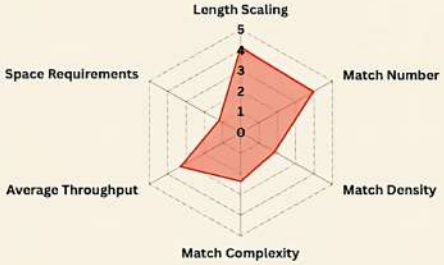
# AHO-CORASICK : THE LIBRARIAN

**Matches patterns like a librarian who knows every book's contents by heart. As each word comes in, they instantly know exactly where it belongs. Fast and accurate, but needs lots of memory.**

## Real life Example

Snort uses an optimized Aho-Corasick automaton and it is also used in text filtering, spell checking, DNA sequence matching and in search engines for matching on large datasets. Its deterministic worst-case makes it predictable so it is the industry standard for NIDS.




**Length Scaling**: Performance remains strong as patterns lengthen because the automaton structure handles size efficiently.
**Match Number:** The algorithm maintains steady stability regardless of how many total matches are found in the text.
**Match Density:** It handles dense clusters of matches well due to its single-pass processing nature.
**Match Complexity:** It scores low here, as it is optimised for exact string matching rather than complex logic.
**Average Throughput:** Speed is high despite the heavy memory access overhead limits raw processing speed.
**Space Requirements:** This is the highest cost, as storing the massive state transition table consumes significant memory.


Aho, A. V., and Corasick, M. J. (1975). Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6), 333-340.



## SUMMARY

As every network packet passes through, the network intrusion detection system (NIDS) carefully monitors the packet for malicious content. Within this, one element is key to the safety of the network, the pattern-matching engine. The efficient operation of such an engine is fundamental to the success of detecting packets containing malicious payloads without slowing network speeds. Signature-based heuristics, the current technology utilised, is no longer sufficient to cover the broad range of network attacks without tradeoffs in time or space, with focus now on future development of artificial-intelligence supported string matching algorithms. Thus, our investigation delves into a comparative analysis by implementing canonical algorithms, benchmarking using current Snort Pattern Rulesets of varying size and length to analyse critical components of performance including; preprocessing time, space utilised, and search time when parsing network packets. This investigation allows us to conclude improvement aspects required and successful elements of each algorithm to drive future development of pattern-matching in NIDS.


# SET-HORSPOOL : THE BRICKIE

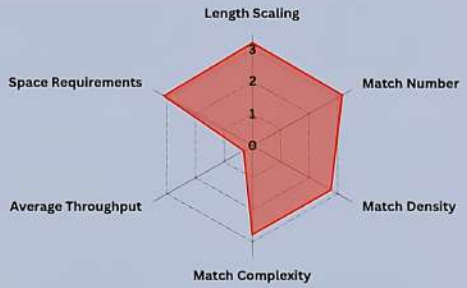
JB

**Builds a wall one brick at a time in small, steady steps. Slow, but every line is perfectly aligned. Prioritises accuracy over speed.**

## Real life Example

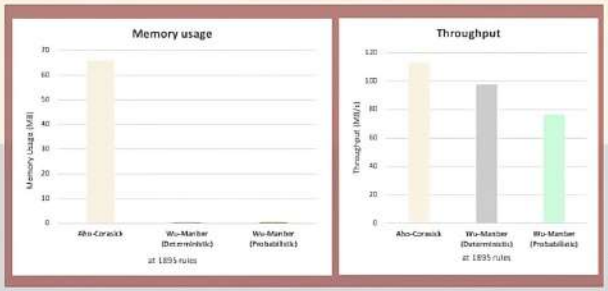
Initially proposed specifically for NIDS and signature matching in medium sets, it is not widely used outside of this scenario. Currently used within the research field for hybrid NIDS which decide when to use SH in its optimal medium-sized sets. Performs well in our datsets for that very reason.

**Length Scaling:** It sees a modest benefit from longer patterns allowing for slightly larger shifts.
**Match Number**: The score is low because performance degrades quickly if too many matches interrupt the shifting process.
**Match Density:** It performs poorly in dense text as it cannot skip ahead effectively.
**Match Complexity:** Designed for simple tasks, it lacks the capability to handle complex pattern requirements.
**Average Throughput:** The processing speed is slow due to its methodical, "brick-by-brick" comparison approach.
**Space Requirements:** This is its greatest strength, requiring almost no memory overhead to function.





Boyer, R. S., Moore, J. S. (1977). A Fast String Searching Algorithm. *Communications of the ACM*, 20(10), 762 - 772

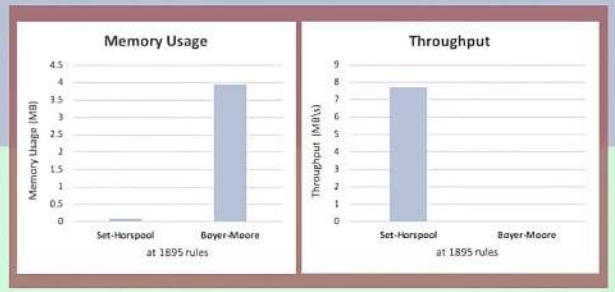




# THE DETECTION WARDEN

## STRING MATCHER

A Comparative Analysis of Multi-Pattern Matching Algorithms for Network Intrusion Detection Systems


**Group 8:**
Tom Chandler
z5479498
Angela He
z5589611
Jonty Bornstein
z5591196
Maggie Zhang
z5591939
Gururam Subramanian
z5636559

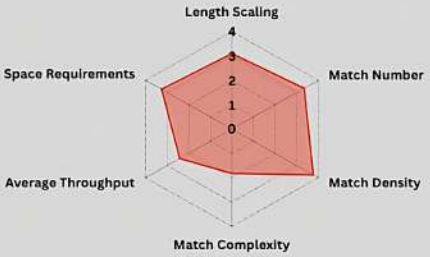

# WU-MANBER DETERMINISTIC

## THE AIRPORT SCANNER

**Processes every bag carefully and never makes mistakes, but the queue gets long. High accuracy, slower speed.**

## Real life Example

Used in WM-q for genomic data to do multiple exact string matching. Also, adapted for language morphology in Uyghur. The deterministic property is useful for fast matching in fixed dictionaries.




**Length Scaling:** It scales reliably well by using block-based hashing to skip sections of text.
**Match Number**: It handles a moderate volume of matches without losing its "careful" scanning accuracy.
**Match Density:** Performance is balanced in dense text, ensuring accuracy is maintained over speed.
**Match Complexity:** Like other exact matches, it scores rather low, focusing on precision over complex rules.
**Average Throughput:** Throughput is moderate, as the focus is on guaranteed accuracy rather than raw speed.
**Space Requirements:** Memory usage is moderate, striking a balance between the "Brickie" and the "Librarian.


Wu, S., and Manber, U. (1994). A fast algorithm for multi-pattern searching. *Technical Report TR 94-17*, Department of Computer Science, University of Arizona.


## CONCLUSION

Through our comparative analysis, we found that each string-matching algorithm offers useful advantages for network intrusion detection systems, but all still face space and time limitations. These constraints persist even as algorithms improve, highlighting ongoing challenges in NIDS pattern-matching engines. As risks involving unprecedented, non-signature based attacks linger in the future, we suggest the following future work: (1) Continued experimental analysis under real-time systems to gain real-time tracking and comprehension of algorithm cost, and (2) The development of heuristics-based detection utilising Neural networks, support vector machines and decision trees. Now, the question lies: **how can we integrate these technologies into string-matching algorithms to ensure accurate detection?** AI will undoubtedly play a growing role in this field, but it should not become the sole point of reliance in such a promising domain.

# WU-MANBER PROBABILISTIC

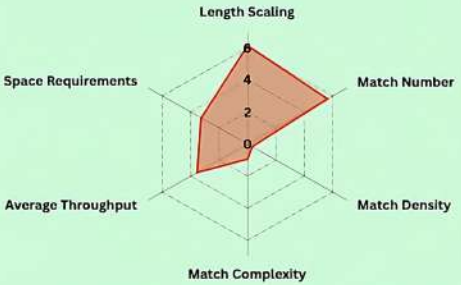
## THE SNIFFER DOG

**A sniffer dog that quickly flags suspicious bags. It sometimes barks at snacks, but catches almost everything important. faster, with low false positives.**

## Real life Example

There are research systems which have implemented and measured probabilistic WM with bloom filters in signature-based IDS environments but this method is not used commercially. Only used in research contexts at the moment.

**Length Scaling:** This category is maximum because longer patterns allow for massive jumps in the text.
**Match Number:** It easily handles high match counts by filtering out safe data rapidly.
**Match Density:** It excels in dense environments, effectively "sniffing out" clusters of interest.
**Match Complexity:** It handles complexity well by using hashing to approximate matches quickly.
**Average Throughput:** It offers the higher speed by ignoring non-suspicious data entirely.
**Space Requirements:** Memory usage is surprisingly low, because it uses compact Bloom filters instead of full tables.





Aldwairi and K. Al-Khamaiseh, Exhaust: Optimizing Wu-Manber pattern matching for intrusion detection using Bloom filters, *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, Sousse, Tunisia, 2015, pp. 1-6, doi: 10.1109/WSWAN.2015.7209081.