Sublinear Algorithms for Parameterized Matching

Leena Salmela **Jorma Tarhio**

July 7th, 2006

Outline

- Definition of parameterized matching
- New algorithms based on the Boyer-Moore-Horspool (BMH) algorithm
- Results of the analysis of the new algorithms
- Some experimental results

Parameterized Matching – Definition

1-dimensional problem:

- Given a pattern of length m and a text of length n find all substrings of text that can be transformed into the pattern by using a bijection on the alphabet.

2-dimensional problem:

- Given a pattern of size $m \times m$ and a text of size $n \times n$ find all $m \times m$ substrings of text that can be transformed into the pattern by using a bijection on the alphabet.

Definitions

Predecessor strings:

- If a character at position i has an earlier occurrence in the position j, the predecessor string contains $i - j$ at position i. Otherwise the predecessor string contains 0.
- Example: 'aabac' is transformed into 0-1-0-2-0
- If two strings p-match then their predecessor strings match exactly.

RGF strings:

- Transform the string into a Restricted Growth Function (RGF): Replace all occurrences of the first occurring character with 1, the second one with 2 and so on.
- Example: 'aabac' is transformed into 1-1-2-1-3
- If two strings p-match then their RGF strings match exactly.

Definitions

q-repetitive patterns:

- A 1-dimensional pattern is q-repetitive if for all substrings of length q there is a character that appears at least twice in the substring.
- A 2-dimensional pattern is q-repetitive if for all substrings of size $q \times q$ there is a character that appears at least twice in the substring.

New 1D Algorithms Based on Boyer-Moore-Horspool

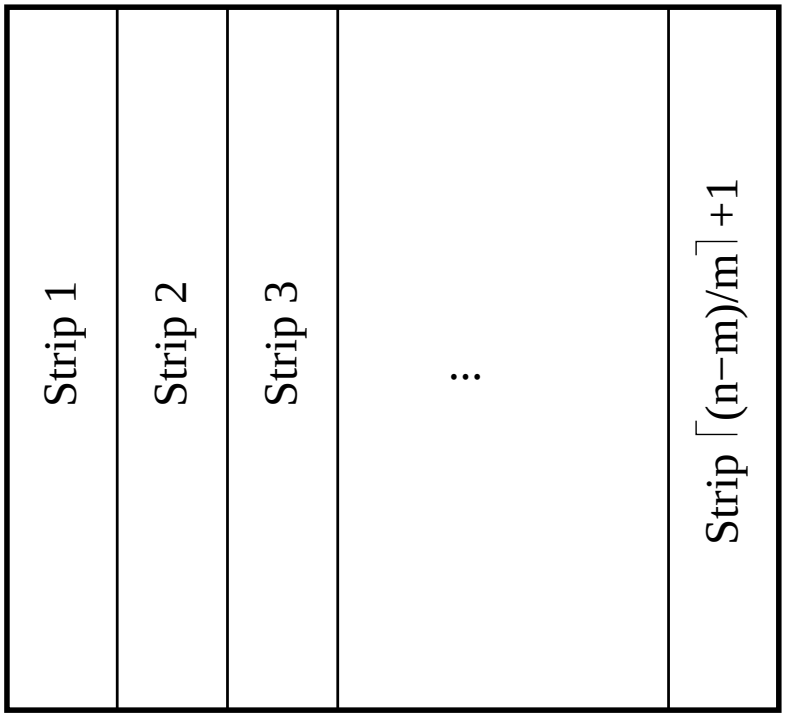
- Boyer-Moore-Horspool makes the shift based on the last character aligned with the pattern.
- In the parameterized version we make the shift based on the last q characters (q-gram) aligned with the pattern
- After the shift that q-gram will be aligned with the last q-gram of the pattern that p-matches it.

How do we index the shift table?

- Transform the q-gram into a RGF string and use the rank of the RGF string as an index. (PBMH-RGF)
  - Memory usage: b_q (the q:th Bell number)
  - The calculation of RGF rank takes some time.
- Transform the q-gram into a predecessor string and reserve enough bits in the index for each character of the predecessor string. (FPBMH)
  - Memory usage: 2^s where $s = \sum_{i=2}^{q} \lceil \log_2 i \rceil$
  - Fast to compute.
- Hashing scheme: Transform the q-gram into a predecessor string and add up all positions in the predecessor string. (PBMH-Hash)
  - Memory usage: $q(q-1)/2 + 1$
  - Memory efficient and fast to compute but two q-grams may have the same hash value thus reducing the length of the shift.

2D Exact String Matching Algorithm by Tarhio ...

- Divide the text into $\lceil (n-m)/m \rceil + 1$ strips:



- Search each strip with a BMH like algorithm and verify the candidates with the trivial algorithm.

. . . 2D Exact String Matching Algorithm by Tarhio

- Three tables used:
  - $M[x]$ is the position where x occurs first in the lowest row of the pattern.
  - N links the occurrences of x in the lowest row of the pattern. M and N are used to find the candidates that have to be verified.
  - $D[x]$ is the occurrence of x closest to the last row but not in the last row pattern. (Shift table)
- The algorithm can be modified to use q-grams ($q \times q$ substrings).

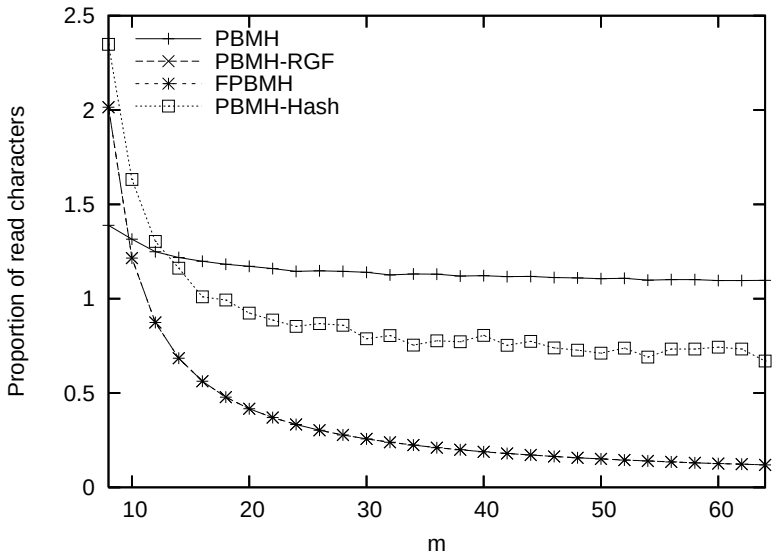
Generalization to Parameterized Matching

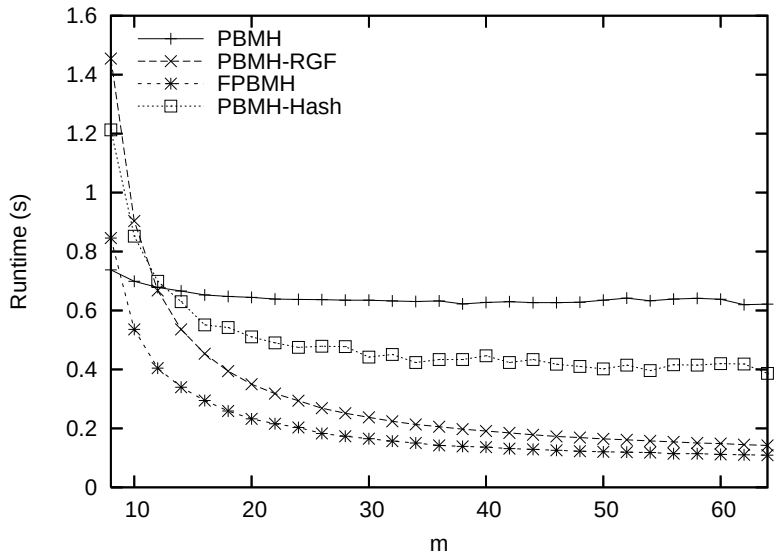
- We generalize the algorithm that uses q-grams.
- Otherwise the algorithm stays unchanged but the q-grams are transformed to RGFs or predecessor strings to index the tables.

Analysis

- 1-dimensional algorithms:
  - Preprocessing: $O(\sigma + mq)$ (σ is the size of the alphabet) plus time to initialize the shift table which is $O(b_q)$ for PBMH-RGF, $O(q^{q-1})$ for FPBMH and $O(q^2)$ for PBMH-Hash.
  - Matching: $O(mn)$ worst-case complexity and $O((qn)/(m-q+1))$ average-case complexity for q-repetitive patterns which is sublinear if $q < (m+1)/2$
- 2-dimensional algorithm:
  - Preprocessing: $O(\sigma + m^2q^2)$ (σ is the size of the alphabet) plus time to initialize the shift table.
  - Matching: $O(m^2n^2)$ worst-case complexity and $O((q^2n^2)/(m-q)^2)$ average-case complexity for q-repetitive patterns which is sublinear if $q < (m+1)/2$

Experimental Results – 1D Algorithms





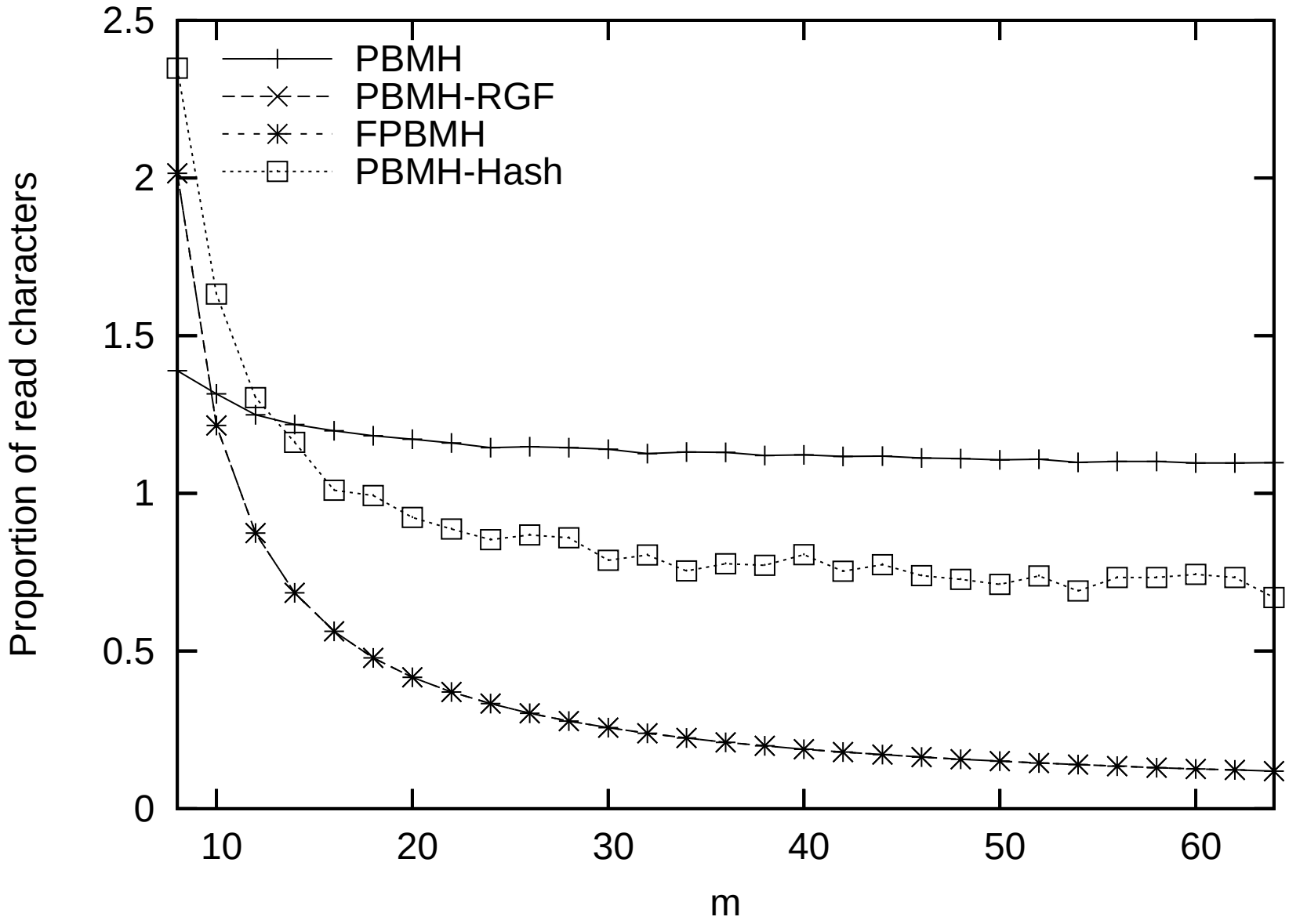
Proportion of read characters (left) and runtime (right) for a text of DNA data and patterns of varying length.

Experimental Results – 2D algorithm

Text	Single-character pattern	Pattern with no repetitions	Pattern with repetitions
Random	0.25	7.90	0.25
Map	1.14	0.25	0.33

Proportion of read characters for two different texts and several different patterns. All the patterns are of size 8×8.

Experimental Results – 1D Algorithms



Experimental Results – 1D Algorithms