

円形デカルト木照合のためのコンパクト複数テキスト索引

Roman Pauli Dominik Köppl

September 2, 2026

Abstract

デカルト木照合は、数値列を値そのものではなくデカルト木の形で比較する構造的パターン照合であり、完全一致が適さない時系列・音楽・ゲノムデータなどに有効である。本講演では、円形デカルト木照合を対象とした BWT ベースの複数テキスト索引 ceBWT の実装と実験評価について述べる。索引は動的に拡張可能な構成と静的圧縮構成を備え、さらに 2/3-fold 複製法とマルチスレッド構築により実用性を高める。ゲノムおよび MIDI データを用いた評価により、静的索引はコンパクトな空間で高速な反復検索を実現する一方、動的索引では逐次併合のコストが実用上の制約となることを示す。

Keywords: デカルト木照合, ceBWT, BWT 索引, 圧縮データ構造, 文字列アルゴリズム, 実験評価

1 はじめに

文字列パターン照合では、完全一致だけでなく、入力値の相対的な構造を保存する照合モデルが重要である。順序保存照合 (order-preserving matching; OPM) は値の相対順序を比較するが、デカルト木照合 (Cartesian tree matching; CTM) は、各区間で選ばれる最小値位置から再帰的に定まる木構造を比較する。したがって CTM は OPM より粗い同値関係であり、局所的な谷やピークの形状を重視する用途にきいている。

本論文の目的は、理論的に提案されていた ceBWT 索引 [9] を実装し、実際のデータでどの程度の構築時間、検索時間、メモリ使用量、スケーラビリティを示すかを明らかにすることである。特に、複数テキストを同時に扱う円形 CTM 索引として、動的索引と静的索引の両方を実装し、完全一致索引、OPM 索引、および KMP 型 CTM アルゴリズムと比較する。

2 関連研究

OPM は、数値列の値そのものではなく、すべての位置対の相対順序を保存する照合モデルである [2]。このモデルに対しては、高速な照合アルゴリズムやコンパクト索引が知られている [5]。

CTM は Park らにより導入された [3]。親距離符号 (parent-distance encoding; PDE) またはデカルトシグネチャを用いることで、長さ m のパターンを長さ n のテキスト中で $O(m+n)$ 時間で検索できる。Kim と Cho は、パターンに依存しないコンパクト CTM 索引を与えた [4]。ceBWT はこれを円形照合と複数テキストへ拡張し、動的な挿入と静的な圧縮表現を扱う [9]。本論文は、この ceBWT

を実装し、理論的な計算量だけでは見えない実用上の挙動を評価した点に特徴がある。

3 主なアイデア

3.1 CTM と PDE

列 X のデカルトは、左端の最小値を根とし、その左側と右側に同じ操作を再帰的に適用して得られる木である。二つの列 X, Y が同じデカルトを持つとき、 $X \approx_{CT} Y$ と書く。PDE $\langle X \rangle$ は、各位置 i について、左側に存在する $X[j] \leq X[i]$ を満たす最大の j との距離を記録し、そのような j が存在しない場合は ∞ を記録する符号である。同じ長さの列では、 $X \approx_{CT} Y$ であることと $\langle X \rangle = \langle Y \rangle$ であることが同値である。

3.2 円形探索空間

ceBWT は、各入力テキストの全円形回転を探索対象にする。パターン P の count 問合せは、すべてのテキストのすべての円形開始位置 q に対して、 P がその位置から始まる無限周期列の長さ $|P|$ 接頭辞と CTM する個数を返す。このため、通常のテキスト境界を越える照合も自然に扱える。

3.3 F 配列, L 配列, LF 写像

通常の FM 索引と同様に、ceBWT も整列された円形回転上で LF 写像を用いる。ただし、通常の文字ではなく、円形デカルトシグネチャに由来する値を F 配列と L 配列に格納する。動的索引は F, L, LCP^∞ を動的 wavelet 木で保持し、静的索引は F, L をビットベクトル族 B_F, B_L に変換する。静的索引は更新できないが、rank/select が高速で、問合せ時間と索引サイズの面で有利である。

3.4 folding 構成

元の ceBWT 構築は、長さ n のテキスト T から補助文字列 $T^4\$$ を作る 4-fold 構成を用いる。これは二つの円形回転の順序比較に十分な長さの文脈を与えるためである。本論文では、実装上の最適化として次の構成を比較する。

4-fold: 元論文に基づく基準構成であり、 $T^4\$$ を用いる。

3-fold: 3 コピーの文脈を用いる構成であり、理論的補題により正当化される。

2/3-fold: 初期索引構築では 2 コピー、比較値の計算では 3 コピーを用いる実装上の削減である。

threaded 2/3-fold: 部分索引を並列に構築し、順序を保って逐次マージする構成である。

これらの削減は、最終的な論理的索引を変えず、構築中に一時的に扱う情報量を減らすための工学的工夫である。

4 評価の要点

評価は、ゲノムデータ上の内部ベンチマークと、MIDI データ上の外部ベンチマークに分けられる。内部評価では、ceBWT 実装の構築法、変換、索引サイズ、検索スケーリング、反復性の影響を調べた。外部評価では、SDSL, r-index, RLCSA, OPM-FM 索引、および KMP 型 CTM 実装と比較した。

4.1 内部評価

同じ総記号数 96,000 で配列長を変えた実験では、配列長が長いほど構築スループットが下がり、メモリ消費が増加した。8,000, 16,000, 24,000 長の設定に対して、スループットはそれぞれ 29,210, 28,238, 27,967 記号/秒、メモリは 26, 30, 33 バイト/記号であった。

構築法の比較では、threaded 2/3-fold が最も高速であり、数列挿入後はおよそ 24,000–28,000 記号/秒を達成した。2/3-fold はおよそ 13,000–17,000 記号/秒であり、3-fold は 2/3-fold と 4-fold の間に位置した。4-fold は最も遅く、9,000 記号/秒未満であった。メモリ面では、初期の少数配列では削減構成が有利であるが、配列数が増えると差は小さくなった。

動的索引から静的索引への変換は、静的索引から動的索引への変換より高速であった。静的検索は動的検索より大幅に速く、パターン長の増加に対して、動的索引は約 8,000 から 238 問合せ/秒へ低下し、静的索引は約 500,000 から 13,000 問合せ/秒へ低下した。これは静的ビットベクトル上の rank/select が、動的 wavelet 木より軽量であることを反映している。

反復性の影響については、rice 由来の反復的部分列と実ゲノム部分列を同じ長さで比較した。静的変換のスループットは反復的数据で 51,176 記号/秒、非反復的数据で 46,345 記号/秒であった。一方、動的構築時のメモリは表示精度ではどちらも 28 バイト/記号であった。主索引のサイズは、動的索引で 1.44 対 1.51 バイト/記号、静的索引でどちらも 0.50 バイト/記号であり、反復性は索引サイズには小さい影響しか持たなかった。ただし、短いパターンの照合数は反復的数据で大きく増加した。

4.2 外部評価

MIDI データでは、ceBWT の構築は完全一致索引や OPM-FM 索引より重かった。threaded 2/3-fold の構築スループットは 10,342 記号/秒、2/3-fold は 8,543 記号/秒であり、OPM-FM の multi-index 構成は 25,675 記号/秒、one-index 構成は $4.24 \cdot 10^5$ 記号/秒であった。構築時ピークメモリは、threaded 2/3-fold で 44 バイト/記号、2/3-fold で 23 バイト/記号であった。

最終索引サイズでは、静的 ceBWT が 0.50 バイト/記号と最小であった。動的 ceBWT は 1.69 バイト/記号であり、SDSL の multi-index 構成 2.87 バイト/記号や OPM-FM の multi-index 構成 3.19 バイト/記号より小さかった。検索時間では、完全一致用の SDSL が最速であるが、これは問題設定が異なる。静的 ceBWT は 2,000 パターンの検索を 0.084 秒で処理し、OPM-FM の multi-index 構成 1.512 秒および one-index 構成 0.150 秒より速かった。動的 ceBWT は 3.239 秒であり、動的構造の rank/select コストが支配的であった。

KMP 型 CTM 基準との比較では、すでに構築済みの ceBWT 索引を用いる場合、静的 ceBWT は 0.08 秒、動的 ceBWT は 3.24 秒であり、KMP-PDE の 7.01 秒および KMP-CTS の 6.82 秒より高速であった。したがって ceBWT は、索引を長期保持し、多数のパターンを繰り返し検索する用途で有効である。一方で、

新しいデータに一回だけ検索する場合には、索引構築を必要としない KMP 型手法が自然な選択となる。

発見される照合数は、照合モデルの粗さを反映する。MIDI 実験では、完全一致索引は 4,418 件、OPM は約 7,038 件、CTM である ceBWT と KMP 基準は 9,745 件を報告した。これは CTM が値の完全一致ではなくデカルト構造を照合するためである。

5 まとめ

本論文は、ceBWT に基づく円形 CTM 索引の初めての体系的な実装評価を与える。静的索引は小さい索引サイズと高速な反復検索を実現し、特に構築済み索引を利用する検索用途で有効である。一方、動的構築は動的 wavelet 木、挿入時の比較値計算、および部分索引の順序付きマージにより重い。threaded 2/3-fold は実用的な改善を与えるが、マージが逐次的であるため、単にワーカ数を増やしても十分な加速は得られない。今後の課題は、動的 rank/select 構造の高速化、マージ処理の並列化、および反復的データに対するより制御された評価である。

References

- [1] P. Ferragina and G. Manzini, “Opportunistic data structures with applications,” Proc. FOCS, pp.390–398, 2000.
- [2] J. Kim, P. Eades, R. Fleischer, S.-H. Hong, C. S. Iliopoulos, K. Park, S. J. Puglisi, and T. Tokuyama, “Order-preserving matching,” Theoretical Computer Science, vol.525, pp.68–79, 2014.
- [3] S. G. Park, A. Amir, G. M. Landau, and K. Park, “Cartesian tree matching and indexing,” Proc. CPM, LIPIcs 128, pp.16:1–16:14, 2019.
- [4] S.-H. Kim and H.-G. Cho, “A compact index for Cartesian tree matching,” Proc. CPM, LIPIcs 191, pp.18:1–18:19, 2021.
- [5] G. Decaroli, T. Gagie, and G. Manzini, “A compact index for order-preserving pattern matching,” Software: Practice and Experience, vol.49, no.6, pp.1041–1051, 2019.
- [6] S. Gog, T. Beller, A. Moffat, and M. Petri, “From theory to practice: Plug and play with succinct data structures,” Proc. SEA, LNCS 8504, pp.326–337, 2014.
- [7] N. Prezza, “A framework of dynamic data structures for string processing,” Proc. SEA, LIPIcs 75, pp.11:1–11:15, 2017.
- [8] T. Gagie, G. Navarro, and N. Prezza, “Fully functional suffix trees and optimal text searching in BWT-runs bounded space,” Journal of the ACM, vol.67, no.1, pp.2:1–2:54, 2020.
- [9] E. M. Osterkamp and D. Koepl, “The ceBWT index: An index for circular Cartesian tree matching on multiple texts,” Theory of Computing Systems, vol.70, no.27, pp.1–27, 2026.