

# An Empirical Analysis of Pattern Matching Algorithms

Dominic Moylett, supervised by Dr. Raphael Clifford, Dr. Markus Jalsenius and Dr. Ben Sach

University of Bristol, Department of Computer Science

## Introduction

In the wake of Big Data, a large amount of recent research in Theoretical Computer Science has been focused on the data streaming model. Under this model, the computer only has access to a window of the input at any time, and the objective is to compute the solution with a sublinear amount of space. There have been a number of theoretical developments in Data Streaming in recent years, but very few of these have ever been implemented to see if they are practical.

I have implemented two algorithms in C for stream-based pattern matching. Note the standard notation for pattern matching, where  $m$  is the length of the pattern and  $\Sigma$  is the alphabet:

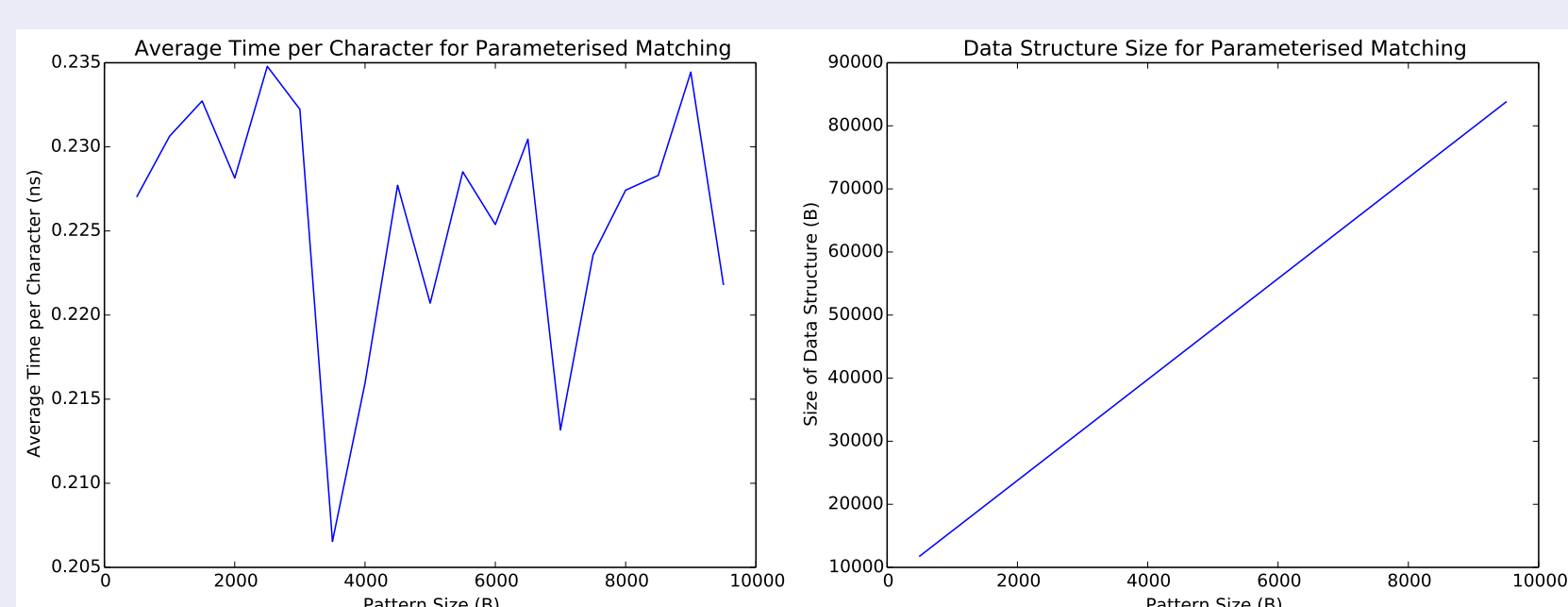
1. An algorithm for parameterised matching with  $O(m + \pi)$  space and  $O(\log \pi)$  amortised time per character, where  $\pi = \min(m, |\Sigma|)$  by Amir et al.<sup>a</sup>
2. An algorithm for exact matching with  $O(\log m)$  space and  $O(1)$  time per character by Breslauer and Galil,<sup>b</sup> and deamortised by Simon's Algorithm.<sup>c</sup> Sublinear space is achieved via Karp-Rabin fingerprints. All of these algorithms have been tested on 50MB of English text from the Pizza and Chili Corpus.

<sup>a</sup>Alphabet Dependence in Parameterized Matching by Amihoud Amir, Martin Farach and S. Muthukrishnan

<sup>b</sup>Real-Time Streaming String-Matching by Dany Breslauer and Zvi Galil

<sup>c</sup>String Matching Algorithms and Automata by Imre Simon

## Parameterised Matching Results

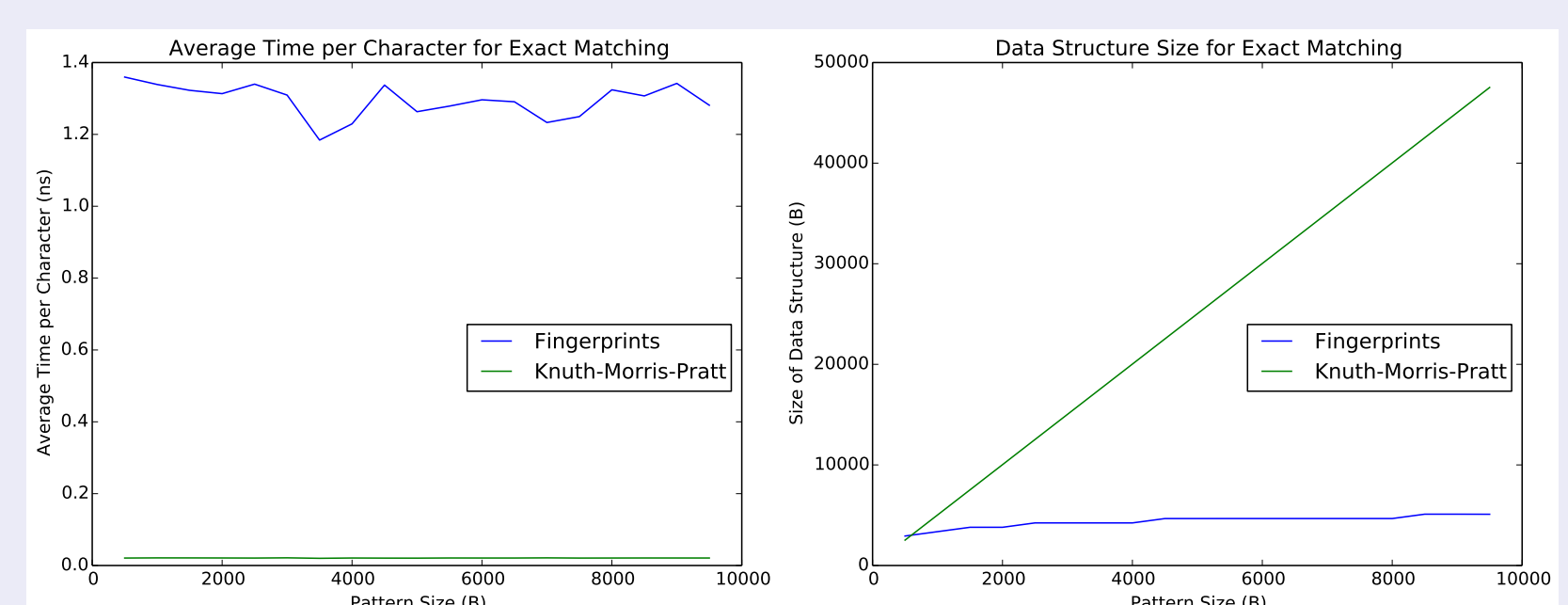


## Parameterised Matching Conclusions

The main bottleneck in Parameterised Matching is that previous occurrences of characters in the text are stored in a search tree, which takes  $O(\log \pi)$  time to query and edit.

Question: Can this be improved with dynamic hashing, for which the best results are  $O(\sqrt{\log \pi / \log \log \pi})$ ?

## Results for Exact Matching



## Exact Matching Conclusions

Exact matching with Fingerprints takes 60-70 times longer per character and a significantly longer build time than Knuth-Morris-Pratt, but requires less space than even storing the pattern.

Question: The practical bottleneck is that the fingerprints use a lot of modular multiplications. Could these be optimised via precomputation or Montgomery Multiplication?

