A note on absolute tradeoffs between label size and decoder efficiency

Jakob Grue Simonsen

Abstract

In this note we prove the following two facts:

- There exist a family of graphs for which there exist an efficient implicit representation without an efficient labeling scheme.
- The main labeling scheme conjecture holds for graph families of speed $2^{O(n^{1/2})}$.

1 Preliminaries

We assume that Turing machines are described by a suitable binary coding as usual. If $M \in \{0,1\}^+$ is a Turing machine, we let $\phi_M : \{0,1\}^+ \rightharpoonup \{0,1\}^+$ denote the partial function computed by M. As usual, we assume a Turing machine may take several arguments by suitable binary coding of pairs, e.g. writing $\phi_M(a.b)$ when M gets more than one argument. If M is run on empty input, we denote this by empty parentheses, e.g. $\phi_M()$.

We fix a universal Turing machine U that is able to simulate every Turing machine with at most linear overhead, that is,

- $\forall M.\forall x \in \{0,1\}^+.\phi_U(M.x) = \phi_M(x)$ (universality), and
- $\exists N \in \mathbb{N}.\forall M.\forall x \in \{0,1\}^+$, U simulates one step of M on input x using at most $N \cdot |M.x|$ steps (linear overhead)

If $n \in \mathbb{N}$ we denote by $\langle n \rangle$ some standard binary representation of n such that $|\langle n \rangle| = O(\log n)$.

2 First Result

The following lemma is a variation of standard results from resource-bounded Kolmogorov complexity (earliest reference apparently Luc Longprés 1986 Cornell PhD thesis).

Lemma 1. *There is a Turing machines Q such that:*

- *For each n, Q, on input $\langle n \rangle$, outputs a string, s_n, of length $n(n-1)/2$*

- *Q runs in time $O(2^{n^2})$.*
- *For each $n \in \mathbb{N}$, there is no Turing machine R such that (i) $\mathrm{TIME}_R() < 2^n$, (ii) $|R| < n(n-1)/2$, and (iii) $\phi_R() = s_n$.*

Proof. On input $\langle n \rangle$, Q successively generates each binary string y of length at most $n(n-1)/2 - 1$ and then uses a copy of the universal machine U as a subroutine to simulate U running on input y for exactly $2^n - 1$ steps. As U simulates each (the Turing machine encoded by) y with linear overhead, each such simulation takes at most $N \cdot |y|(2^n - 1)$ steps where N is a constant independent of y. Hence, the total time taken to simulate all strings is $O(2^{n(n-1)/2} \cdot (n(n-1)/2 - 1)(2^n - 1)) = O(2^{n^2})$; the overhead needed to generate each string and other housekeeping operations is negligible compared to this.

During the simulation, Q stores all strings of length exactly $n(n-1)/2$ output by strings y during the simulation in a set S (i.e., if $|\phi_U(y)| = n(n-1)/2$ and y runs for at most $2^n - 1$ steps on empty input, Q stores $\phi_U(y)$. When all strings have been simulated, Q outputs the lexicographically smallest string of length $n(n-1)/2$ that is *not* in S.

By construction, Q always terminates with output a string of length $n(n-1)/2$ and runs in time $O(2^{n^2})$. Suppose, for contradiction, that there were an $n \in \mathbb{N}$ and a Turing machine R such that (i) $\mathrm{TIME}_R() < 2^n$, (ii) $|R| < n(n-1)/2$, and (iii) $\phi_R() = s_n$. By construction, Q simulates R for exactly $2^n - 1$ steps, and thus $\phi_R() = \phi_U(R) = s_n \in S$. But by construction of Q, $s_n \notin S$. Contradiction. □

Note that $|\langle n \rangle| = O(\log n)$, so the time complexity of Q is $O\left(2^{2^{2\log n}}\right)$, that is, doubly exponential in the square of its input size.

Theorem 1. *There is a family $\mathcal{F}$ of graphs such that:*

- *$\mathcal{F}$ has a labeling scheme such that every element of $\mathcal{F}_n$ has maximum label size $O(\log n)$, the encoding is computable, and the decoder runs in time $O(2^{n^2})$.*
- *If a labeling scheme for $\mathcal{F}$ (i) has maximum label size $\leq n/2 - 2$ for infinitely many n, then no decoder for the labeling scheme can run in polynomial time in n.*

Proof. There is an obvious, computable bijection between the set of (labeled) simple, undirected graphs with n nodes (given by an $n \times n$ symmetric adjacency matrix with zero diagonal) and the set of binary strings of length $n(n-1)/2$: if s is such a string, the first $n-1$ bits is the first row above the diagonal of the adjacency matrix, the next $n-2$ bits the second row, and so forth.

Thus, the Turing machine Q of Lemma 1 produces, for each n, an adjacency matrix for a graph of n nodes. We set $\mathcal{F}_n = \{\phi_Q(\langle n \rangle) : n \in \mathbb{N}\} = \{s_n : n \in \mathbb{N}\}$. Note that $\mathcal{F}_n$ contains exactly one labelled graph[1]

[1]If one wants the family $\mathcal{F}$ to be closed under isomorphism, one can merely add all graphs isomorphic to the ones in $\mathcal{F}_n$. This incurs the cost of re-coding the identifiers of the graph, adding $\log n$ to the label size.

We prove the claims of the theorem in turn:

- Each element $G \in \mathcal{F}$ can be labelled by a Turing machine as follows: Each node receives a label comprising (i) a representation of n (using $\log n$ bits), and (ii) an identifier of the node (using $\log n$ bits), that is, a total of $O(\log n)$ bits. Furthermore, a decoder running in time $O(2^{n^2})$ can be devised as follows: On input the labels of two nodes, the decoder first decodes $\log n$ bits to obtain n, then runs a copy of Q as a subroutine to obtain $\phi_Q(\langle n \rangle)$ (i.e., the adjacency matrix of the unique graph in $\mathcal{F}_n$), then decoding the identifier part of the labels of the two nodes, and finally performing a lookup in the adjacency matrix using the two identifiers. The cost of all operations except for computing $\phi_Q(\langle n \rangle)$ can clearly be done in polynomial time in n. Thus, the total time use is dominated by the time for computing $\phi_Q(\langle n \rangle)$, namely $O(2^{n^2})$.

- Assume, for contradiction, that there is a labeling scheme for $\mathcal{F}$ that has maximum label size $\leq n/2-2$ for all n in some infinite set $I = \{n_1, n_2, \ldots\}$ *and* has a decoder running in time $P(n)$ where P is some polynomial.

  Then, there are constants c, C such that we can build a family of programs p_n each of size $|p_n| \leq c + n^2/2 - n - 1$ such that for each $n \in I$, $\phi_{p_n} = \phi_Q(\langle n \rangle)$ and p_n runs in time $Cn^2(P(n) + n^2)$. To see this, note that one can simply store a string consisting of concatenatation of all the labels of the nodes separated by a fresh symbol #, and use the decoder as a subroutine to ascertain, for each pair of nodes, whether they are adjacent, and thus outputting the adjacency matrix $\phi_Q(\langle n \rangle)$. The string consisting of concatenation of the node labels and separators can be stored using at most $n(n/2-2) + n - 1 = n^2/2 - n - 1$ bits, and the remaining program logic can be stored using c bits for some c, independently of n. Hence, $|p_n| \leq c + n^2/2 - n - 1$. The running time of p_n is bounded above by the time to query all pairs of nodes using the decoder as well as quadratic time in the size of the string used to store all labels (i.e., $O(n^4)$) to move tape pointers into position. Hence, the total running time of p_n is bounded above by $Cn^2(P(n) + n^2)$ for some constant C, and is hence polynomial in n.

  But then there is an N such that for all $n > N$ with $n \in I$, we have (i) $\text{TIME}_{p_n}() = Cn^2(P(n) + n^2 < 2^n$. In addition, for all $n \in I$, we have(ii) $|p_n| < n^2/2 - n - 1 < n(n-1)/2$ and (iii) $\phi_{p_n}() = s_n$. As I was infinite, there is thus at least one n that satisfies (i), (ii) and (iii), contradicting Lemma 1.

□

Corollary 1. *There is a family of finite graphs that has (i) a labeling scheme producing labels of size $O(\log n)$, with a decoder running in time $O(2^{n^2})$, but (ii) if a labeling scheme has a decoder running in time polynomial in the label size (or in n), the maximum label size produced by this scheme is $\Omega(n)$.*

Kannan, Naor and Rudich consider families of finite graphs to have labeling scheme if there is a computable labeling encoder producing labels of size at most $O(\log n)$ and a decoder running in time polynomial in the label size. A fortiori, running in time polynomial in the label size means running in polynomial time in n. Theorem 1 thus has the following corollary:

Corollary 2. *There is a family of finite graphs that does not have a labeling scheme (in the sense of Kannan, Naor and Rudich), but has a labeling scheme if the decoder is allowed to run in time $O(2^{n^2})$.*

3 Second Result

We repeat the main labeling scheme conjecture: All hereditary families of n-vertex graphs of size at most $2^{O(n \log n)}$ have a labeling scheme.

We now show that under a small restriction, this is in fact the case. We define the following graphs of n nodes: The family of graph that consist of cliques as C_n, the family of star forests as T_n and the family of path forests as P_n. For any graph family F we denote by $\bar{F} = \{G : \bar{G} \in F\}$. Denote by $S(n)$ the number of partitions of a set with n indistinguishable elements into nonempty subsets, and note that the growth rate of $S(n)$ is $e^{\Theta(\sqrt{n})}$.

Theorem 2. *For some constant c, all hereditary families of n-vertex graphs of size at most $2^{c(n^{1/2})}$ have a labeling scheme.*

Proof. By Balogh et al. [] hereditary graph families may have either size at most cn^k for some constant c and k, or at least $S(n)$. Moreover, there are exactly six possible families of size $S(n)$: $C_n, T_n, P_n, \bar{C}_n, \bar{T}_n$ and $\bar{P}_n$. If the graph family is of the first kind a labeling scheme can be constructed as follows: Each node receives a label of size $k \log n + c + \log n$ where the first $\log n$ bits are unique identifier of the node in the graph and the remaining bits describe the identifier of the graph in the graph family. As for the second case, there exist six previously defined labeling schemes for the families described. □