Tight Bounds for Top Tree Compression

Philip Bille(✉), Finn Fernstrøm, and Inge Li Gørtz

DTU Compute, Technical University of Denmark,
Richard Petersens Plads, 2800 Kgs. Lyngby, Denmark
{phbi,inge}@dtu.dk, finnfernstroem@gmail.com

Abstract. We consider compressing labeled, ordered and rooted trees using DAG compression and top tree compression. We show that there exists a family of trees such that the size of the DAG compression is always a logarithmic factor smaller than the size of the top tree compression (even for an alphabet of size 1). The result settles an open problem from Bille et al. (Inform. and Comput., 2015).

1 Introduction

Let T be a labeled, ordered, and rooted tree. The overall idea in *top tree compression* [3] is to first construct the *top tree* $\mathcal{T}$ for T [1,2], which is a balanced binary tree representing a hierarchical decomposition of T into overlapping connected subgraphs of T. The top tree $\mathcal{T}$ is then *DAG compressed*, i.e., converted into the minimal DAG representation [4], into the *top DAG*, which we then output as the final compressed representation of T. Top tree compression has several attractive properties. For instance, it achieves almost optimal worst-case compression, supports navigational queries in logarithmic time, and is highly competitive in practice [3,5].

An interesting open question from Bille et al. [3] is how top tree compression compares to classical DAG compression. Let n_G denote the size (vertices plus edges) of the graph G. From Bille et al. [3, Thm. 2] we have that for any tree T, the size of the top DAG $\mathcal{TD}$ of T is always at most a factor $O(\log n_T)$ larger than the size of DAG $\mathcal{D}$ of T. However, it is not known if this bound is tight and answering this question is stated as an open problem. Our main result in this paper is to show that there exists a family of trees such that the DAG is always a factor $\Omega(\log n_T)$ smaller than the top DAG and that this bound can be achieved even for an alphabet of size 1 (i.e. unlabeled trees). This settles this open question and proves that the $O(\log n_T)$ factor is tight.

Due to lack of space we omit a detailed discussion of the related work.

2 Top Trees and Top DAGs

We briefly review top tree compression [3]. Let T be a tree with n_T nodes. Let v be a node in T with children $v_1, \ldots, v_k$ in left-to-right order. Define $T(v)$ to be the

P. Bille and I.L. Gørtz—Partially supported by the Danish Research Council (DFF - 4005-00267).

G. Fici et al. (Eds.): SPIRE 2017, LNCS 10508, pp. 97–102, 2017.
DOI: 10.1007/978-3-319-67428-5_9

subtree induced by v and all proper descendants of v and define $F(v)$ to be the forest induced by all proper descendants of v. For $1 \leq s \leq r \leq k$ let $T(v, v_s, v_r)$ be the tree pattern induced by the nodes $\{v\} \cup T(v_s) \cup T(v_{s+1}) \cup \cdots \cup T(v_r)$.

A *cluster* with *top boundary node* v is a tree pattern of the form $T(v, v_s, v_r)$, $1 \leq s \leq r \leq k$. A *cluster* with *top boundary node* v and *bottom boundary node* u is a tree pattern of the form $T(v, v_s, v_r) \backslash F(u)$, $1 \leq s \leq r \leq k$, where u is a node in $T(v_s) \cup \cdots \cup T(v_r)$. Nodes that are not boundary nodes are called *internal nodes*. Two edge disjoint clusters A and B whose vertices overlap on a single boundary node can be *merged* if their union $C = A \cup B$ is also a cluster using various type of merges (see details in Bille et al. [3]).

A *top tree* $\mathcal{T}$ of T is a hierarchical decomposition of T into clusters. It is an ordered, rooted, labeled, and binary tree defined as follows.

- The nodes of $\mathcal{T}$ correspond to clusters of T.
- The root of $\mathcal{T}$ corresponds to the cluster T itself.
- The leaves of $\mathcal{T}$ correspond to the edges of T. The label of each leaf is the pair of labels of the endpoints of its corresponding edge (u, v) in T. The two labels are ordered so that the label of the parent appears before the label of the child.
- Each internal node of $\mathcal{T}$ corresponds to the merged cluster of its two children. The label of each internal node is the type of merge it represents (out of the five merging options from [3]). The children are ordered so that the left child is the child cluster visited first in a preorder traversal of T.

We construct the top tree $\mathcal{T}$ of height $O(\log n_T)$ for top tree compression bottom-up in $O(\log n_T)$ iterations. At each iteration we maintain an auxiliary rooted ordered tree $\widetilde{T}$ initialized as $\widetilde{T} := T$. The edges of $\widetilde{T}$ will correspond to the nodes of $\mathcal{T}$ and to the clusters of T. The internal nodes of $\widetilde{T}$ will correspond to boundary nodes of clusters in T and the leaves of $\widetilde{T}$ will correspond to a subset of the leaves of T. In each iteration, a constant fraction of $\widetilde{T}$'s edges (i.e., clusters of T) are merged as described below. The precise sequence of merges is essential for obtaining the compression guarantees of top tree compression and enabling efficient navigation.

Step 1: Horizontal Merges . For each node $v \in \widetilde{T}$ with $k \geq 2$ children $v_1, \ldots, v_k$ for $i = 1$ to $\lfloor k/2 \rfloor$, merge the edges (v, v_{2i-1}) and (v, v_{2i}) if v_{2i-1} or v_{2i} is a leaf. If k is odd and v_k is a leaf and both v_{k-2} and v_{k-1} are non-leaves then also merge (v, v_{k-1}) and (v, v_k).

Step 2: Vertical Merges . For each maximal path $v_1, \ldots, v_p$ of nodes in $\widetilde{T}$ such that v_{i+1} is the parent of v_i and $v_2, \ldots, v_{p-1}$ have a single child: If p is even merge the following pairs of edges $\{(v_1, v_2), (v_2, v_3)\}, \ldots, \{(v_{p-2}, v_{p-1})\}$. If p is odd merge the following pairs of edges $\{(v_1, v_2), (v_2, v_3)\}, \ldots, \{(v_{p-3}, v_{p-2})\}$, and if (v_{p-1}, v_p) was not merged in Step 1 then also merge $\{(v_{p-2}, v_{p-1}), (v_{p-1}, v_p)\}$.

Since each iteration shrinks the tree by a constant factor the resulting top tree $\mathcal{T}$ has height $O(\log n_T)$. The top DAG $\mathcal{TD}$ is the minimal DAG representation of $\mathcal{T}$.

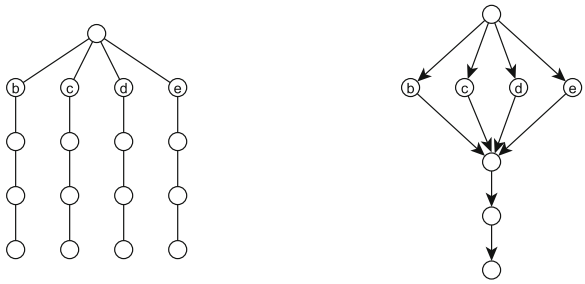

Fig. 1. An example of the labeled construction with $h = 4$ and $\sigma = 5$. On the right is the minimal DAG representation of the tree. Only labels different from a are shown.

2.1 Top DAG Properties

We show a few basic properties of top DAGs that we need for our construction. Let $\widehat{T}$ be a subtree of T. If there are no merges in the first k iterations of the top tree construction algorithm, which include edges from both $\widehat{T}$ and $T \backslash \widehat{T}$, we define $\widehat{T}$ to be *k-local*. A path $v_0, \ldots, v_\ell$ such that v_ℓ is a leaf, v_{i+1} is a child of v_i, and v_i has no other children for $i > 0$, is an *isolated path*. The following results follow directly from the top DAG construction.

Lemma 1. *Any k-local isolated path of length $2^{k-1} < j \leq 2^k$ will be merged into a single cluster in exactly k iterations of the top tree construction algorithm.*

Lemma 2. *Any k-local isolated path of length 2^k, where all nodes except one have identical labels is represented by a subgraph in the top DAG of size $\Theta(k)$, where each node is contained in atleast one path of clusters of length k.*

Lemma 3. *Let $\widehat{T}$ be a k-local subtree of T, which after $j > 0$ iterations of the construction algorithm is an isolated path p in $\widetilde{T}$ of length 2^{k-j}, where all clusters except two are identical. The part of $\mathcal{TD}$, which represents $\widehat{T}$ above level j, then has size $\Theta(k - j)$, and each cluster in $\widetilde{T}$ is contained in a path of clusters of length $k - j$.*

3 Bounds for Large Alphabets

As a warm-up we first consider large alphabets and show the following result.

Theorem 4. *There exists a family of rooted, ordered, and labeled trees, for which $n_{\mathcal{TD}} = \Theta(\log n_T) \cdot n_D$ for alphabets of size $\Omega(\sqrt{n_T})$.*

Proof. Given positive integers k, w, and an alphabet $\Sigma = \{a, b, c, \ldots\}$ of size σ, construct a tree T as the union of w isolated paths of length $h = 2^k$ with a shared root. The second node of each path has its own unique label, $b, c, d, \ldots$. All other nodes are labeled a. We have $n_T = \Theta(wh)$ and $n_{\mathcal{D}} = \Theta(w + h)$ (see Fig. 1).

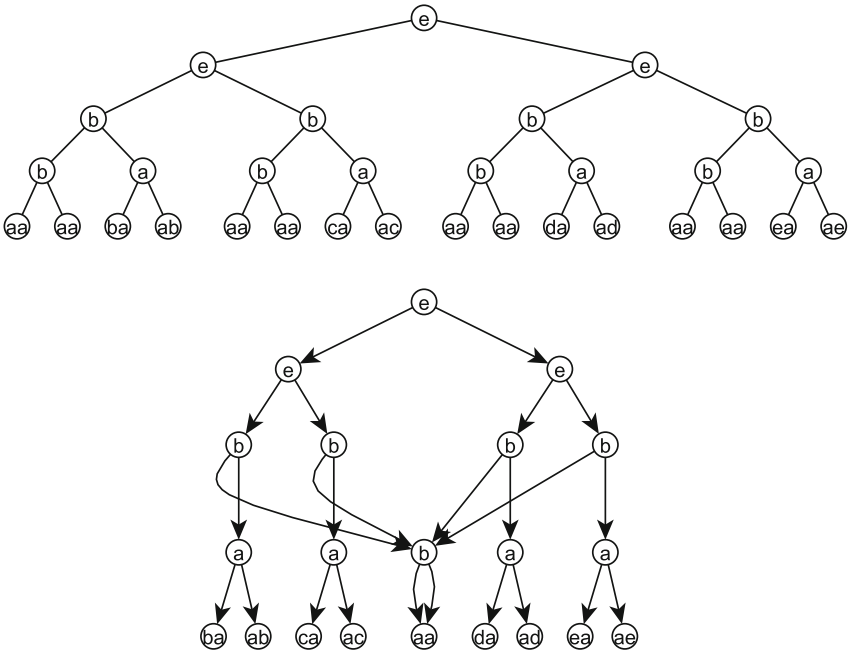

Fig. 2. Top tree and top DAG of the tree in Fig. 1. The label of the leaves corresponding to an edge $(v, p(v))$ in T, where $p(v)$ is the parent of v, is the label of $p(v)$ followed by the label of v. The labels of the internal nodes are the merge types as defined in [3].

Next consider the top DAG. We show that $n_{\mathcal{TD}} = \Theta(w \log h)$. It is easy to verify that the paths are k-local. By Lemma 1 the edges of each path p will be merged into one cluster in the first k iterations. Subsequently these clusters are merged together. By Lemma 2 each path p is represented in $\mathcal{TD}$ by a subgraph of size $\Theta(\log h)$, where the second node v_1 of p is contained in a path of clusters of length $\Theta(\log h)$. These clusters cannot be shared by another path p', since the label of v_1 is unique. We have w paths, so in total this part of the top DAG has size $\Theta(w \log h)$. The merging of the clusters containing the w paths is represented by $\Theta(w)$ clusters in $\mathcal{TD}$. Hence in total $n_{\mathcal{TD}} = \Theta(w \log h)$ (see Fig. 2).

We choose $w = \Theta(\sqrt{n_T})$ and $h = \Theta(\sqrt{n_T})$. Hence we get $n_D = \Theta(\sqrt{n_T})$ and $n_{\mathcal{TD}} = \Theta(\sqrt{n_T} \log n_T)$, which implies $n_{\mathcal{TD}} = \Theta(\log n_T) \cdot n_D$. We need $w + 1$ different characters, hence $\sigma = \Omega(\sqrt{n_T})$. □

4 Bounds for Unlabeled Trees

We now modify the construction from Sect. 3, such that it works for unlabeled trees, and show the following result.

Theorem 5. *There exists a family of rooted, ordered, and unlabeled trees, for which* $n_{\mathcal{TD}} = \Theta(\log n_T) \cdot n_D$.

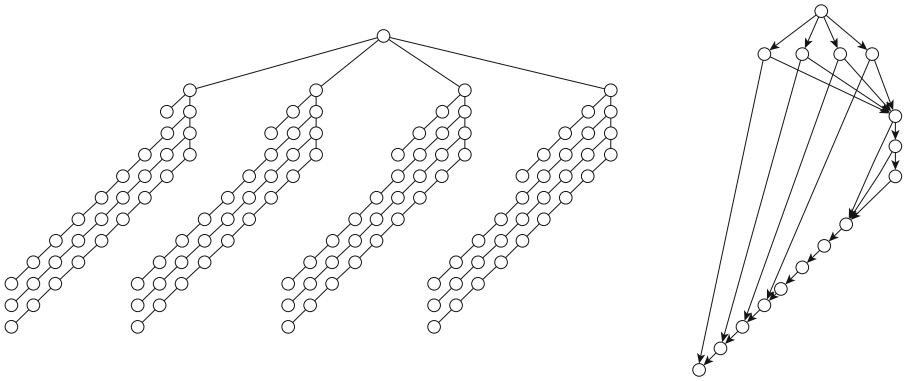

Fig. 3. An example of the unlabeled construction with $w = 4$ and $h = 4$. On the right is the minimal DAG representation of the tree.

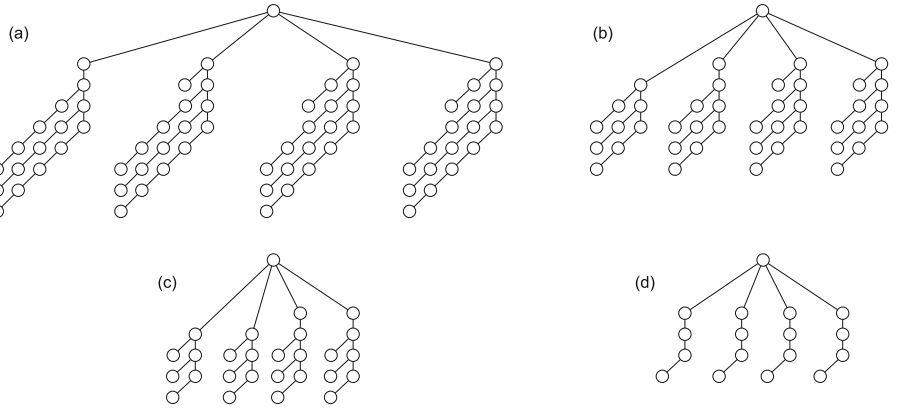

Fig. 4. The auxiliary tree $\widetilde{T}$ of the tree from Fig. 3 after the first 4 iterations: (a) iteration 1, (b) iteration 2, (c) iteration $3 = k_w + 1$, (d) iteration $4 = k_w + 2$.

Proof. Given integers $k_h > 1$ and $k_w > 0$, construct a rooted tree T as the union of $w = 2^{k_w}$ subtrees, $T_1, \ldots, T_w$, connected only at the root of T. Each T_i has a path of length $h = 2^{k_h}$, denoted its *main path* m_i. The second node on m_i has a path on its left side of length i, denoted the *identifying path* of T_i. All other nodes on m_i have a path on their left side of length $2w$, denoted a *side path*. We have $n_T = \Theta(w^2 h)$ and $n_D = \Theta(w + h)$ (see Fig. 3).

We now show that $n_{\mathcal{TD}} = \Theta(w \log h)$. By definition $w = 2^{k_w}$, so all of the side paths have length 2^{k_w+1}. They are all isolated and $(k_w + 1)$-local, so by Lemma 1 each of them will be merged into one cluster in $k_w + 1$ iterations, and by Lemma 2 they are represented in $\mathcal{TD}$ by a subgraph of size $\Theta(\log w)$. Consider now a subtree T_i. In iteration $k_w + 2$ each of the clusters containing the side paths of T_i will be merged with the sibling edge on m_i, except for the

bottom most cluster, which will have nothing to merge with in this iteration. The identifying path of T_i will be merged into one cluster in $j \leq k_w$ iterations. In iteration $j+1$ this cluster will be merged with the sibling edge on m_i, and in iteration $j+2$ this cluster will be merged with the top most edge on m_i. Nothing more happens with this cluster until after iteration $k_w + 2$, at which point T_i has been merged into an isolated path (see Fig. 4).

Each identifying path p is represented in $\mathcal{TD}$ by a node with edges to two clusters, which merged together gives p. It is easy to see that these clusters already exist in $\mathcal{TD}$ in the iteration p is merged into one cluster. Hence all the identifying paths are represented in total by $\Theta(w)$ clusters in $\mathcal{TD}$. When the identifying paths are merged with the first and second edges from the top of the main paths, the corresponding parts of $\mathcal{TD}$ also have in total size $\Theta(w)$.

After k_w+2 iterations $\widetilde{T}$ will consist of w isolated paths (of clusters) of length $h = 2^{k_h}$. It is easy to verify that they are $(k_w + 2 + k_h)$-local. By Lemma 1 the clusters of each path p will be merged into one cluster in the next k_h iterations. Subsequently these clusters are merged together. By Lemma 3 each of these paths, p, is represented in $\mathcal{TD}$ above level $k_w + 2$ by a subgraph of size $\Theta(\log h)$, where the top most cluster C_0 of p is contained in a path of clusters in $\mathcal{TD}$ of length $\Theta(\log h)$. These clusters cannot be shared by another path p', since C_0 is unique. We have w paths, so we get w of these subgraphs. Hence in total this part of the top DAG has size $\Theta(w \log h)$. The merging of the subgraphs is represented by $\Theta(w)$ clusters in $\mathcal{TD}$. Hence in total $n_{\mathcal{TD}} = \Theta(w \log h)$.

We choose $w = \Theta(n^{1/3})$ and $h = \Theta(n^{1/3})$. Hence we get $n_D = \Theta(n^{1/3})$ and $n_{\mathcal{TD}} = \Theta(n^{1/3} \log n_T)$, which implies $n_{\mathcal{TD}} = \Theta(\log n_T) \cdot n_D$. □

References

1. Alstrup, S., Holm, J., de Lichtenberg, K., Thorup, M.: Minimizing diameters of dynamic trees. In: Proceedings of 24th ICALP, pp. 270–280 (1997)
2. Alstrup, S., Holm, J., Thorup, M.: Maintaining center and median in dynamic trees. In: Proceedings of 7th SWAT, pp. 46–56 (2000)
3. Bille, P., Gørtz, I.L., Landau, G.M., Weimann, O.: Tree compression with top trees. Inform. Comput. **243**, 166–177 (2015)
4. Downey, P.J., Sethi, R., Tarjan, R.E.: Variations on the common subexpression problem. J. ACM **27**(4), 758–771 (1980)
5. Hübschle-Schneider, L., Raman, R.: Tree compression with top trees revisited. In: Proceedings of 14th SEA, pp. 15–27 (2015)