technische universität dortmund

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung

An Automaton-Based View on Error-Tolerant Pattern Matching with Backward Search

Dominik Kopczynski* Sven Rahmann†

*Collaborative Research Center SFB 876, Computer Science XI, TU Dortmund, Germany dominik.kopczynski@tu-dortmund.de

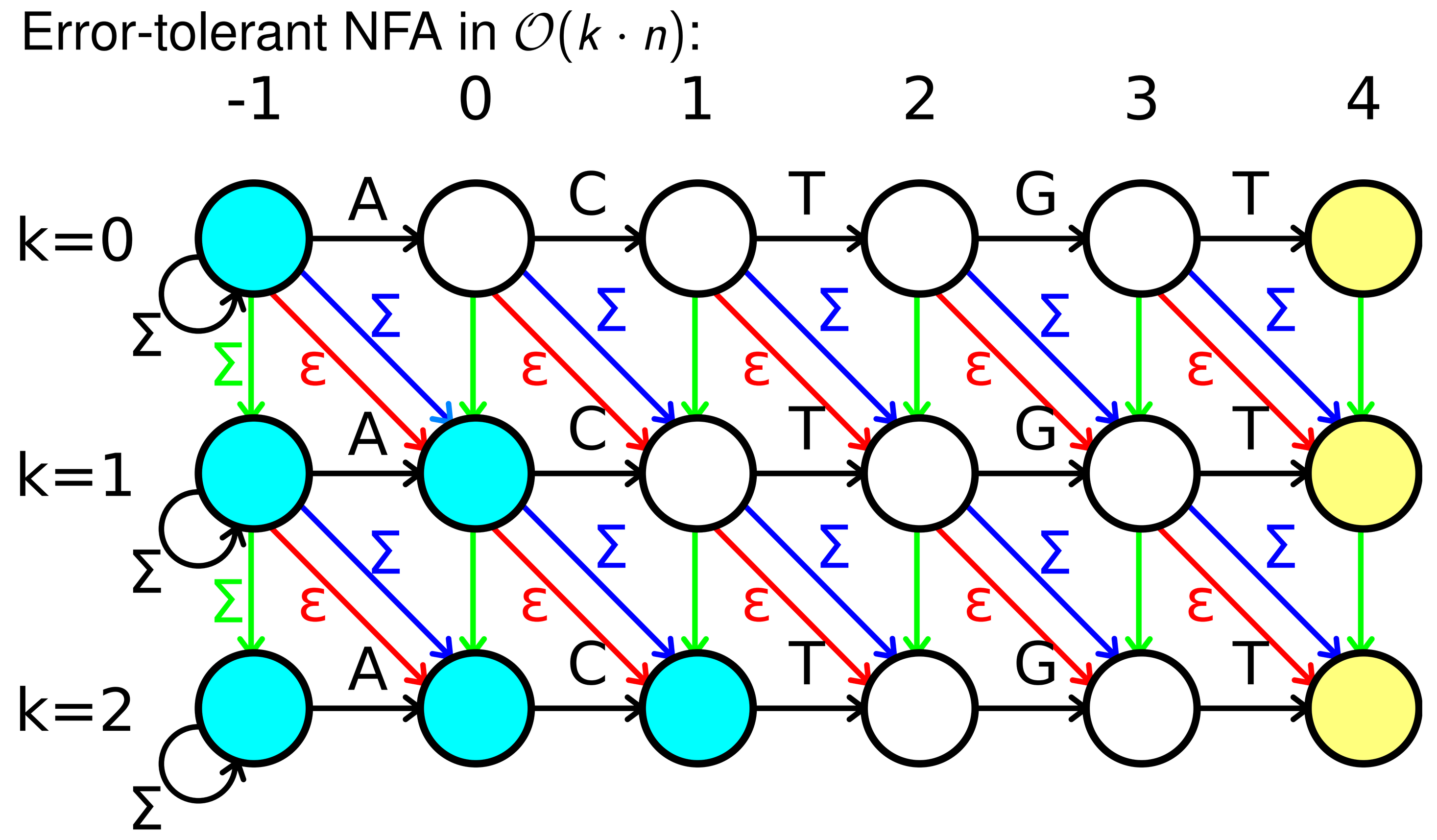
† Genome Informatics, Institute of Human Genetics, Faculty of Medicine, University of Duisburg-Essen, Germany sven.rahmann@uni-due.de

Introduction: Backward search is used as a computational core in many read mapping applications in the context of next generation sequencing data analysis. Here we introduce an automaton-based view on error-tolerant backward search by combining the non-deterministic finite automaton from the error-tolerant NFA with exact backward search. This leads to a conceptually simple, efficient and easily implementable version of error-tolerant backward search.

Background

Input: text T, pattern P, $n = |T|$, $m = |P|$, k errors at most
Output: all occurrences of P in T with $0 \le i \le k$ errors

Error-tolerant NFA in $\mathcal{O}(k \cdot n)$:



Exact *Backward Search* [1] in $\mathcal{O}(m)$:

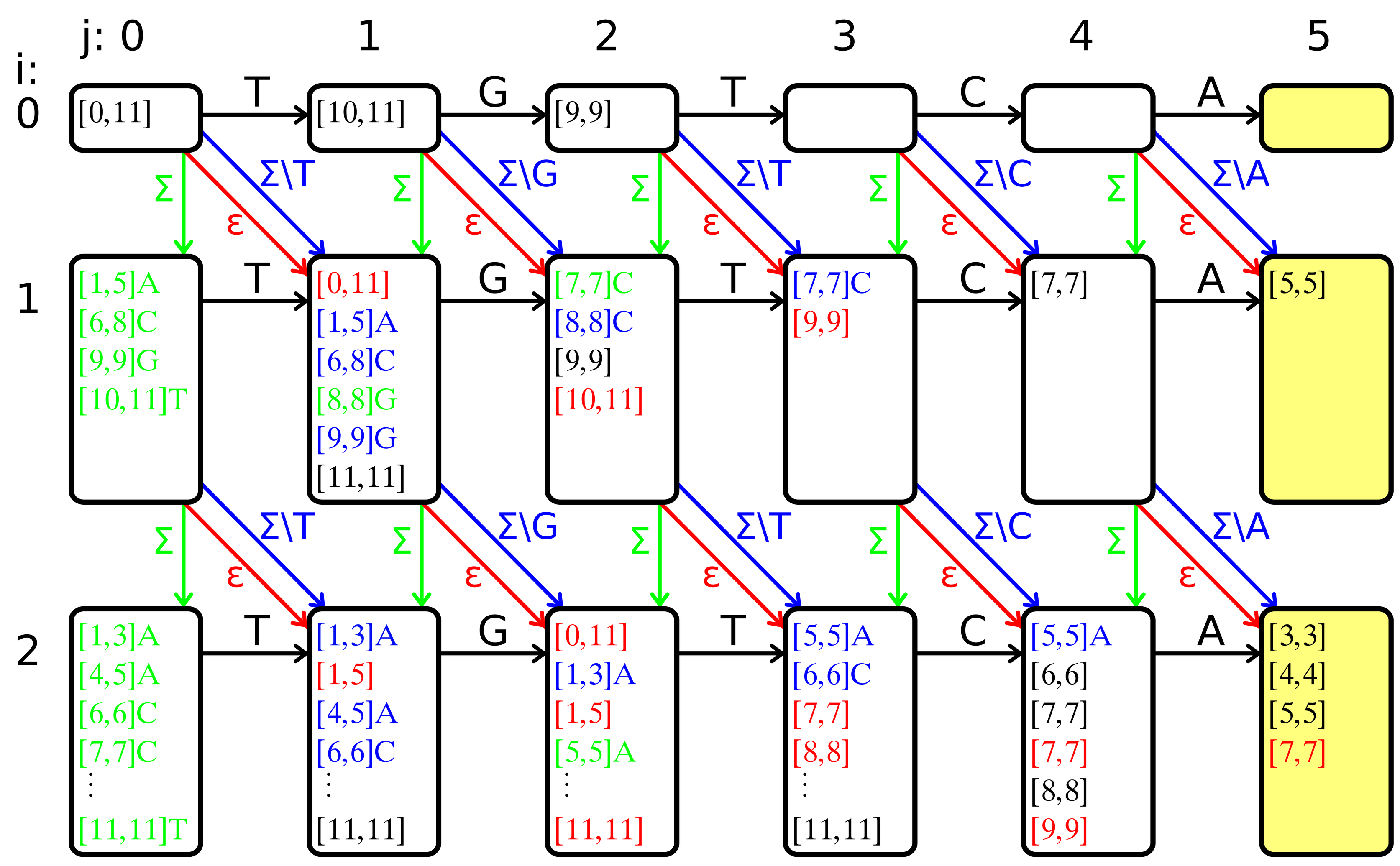
- Uses *suffix array* `pos` of T and *Burrows-Wheeler transform* (BWT)
- Needs auxiliary tables:
  - `less[c]`: number of characters in T lexicographically smaller than c
  - `occ[c][r]`: number of c's in BWT up to index r
- Updates an interval containing possible suffixes in `pos`
- Starts with whole interval $L = 0$, $R = n - 1$ for empty pattern
- Updates interval processing reversed pattern, using:

$$L^+(c) = \texttt{less}[c] + \texttt{occ}[c][L-1]$$
$$R^+(c) = \texttt{less}[c] + \texttt{occ}[c][R] - 1$$

Automaton BS

Automaton-based error-tolerant *Backward Search*:

- Initialize empty matrix M with $(k+1) \times (m+1)$ nodes
- Use reversed pattern P'
- Store full interval $[0, n-1]$ in node $M[0][0]$
- For every interval in every node:
  - If after BS update new interval is valid, perform:
    - A match with $c = P'[j]$ and store in $M[i][j+1]$
    - An insertion with $c \in \Sigma$ and store in $M[i+1][j]$
    - A substitution with $c \in \Sigma \setminus P'[j]$ and store in $M[i+1][j+1]$
  - Perform a deletion, store current interval in $M[i+1][j+1]$
- Example:
  - Text: `AAAACGTACCT$`, pattern: `ACTGT`, $k = 2$
  - No exact match, one match with single error, four matches with two errors



Implementation

Memory saving:
- Only two columns needed, current and subsequent column
- After processing current column all important data stored in subsequent column

Traceback:
- Needs complete matrix M
- Is applicable without considering `pos` and BWT after processing
- Auxiliary data must be stored per interval:
  - Its ancestor interval
  - Operation it was computed (mat, ins, del, sub)
  - Character involved in operation

Reasonable improvements for read mapping:
- Omit computation of first column, exponential growing, insertions at the left and right of a read not reasonable
- Restrict error bound for the first j matches
- Precompute lower bound errors for every suffix in P' (consider $D(\cdot)$ array in [2])

Conclusion: We presented a novel view on error-tolerant pattern matching using backward search, combining error-tolerant NFA with backward search. Certain improvements lead to a dramatically acceleration of computation time. This method is additionally well suited e.g. for teaching in class.

References

[1] P. Ferragina and G. Manzini. Opportunistic data structures with applications. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 390–398. IEEE, 2000.

[2] H. Li and R. Durbin. Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.