Compression of Graphical Structures

Yongwook Choi and Wojciech Szpankowski
Department of Computer Science
Purdue University
W. Lafayette, IN 47907, U.S.A.
Email: ywchoi@purdue.edu, spa@cs.purdue.edu

***Abstract*— F. Brooks argues in [3] there is "no theory that gives us a metric for information embodied in structure". Shannon himself alluded to it fifty years earlier in his little known 1953 paper [14]. Indeed, in the past information theory dealt mostly with "conventional data", be it textual data, image or video data. However, databases of various sorts have come into existence in recent years for storing "unconventional data" including biological data, web data, topographical maps, and medical data. In compressing such data structures, one must consider two types of information: the information conveyed by the *structure itself*, and then the information conveyed by the data labels implanted in the structure. In this paper, we attempt to address the former problem by studying information of graphical structures (i.e., unlabeled graphs). In particular, we consider Erdös-Rényi graphs $\mathcal{G}(n, p)$ over n vertices in which edges are added randomly with probability p. We prove that the *structural entropy* of $\mathcal{G}(n, p)$ is $\binom{n}{2}h(p) - \log n! + o(1) = \binom{n}{2}h(p) - n \log n + O(n)$, where $h(p) = -p \log p - (1 - p) \log(1 - p)$ is the entropy rate of a conventional memoryless binary source. Then, we design a two-stage encoding that optimally compress unlabeled graphs up to the first two leading terms of the structural entropy.**

I. Introduction

In 1948 Shannon introduced a metric for information launching the field of information theory. However, as observed by Brooks [3] and others [12], [17], there is no theory that gives us a metric for information embodied in structure. Shannon himself in his 1953 little known paper [14] argued for an extension of information theory to "non-conventional data" (i.e., lattices). Indeed, data is increasingly available in various forms (e.g., sequences, expressions, interactions, structures) and in exponentially increasing amounts. For example, in biology large amounts of data are now in public domain on gene regulation, protein interactions, and metabolic pathways. Most of such data is multidimensional and context dependent. Therefore, it necessitates novel theory and efficient algorithms to extract meaningful information from non-conventional data structures. Typically, a data file of this new type (e.g., biological data, topographical maps, medical data, volumetric data) is a "data structure" conveying a "shape" and consisting of labels implanted in the structure. In understanding such data structures, one must take into account two types of information: the information conveyed by the structure itself and the data labels implanted in the structure. In this paper, we address the former problem by studying information of graphical structures.

As the first step to understanding information in structure, we restrict our attention to structures on graphs. More specifically, we study unlabeled graphs generated by a memoryless source known as the Erdös-Rényi model [2] in which edges are added randomly with probability p. This model induces a probability distribution on structures from which Shannon entropy can be computed giving us a fundamental limit on lossless unlabeled graph compression. We prove that this *structural entropy* is

$$\binom{n}{2}h(p) - \log n! + o(1) = \binom{n}{2}h(p) - n \log n + O(n),$$

where n is the number of vertices and $h(p) = -p \log p - (1 - p) \log(1 - p)$ is the entropy rate of a conventional memoryless binary source. Then, we design and analyze an asymptotically optimal encoding algorithm that achieves the compression rate

$$\binom{n}{2}h(p) - n \log n + O(n)$$

that matches on average the lower bound up to the first two leading terms. Our algorithm is a two-stage scheme. First, it encodes a structure into two binary strings that are next compressed using an arithmetic encoder. Experimental results on real data networks confirm efficiency and utility of our algorithm.

Literature on graphical structure compression is scarce. The problem of succinct representation of general unlabeled graphs was introduced more than twenty years ago by Turan [18]. Naor [10] provided such a representation when all unlabeled graphs (or structures) are equally probable. There also have been some heuristic methods for real-world graphs compression including Adler and Mitzenmacher [1], who proposed an encoding technique for web graphs, and similar idea has been used in [15] for compressing sparse graphs. Recently, attention has been paid to grammar compression for some data structures: Peshkin [11] proposed an algorithm for a graphical extension of the one-dimensional SEQUITUR compression method. However, SEQUITUR is known not to be asymptotically optimal [13]. Therefore, the Peshkin method already lacks asymptotic optimality in the 1D case. To the best of our knowledge our algorithm is the first provable asymptotically optimal compression scheme for graphical structures.

II. Entropy of a Random Structure

In a random graph model $\mathcal{G}$, the vertex set V consists of n distinguishable vertices and edges between vertices are added at random. In this setting, the graph entropy $H_{\mathcal{G}}$ is defined as

$$H_{\mathcal{G}} = \mathbf{E}[-\log P(G)] = -\sum_{G\in\mathcal{G}} P(G)\log P(G),$$

where $P(G)$ is the probability of a graph G. Throughout the paper, the base of the logarithm is 2.

In this study, we investigate graphical structural entropy. For this purpose, it is convenient to introduce the unlabeled version of a random graph model that we shall call a *random structure model*. In such a model, graphs are generated in the same manner as in $\mathcal{G}$, but they are thought of as unlabeled graphs and those having "the same structure" are considered to be indistinguishable even if their labeled versions are different. A set of all structures will be denoted by $\mathcal{S}$. For a given structure (or an unlabeled graph) $S \in \mathcal{S}$, the probability of S can be computed as $P(S) = \sum_{G\cong S, G\in\mathcal{G}} P(G)$. Here $G \cong S$ means that G and S have the same structure, that is, S is *isomorphic* to G (two labeled graphs G_1 and G_2 are called isomorphic if and only if there is a one-to-one map from $V(G_1)$ onto $V(G_2)$ that preserves the adjacency.) If all isomorphic labeled graphs have the same probability, then for any labeled graph $G \cong S$

$$P(S) = N(S)\cdot P(G), \qquad (1)$$

where $N(S)$ is the number of different labeled graphs that have the same structure as S. The *structural entropy* $H_{\mathcal{S}}$ of a random structure $\mathcal{S}$ can be defined then as

$$H_{\mathcal{S}} = \mathbf{E}[-\log P(S)] = -\sum_{S\in\mathcal{S}} P(S)\log P(S),$$

where the summation is over all distinct structures.

Example: In Figure 1(left), we have all different graphs built on three vertices. In Figure 1(right), we have all different structures that can be generated by $\mathcal{S}$ with $N(S_1) = N(S_4) = 1$ and $N(S_2) = N(S_3) = 3$. ■

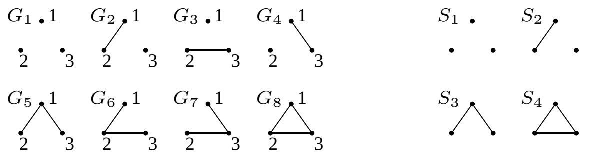


Fig. 1. All different graphs and structures with three vertices

In order to compute the probability of a given structure S, one needs to estimate $N(S)$, representing the number of ways to construct a given structure S. For this, we need to consider the symmetries or automorphisms of a graph. An *automorphism* of a graph G is an adjacency preserving permutation of vertices of G. The collection $\mathrm{Aut}(G)$ of all automorphisms of G is called *the automorphism group* of G. In group theory, it is well known that [7]

$$N(S) = \frac{n!}{|\mathrm{Aut}(S)|}. \qquad (2)$$

We also easily observe that $1 \le |\mathrm{Aut}(S)| \le n!$.

Example: In Figure 2, the graph G on the left has exactly four automorphisms, that is, in the usual cyclic permutation representation: $(v_1)(v_2)(v_3)(v_4)$, $(v_1)(v_4)(v_2v_3)$, $(v_1v_4)(v_2)(v_3)$, and $(v_1v_4)(v_2v_3)$. For example, $(v_1)(v_4)(v_2v_3)$ stands for a permutation π such that $\pi(v_1) = v_1$, $\pi(v_4) = v_4$, $\pi(v_2) = v_3$, and $\pi(v_3) = v_2$. Thus, by (2), G has $4!/4 = 6$ different labelings as shown on the right. ■

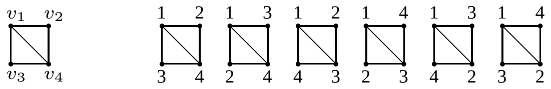


Fig. 2. The six different labelings of a graph

With these preliminary definitions, we are now in the position to present a relationship between $H_{\mathcal{G}}$ and $H_{\mathcal{S}}$.

Lemma 1: If all isomorphic graphs have the same probability, then

$$H_{\mathcal{S}} = H_{\mathcal{G}} - \log n! + \sum_{S\in\mathcal{S}} P(S)\log|\mathrm{Aut}(S)|.$$

Proof. Observe that for any $\mathcal{G}$ and $\mathcal{S}$ the entropy $H_{\mathcal{G}}$ becomes

$$-\sum_{S\in\mathcal{S}}\sum_{\substack{G\cong S,\\ G\in\mathcal{G}}} \frac{P(S)}{N(S)}\log\frac{P(S)}{N(S)} = -\sum_{S\in\mathcal{S}} P(S)\log\frac{P(S)}{N(S)}$$

$$= H_{\mathcal{S}} + \log n! - \sum_{S\in\mathcal{S}} P(S)\log|\mathrm{Aut}(S)|.$$

This proves the lemma. ■

III. Main Results

In order to develop further the idea of information in a random structure, we focus on the binomial random graph model due to Erdös and Rényi [2]. In this model $\mathcal{G}(n,p)$, given a real number p ($0 \le p \le 1$), graphs are generated randomly on the vertex set $V = \{1, 2, \cdots, n\}$ with edges chosen independently with probability p. If a graph G in $\mathcal{G}(n,p)$ has k edges, then $P(G) = p^k q^{\binom{n}{2}-k}$ where $q = 1-p$. Let $\mathcal{S}(n,p)$ be the random structure model corresponding to $\mathcal{G}(n,p)$, that is, the unlabeled version of $\mathcal{G}(n,p)$. By (1) if $S \in \mathcal{S}(n,p)$ has k edges, then $P(S) = N(S)\cdot p^k q^{\binom{n}{2}-k}$.

A. Structural Entropy

To proceed we need to observe some important property of $\mathcal{S}(n,p)$ (or equivalently, $\mathcal{G}(n,p)$) - *asymmetry*. A graph is said to be *asymmetric* if its automorphism group does not contain any permutation other than the identity (i.e., $(v_1)(v_2)\cdots(v_n)$); otherwise it is called *symmetric*. It is known that almost every graph from $\mathcal{G}(n,p)$ is asymmetric [6], [9]. In the sequel, we write $X \ll Y$ to mean $X = o(Y)$ when $n \to \infty$.

Lemma 2 (Kim, Sudakov, and Vu, 2002): For all p satisfying $\frac{\ln n}{n} \ll p$ and $1-p \gg \frac{\ln n}{n}$ (i.e., both the graph and its complement graph are connected graphs with high probability.), a random graph $G \in \mathcal{G}(n,p)$ is symmetric with probability $O\left(n^{-w}\right)$ for any positive constant $w > 1$.

Using this property, we are in the position to present our first main result, namely the structural entropy of $\mathcal{G}(n,p)$.

Theorem 1: Let $H_{\mathcal{S}}$ be the entropy of $\mathcal{S}(n,p)$, that is, the structural entropy of $\mathcal{G}(n,p)$. Then, for large n and all p satisfying $\frac{\ln n}{n} \ll p$ and $1-p \gg \frac{\ln n}{n}$,

$$H_{\mathcal{S}} = \binom{n}{2}h - \log n! + O\left(\frac{\log n}{n^{\alpha}}\right), \qquad \alpha > 0,$$

where $h := h(p) = -p\log p - (1-p)\log(1-p)$.

Proof. Let us first compute the entropy $H_{\mathcal{G}}$ of $\mathcal{G}(n,p)$. In $\mathcal{G}(n,p)$, $m = \binom{n}{2}$ distinct edges are independently selected with probability p, and thus there are 2^m different labeled graphs. That is, each graph instance can be considered as a binary sequence X of length m. Thus,

$$H_{\mathcal{G}} = -\mathbf{E}[\log P(X_1^m)] = -m\mathbf{E}[\log P(X_1)] = \binom{n}{2}h.$$

Let $A = \sum_{S\in\mathcal{S}} P(S)\log|\mathrm{Aut}(S)|$. Since $|\mathrm{Aut}(S)| = 1$ for all asymmetric S,

$$\begin{aligned} A &= \sum_{S\in\mathcal{S}(n,p) \text{ and } S \text{ is symmetric}} P(S)\log|\mathrm{Aut}(S)| \\ &\le \sum_{\substack{S\,\in\,\mathcal{S}(n,p),\\ S \text{ is symmetric}}} P(S)\cdot n\log n \quad (\because |\mathrm{Aut}(S)| \le n! \le n^n) \\ &= O\left(\frac{\log n}{n^{w-1}}\right) \quad \text{for any constant } w > 1 \quad \text{(by Lemma 2).} \end{aligned}$$

Lemma 1 completes the proof. ■

By Shannon's source coding theorem, we conclude that the entropy computed in Theorem 1 is the fundamental limit on the lossless compression of structures $\mathcal{S}(n,p)$. In the next section, we design an asymptotically optimal compression algorithm matching the first two leading terms of the structural entropy.

B. Asymptotically Optimal Compression Algorithm

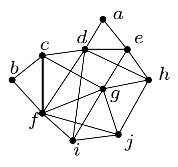
In this section, we present our algorithm that encodes structures (or unlabeled graphs). For a given unlabeled graph G, our algorithm encodes G first into two binary sequences and then compress them by an arithmetic encoder. We shall show in Theorem 2 that the proposed algorithm is asymptotically optimal up to the first two leading terms.

To describe the algorithm precisely, we need some definitions and notations. An *ordered partition* of a set X is a sequence of nonempty subsets of X such that every element in X is in exactly one of these subsets. For example, one ordered partition of $\{a,b,c,d,e\}$ is $\{a,b\},\{e\},\{c,d\}$ that is denoted by $ab/e/cd$. It is equivalent to $ba/e/dc$, but distinct from $e/ab/cd$. Given an ordered partition P of a set X, we also define an order on the elements of X as follows: $a < b$ in P if the subset containing a precedes the subset containing b in P. For example, $a < c$ and $e < c$ in $P = ab/e/cd$, but $e \not< a$. An ordered partition P_1 of a set X is called *finer* than ordered partition P_2 of X if the following two conditions hold: (1) every element (i.e., subset of X) of P_1 is a subset of some element of P_2, and (2) for all $a,b \in X$, $a < b$ in P_1 if $a < b$ in P_2. For example, both $a/b/e/cd$ and $ab/e/d/c$ are finer than $ab/e/cd$. Finally, a subtraction of an element from an ordered partition gives us another ordered partition (e.g., for $P = ab/e/cd$ we find that $P - c$ and $P - e$ are $ab/e/d$ and ab/cd, respectively).

Now we describe in details the proposed algorithm. It runs in n steps. During the course of the algorithm, one ordered partition P of a subset of $V(G)$ is maintained. Let P_i be the ordered partition after the i-th step. At the beginning, $P_0 = V(G)$. In the i-th step, one vertex v is removed randomly from the first subset in P_{i-1}. Then, for each subset U in $P_{i-1} - v$ (in its order), we encode the *number of neighbors* of v in U using $\lceil\log(|U|+1)\rceil$ bits. After that, $P_{i-1} - v$ becomes a finer partition P_i such that for each subset U in $P_{i-1} - v$, U is divided into two smaller subsets U_1 and U_2, and U_1 precedes U_2 in P_i where U_1 is the set of all neighbors of v in U and U_2 is the set of all non-neighbors of v in U. These steps are repeated until P becomes empty.

While the algorithm is running, the binary encodings of the number of neighbors are concatenated in the order they are generated. In the course of the algorithm, we separately maintain two types of encodings - those of length more than one bits (i.e., for subsets $|U| > 1$) and those of exactly one bit (i.e., for subsets $|U| = 1$). The former type of encodings are appended to a binary sequence B_1. Similarly, the latter type of encodings form a binary sequence B_2.

Example: Figure 3 shows the details of our encoding algorithm step by step. In the table, k denotes the step number, and v denotes the randomly chosen vertex in each step. All encodings whose length is bigger than one (denoted by *italic* font) are appended to B_1. The other encodings are appended to B_2. After ten steps, B_1 and B_2 are *01001101000011101 01* and 1001011000000101, respectively. ■



k	v	$P_{k-1} - v$	encoding	P_k
0				$abcdefghij$
1	i	$abcdefghj$	*0100*	$dfgj/abceh$
2	f	$dgj/abceh$	*11, 010*	$dgj/bc/aeh$
3	d	$gj/bc/aeh$	*00, 01, 11*	$gj/c/b/aeh$
4	j	$g/c/b/aeh$	1, 0, 0, *01*	$g/c/b/h/ae$
5	g	$c/b/h/ae$	1, 0, 1, *01*	$c/b/h/e/a$
6	c	$b/h/e/a$	1, 0, 0, 0	$b/h/e/a$
7	b	$h/e/a$	0, 0, 0	$h/e/a$
8	h	e/a	1, 0	e/a
9	e	a	1	a
10	a			

Fig. 3. An example for our encoding algorithm, given the graph on the left

We can easily observe that B_2 is nothing but a binary sequence generated by a binary memoryless source(p) with p being the probability of generating '1' if the input graph is generated by $\mathcal{S}(n,p)$.

At the end of our encoding algorithm, B_1 and B_2 are compressed to $\hat{B}_1$ and $\hat{B}_2$ by an adaptive binary arithmetic encoder [4]. The algorithm also needs the number of vertices n. As easy to see, the computational complexity is $O(n^2)$.

Now we describe our *decoding algorithm* which from n, $\hat{B}_1$, and $\hat{B}_2$ constructs a graph isomorphic to the original graph. First we restore B_1 and B_2 by decompressing $\hat{B}_1$ and $\hat{B}_2$. Then, we create a graph G having n vertices and no edges. The general framework of our decoding algorithm is very similar to that of our encoding algorithm. Again, one ordered partition P of a subset of $V(G)$ is maintained. Let P_i be the ordered partition after i-th step. At the beginning, $P_0 = V(G)$. In i-th step, we remove any vertex v from the first subset in P_{i-1}. Then, for each subset U in $P_{i-1} - v$ (in its order), we extract the first $\ell = \lceil\log(|U|+1)\rceil$ bits from either B_1 (if $|U| > 1$) or B_2 (if $|U| = 1$). Let k be the number that ℓ bits represent.

TABLE I

THE AVERAGE LENGTH OF ENCODINGS (IN BITS)

	Networks	# of nodes	# of edges	our algorithm	adjacency matrix, $\binom{n}{2}$	adjacency list, $e\lceil\log n\rceil$	arithmetic coding
Real-world	US Airports	332	2,126	8,118	54,946	19,134	12,991
	Protein interaction (Yeast)	2,361	6,646	46,912	2,785,980	79,752	67,488
	Collaboration (Geometry)	6,167	21,535	115,365	19,012,861	279,955	241,811
	Collaboration (Erdös)	6,935	11,857	62,617	24,043,645	154,141	147,377
	Genetic interaction (Human)	8,605	26,066	221,199	37,018,710	364,924	310,569
	Internet (AS level)	25,881	52,407	301,148	334,900,140	786,105	396,060
Random	$\mathcal{S}(n,p)$	1,000	$p = 0.001$	2,353	499,500	5,000	5,717
	$\mathcal{S}(n,p)$	1,000	$p = 0.01$	34,431	499,500	50,019	40,414
	$\mathcal{S}(n,p)$	1,000	$p = 0.1$	227,077	499,500	499,367	234,231
	$\mathcal{S}(n,p)$	1,000	$p = 0.3$	432,654	499,500	1,498,467	440,210

Then we select any k vertices in U and make an edge between v and each of those k vertices. After that, $P_{i-1} - v$ becomes a finer partition P_i in the same way as our encoding algorithm. These steps are repeated until P becomes empty.

To measure the performance of our algorithm, let $L(S)$ be the length of the encoding generated by our algorithm, that is, $|\hat{B}_1| + |\hat{B}_2|$. In Section IV we sketch a proof that $\mathbf{E}_{\mathcal{G}}[L(S)]$ matches the first two terms in the structural entropy of $\mathcal{G}(n,p)$.

Theorem 2: The average length $\mathbf{E}_{\mathcal{G}}[L(S)]$ of the compressed sequence does not exceed

$$\binom{n}{2}h - n\log n + (c + \Phi(\log n))\,n + O\left(n^{1-\eta}\right),$$

where $h := h(p)$, c is an explicitly computable constant, $\Phi(\log n)$ is a fluctuating function with a small amplitude, and η is some positive constant.

C. Experimental Results

In order to test our graphical structure compression algorithm on real data, we apply it to both random and real-world networks including biological, social, and technological networks. Table I summarizes the results. For comparison, in the table we list the lengths of three other encodings of graphs, namely, the usual implementations of adjacency matrix of $\binom{n}{2}$ bits, and adjacency list of *at least* $e\lceil\log n\rceil$ bits (normally, $2e\lceil\log n\rceil$ bits) where n is the number of vertices and e is the number of edges. Finally, in the last column of the table we apply the arithmetic encoder to the adjacency matrix.

For "collaboration graphs" of Table I our algorithm achieves more than twice better compression than the standard arithmetic encoder. This seems to be a consequence of a small value of p for these graphs, even if the "collaboration graphs" are not in $\mathcal{G}(n,p)$ but rather generated by a *power law* distribution (we still expect our analysis applies to this model of graph generations). Consider again the structural entropy $H_{\mathcal{S}}$ of the $\mathcal{G}(n,p)$ model when $p \to 0$ satisfying conditions of Theorem 1. Let $p \sim \omega(n)(\log n/n)$ for slowly growing $\omega(n) \to \infty$ as $n \to \infty$. In this case

$$h(p) \sim \omega(n)\frac{\log^2 n}{n},$$

and therefore the structural entropy becomes

$$H_{\mathcal{S}} \sim \frac{1}{2}(n-1)\omega(n)\log^2 n - n\log n + O(n).$$

Clearly, the second leading term $n\log n$ plays significant role in the compression of such graphs.

IV. ANALYSIS

In this section, we analyze the performance of our compression algorithm by first computing the expected lengths of B_1 and B_2, and ultimately proving Theorem 2.

In order to analyze our algorithm, we conveniently introduce a binary tree that better captures the progress of the algorithm. Given a graph G on n vertices, the binary tree T_n is built as follows. At the beginning, the root node contains all n graph vertices, $V(G)$, that one can also visualize as n balls. Then one graph vertex (ball) v is randomly removed from n graph vertices of the root node. The other $n-1$ graph vertices move down to the left or right depending whether they are adjacent vertices in G to v or not; adjacent vertices go to the left child node and the others go to the right child. We create a new child node in T_n if there is at least one graph vertex in that node. At this point, the tree is of height 1 with $n-1$ vertices in the nodes at level 1. Similarly, in the i-th step, we randomly remove one graph vertex (ball) v from the leftmost node at level $i-1$. The other graph vertices at level $i-1$ move down to the left or right depending whether they are adjacent to v or not. We repeat these steps until all vertices are removed.

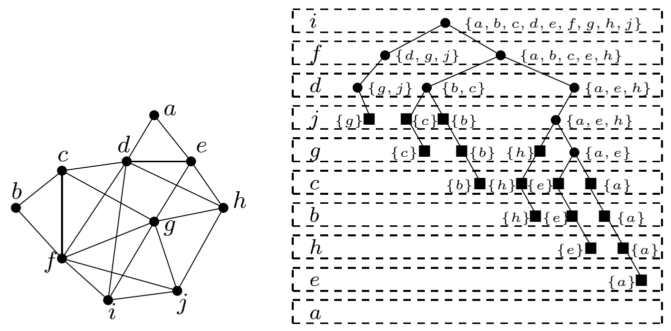


Fig. 4. A graph and one of its corresponding binary trees

The construction of the tree and the progress of the algorithm is presented in Figure 4. In Figure 4(right), selected graph vertices are shown on the left. At each level, the subsets of graph vertices, after removing the chosen vertex, are shown next to the nodes. In this example, the same vertex are selected as in Figure 3. We observe that the subsets of graph vertices at each level (from the left to the right) are the same as the subsets in each step of our algorithm in Figure 3.

Let N_x denote the number of graph vertices (balls) that pass through node x (excluding the vertex that is removed at x, if any.) We observe that our algorithm needs to encode the number of neighbors of a graph vertex among N_x vertices for each node x in T_n. This requires $\lceil \log(N_x+1) \rceil$ bits. Then by the construction

$$|B_1| = \sum_{x \in T_n \text{ and } N_x > 1} \lceil \log(N_x+1) \rceil,$$

and

$$|B_2| = \sum_{x \in T_n \text{ and } N_x = 1} \lceil \log(N_x+1) \rceil = \sum_{x \in T_n \text{ and } N_x = 1} 1.$$

In Figure 4(right) the summations for $|B_1|$ and $|B_2|$ are over all circle-shaped nodes and over all square-shaped nodes, respectively.

We first evaluate $\mathbf{E}[|B_1|]$. We shall prove in the journal version of this paper that

$$\mathbf{E}[|B_1|] \le x_n$$

where x_n satisfies $x_0 = x_1 = 0$ and for $n \ge 2$

$$x_n = \lceil \log(n+1) \rceil + \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} (x_k + x_{n-k}).$$

The above recurrence can be solved using analytic techniques such as generating functions, Mellin transform, and Poissonization [16]. This will lead to the following bound on $|B_1|$.

Theorem 3: For large n,

$$\mathbf{E}[|B_1|] \le \frac{1}{h} (\beta + \Phi(\log n)) n + O\left(n^{1-\eta}\right),$$

where $h := h(p)$, η is a positive constant,

$$\beta = \log e \cdot \sum_{b \ge 2} \frac{\lceil \log(b+1) \rceil}{b(b-1)} = 3.760 \cdots,$$

and $\Phi(\log n)$ is a fluctuating function for $\log p / \log q$ rational with small amplitude and zero otherwise.

Now we estimate the average of $|B_2|$. We shall first prove that

$$\mathbf{E}[|B_2|] = \frac{n(n-1)}{2} - b_n$$

for some b_n. Then we observe that $b_n \ge y_n - n$ for some y_n satisfying $y_0 = 0$ and for $n \ge 0$

$$y_{n+1} = n + \sum_{k=0}^{n} \binom{n}{k} p^k q^{n-k} (y_k + y_{n-k}).$$

In fact, y_n represents the expected path length in a digital search tree over n strings [8], and we adopt here the solution from [8]. In conclusion, we arrive at the following result.

Theorem 4: For large n,

$$\mathbf{E}[|B_2|] \le \frac{n(n-1)}{2} - \frac{n}{h} \log n$$

$$+ \frac{n}{h} \left(h - \frac{h_2}{h} - \gamma + 1 + \alpha - \delta_o(\log n) \right) - \frac{1}{h} \log n + O(1),$$

where $h := h(p)$, $\gamma = 0.577 \cdots$ is the Euler constant, $h_2 = p \log^2 p + q \log^2 q$,

$$\alpha = - \sum_{k=1}^{\infty} \frac{p^{k+1} \log p + q^{k+1} \log q}{1 - p^{k+1} - q^{k+1}},$$

and $\delta_o(\log n)$ is a fluctuating function for $\log p / \log q$ rational with small amplitude and zero otherwise.

Finally, we compute the total expected length of the encoding by observing that the arithmetic coder can compress on average a file of size m up to [4], [5] $mh + \frac{1}{2} \log m + O(1)$, where h is the entropy rate of the binary source.

Acknowledgment

This work was supported in part by the NSF Grants CCF-0513636, DMS-0503742, DMS-0800568, and CCF-0830140, NIH Grant R01 GM068959-01, NSA Grant H98230-08-1-0092, EU Project No. 224218 through Poznan University of Technology, and the AFOSR Grant FA8655-08-1-3018.

References

[1] M. Adler and M. Mitzenmacher, Towards compressing web graphs, *In Proc. of the IEEE Data Compression Conference*, 203–212, 2001.
[2] B. Bollobas, *Random Graphs*, Cambridge University Press, Cambridge, 2001.
[3] F.P. Brooks Jr, Three great challenges for half-century-old computer science, *Journal of the ACM*, 50(1), 25–26, 2003.
[4] T.M. Cover and J.A. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, 2006.
[5] M. Drmota, H.-K. Hwang, and W. Szpankowski, Precise average redundancy of an idealized arithmetic coding, *Proc. Data Compression Conference*, 222-231, 2002.
[6] P. Erdös and A. Rényi, Asymmetric graphs, *Acta Math. Acad. Sci. Hungar.* 14, 295–315, 1963.
[7] F. Harary and E.M. Palmer, *Graphical Enumeration*, Academic Press, 1973.
[8] P. Jacquet and W. Szpankowski, Asymptotic behavior of the Lempel-Ziv parsing scheme and digital search trees, *Theoretical Computer Science*, 144(1&2), 161–197, 1995.
[9] J.H. Kim, B. Sudakov, and V.H. Vu, On the asymmetry of random regular graphs and random graphs, *Random Structures and Algorithms*, 21(3-4), 216–224, 2002.
[10] M. Naor, Succinct representation of general unlabeled graphs, *Discrete Applied Mathematics*, 28(3), 303–307, 1990.
[11] L. Peshkin, Structure induction by lossless graph compression, *In Proc. of the IEEE Data Compression Conference*, 53–62, 2007.
[12] N. Rashevsky. Life, information theory, and topology. *Bull. Math. Biophysics*, 17:229–235, 1955.
[13] S.A. Savari, Compression of words over a partially commutative alphabet, *IEEE Trans. on Information Theory*, 50, 1425-1441, 2004.
[14] C. Shannon. The lattice theory of information. *IEEE Transaction on Information Theory*, 1:105–107, 1953.
[15] J. Sun, E.M. Bollt, and D. Ben-Avraham, Graph compression–save information by exploiting redundancy, *Journal of Statistical Mechanics: Theory and Experiment*, P06001, 2008.
[16] W. Szpankowski, *Average Case Analysis of Algorithms on Sequences*, John Wiley & Sons, New York, 2001.
[17] E. Trucco. A note on the information content of graphs. *Bull. Math. Biophysics*, 18:129–135, 1956.
[18] Gy. Turan, On the succinct representation of graphs, *Discrete Applied Mathematics*, 8(3), 289–294, 1984.