AICR · OPEN PROBLEM

A Constant-Factor Guarantee for Huffman-Style Search Tree Construction

Greedy pairwise merging against the optimal OR tree

Status. Statement: AI-written from Mohammad Etemad, Mohammad Mahmoody and David Evans, *Optimizing Trees for Static Searchable Encryption*, IACR Cryptology ePrint Archive 2018, not yet checked by a human. Nothing is proved about the best-first algorithm's approximation ratio for any class of query distributions: the paper's exhaustive experiments already refute exact optimality, the approximation-ratio question is untouched, and all of the paper's proved bounds concern the level-optimal algorithm instead.

Category. *Searchable Encryption & Encrypted Data Structures.*

Conjecture (informal). *In a tree-based encrypted index every file sits at a leaf tagged with its keyword set, every internal node is tagged with the union of the tags below it, and a query costs one unit for every node it touches on the way down. Choosing which files to put next to each other therefore decides the cost of every future search, and the paper reduces the choice to a clean objective: minimise, over all binary trees on the given leaves, the sum over internal nodes of the probability that a random query is satisfied by that node's tag. The paper's best-performing heuristic for this is the obvious greedy one, borrowed from Huffman coding: repeatedly take the two current nodes whose merged tag has the smallest such probability, make them siblings, and replace them by their parent. It wins every experiment the paper runs, and on eight files it is essentially indistinguishable from brute-force search over all binary trees. What is missing is any theoretical guarantee at all: the paper proves approximation bounds for its other, level-by-level algorithm and states that analysing the greedy one is left open, while recording its expectation that all its algorithms are within a constant factor of optimal. Note that the greedy tree is genuinely not always optimal — exhaustive search beats it on some inputs — so the question really is about an approximation ratio, and it can just as well be settled by exhibiting a workload on which greedy is off by an unbounded factor.*

1 The setting

In tree-based searchable symmetric encryption [Goh03, KP13, BlindSeer] the encrypted index is a binary tree: files sit at the leaves with a vector of the keywords they contain, internal nodes carry the bitwise OR of their children's vectors, and a monotone Boolean query is answered top-down, descending below exactly those visited nodes whose vector satisfies it. Every node costs the same to process, so search cost is the number of visited nodes, and it depends on how the files were grouped. The paper shows that for any OR tree T over L and any distribution Q over monotone queries,

$$\mathop{\mathbb{E}}_{q \leftarrow Q} \left[\#\{\text{nodes visited searching } q \text{ in } T\} \right] \;=\; 1 + 2\,\mathrm{SLC}_Q(T),$$

so designing the index is the combinatorial problem of minimizing SLC_Q over the $\prod_{i=1}^{n-1}(2i-1)$ trees on n leaves. When Q is uniform over single keywords the objective specializes to $\frac{1}{m}\sum_{u\in I(T)}\mathrm{HW}(\bar{u})$, where HW is Hamming weight: minimize the total size of the $n-1$ keyword unions formed while merging the files into a hierarchy. This is a natural cost measure for hierarchical clustering with no encryption in it, and it is closely analogous to the classical problem of building binary search trees tuned to an access distribution [Meh75, Nag97], except that the cost of a node depends on the union of its subtree rather than on a weight given in advance.

The paper's best-first algorithm (Definition 2) is the Huffman-style greedy heuristic for this objective, and it is the one that performs best: across the paper's Enron experiments it produces the cheapest trees of the three heuristics, roughly halving the search cost of a randomly built tree, and on the largest exhaustively checkable instance ($n = 8$, all 135,135 trees enumerated, across 25 experiment sets: five randomly selected sets of eight files crossed with five query distributions) brute force finds a strictly better tree in 13 of 25 trials but improves the cost by at most 0.09. So the greedy tree is not exactly optimal, which fixes the shape of any positive answer: it must be an approximation bound, not an optimality proof.

What the paper proves instead concerns only its second algorithm, the level-optimal algorithm (Algorithm 2), which builds the tree level by level using Blossom minimum-weight matching; the hybrid algorithm is left unanalyzed alongside best-first. For that algorithm it gives an $O(\log n)$ approximation against $\mathrm{Opt}_Q(L)$ when Q is supported on disjunctive queries, together with a matching $\Omega(\log n)$ instance — files with disjoint keyword sets, the i-th having 2^{i-1} keywords — on which the best-first tree costs $O(m)$ while every balanced tree costs $\Omega(m \log n)$. That instance is also the reason the comparison for best-first has to be against $\mathrm{Opt}_Q(L)$ over all trees: greedy is the algorithm that is allowed to go unbalanced, and on that instance it is the one that wins. For the best-first algorithm itself the paper proves nothing, and its Open questions paragraph records both the expectation (all three algorithms are probably within a constant factor of optimal) and the gap (the theoretical analysis of best-first is left to future work). The level-by-level argument the paper does have cannot be transplanted: its proof bounds the zero-weight of the parents of the leaves against that of any balanced

tree layer by layer, which presupposes the layered structure that the greedy algorithm does not have.

Progress is experimental only. On $n = 8$ files, exhaustive enumeration of all 135,135 binary trees beats the best-first tree in 13 of 25 trials, by at most 0.09 in cost. On the Enron corpus with n up to 2^{10} and dictionary sizes up to 11,308, the best-first tree has the lowest cost of the two of the paper's algorithms compared there — it beats the level-optimal tree in every experiment group of the paper's Table 1 and roughly halves the cost of a random tree; the hybrid algorithm is run only for $n > 2^{10}$ and is never compared against best-first. Those comparisons cover single-keyword queries and two- and three-keyword conjunctions and disjunctions. In the paper's $\Omega(\log n)$ separation instance the best-first tree is the cheap one, costing $O(m)$ against $\Omega(m \log n)$ for any balanced tree.

Why it matters. Best-first is the algorithm that actually wins, in this paper's own experiments and in the small-n exhaustive comparison, and it is the one a practitioner would ship for a static encrypted index; it is also the only one of the three with no guarantee of any kind. A constant-factor bound would give the deployed heuristic its first worst-case justification and would say that the cost of an encrypted tree index is, up to a constant, an intrinsic property of the file collection and query workload rather than an artefact of how the tree was built. A counterexample would be equally valuable and arguably more surprising, since it would exhibit a workload on which a Huffman-style greedy is asymptotically wrong for a cost function that looks submodular, and it would tell index builders to look past greedy. Either way the answer is about a self-contained combinatorial optimization problem — approximating the cheapest hierarchical grouping under expected query satisfaction — with an interest of its own beyond encrypted search. The whole statement is bit vectors, bitwise OR, binary trees and one sum of $n - 1$ probabilities; the algorithm is five lines of pseudocode, no cryptography enters the statement at all, and nothing depends on the paper's implementation or data.

2 Notation and parameters

$[n] = \{1, \dots, n\}$.

The dictionary is a set of m keywords $w_1, \dots, w_m$. A vector

$\bar{v} \in \{0,1\}^m$ records a set of keywords, with $\bar{v}[j] = 1$ meaning w_j is present; $\mathrm{OR}(\bar{x},\bar{y}) \in \{0,1\}^m$ is the bitwise OR, i.e. $\mathrm{OR}(\bar{x},\bar{y})[j] = \bar{x}[j] \vee \bar{y}[j]$ for all $j \in [m]$.

$\mathcal{M}_m$ is the set of monotone Boolean formulas over variables $x_1,\ldots,x_m$: formulas built from the variables using $\wedge$ and $\vee$ only, with no negation. For $q \in \mathcal{M}_m$ and $\bar{v} \in \{0,1\}^m$ we write $q(\bar{v}) = 1$ if q evaluates to true under the assignment $x_j := \bar{v}[j]$ for all $j \in [m]$, and $q(\bar{v}) = 0$ otherwise.

$L = \{(f_1,\bar{f_1}),\ldots,(f_n,\bar{f_n})\}$ is a set of n file identifiers f_i with vector labels $\bar{f_i} \in \{0,1\}^m$, where $\bar{f_i}[j] = 1$ iff file f_i contains keyword w_j. For a tree T, $L(T)$ is its set of leaves, $I(T)$ its set of internal nodes, $\pi(u)$ the parent of a non-root node u, and $\Gamma(u)$ the set of children of an internal node u. Each node u of T carries a label $\bar{u} \in \{0,1\}^m$.

Q denotes a distribution over $\mathcal{M}_m$, and $q \leftarrow Q$ a sample from it. For a node u with label $\bar{u}$, its *local cost* is

$$\mathsf{P}_Q(u) \;=\; \Pr_{q\leftarrow Q}\big[\, q(\bar{u}) = 1 \,\big] \;\in\; [0,1],$$

and for an OR tree T over L (Definition 1),

$$\mathrm{SLC}_Q(T) \;=\; \sum_{u\in I(T)} \mathsf{P}_Q(u),$$

$$\mathrm{Opt}_Q(L) \;=\; \min_{T\,:\,L(T)=L} \mathrm{SLC}_Q(T),$$

the minimum ranging over all OR trees over L, of any shape. Finally $\mathsf{BF}(L,Q)$ is the set of trees the best-first algorithm of Definition 2 can output on L and Q.

The parameters are:

- m, the dictionary size, i.e. the number of keywords; labels are vectors in $\{0,1\}^m$. Unbounded and independent of n.
- n, the number of files, hence the number of leaves; any integer $n \geq 2$ (best-first needs no power-of-two restriction).
- L, the leaf set: n file identifiers with their keyword vectors, arbitrary.
- Q, the query distribution, an arbitrary distribution over monotone Boolean formulas on m variables.

- c, the approximation constant, absolute: independent of m, n, L and Q.

Definition 1 (OR tree over L). Let $L = \{(f_1, \bar{f}_1), \ldots, (f_n, \bar{f}_n)\}$ with $\bar{f}_i \in \{0,1\}^m$. An *OR tree over* L is a full binary tree T (every node has either 0 or 2 children) with $L(T) = L$, in which every leaf f_i carries its given label $\bar{f}_i$ and every internal node $u \in I(T)$ with $\Gamma(u) = \{x, y\}$ carries the label $\bar{u} = \mathrm{OR}(\bar{x}, \bar{y})$. The labels are therefore determined by L together with the shape of T and the assignment of files to leaves; $|I(T)| = n - 1$ always.

Definition 2 (the best-first algorithm BF). On input a leaf set L with $|L| = n$ and access to the local cost function $\mathsf{P}_Q(\cdot)$, the algorithm sets $W := L$ and, while $|W| > 1$, does the following. For each unordered pair $x \neq y$ in W let $C(x, y) := \mathsf{P}_Q(z)$, where z is a new node with label $\bar{z} = \mathrm{OR}(\bar{x}, \bar{y})$. Pick a pair $\{x, y\}$ minimizing $C(x, y)$, create the node z with label $\bar{z} = \mathrm{OR}(\bar{x}, \bar{y})$, set $\Gamma(z) = \{x, y\}$ and $\pi(x) = \pi(y) = z$, remove x and y from W and add z to W. When $|W| = 1$ return the tree built, whose root is the single remaining element of W; it is an OR tree over L, not necessarily balanced. We write $\mathsf{BF}(L, Q)$ for the set of all trees obtainable this way, over all choices of minimizing pair at each step.

3 The conjecture

Conjecture 1 (best-first is a constant-factor approximation). *There is an absolute constant $c > 0$ such that the following holds for every dictionary size $m \geq 1$, every $n \geq 2$, every leaf set $L = \{(f_1, \bar{f}_1), \ldots, (f_n, \bar{f}_n)\}$ with $\bar{f}_i \in \{0,1\}^m$, every distribution Q over $\mathcal{M}_m$, and every tree $T \in \mathsf{BF}(L, Q)$:*

$$\mathrm{SLC}_Q(T) \;\leq\; c \cdot \mathrm{Opt}_Q(L).$$

Equivalently, $\mathrm{SLC}_Q(T) = O(\mathrm{Opt}_Q(L))$ with the hidden constant independent of m, n, L and Q. Note that $\mathrm{Opt}_Q(L)$ is the minimum over all OR trees over L, including unbalanced ones, so no restriction on the height of T is imposed on either side.

Remark 1 (the strength of the statement). This is the maximal reading of the paper, and it may well be false. The paper leaves open "a theoretical analysis of the optimality" of best-first without stating a target bound; the constant factor comes from the neighbouring sentence, which says all the paper's algorithms are "probably within a constant factor of optimal", and that sentence is hedged with "when it comes to real data". A worst-case constant-factor claim over all monotone Q therefore goes beyond anything the paper asserts, and a counterexample would settle the problem without contradicting the paper.

Remark 2 (the natural first target). Reviewers who want a version closer to what the paper proves elsewhere should consider Conjecture 1 restricted to Q uniform over single keywords, where the objective is $\frac{1}{m}\sum_{u \in I(T)} \mathrm{HW}(\bar{u})$, the total Hamming weight of the internal labels. That special case is the natural first target and I would regard it as the real content. The objective is also computable in closed form for the other family the paper singles out: for Q uniform over k-keyword conjunctions, $\mathsf{P}_Q(u)$ is a fixed polynomial in $\mathrm{HW}(\bar{u})$.

Remark 3 (tie-breaking). The best-first algorithm is defined only up to the choice of minimizing pair, so its output is a set of trees rather than a single tree, and Conjecture 1 quantifies over *every* $T \in \mathsf{BF}(L, Q)$. A bound proved for one canonical tie-breaking rule would be a partial answer, and worth reporting as such.

Remark 4 (what is known about difficulty). Unlike the paper's

other open question, this one comes with no obstruction named by the authors. The only evidence about difficulty is that greedy is not exactly optimal, and that the paper's layered proof technique does not apply to an unlayered algorithm — the latter being a reading of their proof rather than their statement. Separately, the general problem of computing $\mathrm{Opt}_Q(L)$ may be NP-hard, which the paper never discusses; that would not affect the conjecture, which is about an approximation ratio, but a solver should be aware of it.

4 Bibliography

[BlindSeer] V. Pappas, F. Krell, B. Vo, V. Kolesnikov, T. Malkin, S. G. Choi, W. George, A. Keromytis, S. Bellovin. *Blind Seer: A Scalable Private DBMS.* 35th IEEE Symposium on Security and Privacy, 2014.

[Goh03] E.-J. Goh. *Secure indexes.* Technical report, Cryptology ePrint Archive, Report 2003/216, 2003.

[KP13] S. Kamara, C. Papamanthou. *Parallel and dynamic searchable symmetric encryption.* Financial Cryptography and Data Security, FC, 2013.

[Meh75] K. Mehlhorn. *Nearly optimal binary search trees.* Acta Informatica, 5(4):287–295, 1975.

[Nag97] S. V. Nagaraj. *Optimal binary search trees.* Theoretical Computer Science, 188(1):1–44, 1997.