Succinct Representation of General Unlabeled Graphs

Moni Naor *

IBM Alamden Research Center

650 Harry Road

San Jose CA 95120

Abstract

We show that general unlabeled graphs on n nodes can be represented by $\binom{n}{2} - n\log_2 n + O(n)$ bits which is optimal up to the $O(n)$ term. Both the encoding and decoding require linear time.

1 Introduction

Assume we are given an unlabeled simple graph G on n nodes, and we are to find a short representation of G. This is useful when trying to save storage or when transmitting the graph. An adjacency matrix representation of a graph requires $\binom{n}{2}$ bits, which is the best possible bound for labeled graphs. Let C_n denote the class of unlabeled graphs on n nodes. $\lceil \log_2 |C_n| \rceil$ is a lower bound on the number of bits in G's representation. From [HP] We know that $\log_2 |C_n| = \binom{n}{2} - n\log n + O(n)$. If efficiency considerations in finding the representation are completely ignored, then we can achieve this bound by deciding on some fixed enumeration of all unlabeled graphs on n nodes; given a graph G, find its rank in the enumeration. Conversely, when given a rank we can enumerate all graphs until we reach the rank.

The goal of this paper is to give efficient methods of finding a succinct representation of a graph. We assume that the (unlabeled) graph G is given by an adjacency matrix of an arbitrary labeling

*This work was done while the author was at UC Berkeley and was supported by NSF grants DCR 85-13926 and CCR 88-13632.

of its nodes. This problem was introduced by Turan in [T]. More formally, we are looking for a pair of mappings $(CODE_n, ENCODE_n)$ satisfying:

- $CODE_n : \{0,1\}^{\binom{n}{2}} \mapsto \{0,1\}^*$
- $DECODE_n : \{0,1\}^* \mapsto \{0,1\}^{\binom{n}{2}}$
- Given a graph G with adjacency matrix $A(G)$, $DECODE_n(CODE_n(A(G)))$ should be the adjacency matrix of a graph isomorphic to G.
- $CODE_n$ and $DECODE_n$ are polynomial time computable.

The length of a representation is the function $l(n) = max|CODE_n(G)|$.

Turan [T] commented that there is an efficient method for representing general unlabeled graphs with strings of length $\binom{n}{2} - 1/8n \log n + O(n)$, based on Ramsey theory [GRS]. Here we prove:

Theorem: There is a representation of simple unlabeled graphs satisfying $l(n) = \binom{n}{2} - n \log_2 n + O(n)$, where both $CODE_n$ and $DECODE_n$ are computable in linear time.

This is the best possible up to the $O(n)$ term as mentioned at the beginning of the section.

The key idea of the representation is to encode some bits implicitly by a permutation on the neighborhoods of half of the nodes of the graph. In Section 2 we describe a method for encoding information in a permutation and in Section 3 we show how to use the encoding to achieve the bound in the theorem. Section 4 contains remarks and open problems.

We will write $CODE_n(G)$ instead of $CODE_n(A(G))$ for an unlabeled graph G, when we do not care which of the adjacency matrix representations of G is used as an input for $CODE_n$.

2 Encoding Information in a Permutation

Suppose we are given t numbers $x_1, x_2 \ldots, x_t$ such that $x_1 < x_2 \ldots < x_t$ and a sequence of k bits $B = b_1, b_2, \ldots b_k$ such that $k \leq \log t!$. The x_i's are to be represented explicitly in some permutation τ of their increasing order. We would like to represent B using that permutation, that is given τ we should be able to determine B efficiently.

To achieve this we will use a standard method of random generation of permutations due to Lehmer (see [D]). There is a $1-1$ correspondence between permutations on t elements and sequences of $t-1$ integers of the form $a_1, a_2, ..a_{t-1}$ where $0 \leq a_i \leq t-i$. A sequence $a_1, a_2, ..a_{t-1}$ determines the permutation $\tau = \tau_1 \cdot \tau_2 \ldots \cdot \tau_{t-1}$ where τ_i is the transposition that swaps i and $a_i + 1$, and the multiplication is the product of permutations.

Any integer $1 \leq A \leq t!$ defines a sequence $a_1, a_2, ..a_{t-1}$ by having

$$\begin{aligned} a_1 &= \lfloor A/(t-1)! \rfloor \\ A_1 &= A \bmod (t-1)! \\ a_2 &= \lfloor A_1/(t-2)! \rfloor \\ A_2 &= A_1 \bmod (t-2)! \\ &\vdots \\ a_{t-1} &= A_{t-2} \bmod 2 \end{aligned}$$

If we treat B as an integer in $[1, t!]$ then we get a corresponding τ which encodes B. We then order $x_1, x_2, \ldots x_t$ according to τ.

Decoding: given a sequence $x_{\tau^{-1}(1)}, x_{\tau^{-1}(2)}, \ldots x_{\tau^{-1}(n)}$, determine τ by sorting it. τ can be decomposed uniquely into $\tau_1 \cdot \tau_2 \ldots \cdot \tau_{t-1}$ where τ_i is a transposition that swaps i and $a_i + 1$, $1 \leq a_i \leq t-i$. B is then set to be $a_1(t-1)! + a_2(t-2)! + \ldots + a_{t-1} + 1$.

This method, though involving a number of operations which is linear in t and achieving the best possible bound, might not be considered efficient, since the numbers involved in the computation when determining the sequence $a_1, a_2, ..a_{t-1}$ are t bits long. To get around this problem, we will sacrifice some encoding power. We divide B into $t-1$ successive blocks $B_1, B_2, \ldots B_{t-1}$, where the number of bits in B_i is $\lfloor \log_2 t - i \rfloor$. Each B_i will encode a_i directly. Let $f(t) = \sum_{i=1}^{t-1} \lfloor \log_2 (t-i) \rfloor$. This method enables us to encode bit sequences of length $k \leq f(t)$.

Claim: $f(t) = t \log_2 t - O(t)$.

Recall that $\log_2 t! = t \log_2 t - O(t)$.

$$f(t) = \sum_{i=1}^{t-1} \lfloor \log_2 (t-i) \rfloor \geq \sum_{i=1}^{t-1} \log_2 (t-i) - 1$$

$$\geq \sum_{i=1}^{t-1} \log_2 i - t = \log_2 t! - t = t \log_2 t - O(t).$$

3 The Representation

We describe the representation for n a power of 2, it can be easily generalized to any n. $CODE_n$ determines some ordering of the nodes, which is the order of the nodes $DECODE_n(CODE_n(A(G)))$ produces. A given a graph G with n nodes to be coded is partitioned arbitrarily into two subgraphs on $\frac{n}{2}$ nodes, G_1 and G_2. The nodes of G_1 will appear in indices $1, \ldots, \frac{n}{2}$ of $DECODE_n(CODE_n(A(G)))$, and the nodes of G_2 will appear in indices $\frac{n}{2}+1, \ldots, n$. After the partition, some adjacency matrix representation of G_1 is fixed by computing $CODE_{n/2}(G_1)$ recursively.

For every $v \in G_2$ let Y_v be the binary vector of length $\frac{n}{2}$ representing the neighborhood of v in G_1, that is $Y_v[i] = 1$ if and only if there is an edge between v and the i^{th} node in $DECODE_{n/2}(CODE_{n/2}(G_1))$. After Y_v is determined for each $v \in G_2$, the Y_v are sorted under the lexicographical order. The sorting can be done using bucket sort, which is linear in the total number of bits in the vectors.

Assume first that no two nodes in G_2 are connected to the same set of nodes in G_1 i.e all the Y_v's are distinct. G_2 will be represented in an adjacency matrix, and following the matrix will be the Y_v's in the order the nodes appear in the matrix. We do have the freedom to determine any order on the nodes in G_2 and their corresponding Y_v. Let B be the first $f(\frac{n}{2})$ bits in $CODE_{n/2}(G_1)$, and let $\tau : \{1, \ldots \frac{n}{2}\} \mapsto \{1, \ldots, \frac{n}{2}\}$ be the permutation that represents B as described in Section 2. We will order the nodes of G_2 by the permutation τ on the increasing order of the Y_v's. The rest of $CODE_{n/2}(G_1)$ will be represented explicitly. The number of bits saved is $f(\frac{n}{2}) = \frac{n}{2} \log_2 n - O(n)$ plus the number of bits saved in $CODE_{n/2}(G_1)$.

In case not all Y_v's are distinct, we have less elements to permute and hence can encode fewer bits; on the other hand we can save bits by not repeating the description of similar neighborhoods. Following each Y_v will be a sequence of 1's ending with a '0' denoting the number of nodes having the same neighborhood. We call these the barriers. The nodes sharing Y_v are assumed to be in successive indices in the adjacency matrix of G_2. The encoding of all the barriers can add at most

$\frac{n}{2}$ bits all together. If there are m distinct neighborhoods then we can encode $f(m)$ bits, but we save $(\frac{n}{2} - m)\frac{n}{2}$ bits in the description of the duplicated neighborhoods. It is easy to verify that

$$\min_{1 \leq m \leq \frac{n}{2}} \{f(m) + \left(\frac{n}{2} - m\right)\frac{n}{2}\} = f\left(\frac{n}{2}\right)$$

and the minimum is achieved for $m = \frac{n}{2}$. Hence at least $f(\frac{n}{2})$ bits are saved per recursive call.

Claim: $l(n) = \binom{n}{2} - n\log_2 n + O(n)$

Proof: From the description above the $CODE_n(G)$ contains:

- $CODE_{n/2}(G_1)$ which is $l(\frac{n}{2})$ bits
- the description of the neighborhoods in G_1 of the nodes of G_2 which is $\frac{n}{2} \cdot \frac{n}{2}$ bits
- adjacency matrix representation of G_2 which is $\binom{n/2}{2}$ bits
- $\frac{n}{2}$ bits to determine repetitions of neighborhoods.

On the other hand $f(\frac{n}{2})$ bits of the first two items can be saved . Hence we have

$$l(n) = l\left(\frac{n}{2}\right) + \frac{n}{2} \cdot \frac{n}{2} + \binom{n/2}{2} + n - f\left(\frac{n}{2}\right).$$

Let $c_1 > 0$ be a constant such that $f(t) \geq t\log_2 t - c_1 \cdot t$. Assume inductively that

$$l\left(\frac{n}{2}\right) \leq \binom{n/2}{2} - \frac{n}{2}\log_2\left(\frac{n}{2}\right) + \cdot\frac{n}{2}$$

for some fixed $c_2 > 0$. Then we can conclude that

$$l(n) \leq \binom{n/2}{2} - \frac{n}{2}\log_2\left(\frac{n}{2}\right) + c_2 \cdot \frac{n}{2}$$

$$+\frac{n}{2} \cdot \frac{n}{2} + \binom{n/2}{2} + n - \frac{n}{2}\log_2\left(\frac{n}{2}\right) + c_1 \cdot \frac{n}{2}$$

$$= \binom{n}{2} - n\log_2 n + (\frac{c_2}{2} + \frac{c_1}{2} + 1 + \frac{1}{2} + \frac{1}{2}) \cdot n.$$

Thus, if $c_2 \geq c_1 + 4$ we have that $l(n) \leq \binom{n}{2} - n\log_2 n + c_2 \cdot n$. $\square$

Time Complexity: The most time consuming stage that is performed at each recursive step is sorting the Y_v, but that can be done in linear time in $\binom{n}{2}$, the size of the input. Since the recursive call to $CODE_{n/2}$ is with a graph on $\frac{n}{2}$ nodes the whole procedure takes time linear in $\binom{n}{2}$. Similar consideration hold for $DECODE_n$ as well. Therefore we have the theorem claimed in the introduction.

4 Conclusions and Extensions

We have solved an open problem raised in [T]: find an efficient coding method for general graphs which is optimal up to the $O(n)$. An interesting question is whether the existence of an efficient coding method that achieves the $\lceil \log_2 |C_n| \rceil$ lower bound implies an efficient (randomized) algorithm for graph isomorphism. If C_n were a power of 2 this would have been true, since each unlabeled graph has a unique representation in this case. For a general treatment of the connection between Complexity theory and Compression see [GS].

More sophisticated methods of encoding information in a permutation and their applications are presented in [FN], [FNSS] and [FNSSS]. Those methods allow random access decoding, that is one need not compute the whole permutation to infer what a certain bit is.

References

[D] Devroy, **Non-uniform Random Variate Generation,** Springer-Verlag, New-York, 1986.

[FN] A. Fiat and M. Naor, *Implicit $O(1)$ Probe Search,* twenty-first ACM Symposium on the Theory of Computing, 1989, pp. 336-344.

[FNSS] A. Fiat, M. Naor, J. Schmidt and A. Siegel, *Non-Oblivious Hashing,* twentieth ACM Symposium on the Theory of Computing, 1988, pp. 367-376.

[FNSSS] A. Fiat, M. Naor, A. Schaffer, J. Schmidt and A. Siegel, *Storing and Searching a Multikey Table,* twentieth ACM Symposium on the Theory of Computing, 1988, pp. 344-351.

[GRS] R.L. Graham, B. Rothschild, J.H. Spencer, **Ramsey Theory,** Wiley, New-York, 1980.

[GS] A. Goldberg and M. Sipser, *Compression and Ranking,* Seventeenth ACM Symposium on the Theory of Computing, 1985, pp 440-448.

[T] G. Turan, *On the succinct representation of graphs,* Discrete Applied Math, Vol 15, No. 2, May 1984, pp. 604 – 618.