Bat-LZ Out of Hell

Zsuzsanna Lipták, Francesco Masillo,
Gonzalo Navarro

Universitá di Verona & Universidad de Chile

CPM 2024

The Context: Accessing Highly Compressed Text

- One can represent a text $T[1..n]$ with a (run-length) context-free grammar of size g_{rl}...
- ... and access any $T[i]$ in $O(\log n)$ time.
- But if one represents T with its Lempel-Ziv 1976 (LZ) parse, of size $z \leq g_{rl}$...
- ... there are no known bounds to access $T[i]$.
- Can we do something in this respect?

Accessing LZ-Compressed Text

- If one goes for the simple algorithm of tracking $T[i]$ backwards...
- ... one may fall into a long reference chain, of length $\leq z$.

a	l	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$
0	0	1	0	1	0	1	1	2	0	2	1	2	1	0	1	0

- Can we design an LZ variant where the length of those chains is bounded?
- Say, by a parameter c, so the cost to access $T[i]$ is $O(c)$.

a	l	a	b	a	r	a	l	a	l	a	b	a	r	d	a	$
0	0	1	0	1	0	1	1	0	1	0	1	0	1	0	1	0

- How would this Bounded-Access-Time (BAT)-LZ compress compared to a grammar that accesses in time $O(\log n)$?

BAT-LZ Parsing

- Some definitions:

  In a left-to-right parse $T = T_1 \cdots T_z$, each $T_i = S_i \cdot a_i$, where S_i occurs in T starting before T_i and $a_i \in \Sigma$.

  The chain length of a_i is zero and that of $T_i[j]$ is one plus that of $S_i[j]$.

  A BAT-LZ parse with parameter c is a left-to-right parse where no chain length exceeds c.

- It turns out that the best BAT-LZ is NP-hard and APX-hard [Cicalese & Ugazio 2024].

  A BAT-LZ parse is greedy if each T_i, when obtained left-to-right, is as long as possible.

- A greedy BAT-LZ parse is not necessarily optimal, but it is promising and we can compute it efficiently.

A Greedy BAT-LZ Parse

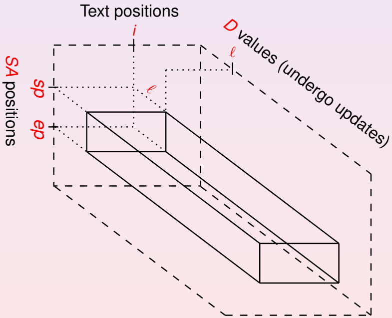
- We parse left-to-right as standard LZ, but put more restrictions in the phrase to form.
- We store the following data:
  1. The suffix array of T, as a wavelet matrix.
  2. The inverse suffix array of T, as a plain array.
  3. An array $C[1..n]$, where $C[i]$ is the chain length of i.
  4. An array $D[1..n]$ where $D[s]$ is the least $d \geq 0$ s.t. $C[s+d] = c$, or else $D[s] = \infty$ (note D changes as we proceed on T).
  5. A dynamic range-maximum-query structure on each level of the wavelet matrix.
- The key observation:

If the source of $T[i..i+\ell-1]$ is $T[s..s+\ell-1]$, then $\ell \leq D[s]$.

A Greedy BAT-LZ Parse

So, we can use $T[s..s+\ell-1]$ as a source for $T[i..i+\ell-1]$, whose SA range is $[sp..ep]$, iff

1. $ISA[s] \in [sp..ep]$ (i.e., $T[s..s+\ell-1] = T[i..i+\ell-1]$),
2. $s < i$ (i.e., it starts before the new phrase), and
3. $\ell \leq D[s]$ (i.e., it does not use forbidden positions).



A Geometric Problem

- We then store each $T[j]$ as a 3D point

  $$(ISA[j], j, D[j]),$$

  and search for points in

  $$[sp, ep] \times [1, i-1] \times [\ell, n].$$

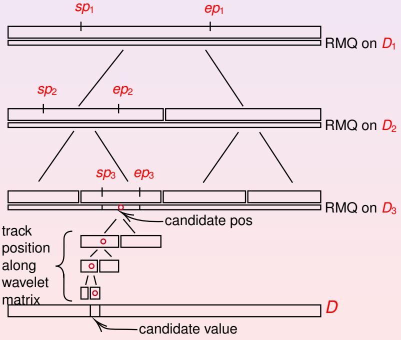
- From i, we find the longest admissible prefix of $T[i..]$.
- That is, we check $T[i..i+\ell-1]$ for consecutive values of ℓ.
- Once we find the longest phrase $T[i..i+\ell-1]$, we:
  1. Set $C[i+l] = C[s+l]+1$ for all $0 \le l < \ell$ and $C[i+\ell] = 0$.
  2. Every time we obtain $C[t] = c$ in this process, we set $D[k] = t-k$ for all $k' < k \le t$, and finally $k' = t$ (initially $k' = 0$ and all $D[\cdot] = \infty$).

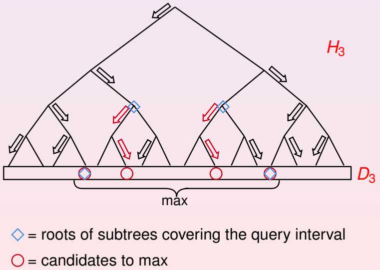
A Geometric Problem

- So we have a 3D orthogonal range search problem where we want one point if it exists.
- The 2nd coordinate of the retrieved point is the desired s.
- The 3rd coordinate is modified along the parse.
- We did not find any proper linear-space solution in the literature (asked experts).
- We propose a linear-space solution supporting operations in time $O(\log^3 n)$.
- Our solution works because the queries on the dynamic coordinate are one-sided.

A Geometric Structure

- The D array permuted in level l of the wavelet matrix is D_l.
- We build a perfectly balanced tree on it; each node tells if the maximum is to the left or to the right, $H_l[1..n]$.
- Given a range $[sp_l..ep_l]$ in D_l, we can identify the $O(\log n)$ maximal subtrees covering it.
- For each subtree, we find its heaviest leaf in $O(\log n)$ time.





A Geometric Structure

- From the heaviest leaf, we find the actual $D[\cdot]$ value by tracking the position downwards in the wavelet matrix, in $O(\log n)$ time.
- In total, we find the largest $D[\cdot]$ value in a range of D_l in $O(\log^2 n)$ time.
- A range search on the wavelet matrix yields $O(\log n)$ ranges across different levels l.
- So our 3D query takes time $O(\log^3 n)$.
- As we query n times, we get $O(n \log^3 n)$ time.
- We actually use exponential search for ℓ, but still the updates require the same time.

A Geometric Structure: Updates

- We track $D[ISA[k]]$ across every wavelet matrix level l.
- We identify all the ancestors $H_l[p/2^h]$ for successive h.
- We always know the (new) maximum below our subtree.
- If the parent node points to the other child, we are done (we always reduce the values of D).
- Else, we must compute that other child's maximum, compare, update the node's direction, and continue.
- Total time is $O(\log^3 n)$ per update, $O(n \log^3 n)$ in total.

The Result

Theorem

A Greedy BAT-LZ parse of a text $T[1..n]$ can be computed using $O(n)$ space and $O(n \log^3 n)$ time.

Theorem

There exists a linear-space data structure that supports five-sided orthogonal range queries on 3D points, plus updates on the one-sided dimension, in time $O(\log^3 n)$ per operation.

Quality of a Greedy Parse

- Our Greedy BAT-LZ parse may not produce the smallest parse (choosing the longest phrase may not be optimal).
- Still, our greedy parser may not produce the smallest greedy parse!
- This is because it may not choose the best source for the longest phrase.

a | l | a b | a r | a l a l | a b a | r d | a $ |
0 | 0 | 1 0 | 2 0 | 1 1 2 0 | 2 1 0 | 1 0 | 1 0 |

a | l | a b | a r | a l a l | a b a r d | a $ |
0 | 0 | 1 0 | 1 0 | 1 1 2 0 | 2 1 2 1 0 | 1 0 |

The Minmax Parsing

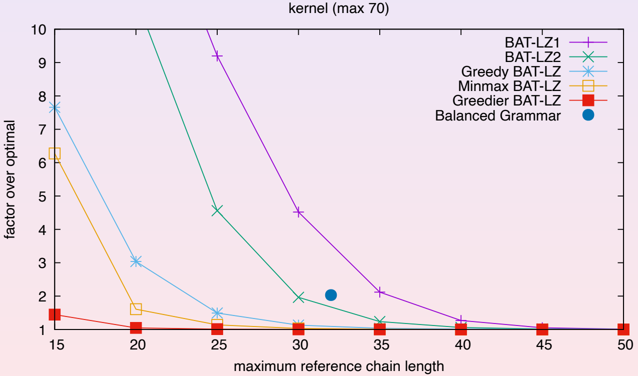
- We develop a Minmax Parse, which chooses a source with least maximum chain length.
- From all the admissible sources for $T[i..i+\ell-1]$, it finds the one with minimum $\max C[s..s+\ell-1]$.
- It annotates the suffix tree nodes, so that we can choose the best descendant of the locus of $T[i..i+\ell-1]$.
- When the values of C change, we must update those annotations for that position and preceding ones.
- Each change in a position updates annotations in the upward path from its suffix tree leaf.
- Parsing time is $O(z'n^2)$, though now we know it can be done in $O(n^2)$ (details omitted).

The Greedier Parsing

- The Minmax parse may sometimes not be greedy, missing potentially longer matches.
- We combine it with our greedy parse, using the dynamic array D again.
- The combined parse is greedy and chooses the “best” phrase.
- Parsing time is $O(z' n^2 \log n)$ (details omitted again).

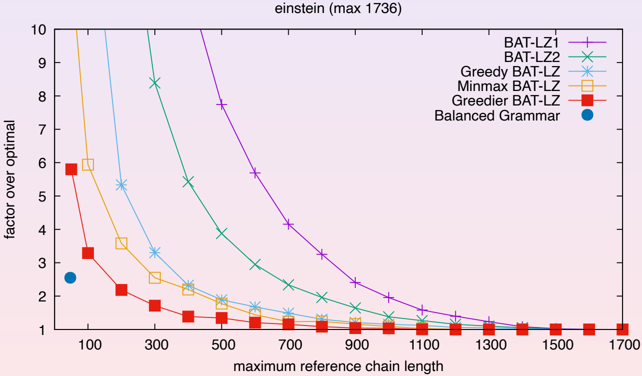
Experiments

- Baseline 1: Cut all LZ phrases where the chain length is divisible by c.
- Baseline 2: Restart the LZ parse whenever this happens.



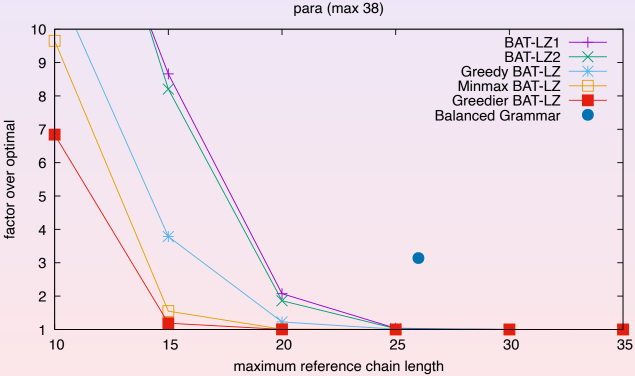
Experiments

- Baseline 1: Cut all LZ phrases where the chain length is divisible by c.
- Baseline 2: Restart the LZ parse whenever this happens.

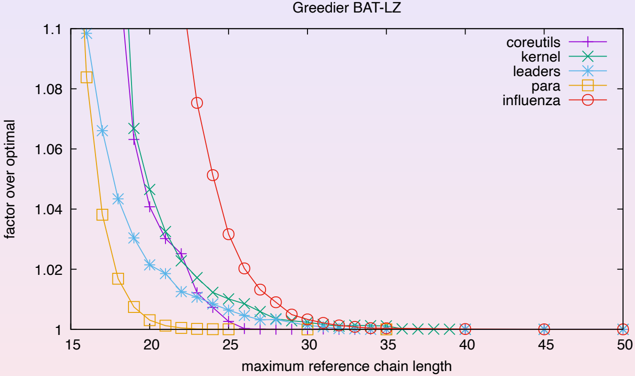


Experiments

- Baseline 1: Cut all LZ phrases where the chain length is divisible by c.
- Baseline 2: Restart the LZ parse whenever this happens.

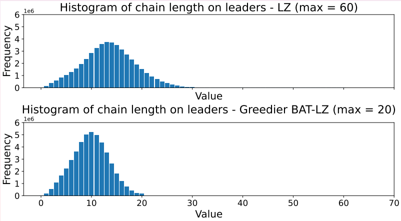


Best Results: Greedier



Epilogue and Discussion

- A simultaneous work by Bannai et al. [ESA B 2024] achieves BAT-LZ greedy parse in $O(n \log \sigma)$ time.
- Likely faster than ours, likely to use more space.
- Our reduction to a geometric problem is also of independent interest.
- We believe we can use it to do the greedier parse in $O(n \log^3 n)$ time.
- Open problem: limit the average reference chain length.



Epilogue and Discussion

- Bannai et al. also show that there is a BAT-LZ parse of size $O(g_{rl})$ if we let $c = \Theta(\log n)$.
- This is nearly optimal given known lower bounds.
- Is there a BAT-LZ parse of size $O(z)$ with $c = \Theta(\log n)$?
- (of course, this would solve the long-standing problem of direct access to LZ)