (ζ_4)	9.559 (ζ_4)	10.461 (ζ_4)	10.435 (ζ_4)
Flickr	21.865	23.958	13.830	13.549	13.496	16.382 (ζ_4)	17.785 (ζ_4)	11.316 (δ)	10.952 (ζ_4)	10.915 (ζ_4)
UK-host	10.826	15.543	8.117	8.218	8.138	10.574 (δ)	14.528 (δ)	7.982 (ζ_4)	8.243 (δ)	8.133 (δ)
IndoChina-host	9.224	10.543	7.340	7.367	7.120	9.753 (ζ_4)	10.823 (ζ_4)	7.262 (ζ_4)	7.310 (δ)	7.126 (δ)
IndoChina	2.025	21.449	-	2.719	2.721	2.349 (δ)	17.603 (ζ_4)	-	2.779 (δ)	2.780 (δ)

Table 2: Performances of the compression techniques under different orderings.

Graph	BV				BL			
	DFS	Undir. DFS	BFS	Undir. BFS	DFS	Undir. DFS	BFS	Undir. BFS
LiveJournal	19.992	20.253	20.763	21.376	12.924 (ζ_4)	13.096 (ζ_4)	13.401 (ζ_4)	13.778 (ζ_4)
UK-host	14.630	14.474	14.903	14.634	13.774 (ζ_4)	13.607 (ζ_4)	13.978 (ζ_4)	13.731 (ζ_4)
IndoChina-host	10.172	10.210	10.231	9.810	10.561 (ζ_4)	10.317 (ζ_4)	10.558 (ζ_4)	10.105 (ζ_4)

Table 4: Performance of the BFS/DFS orderings.

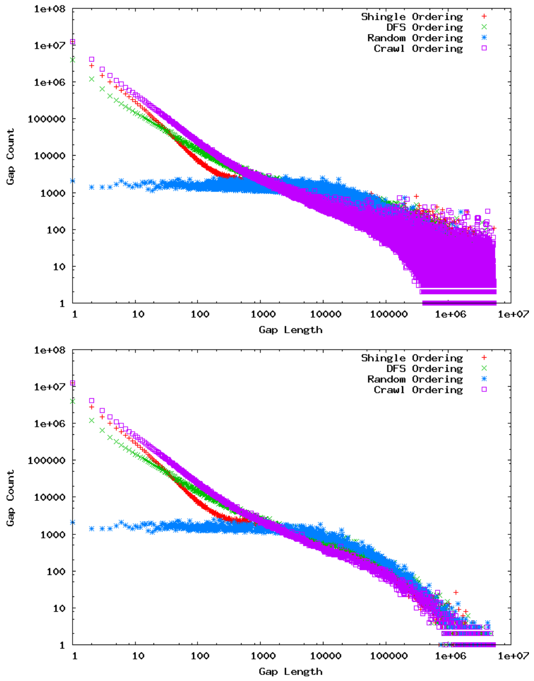

Figure 2: Gap distribution in LiveJournal graph.

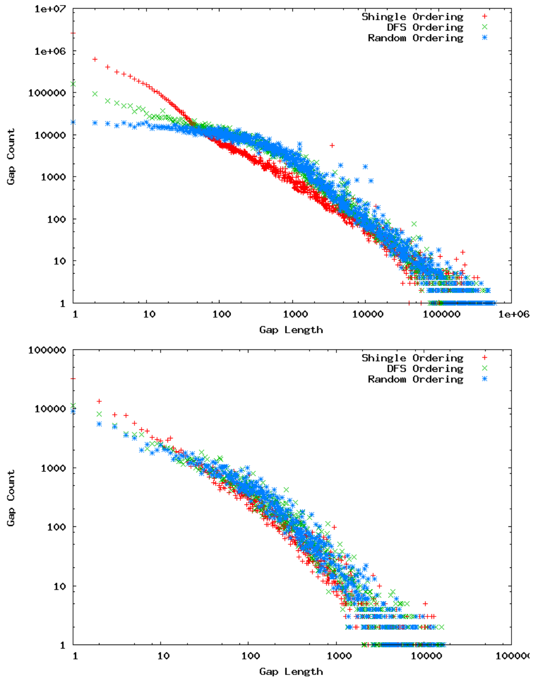

Figure 3: Gap distribution in UK-host and IndoChina-host graphs.

5.6 A cause of incompressibility

We investigate what causes social networks to be far less compressible than Web graphs. We ask the question: is the densest portion of a social network far more compressible than the rest of the graph? To study this, we analyze k-cores of the LiveJournal social network. Recall that a k-core of a graph is the largest induced subgraph whose minimum degree is at least k. For each k, the k-core of LiveJournal was extracted and compressed by itself. Then, the k-core edges were were removed from the original LiveJournal, which was also compressed by itself. The results are shown in Figure 4. It is clear that as k increases, the k-core gets easier to compress but at the same time the remaining graph gets harder and harder to compress. This suggests that the low-degree nodes in social networks are primarily responsible for its incompressibility.

On a separate note, k-cores can also be used to compress the social network. This is done by representing all the nodes in a k-core by a single virtual node, and compressing the k-core graph and the remainder graph (with the virtual node) separately. For the LiveJournal graph, for $k = 50$, we obtain 9.435 bits/link compression. This is a mild improvement over the best numbers in Table 2.

6. CONCLUSIONS

We have considered the compression of social networks, and have found both key similarities and differences between this problem and the related, previously well-studied problem of compressing Web graphs. As with Web graphs, it appears desirable to take advantage of lexicographic locality and similar neighborhoods for compression of social networks. However, it is less clear that there is a natural ordering for social networks comparable to the URL

Graph	BV			BL		
	Geographic	Shingle	Double Shingle	Geographic	Shingle	Double Shingle
LiveJournal (zip)	17.258	17.042	16.975	11.396 (ζ_4)	10.964 (δ)	10.950 (δ)

Table 5: Performance of geographic ordering on LiveJournal (zip).

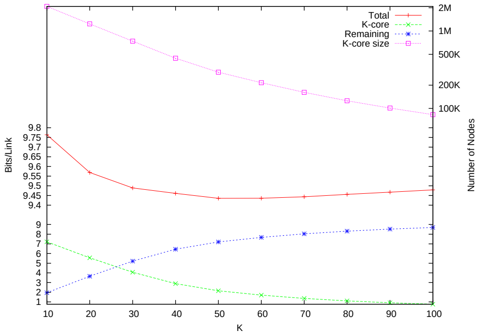

Figure 4: Compressibility of k-cores.

ordering of Web pages. Instead, we have shown that shingle ordering, which is based only on the local linkage of the graph, performs better than other seemingly natural orderings. We also highlight the critical role of reciprocal edges in social network graphs.

We have shown how the optimization of locality-based compression schemes can be formulated as variations of the well-known minimum linear arrangement problem. These variations have their own subtle properties; approximation algorithms (or proving the hardness of approximation) remain interesting open questions. We have also considered lower bounds on compression for general classes of graphs, which suggest that the amazing compression ratios obtainable on the Web may be due to its particular structure, and may not be available for social graphs even given an optimal lexicographic ordering.

7. REFERENCES

[1] M. Adler and M. Mitzenmacher. Towards compressing web graphs. In *DCC*, pages 203–212, 2001.

[2] R. Albert and A.-L. Barabasi. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] C. Ambühl, M. Mastrolilli, and O. Svensson. Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *Proc. 48th FOCS*, pages 329–337, 2007.

[4] D. Bladford and G. Blelloch. Index compression through document reordering. In *Proc. DCC*, pages 342–351, 2002.

[5] P. Boldi, M. Santini, and S. Vigna. Permuting web graphs. In *Proc. 6th WAW*, pages 116–126, 2009.

[6] P. Boldi and S. Vigna. The webgraph framework I: Compression techniques. In *Proc. 13th WWW*, pages 595–602, 2004.

[7] P. Boldi and S. Vigna. The Webgraph framework II: Codes for the world-wide web. In *Proc. DCC*, 2004.

[8] B. Bollobas and O. Riordan. Mathematical results on scale-free random graphs. In S. Bornholdt and H. Schuster, editors, *Handbook of Graphs and Networks*, pages 1–37. Wiley–WCH, 2002.

[9] B. Bollobás, O. Riordan, J. Spencer, and G. E. Tusnády. The degree sequence of a scale-free random graph process. *RSA*, 18(3):279–290, 2001.

[10] A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *JCSS*, 60:630–659, 2000.

[11] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to web graph compression with communities. In *Proc. 1st WSDM*, pages 95–106, 2008.

[12] A. Flaxman, A. M. Frieze, and T. I. Fenner. High degree vertices and eigenvalues in the preferential attachment graph. *Internet Mathematics*, 2(1), 2005.

[13] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[14] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *TCS*, pages 237–267, 1976.

[15] D. Gibson, R. Kumar, and A. Tomkins. Extracting large dense subgraphs in massive graphs. In *Proc. 31st VLDB*, pages 721–732, 2005.

[16] J. Håstad. Some optimal inapproximability results. *JACM*, 48(4):798–859, 2001.

[17] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *Proc. 12th KDD*, pages 611–617, 2006.

[18] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *PNAS*, 102(33):11623–11628, 2005.

[19] C. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] C. McDiarmid. Concentration for independent permutations. *Comb., Prob. & Comp*, 11(2):163–178, 2002.

[21] S. Raghavan and H. Garcia-Molina. Representing web graphs. In *Proc. 19th ICDE*, pages 405–416, 2003.

[22] K. H. Randall, R. Stata, J. Wiener, and R. Wickremesinghe. The Link database: Fast access to graphs of the web. In *Proc. DCC*, pages 122–131, 2002.

[23] S. Rao and A. W. Richa. New approximation techniques for some linear ordering problems. *SICOMP*, 34(2):388–404, 2004.

[24] W.-Y. Shieh, T.-F. Chen, J. J.-J. Shann, and C.-P. Chung. Inverted file compression through document identifier reassignment. *IPM*, 39(1):117–131, 2003.

[25] F. Silvestri. Sorting out the document identifier assignment problem. In *Proc. 29th ECIR*, pages 101–112, 2007.

[26] F. Silvestri, R. Perego, and S. Orlando. Assigning document identifiers to enhance compressibility of web search indexes. In *Proc. SAC*, pages 600–605, 2004.

[27] T. Suel and J. Yuan. Compressing the graph structure of the web. In *Proc. DCC*, pages 213–222, 2001.

[28] I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, 1999.

APPENDIX

A. MLogA VS. MLinA VS. MLogGapA

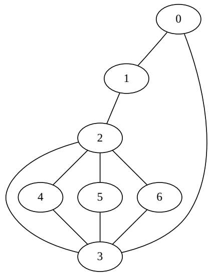

Figure 5: An example showing the difference between MLogA **and** MLinA**.**

The graph in Figure 5 is an example showing that the MLinA and the MLogA problems can have different optimal solutions: there is no ordering that minimizes both the objective functions simultaneously. The best solutions for MLinA have cost 19 whereas the best solutions for MLogA have cost $\lg 180$. It can be checked that among the optimal MLinA orderings (with cost 19), the best for MLogA has cost $\lg 192$ (e.g, the ordering $4, 5, 3, 2, 6, 1, 0$). Among the optimal MLogA ordering (with cost $\lg 180$), the best for MLinA has cost 20 (obtained by swapping 3 and 5 in the previous ordering).

It is easy to similarly show that MLogGapA can have different solutions from both MLinA and MLogA. For instance, consider a star with three leaves. The optimum ordering for MLogGapA will place the center of the star as the first (or the last) node of the ordering, yielding a total cost of 0. On the other hand this solution is suboptimal for both MLinA and MLogA, either of which would place the center of the star as the second (or the third) node in the ordering.

B. HARDNESS OF MLogA

We prove the hardness of MLogA using the inapproximability of MaxCut. Our starting point is that MaxCut cannot be approximated to a factor greater than $\frac{16}{17} + \epsilon$ unless P = NP [16]. In the reduction below we have not attempted to optimize parameters.

We start from a MaxCut instance $(G(V, E), k)$, where the question is if there is a cut of size at least k in G. Let $|V| = n$ and $|E| = m$. We build the graph G' composed by a clique of size n^{100} and a disjoint copy of the complement of G denoted $\bar{G}$. Further, we add an edge between each node of the clique and each node of $\bar{G}$. Each edge of the clique will have multiplicity $n^{500} + 1$ and all other edges will have unit multiplicity.

Now we would like to answer the following question Q: is it possible to find an ordering of G' with an MLogA cost smaller than a given Z? We show that answering questions of the form Q would allow us to approximate the corresponding MaxCut instance. Let $C = \sum_{1 \le i < j \le n^{100}+n+1} \lg(j - i)$ and let $X = n^{500} \sum_{1 \le i < j \le n^{100}} \lg(j - i)$.

First, note that in any ordering of G' for which the answer for Q is yes when $Z = C + X - k \lg n^{100}$, the nodes in the clique must be adjacent. Otherwise, at least one edge of the clique will be enlarged by at least 1. In this case, the overall cost of the clique edges will be at least $X - (n^{500} + 1)(\lg n^{100}) + (n^{500} + 1) \lg(n^{100} + 1)$, which is $X + \Omega(n^{400})$. This is larger than the cost allowed by the question Q.

We show that if the answer to Q when $Z = C + X - k \lg n^{100}$ is positive, then there is a cut in G of size at least $k(1 - \frac{1}{50})$, and otherwise there is no cut of size k. As this allows approximations of MaxCut to a factor better than $16/17$, this shows that we can have an algorithm to answer questions of the form Q only if P = NP, proving the hardness of MLogA. From our previous argument, we now need only consider ordering of G' where the clique nodes are laid out consecutively. Each such ordering naturally gives a cut of the original graph, and the cost of the MLogA objective function is equal to $C + X - \sum_{\{u,v\} \in E(G)} \lg |\pi(u) - \pi(v)|$. Consider the edges in G (corresponding to the missing edges in G') that pass over the clique. Each of these edges will have length at least n^{100}, and hence the cost of the MLogA objective function is smaller than $C + X - k \lg n^{100} = Z$. Hence if the there is a cut of size at least k in G, then the answer to Q is yes.

On the other hand, each of the other missing edges will have length at most n, (the order of G), and hence have cost at most $\lg n$. As the MaxCut cost k is at least $\frac{m}{2}$, if G does not have a cut of size at least $k(1 - \frac{1}{50})$, then the smallest that the MLogA objective function can be is

$$C + X - k \left(1 - \frac{1}{50}\right) \lg(n^{100} + n) - k \lg n > Z,$$

for n sufficiently large. This proves the claim. □

C. HARDNESS OF MLinGapA

We start from the (directed) MLinA problem, which is known to be NP-hard [14]. Let $(G(V, E), k)$ be an MLinA instance where the question is if there is a linear arrangement whose sum of edge lengths is $\le k$. Let $n = |V|$ and $m = |E|$. We create the instance of the (directed) MLinGapA problem as follows.

The graph G' will be composed of $n' = n^{c+1} + 2m$ nodes, for a large enough constant c. For each node $v \in V(G)$, two directed cliques $K_{v,1}$ and $K_{v,2}$ of equal sizes n^c will be created. Also, a clique of n nodes $d_{v,1}, \ldots, d_{v,2n}$ (the "peer nodes" of v) will be created for each $v \in V(G)$. Each node in $K_{v,1}$ and each node in $K_{v,2}$ will point to node $d_{v,i}$ for all $i = 1, \ldots, \deg(v)$ and vice versa.

The set $E(G')$ will contain $2m$ other edges, that we call the "original" edges. In particular, for each edge $(v, u) \in E(G)$ the edges ($d_{v,*}$,$d_{u,*}$) and ($d_{u,*}$,$d_{v,*}$) will be added in such a way that each node $d_{v,*}$ will have outdegree $\le n$.

Given an arbitrary node v, consider the following ordering of its two cliques and of its peer nodes: the first clique laid out on n^c consecutive nodes, followed by its $2n$ peers, and finally the second clique (using a total of $n^c + n$ nodes); we call this ordering *good*. Let F be the cost of the edges of the cliques, and the edges from the cliques to the peers, in this ordering. Note that F can be trivially computed in polynomial time. Now we ask: is there an ordering with MLinGapA cost at most $nF + 3K(2n^c) + 3mn^2 = T$?

If there is an MLinA ordering π of cost at most K, then it is easy to find an MLinGapA ordering of cost at most T. If v is the first node of π, place the first clique of v followed by the peers of v and the second clique of v at the beginning. Then do the same for the second node of π, and so on, until all nodes have been placed. Now we compute the total MLogGapA cost. We have a fixed cost of nF (the ordering of the "nodes structures") for the non-original edges. As for the original edges, note that each node from which an original edge starts has out-degree 1, thus encoding the "gap" induced by that edge has the same cost of encoding its length. Note that the number of cliques that an edge (that had length ℓ in π) passes over in the new ordering is 2ℓ and each such clique has size

n^c. Thus, the cost in the new ordering of the edge will be at most $\ell 2n^c + \xi$, where ξ is an error term that equals n^2 (the total number of peer nodes). Now for any edge of length in the MLinA, there are three gaps of cost at most $\ell 2n^c + n^2$. The total cost will thus be at most $nF + 3K(2n^c) + 3mn^2 = T$.

Now suppose we have an MLinGapA ordering with cost at most T. We then show that there is an MLinA ordering of cost at most K. To show this, we first prove that for each v, the ordering will be such that (i) the distance between any two nodes of $K_{v,1}$ (resp., $K_{v,2}$) will be at most $n^c + n^4$, i.e., the cliques will not be spread out, (ii) the distance between each single peer of v and its nearest node of $K_{v,1}$ ($K_{v,2}$) will be at most n^4.

Suppose this statement is true. We show by contradiction that there must exist an MLinA ordering of cost at most K. First, notice that the minimum cost that we have to pay for the edge between nodes in $V(G')$ that are generated from one node v is at least F, since in any ordering the gaps are of length at least 1 and for any ordering, the sum of the cost of the backward edges is at least that of in the good ordering. Furthermore, it follows from (i) and (ii) that in all valid solutions, for each $v \in V(G)$, each peer node of v must be placed at distance at most $n^c + 2n^4$ from each clique node of v. Now, the number of nodes of the cliques generated by v is $2n^c$ so it is necessary that each peer node has to be placed after at least $n^c - 2n^4$ nodes of one of its two cliques and before $n^c - 2n^4$ nodes of its other (as each peer node has to be at distance $\leq n^c + 2n^4$ from each node of its cliques). Hence, the total cost for any ordering of cost $K+1$ for the MLinA problem is at least $nF + 3(K+1)(2n^c - 4n^4) > T$, a contradiction.

To finish the proof, we show (i): if the maximum distance between two nodes in any of the K_v cliques is $> n^c + n^4$, then the total cost of the ordering is $> T$. Indeed if the distance between any two nodes of K_v is more than $n^c + n^4$, then the cost for the edges between the clique and peer nodes of v will be $\geq F + n^{c+4} - n^c$ where the first term of the sum follows since all the gaps are of length at least one, and there are at least $n^c + n$ backlinks. The second term of the sum is the added cost due to the spread of the clique, which is $\geq n^4$, and since, say, the rightmost node of the clique must go across all the non-clique nodes between clique nodes, for a total of at least $n^c - 1$ links. Hence, the cost of the ordering would be $\geq nF + n^{c+4} - n^c > T$, contradicting the validity of the solution since $K = O(n^2)$.

Finally we have to prove (ii): for each $v \in V(G)$, no peer node of v is at distance $\geq n^4$ from each of the cliques of v. Proceeding as before, we lower bound the cost of the ordering for the edges between the nodes of the peers and the cliques of v. The cost of the ordering will be F plus the cost due to the enlargement of the gaps between v and K_v. Thus, the total cost of the ordering is $\geq nF + n^{c+4} > T$, again a contradiction. □

D. LOWERBOUND: MLogA FOR EXPANDERS

Let $G = (V, E)$ be a simple graph with no isolated nodes and let $|E| = m$. For the edge expansion case, note that for any $S \subseteq V$ such that $|S| = n/2$, we have that $\Omega(n)$ edges are in the cut $(S, G \setminus S)$. If $\Omega(n)$ edges are in the cut then there are $\Omega(n)$ edges of length at least $\sqrt{n}$ because the graph is simple. Each such edge will have cost $\Omega(\log n)$. The edge expansion claim follows. The node expansion case is analogous.

Now suppose that the conductance is $\Omega(1)$. First note that the total degree of the graph is $\geq 2n - 2$, since the graph has to be connected. We claim that there is an $[S, T]$ cut with $m - n + 1 < |S| \leq m$. To see this, consider any line embedding of the graph. Scan the nodes from the left and stop as soon as the total degree of the nodes from the nodes seen so far to the unseen nodes is more than m. Remove the current node, whose degree is $\leq n - 1$, letting S be the set of nodes seen up until the current node. Then, $|S| \geq \Omega(m)$, and at least $\Omega(m)$ edges are in the cut. Hence, at least $\Omega(m)$ of those edges will have length $\Omega(\sqrt{m})$. Thus their total cost will be $\Omega(m \log m) = \Omega(m \log n)$. □

E. SHINGLE ORDERING IN PA MODEL

We need the following concentration inequality, proved in a stronger form by McDiarmid [20].

Theorem 8. *Let X be a non-negative random variable not identically 0, which is determined by an independent random permutation σ, satisfying the following for some $c, r > 0$: interchanging two elements in the permutation can affect X by at most c, and for any s, if $X \geq s$, then there is a set of at most rs coordinates of σ whose values certify that $X \geq s$. Then, for any $0 \leq t \leq E[X]$,*

$$\Pr[|X - E[X]| > t + 60c\sqrt{rE[X]}] \leq 4\exp\left(-\frac{t^2}{8c^2 rE[X]}\right).$$

Using this, we prove Theorem 6. Given an ordering and an arbitrary node v, we say that the edge (v, w) is *shingled* if the position of v is determined by w, i.e., if the minimum out-neighbor of v, according to the random shingle permutation, is w. Also, we say that a node v is *shingled by* w if w is the minimum out-neighbor of v according to the random shingle permutation. A node $v \in S$ is *good* if there is a node $v' \in S$, $v \neq v'$, such that v and v' are shingled by the same node.

Let X be the number of good nodes. By property (ii), each node v in S has a common out-neighbor with at least another node in S. As all nodes in S have outdegree bounded by k and hence with probability $1/(2k-1) \geq 1/(2k)$, one of their common out-neighbors will be the smallest of both their out-neighborhoods according to the random shingle permutation, i.e., they will be shingled together. Thus, $E[X] \geq |S|/(2k)$.

We will first assume $X \geq \Omega(|S|)$ whp, and claim that at least $\Omega(|S|)$ edges are copied. Indeed, partition the good nodes in S according to their shingling node. Each part will contain at least two nodes (by the definition of good nodes), and in each part all the nodes, but the first, will copy their edge pointing to their shingling node. Thus, the fraction of good nodes in a part copying at least one of their edges is $\geq 1/2$. The claim follows.

To obtain the high probability lower bound on X we use Theorem 8. Note that here we only have the random shingle permutation (i.e., no random trials). In order to use Theorem 8 we have to choose suitable c, r. Using property (iv), we can upperbound the effect on X of a swap of two elements with $c = 2n^{\frac{1}{2}-\epsilon}$. To see this, the only nodes that can change their good or bad statuses are the in-neighbors of the two swapped nodes and these can be upperbounded by $2n^{\frac{1}{2}-\epsilon}$. If a node $v \in S$ is good, then there exists a node $v' \in S$ with the same shingling node w. Thus, to certify that v is good it suffices to reveal the positions of the nodes in $N^+(v) \cup N^+(v')$ — v is good iff w is the first of the nodes in $N^+(v) \cup N^+(v')$. As the degrees of v, v' are bounded by k, we can safely choose $r = 2k$. By plugging c, r into Theorem 8 we get the high probability lower bound on X. □